SE 3XA3: Requirements Specification
Rogue Reborn

Group #6, Team Rogue++

Ian Prins	prinsij
Mikhail Andrenkov	andrem5
Or Almog	almogo

Due Friday, October 7th, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
09/28/16	1.0	initial setup
10/02/16	1.0	Continued setup

This document describes the requirements for .... 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 goal of the project is to produce a reimplementation of the original Rogue computer game, originally developed by Michael Toy, Glenn Wichman, and Ken Arnold in 1980. The gameplay of the reimplementation should mimic that of the original whenever possible. The objective of the rewrite is to produce a copy in a modern language, using modern design principles, with superior documentation and a full test suite. The original Rogue is of historical interest as it forms the foundation and is the namesake of the roguelike genre of games, typified by their randomized environments, difficulty, and permadeath features. The motivation for this project is the poor condition of the original source code. The original source was not written with readability in mind, and designed for extremely low-performance systems who required some unusual design patterns. The version of C in which it was written is very old, which hinders compilation or feature extension. The intended audience for this document is the stakeholders of this project, especially Dr Smith and the 3XA3 TAs.

1.2 The Stakeholders

1.2.1 The Client

The client of the project is Dr Spencer Smith. Dr Smith commissioned the project and will be overseeing its production. Dr Smith provides the specifications for this document, as well as other aspects of the project, including the test suite, and all documentation. In addition he will be evaluating the final product.

1.2.2 The Customers

The project customers are the players of the game. It is expected that this will consist primarily of players of the original, as well as players and developers of later roguelike games. The roguelike community has a strong open-source tradition, so a modern, well-documented Rogue could be a valuable starting point or inspiration for projects by other teams.

1.2.3 Other Stakeholders

1.3 Mandated Constraints

1.4 Naming Conventions and Terminology

1.5 Relevant Facts and Assumptions

User characteristics should go under assumptions.

2 Functional Requirements

2.1 The Scope of the Work and the Product

Not sure about this section

2.1.1 The Context of the Work

2.1.2 Work Partitioning

2.1.3 Individual Product Use Cases

2.2 Functional Requirements

This section will specify the functional requirements of the Rogue++ project. They are numerous, scattered, and interdependent, therefore an attempt shall be made to organize them into cascading, logical segments.

2.2.1 Basic mechanics

- The player should be able to start a new game
- The player should be able to save the current game by name

- The player should be able to load previous games by name
- The player should be able to quit the game
- The player must always begin with the default level 1 hero
- The player must always see their hero's statistics
- The game must wait until the user takes an action to manipulate the environment

2.2.2 Interaction

- The player should be able to view detailed information about:
  - The hero
  - The surrounding environment
- The player should be able to pass the turn
- The player should be able to walk around
- The player should be able to open and close doors

2.2.3 The Dungeon

- The player must begin at the dungeon's first level
- The game must generate each dungeon level one at a time
- Each level must have a downwards staircase
- Every level must generate rooms, corridors, monsters, treasure, and traps
- The player must be able to see in a 3x3 square centered on the hero
- The player must be able to see the entire room the hero is in, if the hero is in a room
- The player should see the outline of dungeon areas previously explored
- The player should be able to search for hidden doors

- The player should not be able to see hidden doors without explicitly searching for them

2.2.4 Equipment

- The game should maintain an inventory of player items
- The player should be able to view the inventory
- The game should limit the player's inventory based on the weight of its contents
- The player should be able to add, drop, use, hold, and remove objects from the inventory
- Scrolls, rings, and wands should have meaningless names until identified
- The player should be able to identify items
- The player should not be able to remove cursed items

2.2.5 Combat

- Each monster must have its own statistics
- Each monster must calculate a plan of action
- Monsters must only attack the player, not other monsters

3 Non-functional Requirements

3.1 Look and Feel Requirements

3.2 Usability and Humanity Requirements

3.3 Performance Requirements

3.4 Operational and Environmental Requirements

3.5 Maintainability and Support Requirements

3.6 Security Requirements

3.7 Cultural Requirements

3.8 Legal Requirements

3.9 Health and Safety Requirements

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

4.2 Off-the-Shelf Solutions

4.3 New Problems

4.4 Tasks

4.5 Migration to the New Product

4.6 Risks

4.7 Costs

4.8 User Documentation and Training

4.9 Waiting Room

4.10 Ideas for Solutions

5 Appendix

This section has been added to the Volere template. This is where you can place additional information.

5.1 Symbolic Parameters

The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.