Software Requirements Specification

for

REIYA

Version 1.1

Prepared by: Team Tessman

IIIT Kottayam, Kerala

11/10/2020

Table of Content

Revision History

Name	Date	Reason For Changes	Version
First revision of SRS	30/09/20	Initial SRS Document	1.0
Altered SRS as required.	11/10/20	Changed Title page Removed table of contents from table of contents Cut down on functional requirements Edited references Moved items from interfaces to operating environment Modified functional requirements to tabular form wherever possible	1.1

1. Introduction

1.1. Purpose

This is the Software Requirements Specification (SRS) document (Version 1.0) for the game Reiya. The purpose of this document is to communicate the requirements which are functional, non-functional and others of this Game to the reader.

This document provides:

a) An overall description of the game Reiya.
b) A definition of the external interface requirements.
c) A specification of the system features (functional requirements), non-functional requirements and other requirements.

1.2. Document Conventions

As the Development team is responsible for the SRS Document, no ambiguity arises from its usage. There is a clear distinction, however, between the use of the word 'player' and 'character'. The 'player' is the human being interacting with the game in the real world while the character is the in-game player avatar being manipulated. This document is too printed on A4 paper in Time New Roman font. Normal text size is 11 blocks while section headings are size 18 bolded blocks. Subheadings are size 14 bolded blocks and third subheadings are size 12 bolded blocks.

1.3. Intended Audience and Reading Suggestions

The Intended Audience for this document are Developers, Game Modders, Project Managers, Marketing Staff, Users, Testers, and Documentation Writers.

The next section (Section 2) provides an overall description of the game Reiya which includes the product perspective, its functions, user classes along with its characteristics, the operating environment, design and implementation constraints, user documentation, and assumptions and dependencies of the game Reiya. Section 3 gives information on the external interface requirements of Reiya. This includes user interfaces, hardware interfaces, software interfaces and communication interfaces.

System features and their functional requirements are specified in Section 4.

All non-functional requirements regarding safety, security, quality and business attributes of Reiya are specified in Section 5.

Other additional requirements are given in Section 6.

All the formal terminologies used in this SRS document are defined in Appendix A: Glossary.

All the models developed as part of system analysis are described in Appendix B: Analysis Models.

All the To-Be-Determined items are given in the Appendix C: TBD List.

1.4. Product Scope

The game Reiya is an interactive 2-D Puzzle Platformer computer game. The goal in this game is to complete all the levels. The objective is to enhance the imaginative vision of the player along with the focus to solve the puzzle while having fun as the player have to imagine a completely new dimension which is inaccessible to the eyes.

1.5. References

1.5.1. IEEE recommended practices for SRS:
https://www.midori-global.com/downloads/jpdf/jira-software-requirement-specification.pdf

Author: John Doe
Revision: 29/12/2011

1.5.2. SRS for Dragon Adventure Game

http://www.dgp.toronto.edu/~ppacheco/course/444/spec.pdf
Document # CSC444-SRS-001A

Author: unknown
Revision: A

2. Overall Description

2.1. Product Perspective

The game mechanics of Reiya takes a dive into the nature of understanding higher dimensions within the constraints of lower dimensions. The game is set in a 3D world where the player controls a character who lives in a 2D cross-section of the world with the special ability to navigate through the third spatial dimension. The main objective of the player being to navigate through the 3D world that it cannot directly observe, and solve puzzles to get to higher levels.
It takes inspiration from classics from the 90s such as Dave and Mario and will contain largely the same abilities for the player such as moving left and right, jumping and the main additional skill being the ability to move through an extra dimension.
As with such games, Reiya will be a single-player game but it will be open-source and as such encourages players and developers to modify it and/or make their own levels pertaining to their interests.

2.2. Product Functions

There are essentially 3 main functionalities:

- Menu and Level Selector: The player is introduced to the main menu when the game starts where he/she can access levels, settings etc. and an automatic system that progresses to the next level within the game as you clear them.

- Character management: The program uses input from bound keys of the keyboard to control the motion of the character. Interaction between the player and the character in game is done in real-time where key presses result in an action taken by the character such as motion in the cross-section such as walking towards left and right, jumping and changing the value of the third axis to move across different cross-sections of the game.

- Collision management: The character and objects in the game have to interact visually and for this purpose collisions are to be defined so that the character can jump on objects and navigate through the world without falling off it or phasing through objects.

2.3. User classes and characteristics

The game should be playable by any players. No special knowledge or skill is assumed on the part of the players. Players are expected to have minimum hardware required to run the game. They are also required to learn the basic controls at the start of the game.

2.4. Operating Environment

The game will be set to run on a PC environment that either runs windows or Linux operating systems. The computer must have hardware that supports OpenGL rendering as that is the primary method used by Godot to render the objects.
Godot v3.0 is required to play the game on these operating systems. No other software is required for the program to run on the Device.

OpenGL capable GPUs (mostly NVIDIA GPUs) can offer better/faster graphics capabilities
Normal GPUs can play it on graphics with less details and cosmetics (effects)
The android platform is graphically adaptable with 2D graphics library and a 3D graphics library based on OpenGL ES 2.0 specifications as well as hardware orientation and accelerated 3D graphics

All additional libraries will be provided by the Godot build system.
Open Gl 2.1/Open Gl ES 2.0 is required if the build system is not being used or if Mono version of Godot Engine is used.
CSG (Constructive Solid Geometry): It is a tool to combine basic shapes or custom meshes to create more complex shapes. In 3D modelling software, CSG is mostly known as "Boolean Operators". In this game it is being used to make shapes and to make cross sections of those shapes dynamically.
GDSript: The programming language used in Godot. It is similar to python in syntax but it converts dynamically to c++.

2.5. Design and Implementation Constraints

No regulatory restrictions will be set on developing and distributing the game as we are exclusively using open-source software and licensing the game under GNU General Public License v3.0.
The only constraints that exist will be that of missing functionalities in godot compared to other proprietary game development engines.

2.6. User Documentation

README.MD
GitHub repository: https://github.com/tessman322/Reiya

2.7. Assumptions and Dependencies

The primary assumption here is that all PCs will be able to handle it. This could very well end up being untrue as a moderate GPU is required to render meshes and calculations regarding them in real time. Trying to use a CPU for these tasks could result in glitches.

CSG is relatively new concept in terms of meshes and can be fairly buggy during runtime and can result in artifacts and glitches.

3. External Interface Requirements

3.1 User Interface:

It is a game with graphical user interface.

- **Menu screen:** For selecting levels, volume controls, graphics settings, video settings, composition(full-screen or windowed mode), quitting the game.
- **Movement controls:** Arrow keys for movement for the player. W key for moving in positive Z-axis, S key for moving in negative Z-axis. Space-bar for jump. Q for dash (optional requirement), E for pushing objects (optional requirement)
- **Z-axis slider(optional requirement):** Same functionality as W and S keys in movement controls but controlled with either a mouse or touch interface which also gives information about current position in Z-axis.
- **Pause screen:** Same as Menu screen with additional option of resuming the game.
- **Level selector:** Levels to select on a grid with personal best time under each level.

3.2 Hardware Interfaces:

No Hardware Interfaces will be used.

3.3 Software Interfaces:

No Software Interfaces will be used

3.4 Communications Interfaces:

None (HTTP with Sockets for multiplayer(optional)) (No plans for multiplayer or external communications finalized, it will be updated)

4. SYSTEM FEATURES

4.1. General Movement

4.1.1. Description and Priority

This is also an important part of game-play and has the highest priority. This mechanic allows the player to move the character left of right and interact with part of objects that are in the same dimension as him.

4.1.2 Stimulus/Response Sequences

Step 1: The player presses arrow keys or the custom keys set by the player.
Step 2: The character moves on the direction assigned to the key with constant velocity.

4.1.3 Functional Requirements

4.1.1 General Movement/Move player	
Function	Move player
Description	allows the player to move the character left of right and interact with part of objects that are in the same dimension as him
Inputs	Left, Right Arrow keys
Source	Keyboard
Outputs	Player is moved in the direction indicated by arrow key
Destination	Screen
Action	If player can be moved in the desired direction, move. Otherwise, do not move.
Requirements	None
Pre-condition	None
Post-condition	None
Side-effects	None

4.2. Inter-Dimensional Movement

4.1.1. Description and Priority

This is the primary method of player input and has the same priority as general movements. This mechanic allows the player to move the character through the dimension inaccessible to him directly and interact with part of objects that character encounter during the motion.

4.2.2 Stimulus/Response Sequences:

Step 1: The player presses up/down arrow keys or the custom keys set by the player or uses the control buttons shown on screen.
Step 2: The character moves on the direction assigned to the key with constant velocity.

4.2.3 Functional Requirements

4.2.1 Inter-Dimensional Movement/Move across dimensions	
Function	Move player across dimensions
Description	allows the player to move the character to and fro in a dimension and allows to interact with part of objects that are in the same dimension as him
Inputs	Up, Down arrow keys
Source	Keyboard
Outputs	Player is moved in the direction indicated by the key
Destination	Screen
Action	If player can be moved in the desired direction, move. Otherwise, do not move.
Requirements	None
Pre-condition	None
Post-condition	None
Side-effects	None

4.3. Jumping

4.3.1. Description and Priority

Being the secondary method of character input, jumping maintains the second-highest game-play priority. This mechanic allows the character to navigate gaps without getting stuck.

4.3.2. Stimulus/Response Sequences

Step 1: The player presses the jump button located on the side of the screen.
Step 2: The character jumps with a constant force.

4.3.3. Functional Requirements

4.3.1 Jumping/Jump	
Function	Make player jump up
Description	allows the player to jump and interact with part of objects that are in the same dimension as him
Inputs	Space key
Source	Keyboard
Outputs	Player is made to jump up
Destination	Screen
Action	If player can be moved in the desired direction, move. Otherwise, do not move.

Requirements	None
Pre-condition	None
Post-condition	None
Side-effects	None

4.4. Death Zones

4.4.1. Descriptions and Priority

Death zones are the opposite of level completion zones in that when touched, they send the character back to the starting point of the level.
Some death zones take on the form of environmental obstacles; others are zones with dangerous game characters which are impossible to defeat with current tool-set of our character. These zones increase the game's difficulty and incentivize good performance via negative reinforcement. They should be prioritized in level design, as they bring the game-play from "playable" to "enjoyable."

4.4.2. Stimulus/Response Sequences

Step 1: The character comes into contact with a death zone.
Step 2: The character is immediately returned to the beginning of the level.

4.4.3. Functional Requirements

4.4.1 Death zones/kill character	
Function	Kill the character
Description	Sends the character to the beginning of the level and reduce it's lives by one.
Inputs	No Input
Source	Designed inside the level
Outputs	Character dies when touched
Destination	Screen
Action	Player must evade it to complete the level.
Requirements	No death zone should be placed so as to make its respective level impossible to complete.
Pre-condition	No Condition
Post-condition	No Condition
Side-effects	No Side-effects

4.5. Title Screen

4.5.1. Descriptions and Priority

The title screen is the screen the player will see every time upon playing the game. Through this interface, the player can choose to start a new game, play from saved data, or adjust the options. Since the title screen is the “hub” for all activities in the project, it must be included.

4.5.2. Stimulus/Response Sequences

Step 1: The player launches the game from their portable device.
Step 2: The start screen loads and appears, prompting the player with three buttons “Play Game”, “Options”, and “Exit”.
Step 3: The player presses one of the buttons, triggering its respective function.

4.5.3. Functional Requirements

4.5.FR1. Play Button	
Function	Display the selected option
Description	Allows the user to select an option by clicking on it or navigating to it by keyboard and will take the user to the level selector
Inputs	Play Game button
Source	Button displayed on the screen
Outputs	None
Destination	Level selector menu
Action	Launches the level selector
Requirements	None
Pre-condition	None
Post-condition	None
Side effects	None

4.5.FR2. Options button	
Function	Display the options menu
Description	Allows the user to enter the settings screen.
Inputs	Options button
Source	Button displayed on the screen
Outputs	None
Destination	Options/Settings menu
Action	Launches the settings screen
Requirements	None
Pre-condition	None
Post-condition	None

Side effects	None

4.5.FR3. Exit	
Function	Exit game
Description	Displays a confirmation message before exiting game
Inputs	Exit button
Source	Button displayed on the screen
Outputs	None
Destination	None
Action	Closes the game
Requirements	None
Pre-condition	None
Post-condition	Progress is saved
Side effects	None

4.6. Pause Menu

4.6.1. Descriptions and Priority

The player should be able to pause anytime during game-play, and this screen fulfils that requirement. The pause menu also allows the player to navigate between game-play and the level selection and title screens. The portable nature of the console renders player convenience paramount, so this feature must be included.

4.6.2. Stimulus/ Response Sequences

Step 1: the player presses the pause button on in game interface.
Step 2: the level pauses, drawing up the pause menu which prompts the player with three options “Resume game”, “options”, and “exit game”.
Step 3: the player presses one of those buttons, triggering it’s respective function.

4.6.3. Functional Requirements

4.6.1 Pause Menu/return to title screen	
Function	Returns to title screen
Description	Quits level and returns to title screen
Inputs	‘Return to title screen’ button
Source	Button displayed on pause menu
Outputs	None
Destination	Title Screen

Action	Quits the level and returns to title screen
Requirements	None
Pre-condition	None
Post-condition	None
Side effects	Progress is saved in quicksave

4.6.2 Pause Menu/resume	
Function	Resumes the game
Description	Resumes the level
Inputs	‘Resume’ button
Source	Button displayed on pause menu
Outputs	None
Destination	Continued level
Action	Closes pause menu and resumes the level
Requirements	None
Pre-condition	None
Post-condition	None
Side effects	None

5. Other Nonfunctional Requirements

5.1. Performance Requirements

Based on the capabilities of current computer systems, performance should not be an issue. However, computers with older hardware may incur some difficulties and potentially run slower.

5.2. Safety Requirements

No Safety Requirements

5.3. Security And Privacy Requirements

No security and privacy requirements as the game does not request personal data.

5.4. Software Quality Attributes

To ensure reliability and correctness, the game will respond to the player’s commands in a timely manner. When the player makes a jump the gravity and velocity components would response immediately, the player should see the effects of this command in the next frame.

For adaptability and flexibility, the game will automatically save the player's progress after every level completion. That way, in the event that the player's device runs out of battery during game-play, the player can resume game-play at a reasonable starting point.

In terms of usability, the Graphical User Interface will be very intuitive for the player to use. The beginning levels of the game will also slowly introduce each of the unique commands available to the player as well as any other game mechanics the player may need to know to complete the level. In more complicated and elaborate levels, these introductions and hints will not be available because the player will have already been introduced to all the tools needed to complete the level. This method will ensure that the player gradually learns all the game mechanics and also enjoys a challenging game-play experience.

Our game focuses on both ease of use and ease of learning while not leaning towards one or the other. the game should be a game that any player can pick up and play instantly without much downtime in figuring out the controls. The control will be easy and intuitive to use and the player will not be bombarded by having to use all the controls at once in the beginning levels. The game play will ease the player into learning and using each of the commands by gradually introducing each command as the player progresses through the game and completes each level. By the end of any given level, the player should be familiar with and fully able to use the command introduced in that level and all the commands introduced in previous levels.

Appendix A

Glossary

1) CSG: Constructive Solid Geometry is a technique used in solid modeling. In 3D computer graphics and CAD, CSG is often used in procedural modeling. CSG can also be performed on polygonal meshes, and may or may not be procedural and/or parametric.

2) Platformer: Platformers / platform games is a sub-genre of action games. They are characterized by their heavy use of jumping and climbing to navigate the player's environment and reach their goal. Levels and environments tend to feature uneven terrain and suspended platforms of varying height that requires use of the character's abilities in order to traverse.

3) GDScript: is a dynamically typed and a high-level programming language. It is used to create content. It is optimized and tightly integrated with Godot Engine, allowing great flexibility for content creation and integration.

4) Mesh: In 3D computer graphics and solid modeling, a mesh is a collection of vertices, edges and faces that defines the shape of a polyhedral object. The faces usually consist of triangles, quadrilaterals, or other simple convex polygons, since this simplifies rendering, but may also be more generally composed of concave polygons, or even polygons with holes.

5) Libre software: Libre means free in Spanish but not free of cost. Gratis is Spanish for "no cost," while libre means free of restrictions, more akin to freedom.

Appendix C
(To be Determined)

- Multiplayer support
- Wall jumping
- Dialogue management
- Fights and action sequences