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	0.1	Initial Setup
10/02/16	0.1.5	Continued Setup
10/07/16	0.2	Added Project Drivers
10/07/16	0.3	Added Functional Requirements and Risks
10/09/16	0.4	Added Non-Functional Requirements
10/10/16	0.5	Added 4.1-4.5
10/11/16	0.6	Added 4.9,4.10, and 2.1.*
10/11/16	0.7	Proofread and Editing

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 in accordance with the Volere template.

1 Project Drivers

1.1 The Purpose of the Project

The goal of the Rogue Reborn project is to produce a reimplementation of the original *Rogue* computer game, originally developed by Michael Toy, Glenn Wichman, and Ken Arnold in 1980 (Petri Kuittinen, 2001). This project was undertaken as a consequence of *Rogue*'s historical significance, as it is the namesake of the roguelike game genre. The motivation behind the remake is characterized by the poor condition of the original source code: it was not written with readability in mind and was designed for extremely low-performance systems that required unusual design patterns. Additionally, the source code was written in an old version of C, which hinders compilation and the possibility of feature extensions. In response to these issues, the Rogue Reborn project aims to develop a fresh copy of the game in a modern language with contemporary design principles, comprehensive documentation, and a full test suite. The intended audience for this document includes all the stakeholders of the project, especially Dr. Smith and the SFWRENG 3XA3 teaching assistants (TAs).

1.2 The Stakeholders

1.2.1 The Client

The primary client of the project is Dr. Spencer Smith. Dr. Smith is responsible for commissioning the project, overseeing its production, and specifying all documentation requirements. He will also be evaluating the final product and the progress showcased throughout the intermediate milestones.

1.2.2 The Customers

The customers of the project are simply the players of the Rogue Reborn game. It is expected that the majority of this population will be players of the original game, in addition to the players and developers of the later roguelike games. Since the roguelike community has a strong open-source tradition, it is possible that a modern, well-documented *Rogue* remake could serve as a valuable starting point or inspiration for projects created by other teams in the future.

1.2.3 Other Stakeholders

The other stakeholders in the Rogue Reborn project include the play testers of the game and the SFWRENG 3XA3 TAs. Specifically, play testers will be recruited to judge the game before its initial release and provide feedback with respect to any discovered bugs and questionable design decisions. On the other hand, the SFWRENG 3XA3 TAs will guide the project development process and incrementally evaluate the success of the project.

1.3 Mandated Constraints

As a constraint imposed by the project client, there are a number of deadlines the project will be expected to meet throughout the course of its development. In particular, the functionality of the final project will be demonstrated on November the 30^{th}, 2016 and the final draft of the project documentation must be produced by December 8^{th}, 2016. Another constraint involves replicating the gameplay of the original product without any significant adaptations; this restricts the potential development platforms, as the interface for the original *Rogue* is extremely ill-suited to touch-input environments such as phones and tablets.

1.4 Naming Conventions and Terminology

Listed below is a glossary of uncommon terms that are used throughout the document.

- **Cursed Equipment**: Equipment that, once used, reveals itself to be harmful to the player character. This type of equipment is typically difficult to remove.

- **Dungeon**: A stack of FINAL_LEVEL floors (levels); it forms the game world in *Rogue*.
- **Experience**: A positive integer value that is increased by defeating monsters; sufficient quantities of experience will cause the player character to level up.
- **Gold**: A positive integer value that serves as the primary basis for measuring a user's score. Gold coins contribute to this quantity and are found throughout the dungeon.
- **Hitpoints**: A positive integer value that measures the health of a character or monster (the amount of "damage" the entity can endure before death).
- **Item Identification**: A common feature of roguelike games whereby item names are randomly associated with an effect at the beginning of each game. These items can be identified by the player by simply using the item and observing their effect (item names are consistent throughout a single game). For example, blue potions may be potions of healing in one game, but in the next game, they could be sleeping gas. Item identification also refers to the act of determining whether or not a given item is cursed.
- **Level**: A positive integer denoting either the floor of the dungeon or to the player character's experience level (which determines their maximum hitpoints).
- ***libtcod***: A.K.A. "The Doryen Library", *libtcod* is a popular graphics library designed for the development of roguelike games. It includes bindings for C, C#, C++, Lua, and Python (Lib).
- **Permadeath**: A feature of roguelikes where the game must be restarted from the beginning upon character death.
- **Rogue**: The name of the 1980 computer game as well as the reference to the player character (the term "player character" will always be explicitly stated in the presence of ambiguity).

- **Roguelike**: A genre of games similar to *Rogue*. Membership in the roguelike genre is characterized by procedurally-generated environments, demanding difficulty, and permadeath features (Slash).
- **Searching**: An action that can be performed by the player character to reveal adjacent invisible features (such as traps or hidden doors).
- **Strength**: An attribute of the player character; determines the likelihood of the player character successfully landing a hit with a melee weapon and how much damage it will inflict on the enemy.

1.5 Relevant Facts and Assumptions

It is assumed players will be utilizing the product in a 64-bit Linux environment, equipped with a keyboard and a monitor that exhibits a display resolution of at least WIDTH_RESOLUTION x HEIGHT_RESOLUTION pixels. Players are also assumed to be moderately familiar with roguelike games, as no extra material describing the game mechanics is provided with the Rogue Reborn distribution. Note that this decision is made in the interest of creating an exploratory and curious atmosphere about the game.

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* release in 1980. Firstly, computers have advanced by an immense margin. In the 1980's, computers were mostly an alien concept and there were few and far between. Today, computers play an irreplaceable role in our society and have received full adoption from the general population. Consequently, computers have seen exposure to an extremely large diversity of industries and academic subjects, and the user market has exploded to unprecedented heights.

From an economic perspective, the video game industry has grown tremendously into a international multi-billion dollar industry. The task of developing the next blockbuster game has expanded to cover entire studios of professional graphic artists, writers, and programmers. Several popular titles have also outgrown their medium and penetrated the book and cinematic markets. Now, the humble *Rogue* is faced with many giants in the field, and while none capture the same magic as the original dungeon crawler, there are certainly successful in their own regards.

The final contextual aspect to consider is the thematic inspirations of *Rogue*. *Rogue* takes place in a realm of fantasy, inspired by the high-fantasy settings of Dungeons and Dragons, which itself has drawn much from various narratives, such as Tolkien's *The Lord of the Rings*, *The Hobbit*, and *The Silmarillion*. Since the release of *Rogue*, many modern pieces in the same thematic genre 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* from 1983.

2.1.2 Work Partitioning

The work required to complete this project has been partitioned between the Rogue++ team of Ian, Mikhail, and Or. Each member has been assigned a highly-cohesive, loosely-coupled segment of the code to be written. It was

unanimously agreed upon that each person is to present their API to the rest of the team as soon as it is ready. This API materializes as a C++ header file which other modules can interpret and use.

- Ian is responsible for the game state control, flow, and graphics. This includes handling user input, rendering graphics, and displaying menus and high scores.

- Mikhail is responsible for most player-tangibles. This includes eating, quaffing potions, handling weaponry, as well as using armor, rings, wands, and scrolls. This work will also transfer over to monster actions, which Mikhail will also actively develop.

- Or is responsible for dungeon and feature generation. This includes generating room, corridor, wall, and door locations, in addition to handling vision and the placement of treasure and traps.

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 (to supply entertainment to the user), and will suffice for the majority of the project customers. However, another subset of the Rogue Reborn consumer market may use the product in a different way. During the 1980's, a group of college students built a piece of software with one goal: beat the original *Rogue* game (Rog, 1985). Now, with the ever-growing advancements in artificial intelligence (AI) of today's modern world, it is well within the realm of possibility that a similar AI could be designed for the Rogue Reborn edition. One could even argue that if a new AI system were to be designed to beat *Rogue*, its designers may feel tempted to seek out this new Rogue Reborn version, as it can supply a well-documented API to facilitate system interaction.

2.2 Functional Requirements

This section will specify the functional requirements of the Rogue Reborn project. These requirements are organized in cascading, logical segments to conquer their numerous, scattered, and interdependent nature.

2.2.1 Basic mechanics

- FR.**1** The user shall be able to start a new game.
- FR.**2** The user shall be able to save the current game by name.
- FR.**3** The user shall be able to load previous games by name.
- FR.**4** The user shall be able to quit the game.
- FR.**5** The player character must always start with a default set of attributes.
- FR.**6** The user shall always see the player character's statistics.
- FR.**7** The game shall wait until the user takes an action before it manipulates the environment.
- FR.**8** The game shall be able to display a help menu.

2.2.2 Interaction

- FR.**9** The user shall be able to view detailed information about the player character.
- FR.**10** The user shall be able to view detailed information about the surrounding environment.
- FR.**11** The player character shall be able to pass their turn.
- FR.**12** The player character shall be able to walk to adjacent cells.
- FR.**13** The player character shall be able to open and close doors.
- FR.**14** The player character shall be able to fall under status effects.
- FR.**15** The player character shall activate a trap every time they walk onto the cell with the trap.

2.2.3 The Dungeon

FR.**16** The player character shall begin at level START_LEVEL.
FR.**17** The game shall generate each dungeon level as it is needed.
FR.**18** Each level shall have a downwards staircase.
FR.**19** Each level shall contain a combination of rooms, corridors, monsters, treasure, and (optionally) traps.
FR.**20** The user shall be able to view the environment within VIEW_DISTANCE cells away (diagonals included) from the location of the player character.
FR.**21** If the player character is in a room, the user shall be able to view the entire room the player character is occupying.
FR.**22** The user must be able to see the *outline* of dungeon areas that were previously explored.
FR.**23** The player character shall be able to search for hidden doors and traps/
FR.**24** The player character shall not be able to see hidden doors without explicitly searching for them.
FR.**25** The *Amulet of Yendor* shall be generated in level FINAL_LEVEL.

2.2.4 Equipment

FR.**26** The game shall maintain an inventory of the player character's collected items.
FR.**27** The player character shall be able to view the inventory.
FR.**28** The game shall limit the player character's inventory based on the weight of its contents.
FR.**29** The player character shall be able to add, drop, use, hold, throw, and remove objects from their inventory.
FR.**30** Scrolls, rings, and wands shall have meaningless names until they are identified.
FR.**31** Scrolls, rings, and wands shall be usable.
FR.**32** The player character shall be able to identify items.
FR.**33** The player character shall not be able to remove cursed items.
FR.**34** The player character's armor shall be able to deteriorate.

2.2.5 Combat

FR.**35** Each monster shall be able to calculate a plan of action during their turn.

FR.**36** Monsters shall only attack the player (not other monsters).

FR.**37** The player character shall be able to defeat every monster.

FR.**38** The player character shall restore lost health over a time interval if they are not engaged in combat.

FR.**39** Armor shall reduce the damage taken by the player character.

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:	The Rogue Reborn game 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 MAXIMUM_ENTERTAINMENT_TIME minutes.

3.2.2 Personalization and Internationalization Requirements

Non-Functional Requirement # 3	
Description:	The Rogue Reborn game 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 and should be 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:	The Rogue Reborn game 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 RESPONSE_SPEED milliseconds 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:	The Rogue Reborn game 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:	The Rogue Reborn game 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:	The Rogue Reborn game shall be able to record the high scores of up to HIGH_SCORE_CAPACITY 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 HIGH_SCORE_CAPACITY 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:	The Rogue Reborn game 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:	The Rogue Reborn game 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:	The Rogue Reborn game shall successfully run on a modern 64-bit Linux operating system.
Rationale:	It is assumed that the product testers and consumers will have access to a 64-bit Linux operating system.
Fit Criterion:	The game must display stable behaviour on a 64-bit Ubuntu 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:	The Rogue Reborn game 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:	The Rogue Reborn game 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:	The Rogue Reborn game 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 LUMINOSITY_DELTA between two successive frames.

4 Project Issues

4.1 Open Issues

The most pressing issue is whether or not this product will support a Windows 10 port. Currently, the development team has experienced issues with linking with the *libtcod* library on Windows. Another issue concerns whether the Rogue Reborn save files will be compatible with the original *Rogue* save files; this is an expensive feature with respect to the value it adds to the project.

4.2 Off-the-Shelf Solutions

The Rogue++ team has chosen to use the *libtcod* library for this project as an off-the-shelf solution for some problems in the product. *libtcod* provides 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 also a number of *Rogue* ports for various platforms, including a variation that utilizes graphical tiles in the game to enhance the visual experience.

4.3 New Problems

As long as the project requirements are satisfied, especially the health and safety requirements, the product should not adversely affect the user. There may also be future issues involving the deploying the project, as the Rogue++ team has not tested the executable without building the source files locally; this could potentially require a partial rewrite of the project. Finally, it may also be possible for the game to corrupt the user's files in some manner while 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, and then the 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 presented in the proof of concept, but more advanced combat such as thrown/ranged weapons and monster abilities will be left for a later development cycle. The full development should consist largely of populating these systems, creating tests, and adding 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 consult the Development Plan document.

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, this is unlikely to be a major issue for users since a single *Rogue* adventure rarely last longer than a few hours. Next, it is a goal of the project that the user interface of the product should be unchanged from the original; this is meant to facilitate the learning process of migrating users. Users that are unfamiliar with the original may find the product (particularly the user interface) slightly confusing, but since *Rogue* was released over 30 years ago, there are a number of resources available online to help explain the UI and the basic gameplay. On a final note, 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 (GNU) C++ compiler. This will streamline the installation process and encourage more developers to try their hand at understanding the code.

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 usage.
  - When using a computer, there is an ergonomic risk involved. Improper operation of the computer can lead to aches in various parts of the body, including the 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 the fantasy realm, including 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 Rogue Reborn game is *not* easy. Frustration could easily overcome the user, especially when they have made significant progress into the game. Anger management issues are widespread, and evidence of anger due to video games is easily found (Gary W. Giumetti and Patrick M. Markey, 2007).

4.7 Costs

The cost of this project is extremely limited. Given that the original game is open source, there are no paid licensing concerns. In addition, all of the software used in the Rogue Reborn project is free to use without attribution. The only potential costs involved is the electricity required to run the development machines.

4.8 User Documentation and Training

If a modern user attempted to play the original *Rogue*, they would not have an easy time: the controls and the interface are unintuitive and foreign. Luckily for the user, the final product will feature an in-game help menu to aid newer players in becoming acquainted with the game (this menu's primary purpose will be to inform the user of the controls). After reviewing the menu, the user should be capable of playing the game and discovering the rest of the game's functionalities for themselves.

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:

- **More Monsters**: The original *Rogue* has 1 monster for every letter of the English alphabet. This is a very small number, especially when compared to modern video games. By using different colours or a tile-set (see 4.10), it is possible to achieve a far greater number of enemies to challenge the user.

- **Infinite Descent**: The concept of a never-ending game is not by any means original. When dungeon levels are generated on the fly, this would not be difficult to implement either. Currently, the "best" game performance is the one that accomplishes the end goal in the fewest moves while obtaining the most gold along the way. With infinite descent, the best run would eliminate the end goal from the equation, turning the objective of the game into a strict function of acquired wealth. As the player descends, the levels would become progressively more difficult and generate more gold for the player to collect as a reward for their progress.

- **Seed Sharing**: A seed is a sequence of bits that dictate which numbers the pseudo-random number generator will create. If two separate pseudo-random number generators use the same seed for instantiation, they will generate the same sequence of numbers. Consequently, this yields the ability to generate the same dungeons from a single seed. This could be used for more accurately comparing the competence of

Rogue players, as some dungeons may be naturally more difficult or longer.

As a side note, for this feature to work, the dungeon generation must not rely on the state of the player for surely different players may take on the same dungeon in different ways.

4.10 Ideas for Solutions

- **Graphics**: Modern video games have engaging animations, special effects, life-like detail, and an overall immersive user experience. In contrast, *Rogue*'s peak graphical experience is an ASCII symbol surrounded by some capital letters; there is no question that graphics are a primary bottleneck for attracting users. For this reason, it would be wise to inquire about using a tile-set for graphics (a capability of *libtcod*). Using a 16x16, 32x32, or even 64x64 tileset could vastly improve the graphical user experience.

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

- **Tutorial Mode**: For better or for worse, *Rogue* is a difficult game. It is frustrating and hard to understand, yet the reward at the end is arguably worth the effort. As such, overcoming the initial play barrier is critical. Introducing a tutorial mode may prove to be extremely beneficial for new players learning the ropes.

References

[Libtcod] Overview. https://bitbucket.org/libtcod/libtcod/. Accessed: 2016-10-07.

An expert system outperforms mere mortals as it conquers the feared Dungeons of Doom. *Scientific American*, 252:18–21, February 1985. Accessed: 2016-10-11.

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

Gary W. Giumetti and Patrick M. Markey. Violent video games and anger as predictors of aggression. *Journal of Research in Personality*, 2007. Accessed: 2016-10-11.

Petri Kuittinen. Rogue - Exploring the Dungeons of Doom (1980). https://web.archive.org/web/20071217100401/http://users.tkk.fi/~eye/roguelike/rogue.html, 2001. Accessed: 2016-10-07.

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

Slash. What is a Roguelike. http://www.roguetemple.com/roguelike-definition/. Accessed: 2016-10-07.

5 Appendix

5.1 Symbolic Parameters

Table 2: **Symbolic Parameter Table**

Parameter	**Value**
FINAL_LEVEL	26
WIDTH_RESOLUTION	1280
HEIGHT_RESOLUTION	400
VIEW_DISTANCE	2
START_LEVEL	1
MINIMUM_ENTERTAINMENT_TIME	20
MINIMUM_RESPONSE_SPEED	30
HIGH_SCORE_CAPACITY	15
LUMINOSITY_DELTA	0.5