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

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 (**?**). 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.

- libtcod: a.k.a. "The Doryen Library", libtcod is a popular and feature-rich library for roguelike development, with bindings for C, C++, Python, Lua, and C#.
- 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 1280x400 pixels. Users are assumed to be at least moderately familiar with the original or similar games, as no extra material describing how to play the game is planned to be produced.

2 Functional Requirements

2.1 The Scope of the Work and the Product

2.1.1 The Context of the Work

The context of the work has vastly changed since the original Rogue came out in 1980. Firstly, the times are very different now. Whereas in the 1980's computers were far and few to find, today they play an irreplaceable part of our society. People are on average a lot more familiar with computers than they were back then, therefore the possible market of users is significantly larger.

On the topic of markets, the video game industry has grown tremendously into a international multi-billion dollar industry. The humble Rogue is faced with giants in the field, and while none capture the same magic as the original dungeon crawler, there are certainly other large players on the field.

The final contextual aspect to consider is the thematic inspirations of Rogue. Rogue takes place in a realm of fantasy, drawn up primarily from some high-fantasy setting of Dungeons and Dragons, which itself has drawn much from various works, such as Tolkien's *The Lord of the Rings*, *The Hobbit*, and *The Silmarillion.* Since the release of Rogue in 1980, many more modern pieces in the genera have been released, such as George R. R. Martin's *A Song of Ice and Fire*, and the collective works of R.A. Salvatore. The influence of these new works can be found in extensions over the original Rogue, such as *Moria* (1983).

2.1.2 Work Partitioning

The work required to complete this project has been divided up between the three group members Ian, Mikhail, and Or. Each has been assigned a highly-cohesive, loosely coupled segment of the code that is to be written. It was unanimously agreed that each team member is to present his API to the rest of the team as soon as time permits. This "API" materializes as a C++ header file, with which other modules in the code can interact.

- Or will be in charge of dungeon generations. This includes generating rooms, corridors, walls, doors, handling vision, and the placement of treasure and traps.
- Mikhail will be in charge of most player-tangibles. This includes eating, quaffing potions, handling weaponry, using armor, rings, wands, and scrolls. Much of this realm also crosses over to monster actions, which Mikhail will be in charge of as well.
- Ian will be in charge of the game's state control. The flow of the game, the timing of events, and highscores will all fall under his domain.

2.1.3 Individual Product Use Cases

The product will have one primary use: playing the game. This is the most direct path to completion of the objective, which is to supply entertainment to the user. Most users, as may be anticipated, will do nothing with the project besides this. However, as experience always shows, alternative uses exist for everything. During the 1980's, a group of college students built a piece of software that had one goal: beat the original Rogue game. With the ever-growing advancements in artificial intelligence of today's modern world, it would not be completely foolish to suggest that an AI could potentially be built for this edition. In fact, one could argue that if a new AI system were to be designed to beat Rogue, its designers would seek out this new version, as it would supply a well-documented API with which the system could interact.

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

FR.**1** The player should be able to start a new game
FR.**2** The player should be able to save the current game by name
FR.**3** The player should be able to load previous games by name
FR.**4** The player should be able to quit the game
FR.**5** The player must always begin with the default level 1 hero
FR.**6** The player must always see their hero's statistics
FR.**7** The game must wait until the user takes an action to manipulate the environment

2.2.2 Interaction

FR.**8** The player should be able to view detailed information about the hero
FR.**9** The player should be able to view detailed information about the surrounding environment
FR.**10** The player should be able to pass the turn
FR.**11** The player should be able to walk around
FR.**12** The player should be able to open and close doors

2.2.3 The Dungeon

FR.**13**	The player must begin at the dungeon's first level
FR.**14**	The game must generate each dungeon level one at a time
FR.**15**	Each level must have a downwards staircase
FR.**16**	Every level must generate rooms, corridors, monsters, treasure, and traps
FR.**17**	The player must be able to see in a 3x3 square centered on the hero
FR.**18**	The player must be able to see the entire room the hero is in, if the hero is in a room
FR.**19**	The player should see the outline of dungeon areas previously explored
FR.**20**	The player should be able to search for hidden doors
FR.**21**	The player should not be able to see hidden doors without explicitly searching for them
FR.**22**	The Amulet of Yendor must be generated in level 26

2.2.4 Equipment

FR.**23**	The game should maintain an inventory of player items
FR.**24**	The player should be able to view the inventory
FR.**25**	The game should limit the player's inventory based on the weight of its contents
FR.**26**	The player should be able to add, drop, use, hold, and remove objects from the inventory
FR.**27**	Scrolls, rings, and wands should have meaningless names until identified
FR.**28**	Scrolls, rings, and wands should be usable
FR.**29**	The player should be able to identify items
FR.**30**	The player should not be able to remove cursed items
FR.**31**	Player armor should be able to deteriorate

2.2.5 Combat

FR.**32**	Each monster must have its own statistics
FR.**33**	Each monster must calculate a plan of action
FR.**34**	Monsters must only attack the player, not other monsters
FR.**35**	Every in-game entity must be defeatable

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 (?).
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

The most pressing issue is whether the project will include a Windows version of the product. Issues linking with the libtcod library on Windows have put this item into doubt. Whether save compatability will be maintained with the original is also an open issue, this is likely to be an expensive feature in relation to the value it adds to the project.

4.2 Off-the-Shelf Solutions

We have chosen to use the libtcod library for this project as an off-the-shelf solution to some problems in the product. Libtcod providers a high-level, cross-platform abstraction over rendering and user input, as well as a number of utilities such as line-drawing and pathfinding. There are a number of ports of Rogue to various platforms, including one that upgrades to graphics of the game to graphical tiles, but as of writing we are not aware of any code-cleanup focused rewrite.

4.3 New Problems

So long as the project requirements are met, especially the health and safety requirements, the product should not adversely affect the user. There may be issues with building and deploying the project, as of writing the product has not been tested without building from source. This could potentially require a partial rewrite of the project. It is also unlikely but possible that the product may corrupt the user's files in some way when attempting to save or load a game.

4.4 Tasks

As outlined by the project client, the project is split into a number of development phases. An early proof of concept will be produced first, followed by a test plan for the product, then final development and documentation. This proof of concept phase will consist largely on laying the foundation for the various systems in the product. For example, basic combat will be in the proof of concept, but more advanced combat such as thrown/ranged weapons and monster abilities will be left for later. The full development should consist largely of fleshing in these systems, developing tests, and more advanced features. Development tasks within a phase will be partitioned among team members as the team leader sees fit. For more detail on the proof of concept and other aspects of the development see the Development Plan.

4.5 Migration to the New Product

Migration to the new product should not be an issue for users of the original Rogue. While it is an open issue whether save files will be compatible across versions, since games of Rogue rarely last longer than a few hours this is unlikely to be a major issue for users. It is a goal of the project that the user interface of the product should be unchanged from the original, so migrating users should have minimal issues learning to use the product. Users not familiar with the original may find the product (particularly the user interface), somewhat confusing, but since Rogue was released over 30 years ago, there are a number of resources available online which explain the interface and the basic gameplay. It is the intention of this project for the product to be available in a format that can be simply compiled by any 64-bit Linux system with a C++ compiler, so the installation process should not be a burden.

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

The project's costs will be extremely limited. With the original being open source, there are no licensing concerns to worry about. In addition, all software used in the project is free and potentially open-source. The only potential costs involved is the electricity required to run the development machines.

4.8 User Documentation and Training

If a modern user tried to play the original Rogue, they would not have an easy time getting started. The controls are not intuitive, and interface even less so. With the final product, a brief document explaining the game will be provided. The document will include things like controls, goals, and basics on how the game works. After reading the document, the user should be fairly capable of playing the game. Of course, with time the user shall become more proficient.

4.9 Waiting Room

The waiting room prescribes objectives, requirements, and features that could be implemented in future iterations. The following is a list of such features, for which consideration was given but time could not allow for:

- **Language Translations** - The project is presently written in English, but support for more languages is a reasonable feature to have. Having more language support would open up accessibility to more users and encourage engagement.

- **Tutorial Mode** - There is no denying it: Rogue is a difficult game. It is frustrating and hard to understand, yet rewarding at the end of it all. Overcoming the initial barrier to play is critical. Introducing a tutorial mode would be supremely beneficial to new players learning the ropes.

4.10 Ideas for Solutions

- **Graphics** - Many modern remakes of the original Rogue feature modern graphics, using 16x16, 32x32, or even 64x64 tilesets. These vastly improve the experience, at the cost of no longer being able to run in the terminal.
- **More Monsters** - The original Rogue has 1 monster per letter of the alphabet, for a total of 26. This is a pretty small number of enemies, and is definitely something that could be expanded.

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.

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
10/10/16	1.3	Added 4.1-4.5
10/11/16	1.4	Added 4.9,4.10,2.1.*