

# SE 3XA3: Test Plan Rogue Reborn

Group #6, Team Rogue++

Ian Prins	prinsij
Mikhail Andrenkov	andrem5
Or Almog	almogo

Due Wednesday, December 7<sup>th</sup>, 2016

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Overview . . . . .	4
1.2	Sections . . . . .	4
<b>2</b>	<b>Functional Requirements Evaluation</b>	<b>5</b>
<b>3</b>	<b>Non-Functional Requirements Evaluation</b>	<b>7</b>
3.1	Usability and Aesthetics . . . . .	7
3.2	Performance . . . . .	8
3.3	Robustness, Maintainability, and Compatibility . . . . .	9
3.4	Legality and Safety . . . . .	10
<b>4</b>	<b>Comparison to Existing Implementation</b>	<b>12</b>
<b>5</b>	<b>Unit Testing</b>	<b>13</b>
<b>6</b>	<b>Changes Due to Testing</b>	<b>15</b>
<b>7</b>	<b>Automated Testing</b>	<b>16</b>
7.1	Automated Testing Strategy . . . . .	16
7.2	Specific System Tests . . . . .	16
<b>8</b>	<b>Trace to Requirements</b>	<b>46</b>
<b>9</b>	<b>Trace to Modules</b>	<b>47</b>
<b>10</b>	<b>Code Coverage Metrics</b>	<b>49</b>

## List of Tables

1	Revision History . . . . .	3
3	Test-Requirement Trace . . . . .	46
5	Module Hierarchy . . . . .	47
6	Test-Module Trace . . . . .	48

## List of Figures

1	Testable Class Interface . . . . .	13
---	------------------------------------	----

Table 1: **Revision History**

Date	Version	Notes
12/06/16	0.1	Initial Draft
12/06/16	0.2	Automated Tests To PlayerChar
12/06/16	0.3	Functional Requirements Evaluation
12/07/16	0.4	Introduction
12/07/16	0.5	Finished Automated Tests
12/07/16	0.6	Non-Functional Requirements Evaluation
12/07/16	0.7	Unit Testing

# 1 Introduction

## 1.1 Overview

The primary objective of this document is to provide a comprehensive summary of the verification process with respect to the Rogue Reborn project. Interested parties are welcome to analyze this paper as a means of evaluating the success of the final application regarding the requirements described in the [SRS](#) and tests prescribed in the [Test Plan](#). After reviewing the document, the reader should understand the strengths and weaknesses of the Rogue Project as it relates to the expectations of the client.

## 1.2 Sections

A brief description of each Test Report section is provided below:

- §1 Brief overview of the Test Report
- §2 Functional evaluation of Rogue Reborn
- §3 Non-functional evaluation of Rogue Reborn
- §4 Description of relationship to original *Rogue* with respect to testing
- §5 Explanation of unit testing in Rogue Reborn
- §6 List of changes that were performed as a consequence of testing
- §7 Tabular depiction of automated tests
- §8 Justification of test files with respect to functional requirements
- §9 Decomposition of modules and trace to test files
- §10 Summary of code coverage metrics

## 2 Functional Requirements Evaluation

Overall, an evaluation of functional requirements reveals near, if not complete coverage. The tests written for the projects turned out to be quite useful, as many caught bugs or business-errors that would have otherwise gone unnoticed. Those will be discussed below. As for the rest of the functional requirements, many were mundane, generic, or crucial enough to have already been satisfied before tests were considered. Those will not be discussed, as their complete satisfaction has already been verified countless times.

The list below refers to each functional requirement by its numerical identifier, as listed in the System Requirements Specification. Please refer to the [SRS](#) document if any confusion arises.

**FR.16:** When performing level tests, a strange anomaly led to one test constantly failing. The test revealed that the player, in fact, did not begin at the first level. Due to an off-by-one error and slight miscommunication between developers, the current level depth the player was on was  $i$  in some places and  $i + 1$  in others. As soon as the test revealed this, the problem was remedied globally.

**FR.19:** Whenever the player uncovers a new dungeon level (including the very first level), an algorithm decides on a position in which to place the user initially. This algorithm while appearing flawless, actually had a very slight chance of placing the player in an unreachable location, surrounded by walls, doomed forever. With the automatic tests running thousands upon thousands of simulations, the bug was quickly revealed, and remedied.

**FR.35:** Some monsters in Rogue Reborn follow a simple AI consisting of steps such as: Look for the player, chase the player, and if you can, hit the player. This is a very simple and easily-implemented AI to both invent and implement, but the enemies in Rogue Reborn go beyond such simple schemes. Some monsters do not seek to kill the player, but rather steal their precious items. One such pest is the Leprechaun, known as the symbol "L". The Leprechaun necessitated the implementation of a variety of methods that were previously unneeded, such as *getNearestGold()*. The testing of this function revealed some very serious performance issues in the pathfinding algorithm used throughout

the project, in which infinite path traces were possible. It also revealed another bug in which the coordinates of several level features (items mostly) were accidentally set to 0, rendering them impossible to reach by pathfinding and once again causing a pathfinding failure. This was due to the assumption that all items are placed on reachable blocks. Fortunately, thanks to the various tests we implemented, these bugs were caught and fixed.

**FR.39:** Working with C++ has its benefits, but also its drawbacks. An anomaly in the way C++ handles integers revealed a very serious bug in the code, in which player armor could reach utterly ridiculous values, rendering the player effectively invincible. By simulating every possibility of armor that can be made, this bug was caught and patched. To elaborate, the reason the bug even existed was because an unsigned integer was allowed to be reduced to a negative value, which of course means that it was not reduced to a negative number and instead went to the highest value an integer can be.

### 3 Non-Functional Requirements Evaluation

The following subsections evaluate the significant non-functional qualities of Rogue Reborn. To simplify notation, NFRT  $i$  is used to denote “Non-Functional Requirements Test #  $i$ ” from the [Test Plan](#) document. Unfortunately, the usability and playtesting surveys referenced in NFRT 1, NFRT 2, NFRT 4, NFRT 7, NFRT 9, NFRT 10, NFRT 12, and NFRT 14 were not performed as a direct consequence of the time constraints imposed on the project (the [Gantt Chart](#) schedules this survey to be released in early January 2017). Another hindering factor is the sophisticated software environment required for compilation: publicly distributing a single executable file is not feasible since Rogue Reborn makes use of a library that exclusively compiles for Linux at the present moment. Despite these inconveniences, an effort has been made to mention several weaker versions of these tests under their corresponding sections.

#### 3.1 Usability and Aesthetics

Overall, the visual appearance of the application was well-received and was universally praised as an improvement over the original *Rogue*. This conclusion was derived from the interactions between the Rogue++ team and the *SFWRENG 3XA3* instructor staff, as well as informal conversations with other colleagues. Unfortunately, the usability survey described in NFRT 1 will be carried out in the future, so the impressions of the general public are not yet known.

Since the usability of the original *Rogue* was relatively poor due to its arbitrary key bindings, the Rogue++ team took deliberate actions to improve this area. Specifically, the application featured arrow key bindings for the player character movements in order to accommodate a more standard and intuitive keyboard layout. Additionally, Rogue Reborn featured a convenient help menu inside the game that listed all actionable keys and their corresponding interpretations. However, due to the sheer quantity of key bindings, Rogue Reborn was *not* successful in alleviating this issue completely. With respect to the [Test Plan](#), a report detailing NFRT 3 is given below.



---

### Non-Functional Requirement Test # 3 Report

---

*Execution:* All strings in the Rogue Reborn source code were programmatically extracted and stored in a text file for later inspection with Microsoft Word. The Python script used to populate the text file is located at [src/misc/strextact.py](#).

*Results:* The aforementioned script discovered approximately 1400 strings. After manually verifying the grammatical correctness and spelling of each string, it was determined that the GUI output is free of linguistic errors.

---

## 3.2 Performance

In general, the technical performance of the Rogue Reborn client was exceptional as a consequence of the developers' decision to use C++ as opposed to a less efficient language such as Python or Java. According to *libtcod*, the application managed to average over 120 FPS (Frames Per Second) and delivered a smooth experience even while on a VM (Virtual Machine). During the final stages of development, the Rogue++ team decided to profile the application using the GDB (GNU Debugger) with respect to peak memory usage and pleasantly discovered that the maximum amount of RAM consumed by Rogue Reborn was 1 MB. With respect to the [Test Plan](#), a report detailing NFRT 5 and NFRT 6 is portrayed below.

---

### Non-Functional Requirement Test # 5 Report

---

*Execution:* The Rogue Reborn application was compiled with a special debug parameter that enabled particular sections of the MasterController module to measure the average maximum execution time between successive frames.

*Results:* The average execution time between successive user actions appeared to stabilize around 20 ms. Clearly, this is appreciably lower than the maximum allowable delay of [MINIMUM\\_RESPONSE\\_SPEED](#) (currently  $\sim 30$  ms).

---

---

### Non-Functional Requirement Test # 6 Report

---

*Execution:* The following regular expression was applied across the source code to extract all integer-typed declarations:

`(unsigned|int|long)\s+[A-Za-z]+\s*(,|;|=.*)\s*`

The Python script that performed the declaration extraction is located at [src/misc/intdeclare.py](#).

*Results:* The aforementioned script identified approximately 170 candidate declarations. Among these, there were no obvious candidates for integer overflow.

---

### 3.3 Robustness, Maintainability, and Compatibility

As discussed during the formal presentation, the Rogue Reborn project excelled in the domains of robustness and maintainability. One justification for this claim is that the developers implemented a CI (Continuous Integration) pipeline to immediately flag deficient commits. In addition, compiling the Rogue Reborn source code did not generate any warnings and the system documentation is thorough, expansive, and relevant. That being said, Rogue Reborn has yet to be released in the playtester community. As such, it is likely that several undiscovered bugs still reside within the code base.

Regarding compatibility, the project was significantly less successful: only Linux distributions are supported and SDL (Simple Directmedia Layer) must be installed on the developer's machine to compile the application ([SDL](#)). As well, the compilation and execution of Rogue Reborn was only tested on Intel x64 processors (the application is not tested on other brands or architectures). With respect to the [Test Plan](#), a report detailing NFRT 8, NFRT 11, and NFRT 13 is conveyed below.

---

### Non-Functional Requirement Test # 8 Report

---

*Execution:* The high score file was manually edited by a developer to include more than `HIGH_SCORE_CAPACITY` records.

*Results:* Only the top `HIGH_SCORE_CAPACITY` records with the highest score were displayed on the `RipScreen` module; the rest of the records were internally acknowledged but otherwise ignored.

---

---

### Non-Functional Requirement Test # 11 Report

---

*Execution:* All documented “Bug” issues on the Rogue Reborn GitLab page were examined to ensure they were closed within 30 days of their creation.

*Results:* A total of 10 issues were discovered under the “Bug” label; all of these entries were closed within two weeks of their posting.

---

---

### Non-Functional Requirement Test # 13 Report

---

*Execution:* The high score file was manually edited by a developer such that several records violated the expected record format.

*Results:* All valid records were processed displayed on the `RipScreen` module; the nonsensical records were internally acknowledged but otherwise ignored.

---

## 3.4 Legality and Safety

Every public software project should include a license to govern the software’s terms of use, development, and distribution. Rogue Reborn was no exception and thus contained the `licensing agreement` from the original *Rogue*.

Regarding health and safety, the Rogue++ team was primarily concerned with the possibility of inducing a seizure by displaying two successive frames with excessively different contrasts. With respect to the [Test Plan](#), a report detailing NFRT 15 is pictured below.

---

### Non-Functional Requirement Test # 15 Report

---

*Execution:* Two consecutive frames with the largest estimated contrast difference were captured and later processed with a Python script to compute their respective monochrome luminosities. All resources pertaining to this test are located in the [src/misc/luminosity](#) directory.

*Results:* The output of [lumtester.py](#) indicated that the frame illustrated in [minlum.png](#) was characterized by an average luminosity of 0.03105, while the [maxlum.png](#) frame possessed an average luminosity of 0.59774. Since the difference between these two values is 0.56669 and is therefore less than [LUMINOSITY\\_DELTA](#), the application is relatively safe for epileptic users (although a cautionary notice should be included in the final distribution package).

---

## 4 Comparison to Existing Implementation

According to all collected resources of the original implementation, there were no tests done to verify the accuracy of the original product. This is somewhat understandable, all things considered. After all, the original product was released in 1980, almost 40 years ago. Since then, standards of software development were transformed from infancy to the rigorous forms they take today.

As such, the tests we have written have nothing to be compared to. If there were tests to compare to, we would take a look at the following:

- How does the coverage of the existing test cases compare to the coverage we implemented?
- For cases where the modules/classes are similar in both versions, how do the tests compare? Are there any significant benefits or drawbacks to one or the other?
- Does any one of the implementations completely neglect an aspect of another? Is the remade implementation missing something critical that the original did test for?
- How are random tests approached by the existing implementation? Are random numbers ever used in the testing phase?

## 5 Unit Testing

As a means of verifying the implementation of various functions and components, the Rogue++ developers devised a suite of unit tests. Unit testing was employed as a consequence of its superior ability to localize errors and encourage contributors to write code that is highly modular and relatively free of side effects. Broadly speaking, the Rogue Reborn unit tests were designed to gain confidence in the code with respect to the satisfaction of the functional requirements.

To mitigate the significant overhead of integrating a professional unit testing framework, the Rogue Reborn team decided to code their own custom test runner. The implemented solution required for all tests to extend an abstract class `Testable` with the interface described in Figure 1. This restriction enabled the test driver class `test.main.cpp` to invoke all of the concrete unit test classes in a uniform manner. In addition, to ensure that the unit tests remained relevant and visible throughout development, the CI pipeline included a stage to build, run, and evaluate the output of the testing executable.

Figure 1: **Testable Class Interface**

```
class Testable {
public:
    virtual Testable();
    virtual ~Testable();

    // Test entry point
    virtual void test() = 0;

    void assert(bool condition, std::string comment);
    void comment(std::string comment);
};
```

The coverage of the tests encompassed all item abilities, various player actions and monster mechanics, virtually all level generation and processing algorithms, and even several UI functions. Each assertion was also complemented with a comment to explain the purpose of the test. For the convenience of the reader, a descriptive summary of every implemented test is

included in the [Automated Testing](#) section of this report.

On the whole, these tests served well to guarantee a minimal level of functionality for each documented feature across various builds. Naturally, the suite did not consider the non-functional requirements of the project since these qualities are much more practical to test manually.

## 6 Changes Due to Testing

Mikhail



## 7 Automated Testing

### 7.1 Automated Testing Strategy

For this project we elected not to use a 3rd party testing library. We made this decision to ease configuration/installation problems and reduce our dependencies, as we judged it would not be necessary. Instead a series of files (labeled test.foobar.cpp) in the repository hold tests, which are run by our custom test runner. These automated tests are run on command by executing the produced executable, or by the continuous integration script run whenever changes are pushed to the central repository. The results of these tests are automatically reported, resulting in a failed or successful build.

### 7.2 Specific System Tests

The following is a list of all system tests in the project.

<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Amulet Construction None Coordinate, context value Amulet object in valid initial state
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Armor Construction 1 None Coordinate Armor object in valid initial state
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Armor Construction 2 None Coordinate, context value, type value Armor object in valid initial state
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Armor Identification Cursed Armor None Verification that armor is identified

<b>Name:</b>	Armor Curse
<b>Initial State:</b>	Cursed Armor
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that armor is cursed
<b>Name:</b>	Armor Enchantment
<b>Initial State:</b>	Cursed Armor
<b>Input:</b>	Curse level
<b>Expected Output:</b>	Verification that armor enchantment is correct
<b>Name:</b>	Armor Rating
<b>Initial State:</b>	Cursed Armor
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that armor rating is correct
<b>Name:</b>	Coordinate Ordering
<b>Initial State:</b>	None
<b>Input:</b>	(0,0) coordinate and (1,1) coordinate
<b>Expected Output:</b>	Verification that (0,0) < (1,1)
<b>Name:</b>	Coordinate Equality
<b>Initial State:</b>	None
<b>Input:</b>	Two (0,0) coordinates
<b>Expected Output:</b>	Verification that the two inputs are equal
<b>Name:</b>	Coordinate Inequality
<b>Initial State:</b>	None
<b>Input:</b>	(0,0) coordinate and (1,1) coordinate
<b>Expected Output:</b>	Verification that the two inputs are not equal
<b>Name:</b>	Coordinate Addition
<b>Initial State:</b>	None
<b>Input:</b>	(2,3) coordinate and (1,2) coordinate
<b>Expected Output:</b>	(3,5) coordinate
<b>Name:</b>	Coordinate Subtraction

<b>Initial State:</b>	None
<b>Input:</b>	(2,3) coordinate and (1,2) coordinate
<b>Expected Output:</b>	(1,1) coordinate
<b>Name:</b>	Feature Construction
<b>Initial State:</b>	None
<b>Input:</b>	Symbol, coordinate, visibility, color
<b>Expected Output:</b>	Feature object in valid initial state
<b>Name:</b>	Feature Symbol Check
<b>Initial State:</b>	Feature with given symbol
<b>Input:</b>	Symbol
<b>Expected Output:</b>	Verification that feature's symbol matches given
<b>Name:</b>	Feature Invisibility Check
<b>Initial State:</b>	Invisible feature
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that feature is invisible
<b>Name:</b>	Feature Visibility Check
<b>Initial State:</b>	Visible feature
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that feature is visible
<b>Name:</b>	Feature Location Check
<b>Initial State:</b>	Feature with given location
<b>Input:</b>	Coordinate
<b>Expected Output:</b>	Verification that feature's location matches given coordinate
<b>Name:</b>	Food Construction
<b>Initial State:</b>	None
<b>Input:</b>	Coordinate and context value
<b>Expected Output:</b>	Food object in valid initial state
<b>Name:</b>	Food Eating
<b>Initial State:</b>	Food and player objects

<b>Input:</b>	None
<b>Expected Output:</b>	Verification that food has increased the player's food life by an appropriate amount
<b>Name:</b>	GoldPile Construction
<b>Initial State:</b>	None
<b>Input:</b>	Coordinate, gold amount value
<b>Expected Output:</b>	GoldPile object in valid initial state
<b>Name:</b>	GoldPile Quantity Check
<b>Initial State:</b>	GoldPile with given amount of gold
<b>Input:</b>	Amount of gold value
<b>Expected Output:</b>	Verification that gold's amount matches given amount
<b>Name:</b>	Item Construction 1
<b>Initial State:</b>	None
<b>Input:</b>	Symbol, coordinate, context value, item class specifier, name value, psuedoname value, item type specifier, item stackability value, item throwability value, weight value
<b>Expected Output:</b>	Item object in valid initial state
<b>Name:</b>	Item Construction 2
<b>Initial State:</b>	None
<b>Input:</b>	Symbol, coordinate, context value, item class specifier, name value, psuedoname value, item type specifier, item stackability value, item throwability value, weight value
<b>Expected Output:</b>	Item object in valid initial state
<b>Name:</b>	Name Vector Check
<b>Initial State:</b>	None
<b>Input:</b>	Vector of item names
<b>Expected Output:</b>	Shuffled vector of item names
<b>Name:</b>	Item Curse Check
<b>Initial State:</b>	Uncursed item
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that item is uncursed

<b>Name:</b>	Item Curse/Effect Check 1
<b>Initial State:</b>	Uncursed item to which the cursed effect has been applied
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that item is cursed
<b>Name:</b>	Item Curse/Effect Check 2
<b>Initial State:</b>	Cursed item whose curse effect has been removed
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that item is uncursed
<b>Name:</b>	Item Unidentified Check
<b>Initial State:</b>	Identified item
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that item is unidentified
<b>Name:</b>	Item Identified Check
<b>Initial State:</b>	Unidentified item
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that item is identified
<b>Name:</b>	Item Display-Name Check 1
<b>Initial State:</b>	Unidentified item
<b>Input:</b>	Psuedoname
<b>Expected Output:</b>	Verification that item's display name matches psuedoname
<b>Name:</b>	Item Display-Name Check 2
<b>Initial State:</b>	Identified item
<b>Input:</b>	True name
<b>Expected Output:</b>	Verification that item's display name matches true name
<b>Name:</b>	ItemZone Containment Check 1
<b>Initial State:</b>	ItemZone with 2 items
<b>Input:</b>	None

<b>Expected Output:</b>	Verification that ItemZone contains the first item
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	ItemZone Containment Check 2 ItemZone with 2 items None Verification that ItemZone contains the second item
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	ItemZone Empty Check ItemZone with 2 items None Verification that ItemZone is not empty
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	ItemZone Size Check ItemZone with 2 items None Verification that ItemZone's size is 2
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	ItemZone Keybind Check 1 ItemZone with 2 items None Verification that first item is bound to 'a' key
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	ItemZone Keybind Check 2 ItemZone with 2 items None Verification that second item is bound to 'b' key
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	ItemZone Contents Retrieval 1 ItemZone with 2 items None Item map with exactly 1 copy of first item
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	ItemZone Contents Retrieval 2 ItemZone with 2 items None Item map with exactly 1 copy of second item

<b>Name:</b>	ItemZone Removal
<b>Initial State:</b>	ItemZone with 2 items
<b>Input:</b>	Removal command
<b>Expected Output:</b>	ItemZone with only second item
<b>Name:</b>	ItemZone Keybind Persistence
<b>Initial State:</b>	ItemZone with first item removed
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that second item is still bound to 'b'
<b>Name:</b>	ItemZone Weight Enforcement
<b>Initial State:</b>	Empty ItemZone
<b>Input:</b>	Attempt to add 500 pieces of armor to ItemZone
<b>Expected Output:</b>	ItemZone with max-weight worth of armor
<b>Name:</b>	Level Construction
<b>Initial State:</b>	None
<b>Input:</b>	Depth, player object
<b>Expected Output:</b>	Level object in valid initial state
<b>Name:</b>	Level Depth Check
<b>Initial State:</b>	Level with given depth
<b>Input:</b>	Depth value
<b>Expected Output:</b>	Verification that level's depth matches given value
<b>Name:</b>	Level BFSPerp Diagonal Small
<b>Initial State:</b>	Empty level object
<b>Input:</b>	Pair of coordinates diagonally adjacent
<b>Expected Output:</b>	Path between coordinates with expected length, utilizing taxicab movement
<b>Name:</b>	Level BFSPerp Horizontal
<b>Initial State:</b>	Empty level object
<b>Input:</b>	Pair of coordinates with equal y-values

<b>Expected Output:</b>	Path between coordinates with expected length, utilizing taxicab movemen
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Level BFSPerp Vertical Empty level object Pair of coordinates with equal x-values Path between coordinates with expected length, utilizing taxicab movemen
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Level BFSDiag Horizontal Empty level object Pair of coordinates with equal y-values Path between coordinates with expected length, utilizing orthogonal movement
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Level BFSDiag Vertical Empty level object Pair of coordinates with equal x-values Path between coordinates with expected length, utilizing orthogonal movement
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Level BFSPerp Diagonal Empty level object Pair of coordinates on diagonal line Path between coordinates with expected length, utilizing taxicab movement
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Level Starting Position Empty level object None Valid starting position coordinate
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Level getAdjPassable Empty level object Coordinate List of coordinates orthogonally adjacent to given coordinate



<b>Name:</b>	Level Path Generation
<b>Initial State:</b>	Player object and generated level
<b>Input:</b>	Series of path requests between random coordinates
<b>Expected Output:</b>	Valid paths between locations
<b>Name:</b>	Level Connectedness
<b>Initial State:</b>	Player object and generated level
<b>Input:</b>	Series of path requests between all rooms in the level
<b>Expected Output:</b>	Valid paths between each room
<b>Name:</b>	Level Staircase Check
<b>Initial State:</b>	Player object and generated level
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that level contains a staircase
<b>Name:</b>	Level GoldPile Check
<b>Initial State:</b>	Player object and generated level
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that level contains at least one goldpile
<b>Name:</b>	Monster Construction
<b>Initial State:</b>	None
<b>Input:</b>	Symbol, coordinate, armor value, HP value, exp value, level value, maxHP value, name value
<b>Expected Output:</b>	Monster object in valid initial state
<b>Name:</b>	Dice-Math 1
<b>Initial State:</b>	None
<b>Input:</b>	1 1-sided die
<b>Expected Output:</b>	Sum of values of 1
<b>Name:</b>	Dice-Math 2
<b>Initial State:</b>	None
<b>Input:</b>	2 1-sided die
<b>Expected Output:</b>	Sum of values of 2

<b>Name:</b>	Dice-Math 3
<b>Initial State:</b>	None
<b>Input:</b>	1 2-sided die
<b>Expected Output:</b>	1 j= Sum of values j= 2
<b>Name:</b>	Dice-Math 4
<b>Initial State:</b>	None
<b>Input:</b>	3 4-sided die
<b>Expected Output:</b>	3 j= Sum of values j= 12
<b>Name:</b>	Mob Armor Check
<b>Initial State:</b>	Mob object
<b>Input:</b>	None
<b>Expected Output:</b>	Verification mob armor is in valid range
<b>Name:</b>	Mob HP Check 1
<b>Initial State:</b>	Mob with given HP value
<b>Input:</b>	HP value
<b>Expected Output:</b>	Verification mob has correct HP value
<b>Name:</b>	Mob MaxHP Check
<b>Initial State:</b>	Mob with given MaxHP value
<b>Input:</b>	MaxHP value
<b>Expected Output:</b>	Verification mob has correct MaxHP value
<b>Name:</b>	Mob Level Check
<b>Initial State:</b>	Mob with given level value
<b>Input:</b>	Level value
<b>Expected Output:</b>	Verification mob has correct level value
<b>Name:</b>	Mob Location Check
<b>Initial State:</b>	Mob with given location
<b>Input:</b>	Coordinate
<b>Expected Output:</b>	Verification mob has correct location
<b>Name:</b>	Mob Name Check

<b>Initial State:</b>	Mob with given name
<b>Input:</b>	Name value
<b>Expected Output:</b>	Verification mob has correct name
<b>Name:</b>	Mob setMaxHP
<b>Initial State:</b>	Mob with default MaxHP
<b>Input:</b>	setMaxHP command with MaxHP value
<b>Expected Output:</b>	mob with given MaxHP value
<b>Name:</b>	Mob setCurrentHP
<b>Initial State:</b>	Mob with default currentHP
<b>Input:</b>	setCurrentHP command with currentHP value
<b>Expected Output:</b>	mob with given currentHP value
<b>Name:</b>	Mob Dead Check 1
<b>Initial State:</b>	Living Mob object
<b>Input:</b>	None
<b>Expected Output:</b>	Verification mob is alive
<b>Name:</b>	Mob HP Check 2
<b>Initial State:</b>	Living Mob object
<b>Input:</b>	Hit command for $i$ mob's current HP
<b>Expected Output:</b>	Verification mob has HP $j = 0$
<b>Name:</b>	Mob Dead Check 2
<b>Initial State:</b>	Dead mob object
<b>Input:</b>	None
<b>Expected Output:</b>	Verification mob is dead
<b>Name:</b>	Monster Construction
<b>Initial State:</b>	None
<b>Input:</b>	Symbol, coordinate
<b>Expected Output:</b>	Monster object in valid initial state
<b>Name:</b>	Monster Flag/Invisibility
<b>Initial State:</b>	Visible monster object
<b>Input:</b>	SetFlag command to make monster invisible

<b>Expected Output:</b>	Invisible monster object
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Monster Aggrevate Idling, sleeping monster object Aggrevate command Awake, chasing monster object
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Monster Damage Calculation Monster object calculateDamage command Correct amount of damage
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Monster Hit Chance Monster and player objects calculateHitChange command Hit chance in valid range
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Monster Armor Check Monster object None Verification that monster armor is in valid range
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Invisible Monster Name Check Invisible uonster object None Verification monster has hidden name
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Visible Monster Name Check Invisible monster object RemoveFlag command to make monster invisible Verification monster has real name
<b>Name:</b> <b>Initial State:</b> <b>Input:</b>	Monster Symbol/Level Association None Depth value

<b>Expected Output:</b>	Set of symbols for monsters that are valid candidates for given depth
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Monster Symbol/Treasure/Level Association None Depth value Set of symbols for monsters that are valid candidates for given depth for a treasure room
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	PlayerChar Initial Amulet Check Just initialized playerchar object None Verification the game does not believe the player has the amulet
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	PlayerChar Initial HP Check Just initialized playerchar object None Verification playerchar has full hp
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	PlayerChar Level-Up Exp Playerchar object at initial level Exp input into playerchar object Playerchar object with increased level
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	PlayerChar Level-Up Manual Playerchar object Level-up command Playerchar object with increased level
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	PlayerChar Damage Playerchar object at full hp Series of damage commands applied to playerchar object Playerchar object with less than full hp
<b>Name:</b> <b>Initial State:</b>	PlayerChar UnArmed 1 Unarmed playerchar object

<b>Input:</b>	calculateDamage command
<b>Expected Output:</b>	0 damage value
<b>Name:</b>	PlayerChar Armed
<b>Initial State:</b>	Playerchar object armed with weapon
<b>Input:</b>	calculateDamage command
<b>Expected Output:</b>	Damage value $\geq 0$
<b>Name:</b>	PlayerChar Stow Weapon
<b>Initial State:</b>	Playerchar object armed with uncursed weapon
<b>Input:</b>	removeWeapon command
<b>Expected Output:</b>	PlayerChar object unarmed
<b>Name:</b>	PlayerChar UnArmed 2
<b>Initial State:</b>	Armed playerchar object
<b>Input:</b>	removeWeapon command, then calculateDamage
<b>Expected Output:</b>	0 damage value
<b>Name:</b>	PlayerChar Remove Non-Armor
<b>Initial State:</b>	Playerchar object with no armor
<b>Input:</b>	removeArmor command
<b>Expected Output:</b>	Boolean indicating failure to remove armor
<b>Name:</b>	PlayerChar Remove Armor
<b>Initial State:</b>	Playerchar object with uncursed armor
<b>Input:</b>	removeArmor command
<b>Expected Output:</b>	Playerchar object without armor
<b>Name:</b>	Potion Construction 1
<b>Initial State:</b>	None
<b>Input:</b>	Coordinate
<b>Expected Output:</b>	Potion object in valid initial state
<b>Name:</b>	Potion Construction 2
<b>Initial State:</b>	None
<b>Input:</b>	Coordinate, item context value, item type specifier

<b>Expected Output:</b>	Potion object in valid initial state
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Potion of Strength Player object Potion of strength Player with strength increased by 1
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Potion of Restore Strength Player object with reduced strength Potion of restore strength Player object with pre-reduction strength
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Potion of Healing Player object with full hp Potion of healing Player object with maxHP increased by 1
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Potion of Extra Healing Player object with full hp Potion of extra healing Player object with maxHP increased by 2
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Potion of Poison Player object with strength $\geq 0$ Potion of poison Player object with reduced strength
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Potion of Raise Level Player object with less than max level Potion or raise level Player object with level + 1
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Potion of Blindness Player object without the blindness condition Potion of blindness Player object with the blindness condition

<b>Name:</b>	Potion of Hallucination
<b>Initial State:</b>	Player object without the hallucination condition
<b>Input:</b>	Potion of hallucination
<b>Expected Output:</b>	Player object with the hallucination condition
<b>Name:</b>	Potion of Detect Monster
<b>Initial State:</b>	Player object without the detect-monsters condition
<b>Input:</b>	Potion of detect monsters
<b>Expected Output:</b>	Player object with the detect-monsters condition
<b>Name:</b>	Potion of Detect Object
<b>Initial State:</b>	Player object without the detect-objects condition
<b>Input:</b>	Potion of detect objects
<b>Expected Output:</b>	Player object with the detect-objects condition
<b>Name:</b>	Potion of Confusion
<b>Initial State:</b>	Player object without the confusion condition
<b>Input:</b>	Potion of confusion
<b>Expected Output:</b>	Player object with the confusion condition
<b>Name:</b>	Potion of Confusion
<b>Initial State:</b>	Player object without the confusion condition
<b>Input:</b>	Potion of confusion
<b>Expected Output:</b>	Player object with the confusion condition
<b>Name:</b>	Potion of Levitation
<b>Initial State:</b>	Player object without the levitation condition
<b>Input:</b>	Potion of levitation
<b>Expected Output:</b>	Player object with the levitation condition
<b>Name:</b>	Potion of Haste
<b>Initial State:</b>	Player object without the haste condition
<b>Input:</b>	Potion of haste
<b>Expected Output:</b>	Player object with the haste condition
<b>Name:</b>	Potion of See-Invisible



<b>Initial State:</b>	Player object without the invisible-sight condition
<b>Input:</b>	Potion of invisible
<b>Expected Output:</b>	Player object with the invisible-sight condition
<b>Name:</b>	Random Range 1
<b>Initial State:</b>	None
<b>Input:</b>	Upper and lower bounds 0,0
<b>Expected Output:</b>	0
<b>Name:</b>	Random Range 2
<b>Initial State:</b>	None
<b>Input:</b>	Upper and lower bounds 5,5
<b>Expected Output:</b>	5
<b>Name:</b>	Random Range 3
<b>Initial State:</b>	None
<b>Input:</b>	Upper and lower bounds 0,60, repeated 40 times
<b>Expected Output:</b>	0 j= result j= 60
<b>Name:</b>	Random Float
<b>Initial State:</b>	None
<b>Input:</b>	40 repeats
<b>Expected Output:</b>	0 j= result j= 1
<b>Name:</b>	Random Boolean
<b>Initial State:</b>	None
<b>Input:</b>	10 repeats
<b>Expected Output:</b>	Both true and false are generated
<b>Name:</b>	Random Percent
<b>Initial State:</b>	None
<b>Input:</b>	40 repeats
<b>Expected Output:</b>	0 j= result j= 100
<b>Name:</b>	Random Position
<b>Initial State:</b>	None

<b>Input:</b>	Two coordinates, as top-left and bottom-right of rectangle, 10 repeats
<b>Expected Output:</b>	Random coordinates within the bounds
<b>Name:</b>	Ring Construction 1
<b>Initial State:</b>	None
<b>Input:</b>	Coordinate
<b>Expected Output:</b>	Ring object with valid initial state
<b>Name:</b>	Ring Construction 2
<b>Initial State:</b>	None
<b>Input:</b>	Coordinate, item context value, type identifier
<b>Expected Output:</b>	Ring object with valid initial state
<b>Name:</b>	Ring of Stealth
<b>Initial State:</b>	Player object without stealth condition
<b>Input:</b>	Ring of stealth
<b>Expected Output:</b>	Player object with the stealth condition
<b>Name:</b>	Ring of Stealth Deactivate
<b>Initial State:</b>	Player object with ring of stealth
<b>Input:</b>	Remove ring
<b>Expected Output:</b>	Player object without the stealth condition
<b>Name:</b>	Ring of Teleportation
<b>Initial State:</b>	Player object without random teleportation condition
<b>Input:</b>	Ring of teleportation
<b>Expected Output:</b>	Player object with the random teleportation condition
<b>Name:</b>	Ring of Teleportation Deactivate
<b>Initial State:</b>	Player object with ring of teleportation
<b>Input:</b>	Remove ring
<b>Expected Output:</b>	Player object without the random teleportation condition
<b>Name:</b>	Ring of Regeneration
<b>Initial State:</b>	Player object without regeneration condition

<b>Input:</b>	Ring of regeneration
<b>Expected Output:</b>	Player object with the regeneration condition
<b>Name:</b>	Ring of Regeneration Deactivate
<b>Initial State:</b>	Player object with ring of regeneration
<b>Input:</b>	Remove ring
<b>Expected Output:</b>	Player object without the regeneration condition
<b>Name:</b>	Ring of Digestion
<b>Initial State:</b>	Player object without digestion condition
<b>Input:</b>	Ring of digestion
<b>Expected Output:</b>	Player object with the digestion condition
<b>Name:</b>	Ring of Digestion Deactivate
<b>Initial State:</b>	Player object with ring of digestion
<b>Input:</b>	Remove ring
<b>Expected Output:</b>	Player object without the digestion condition
<b>Name:</b>	Ring of Dexterity
<b>Initial State:</b>	Player object
<b>Input:</b>	Ring of dexterity
<b>Expected Output:</b>	Player object with dexterity increased by the appropriate amount
<b>Name:</b>	Ring of Dexterity Deactivate
<b>Initial State:</b>	Player object with ring of dexterity
<b>Input:</b>	Remove ring
<b>Expected Output:</b>	Player object with normal dexterity
<b>Name:</b>	Ring of Adornment
<b>Initial State:</b>	Player object
<b>Input:</b>	Ring of adornment
<b>Expected Output:</b>	Identical player object
<b>Name:</b>	Ring of Adornment
<b>Initial State:</b>	Player object with ring of adornment
<b>Input:</b>	Remove ring

<b>Expected Output:</b>	Identical player object
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Ring of See-Invisible Player object without the see-invisible condition Ring of see-invisible Player object with the see-invisible condition
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Ring of See-Invisible Deactivate Player object with ring of see-invisible Remove ring Player object without the see-invisible condition
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Ring of Maintain-Armor Player object without the maintain-armor condition Ring of maintain-armor Player object with the maintain-armor condition
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Ring of Maintain-Armor Deactivate Player object with ring of maintain-armor Remove ring Player object without the maintain-armor condition
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Ring of Searching Player object without the auto-search condition Ring of searching Player object with the auto-search condition
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Ring of Searching Deactivate Player object with ring of searching Remove ring Player object without the auto-search condition
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Room Construction Check 1 Randomly generated room None Verification that room's size is in valid range

<b>Name:</b>	Room Construction Check 2
<b>Initial State:</b>	Randomly generated room
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that room edges are within valid bounds
<b>Name:</b>	Scroll Construction 1
<b>Initial State:</b>	None
<b>Input:</b>	Coordinate
<b>Expected Output:</b>	Scroll object in valid initial state
<b>Name:</b>	Scroll Construction 2
<b>Initial State:</b>	None
<b>Input:</b>	Coordinate, item context value, type identifier
<b>Expected Output:</b>	Scroll object in valid initial state
<b>Name:</b>	Scroll PseudoNames
<b>Initial State:</b>	Scrolls are uninitialized
<b>Input:</b>	initializeScrollNames command
<b>Expected Output:</b>	Vector of valid scroll psuedonames
<b>Name:</b>	Scroll of Protect Armor
<b>Initial State:</b>	Player with cursed armor
<b>Input:</b>	Scroll of protect armor
<b>Expected Output:</b>	Player with uncursed armor with protect-armor effect
<b>Name:</b>	Scroll of Hold Monster
<b>Initial State:</b>	Monster without the held flag
<b>Input:</b>	Scroll of hold monster
<b>Expected Output:</b>	Monster with the held flag
<b>Name:</b>	Scroll of Enchant Weapon
<b>Initial State:</b>	Player with weapon
<b>Input:</b>	Scroll of enchant weapon
<b>Expected Output:</b>	Player with uncursed weapon with higher enchant level
<b>Name:</b>	Scroll of Enchant Armor

<b>Initial State:</b>	Player with armor
<b>Input:</b>	Scroll of enchant armor
<b>Expected Output:</b>	Player with uncursed armor with higher enchant level
<b>Name:</b>	Scroll of Identity
<b>Initial State:</b>	None
<b>Input:</b>	Scroll identity
<b>Expected Output:</b>	No exceptions
<b>Name:</b>	Scroll of Teleportation
<b>Initial State:</b>	Player at coordinate (0,0)
<b>Input:</b>	Scroll of teleportation
<b>Expected Output:</b>	Player at coordinate != (0,0)
<b>Name:</b>	Scroll of Sleep
<b>Initial State:</b>	Player without the sleep condition
<b>Input:</b>	Scroll of sleep
<b>Expected Output:</b>	Player with the sleep condition
<b>Name:</b>	Scroll of Scare Monster
<b>Initial State:</b>	None
<b>Input:</b>	Scroll of scare monster
<b>Expected Output:</b>	No exceptions
<b>Name:</b>	Scroll of Remove Curse
<b>Initial State:</b>	Player with cursed weapon
<b>Input:</b>	Scroll of remove curse
<b>Expected Output:</b>	Player with uncursed weapon
<b>Name:</b>	Scroll of Create Monster
<b>Initial State:</b>	Level object
<b>Input:</b>	Scroll of create monster
<b>Expected Output:</b>	Level with 1 additional monster
<b>Name:</b>	Scroll of Aggravate Monster
<b>Initial State:</b>	Level with sleeping monsters
<b>Input:</b>	Scroll of aggravate monster

<b>Expected Output:</b>	Level with no sleeping monsters
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Scroll of Magic Mapping Unrevealed level Scroll of magic mapping Level where all tiles have been revealed
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Scroll of Confuse Monster Player without the confuse-monster condition Scroll of confuse monster Player with the confuse-monster condition
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Stair Construction None Coordinate, direction value Stair object in valid initial state
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Stair Direction Check Stair constructed with direction Direction value Verification stair has given direction value
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Floor Passability Check Floor object None Verification floor is passable
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Floor Symbol Check Floor object None Verification floor has correct symbol
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Floor Transparency Check Floor object None Verification floor is transparent

<b>Name:</b>	Wall Passability Check
<b>Initial State:</b>	Wall object
<b>Input:</b>	None
<b>Expected Output:</b>	Verification wall is not passable
<b>Name:</b>	Wall Symbol Check
<b>Initial State:</b>	Wall object
<b>Input:</b>	None
<b>Expected Output:</b>	Verification wall has correct symbol
<b>Name:</b>	Wall Opacity Check
<b>Initial State:</b>	Wall object
<b>Input:</b>	None
<b>Expected Output:</b>	Verification wall is transparent
<b>Name:</b>	Corridor Passability Check
<b>Initial State:</b>	Corridor object
<b>Input:</b>	None
<b>Expected Output:</b>	Verification corridor is passable
<b>Name:</b>	Corridor Symbol Check
<b>Initial State:</b>	Corridor object
<b>Input:</b>	None
<b>Expected Output:</b>	Verification corrido has correct symbol
<b>Name:</b>	Corridor Transparency Check
<b>Initial State:</b>	Corridor object
<b>Input:</b>	None
<b>Expected Output:</b>	Verification corridor has special corridor transparency
<b>Name:</b>	Door Passability Check
<b>Initial State:</b>	Door object
<b>Input:</b>	None
<b>Expected Output:</b>	Verification door is not passable
<b>Name:</b>	Door Symbol Check



<b>Initial State:</b>	Door object
<b>Input:</b>	None
<b>Expected Output:</b>	Verification corridor has correct symbol
<b>Name:</b>	Door Transparency Check
<b>Initial State:</b>	Door object
<b>Input:</b>	None
<b>Expected Output:</b>	Verification Door has special corridor transparency
<b>Name:</b>	Door Trap
<b>Initial State:</b>	Player and level
<b>Input:</b>	Door trap
<b>Expected Output:</b>	Player at a level with depth + 1
<b>Name:</b>	Rust Trap
<b>Initial State:</b>	Player with enchanted weapon
<b>Input:</b>	Rust trap
<b>Expected Output:</b>	Player with unenchanted weapon
<b>Name:</b>	Sleep Trap
<b>Initial State:</b>	Player without the sleep condition
<b>Input:</b>	Sleep trap
<b>Expected Output:</b>	Player with the sleep condition
<b>Name:</b>	Bear Trap
<b>Initial State:</b>	Player without the immobilized condition
<b>Input:</b>	Bear trap
<b>Expected Output:</b>	Player with the immobilized condition
<b>Name:</b>	Teleport Trap
<b>Initial State:</b>	Player
<b>Input:</b>	Teleport trap
<b>Expected Output:</b>	Player at a different location
<b>Name:</b>	Dart Trap
<b>Initial State:</b>	Player
<b>Input:</b>	Dart trap

<b>Expected Output:</b>	Player with less hp
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Tunnel Digging Level and pair of unconnected rooms Dig command Valid path between the two rooms
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Open Inventory Screen Playstate, player, empty level Inventory key Inventory screen
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Close Inventory Screen Inventory screen, player, empty level Exit key Playstate
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Movement Playstate, player, empty level Movement key Player should be in expected location in the level
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Open Status Screen Playstate, player, empty level Status key Status screen
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Exit Status Screen Status Screen, player, empty level Exit key Playstate
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	No Wand Zap Playstate, player with no wand Zap key Unchanged playstate

<b>Name:</b>	Zap Wand Select
<b>Initial State:</b>	Playstate, player with wand, empty level
<b>Input:</b>	Zap key, then direction key
<b>Expected Output:</b>	Inventory Screen
<b>Name:</b>	Zap Wand Fire 1
<b>Initial State:</b>	Inventory wand select
<b>Input:</b>	Item select hotkey
<b>Expected Output:</b>	Playstate
<b>Name:</b>	Zap Wand Fire 2
<b>Initial State:</b>	Inventory wand select
<b>Input:</b>	Item select hotkey
<b>Expected Output:</b>	wand with charges - 1
<b>Name:</b>	Game Quit
<b>Initial State:</b>	Playstate
<b>Input:</b>	Quit key and confirmation key
<b>Expected Output:</b>	RIPScreen
<b>Name:</b>	Wand Construction 1
<b>Initial State:</b>	None
<b>Input:</b>	Coordinate
<b>Expected Output:</b>	Wand in valid initial state
<b>Name:</b>	Wand Construction 2
<b>Initial State:</b>	None
<b>Input:</b>	Coordinate, item context value, type specifier
<b>Expected Output:</b>	Wand in valid initial state
<b>Name:</b>	Wand of Teleport Away
<b>Initial State:</b>	Player, nearby monster
<b>Input:</b>	Wand of teleport away
<b>Expected Output:</b>	Monster has distance to player $\leq 20$
<b>Name:</b>	Wand of Slow Monster

<b>Initial State:</b>	Player, monster without slowed flag
<b>Input:</b>	Wand of slow monster
<b>Expected Output:</b>	Monster has slowed flag
<b>Name:</b>	Wand of Invisibility
<b>Initial State:</b>	Player, monster without invisible flag
<b>Input:</b>	Wand of invisibility
<b>Expected Output:</b>	Monster with invisible flag
<b>Name:</b>	Wand of Polymorph
<b>Initial State:</b>	Player, monster
<b>Input:</b>	Wand of polymorph
<b>Expected Output:</b>	Different monster at previous monster's locations
<b>Name:</b>	Wand of Haste Monster
<b>Initial State:</b>	Player, monster without haste flag
<b>Input:</b>	Wand of haste monster
<b>Expected Output:</b>	Monster with haste flag
<b>Name:</b>	Wand of Magic Missile
<b>Initial State:</b>	Player, monster
<b>Input:</b>	Wand of magic missile
<b>Expected Output:</b>	Monster with reduced hp
<b>Name:</b>	Wand of Cancellation
<b>Initial State:</b>	Player, monster without cancelled flag
<b>Input:</b>	Wand of cancellation
<b>Expected Output:</b>	Monster with cancelled flag
<b>Name:</b>	Wand of Do Nothing
<b>Initial State:</b>	Player, monster
<b>Input:</b>	Wand of do nothing
<b>Expected Output:</b>	No exceptions
<b>Name:</b>	Wand of Drain Life
<b>Initial State:</b>	Player with reduced health, monster
<b>Input:</b>	Wand of drain life

<b>Expected Output:</b>	Player with increased health, monster with reduced health
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Wand of Cold Player, monster Wand of cold No exceptions
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Wand of Fire Player, monster Wand of fire No exceptions
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Weapon Construction 1 None Coordinate Weapon in valid initial state
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Weapon Construction 2 None Coordinate, item context value, type specifier Weapon in valid initial state
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Weapon Identification Check Identified weapon None Verification that weapon is identified
<b>Name:</b> <b>Initial State:</b> <b>Input:</b> <b>Expected Output:</b>	Weapon Curse Check Cursed weapon None Verification that weapon is cursed
<b>Name:</b> <b>Initial State:</b> <b>Input:</b>	Weapon Name Check Weapon None

<b>Expected Output:</b>	Verification that weapon has valid name
<b>Name:</b>	Weapon Enchantment Check
<b>Initial State:</b>	Cursed weapon
<b>Input:</b>	None
<b>Expected Output:</b>	Verification that weapon has expected enchantment values

## 8 Trace to Requirements

The following table maps each implemented test file to a set of functional and non-functional requirements

Table 3: **Test-Requirement Trace**

File	Related Requirement(s)
test.amulet.cpp	FR.25
test.armor.cpp	FR.29, FR.34, FR.39,
test.coord.cpp	FR.17
test.feature.cpp	FR.5, FR.13, FR.14, FR.15, FR.25, FR.31
test.food.cpp	FR.5, FR.31
test.goldpile.cpp	FR.5
test.item.cpp	FR.5, FR.13, FR.14, FR.15, FR.25, FR.30 FR.31
test.itemzone.cpp	FR.5, FR.9, FR.26
test.level.cpp	FR.16-19
test.levelgen.cpp	FR.16-19
test.main.cpp	Put everything together
test.mob.cpp	FR.37, FR.38, FR.39
test.monster.cpp	FR.35-39
test.playerchar.cpp	FR.9-15, FR.26-34, NFR.5
test.potion.cpp	FR.5, FR.13, FR.14, FR.15, FR.25, FR.31
test.ring.cpp	FR.5, FR.13, FR.14, FR.15, FR.25, FR.31
test.room.cpp	FR.17, FR.18, FR.19, FR.21
test.scroll.cpp	FR.5, FR.13, FR.14, FR.15, FR.25, FR.31
test.stairs.cpp	FR.18, FR.19
test.terrain.cpp	FR.13, FR.15, FR.18, FR.19, FR.23, FR.24
test.testable.cpp	Defines test-suite
test.testable.h	Defines test-suite
test.trap.cpp	FR.12, FR.15, FR.19, FR.20, FR.23, FR.24, FR.34
test.tunnel.cpp	FR.17, FR.19
test.uistate.cpp	FR.1-4, FR.6-10, NFR.1, NFR.3, NFR.5
test.wand.cpp	FR.5, FR.13, FR.14, FR.15, FR.25, FR.31
test.weapon.cpp	FR.5, FR.13, FR.14, FR.15, FR.25, FR.31

## 9 Trace to Modules

The following table re-iterates the modules of the project, along with their respective domain and module ID. The module IDs are used to refer to modules in the trace. More about the modules can be found in the Module Guide.

Table 5: **Module Hierarchy**

<b>Level 1</b>	<b>Level 2</b>	
Hardware-Hiding Module	BasicIO	<b>M1</b>
	Doryen	<b>M2</b>
	Input Format	<b>M3</b>
Behaviour-Hiding Module	External	<b>M4</b>
	Item	<b>M5</b>
	Level	<b>M6</b>
	LevelGen	<b>M7</b>
	MainMenu	<b>M8</b>
	Mob	<b>M9</b>
	Monster	<b>M10</b>
	PlayerChar	<b>M11</b>
	RipScreen	<b>M12</b>
	PlayState	<b>M13</b>
	SaveScreen	<b>M14</b>
	UIState	<b>M15</b>
Software Decision Module	Coord	<b>M16</b>
	Feature	<b>M17</b>
	ItemZone	<b>M18</b>
	MasterController	<b>M19</b>
	Random	<b>M20</b>
	Terrain	<b>M21</b>

The following table maps test files, which implement tests, to specific modules, given by their IDs.



Table 6: **Test-Module Trace**

File	Related Module(s)
test.amulet.cpp	M7, M13, M15
test.armor.cpp	M5, M9, M11
test.coord.cpp	M2, M5, M6, M7, M16, M20
test.feature.cpp	M5, M11, M17, M18
test.food.cpp	M5, M6, M7, M11, M13
test.goldpile.cpp	M5, M6, M7, M10, M11, M17, M18
test.item.cpp	M5, M17
test.itemzone.cpp	M5, M6, M16, M17, M18
test.level.cpp	M5, M6, M10, M11, M16, M17, M20
test.levelgen.cpp	M5, M6, M10, M16, M17, M20, M21
test.main.cpp	None (Puts everything together)
test.mob.cpp	M9, M10, M11, M13, M15, M16
test.monster.cpp	M9, M10, M16
test.playerchar.cpp	M5, M6, M9, M11, M12, M13, M15, M16, M17, M18, M19
test.potion.cpp	M5, M6, M7, M10, M11, M17, M18
test.ring.cpp	M5, M6, M7, M10, M11, M17, M18
test.room.cpp	M6, M7, M16, M20
test.scroll.cpp	M5, M6, M7, M10, M11, M17, M18
test.stairs.cpp	M7, M17, M19, M21
test.terrain.cpp	M6, M7, M20, M21
test.testable.cpp	Defines test-suite
test.testable.h	Defines test-suite
test.trap.cpp	M6, M7, M11, M15, M17
test.tunnel.cpp	M5, M6, M16
test.uistate.cpp	M4, M8, M12, M13, M15, M19
test.wand.cpp	M5, M6, M7, M10, M11, M17, M18
test.weapon.cpp	M5, M6, M7, M10, M11, M17, M18

## 10 Code Coverage Metrics

By looking at the test-requirements matrix, and also cross-referencing the test-module trace above with the module-requirements trace given in the Module Guide, it is possible to determine exactly which functional and non-functional requirements were satisfied with the test cases we created.

As can be expected, near **complete coverage** of both functional and non-functional requirements is achieved. Except for a few non-functional requirements, the modules and direct requirements reflected in the test cases offer a complete coverage of the requirements. Some (in particular, non-functional) requirements are nigh impossible to test using code. An example includes NFR.2: "The Rogue Reborn game shall be fun and entertaining." Whatever software exists that can determine such a thing would never pass the Turing test, and thus can be deemed an impossibility as of current technology. But while it is impossible to test with code, such a thing is easily testable with human playtesters.

Along with NFR.2, several non-functional requirements were not feasible to assert with software, but all were correctly proven by other means, most of which involved manual human labor.

To expand on the previous statements, we encountered some requirements where the achievable target was difficult to materialize, but still algorithmic and computational in nature. A prime example of this is the luminosity constraint, which ruled that no two consecutive frames may have a change in brightness greater than some defined delta. In order to properly measure this, we had to go outside of the program, and write a separate script to do the hard work. We used python to calculate the pixel-accurate luminosity of some key screenshots, and using the calculation proposed by the non-functional requirement, arrived at correct results. The results were deemed close enough to the predefined delta, which itself was based more or less on our intuition.

## References

About SDL. <https://www.libsdl.org/index.php>. Accessed: December 7, 2016.