SE 3XA3: Requirements Specification
Rogue Reborn

Group #6, Team Rogue++

Ian Prins	prinsij
Mikhail Andrenkov	andrem5
Or Almog	almogo

Due Tuesday, October 11th, 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
10/07/16	1.1	Added Project Drivers
10/07/16	1.1	Added Functional Requirements and Risks
10/09/16	1.2	Added Non-Functional Requirements

This document describes the requirements for the Rogue Reborn project. The template for the Software Requirements Specification (SRS) is a subset of the Volere template (Robertson and Robertson, 2012). For the convenience of the readers, the sections pertaining to the non-functional requirements have been expanded into their respective subsections with respect to the Volere template.

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

Other stakeholders include playtesters of the game, as well as the 3XA3 TAs. Playtesters of the game will be recruited to play the game, and therefore have stake in the success of the project. The 3XA3 TAs will be evaluating the success of the project, as well as providing feedback and guiding the project while it is still in development.

1.3 Mandated Constraints

As a constraint imposed by the project client, there are a number of deadlines for the project throughout its development. In particular, the final demonstration of functionality will be on november the 30th, and the final draft of the project documentation must be produced by the 8th of december. The goal of replicating the gameplay of the original without significant change restricts the platforms for which the project can be developed. In particular, the interface for the original is extremely ill-suited to touch-input environments such as phones and tablets.

1.4 Naming Conventions and Terminology

Listed below are a number of video game and/or roguelike specific terms used in this document.

- Rogue: Both the name of the 1980 computer game, and the a reference to the player character (we will always use the term player character).

- Roguelike: A genre of games similar to Rogue. Membership in the roguelike genre is largely determined by the presence or absence of permadeath, but many games feature many more similarities.

- Permadeath: A feature of roguelikes where the game is restarted from the beginning upon character death.
- Hitpoints: A positive integer value that measures the health of a character (more is healthier).
- Strength: A key statistic of the player character, strength determines how likely they are to successfully land a hit with a melee weapon and how much damage it is likely to do.
- Item Identification: A common feature of roguelikes where items are scrabbled at the beginning of a game, with the player not knowing which corresponding to which effect. Certain effects or simply using these items can identify items. For example, a blue potions may be potions of healing in one game, but in the next they could be sleeping gas. Item identification also refers to determining whether a given item is cursed.
- Cursed equipment: Equipment that once used reveal itself to be harmful to their user and difficult to remove.
- Dungeon: Consisting of a stack of 30 floors, the dungeon forms the game world in Rogue.
- Gold: Gold coins can be found throughout the dungeon, the number of gold coins collected is the primary basis for the player's score.
- Level: Can refer to a floor of the dungeon, or to the player character's experience level, which determines their hitpoints.
- Experience: Experience is gained by defeating monsters, and sufficient quantities will cause the player character to level up.
- Searching: Certain features of the dungeon, such as traps and hidden doors are not immediately visible to the player. The player character can explicitly search their immediate surroundings for such features.

1.5 Relevant Facts and Assumptions

It is assumed users will be utilizing the product in a 64 bit Linux environment, with a keyboard and monitor of at least [INSERT DIMENSIONS].

Users are assumed to be at least moderately familiar with the original, no extra material describing how to play the game is planned to be produced.

User characteristics should go under assumptions. [DELETE]

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

3.1.1 Appearance Requirements

Non-Functional Requirement # 1	
Description:	The Rogue Reborn UI shall closely resemble the original *Rogue* UI.
Rationale:	The new game should be visually similar to the old game.
Fit Criterion:	The new UI must have similar locations for all GUI elements and must use ASCII symbols for all graphical components.

3.1.2 Style Requirements

There are no significant requirements that are applicable to this category.

3.2 Usability and Humanity Requirements

3.2.1 Ease of Use Requirements

Non-Functional Requirement # 2	
Description:	Rogue Reborn shall be fun and entertaining.
Rationale:	Games are developed for enjoyment purposes.
Fit Criterion:	The game must be able to hold the interest of a new user for at least 20 minutes.

3.2.2 Personalization and Internationalization Requirements

Non-Functional Requirement # 3	
Description:	Rogue Reborn shall target an anglophone audience.
Rationale:	The game will be developed and tested by an anglophone population.
Fit Criterion:	All game text must be written in English, free of any grammar or spelling mistakes.

3.2.3 Learning Requirements

Non-Functional Requirement # 4	
Description:	The Rogue Reborn game shall be easy to learn and play.
Rationale:	Users may prematurely lose interest in the game if the controls are difficult or frustrating.
Fit Criterion:	The game must use an intuitive keyboard layout and possess an in-game mechanism to view all key bindings.

3.2.4 Understandability and Politeness Requirements

There are no significant requirements that are applicable to this category.

3.2.5 Accessibility Requirements

There are no significant requirements that are applicable to this category.

3.3 Performance Requirements

3.3.1 Speed and Latency Requirements

Non-Functional Requirement # 5	
Description:	Rogue Reborn shall appear responsive to user input.
Rationale:	Slow update times may induce frustration.
Fit Criterion:	On average, the game UI must be updated within at least 33ms of a visible user action.

3.3.2 Safety-Critical Requirements

There are no significant requirements that are applicable to this category.

3.3.3 Precision or Accuracy Requirements

Non-Functional Requirement # 6	
Description:	Rogue Reborn shall use integer types with an appropriate level of precision.
Rationale:	Integer overflow may cause unexpected behaviour.
Fit Criterion:	All integer values in the game with an unknown upper bound must be at least 32 bits in size.

3.3.4 Reliability and Availability Requirements

Non-Functional Requirement # 7	
Description:	Rogue Reborn shall not crash under normal operating circumstances.
Rationale:	Frequent crashes may frustrate users and diminish their experience.
Fit Criterion:	Every reproducible event that causes the game to crash must be documented, root-caused, and resolved.

3.3.5 Robustness or Fault-Tolerance Requirements

There are no significant requirements that are applicable to this category.

3.3.6 Capacity Requirements

Non-Functional Requirement # 8	
Description:	Rogue Reborn shall be able to record the high scores of up to 15 users.
Rationale:	Allows for a variety of users to directly compete against one another.
Fit Criterion:	The game must be able to load and display the high scores of 15 previous performances.

3.3.7 Scalability or Extensibility Requirements

There are no significant requirements that are applicable to this category.

3.3.8 Longevity Requirements

There are no significant requirements that are applicable to this category.

3.4 Operational and Environmental Requirements

3.4.1 Expected Physical Environment

Non-Functional Requirement # 9	
Description:	Rogue Reborn shall successfully run on any modern laptop or desktop computer with an Intel x64 processor.
Rationale:	Most potential users will have access to this hardware environment.
Fit Criterion:	The game must display stable behaviour on a computer with an Intel x64 processor (equipped with a keyboard, mouse, and monitor).

3.4.2 Requirements for Interfacing with Adjacent Systems

There are no significant requirements that are applicable to this category.

3.4.3 Productization Requirements

Non-Functional Requirement # 10	
Description:	Rogue Reborn shall be distributed as a compressed folder containing a single executable file along with any necessary licenses.
Rationale:	This is a simple approach to the distribution process.
Fit Criterion:	The game must be distributed as a folder containing a collection of applicable licenses in addition to a single executable file that is able to run on a fresh system without any external dependencies.

3.4.4 Release Requirements

There are no significant requirements that are applicable to this category.

3.5 Maintainability and Support Requirements

3.5.1 Maintenance Requirements

Non-Functional Requirement # 11	
Description:	All reported bugs shall be resolved within a month of their submission.
Rationale:	Immediately concentrating effort on subcritical bugs may distract developers.
Fit Criterion:	Every incident featured in the GitLab ITS must be closed within a month of its creation.

3.5.2 Supportability Requirements

There are no significant requirements that are applicable to this category.

3.5.3 Adaptability Requirements

Non-Functional Requirement # 12	
Description:	Rogue Reborn shall successfully run on a modern Linux x64 operating system.
Rationale:	It is assumed that the product testers and consumers will have access to a Linux x64 operating system.
Fit Criterion:	The game must display stable behaviour on an Ubuntu x64 distribution.

3.6 Security Requirements

3.6.1 Access Requirements

There are no significant requirements that are applicable to this category.

3.6.2 Integrity Requirements

Non-Functional Requirement # 13	
Description:	Rogue Reborn shall verify the validity of the saved high score file before displaying its contents.
Rationale:	Malicious users may attempt to inject false records into this file.
Fit Criterion:	The game must display no previous high scores if it detects a flaw in the records file.

3.6.3 Privacy Requirements

There are no significant requirements that are applicable to this category.

3.6.4 Audit Requirements

There are no significant requirements that are applicable to this category.

3.6.5 Immunity Requirements

There are no significant requirements that are applicable to this category.

3.7 Cultural Requirements

There are no significant requirements that are applicable to this category, since Rogue Reborn does not modify any cultural aspects from the original *Rogue*.

3.8 Legal Requirements

3.8.1 Compliance Requirements

Non-Functional Requirement # 14	
Description:	Rogue Reborn shall be distributed with an accompanying LICENSE.txt file.
Rationale:	This license must be distributed with projects that are a modification of the original *Rogue* source code.
Fit Criterion:	The corresponding LICENSE.txt file is included in the distribution package.

3.8.2 Standards Requirements

There are no significant requirements that are applicable to this category.

3.9 Health and Safety Requirements

Non-Functional Requirement # 15	
Description:	Rogue Reborn shall not contain visual sequences that are likely to trigger seizures.
Rationale:	Individuals with photosensitive epilepsy may feel disoriented, uncomfortable, or unwell (Epilepsy Society).
Fit Criterion:	The average luminosity of the game UI cannot change by more than 0.5 between two successive frames.

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

- **Computer Usage Risks** - There are several risks associated with computer usage. This is often a subject matter that is discussed thoroughly in an office environment, where computers see frequent, daily usage.

  - When using a computer, there is an ergonomic risk involved. Improper usage of the computer can lead to aches in various parts of the body, including back, neck, hands, and chest.
  - There is also a significant risk of eye aches, along with other vision problems.
  - Repetitive motion is another factor that could cause discomfort when using a computer.

- **Offensive Content** - The game draws heavily from fantasy, involving themes of violence, fear, and witchcraft. While these elements are only displayed in a textual context, certain cultures and societies may find such elements offensive or disturbing.

- **Anger** - The game is not easy. Frustration could easily overcome the player, especially when he/she has progressed far into the game. Anger management issues are widespread, and evidence of anger due to video games is easily found.

4.7 Costs

4.8 User Documentation and Training

4.9 Waiting Room

4.10 Ideas for Solutions

References

Epilepsy Society. Photosensitive epilepsy. https://www.epilepsysociety.org.uk/photosensitive-epilepsy#.V_pne0ArKCg. Accessed: 2016-10-09.

James Robertson and Suzanne Robertson. *Volere Requirements Specification Template*. Atlantic Systems Guild Limited, 16 edition, 2012.

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.