

SE 3XA3: Test Plan Rogue Reborn

Group #6, Team Rogue++

| | |
|-------------------|---------|
| Ian Prins | prinsij |
| Mikhail Andrenkov | andrem5 |
| Or Almog | almogo |

Due Thursday, December 8th, 2016

Contents

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

List of Tables

| | | |
|---|----------------------------------|----|
| 1 | Revision History | 3 |
| 3 | Test-Requirement Trace | 46 |
| 4 | Module Hierarchy | 47 |
| 5 | 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 | Added Automated Tests To PlayerChar |
| 12/06/16 | 0.3 | Added FR Evaluation |
| 12/07/16 | 0.4 | Added Introduction |
| 12/07/16 | 0.5 | Finished Automated Tests |
| 12/07/16 | 0.6 | Added Metrics and Comparison |
| 12/07/16 | 0.7 | Added NFR Evaluation |
| 12/07/16 | 0.8 | Added Unit Testing |
| 12/08/16 | 0.9 | Added Changes Due to Testing |
| 12/08/16 | 1.0 | Proofread and Editing |

1 Introduction

1.1 Overview

The primary objective of this document is to provide a comprehensive report detailing the results of the verification process applied to the Rogue Reborn project. Interested parties are welcome to analyze this paper as a means of evaluating the success of the final application with respect to 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 Reborn project relative to the expectations of the client.

1.2 Sections

A brief description of the Test Reports sections 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

With the implementation of the application drawing to close, it can be witnessed that Rogue Reborn is functionally complete. One of the primary driving forces behind reaching this goal is the elaborate unit testing suite. It was discovered that most of these tests were quite useful and discovered bugs as well as logic errors that might have otherwise gone unnoticed; these tests are discussed below. However, there were also many tests that were relatively mundane, generic, or crucial enough to have already been satisfied before testing was even considered. As a result, those tests will not be explicitly mentioned given that 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 [SRS](#).

FR.16: While performing tests over the [Level](#) module, a strange anomaly led to one test constantly failing. This test later revealed that the player character did not actually begin at the first level. Due to an off-by-one error and slight miscommunication between developers, the current level depth of the player character was i in some places and $i + 1$ in others. As soon as this bug was uncovered, the problem was remedied globally across the code.

FR.19: Whenever the player uncovers a new dungeon level (including the very first level), an algorithm decides on the player character's initial position. Although this algorithm appeared flawless in the past, there was actually a slight chance of placing the player character in an unreachable location, (i.e., surrounded by walls and therefore doomed forever). With the automatic tests running thousands upon thousands of simulations, this glitch was quickly exposed and resolved.

FR.35: Some monsters in Rogue Reborn follow a simple AI decision algorithm consisting of steps such as:

1. Check if the monster is sleeping or passive
2. If the monster is aggravated, attempt to chase the player character
3. If the monster is aggravated and adjacent to the player character, attempt to hit the player character

Although this is a very simple recipe and is easy to both invent and implement, several enemies in Rogue Reborn go beyond such simple schemes. For instance, some monsters do not seek to kill the player character, but rather steal their precious items or gold. One such pest is the Leprechaun (depicted by the “L” symbol). The Leprechaun necessitated the implementation of a variety of new methods, such as `getNearestGold()`. The tests related to this function revealed some very serious performance issues in the pathfinding algorithm used throughout the project, such as infinite path traces. It also revealed another bug in which the coordinates of several level features (items mostly) were accidentally set to (0,0), rendering them impossible to reach. These issues were due in part by the assumption that all items are placed on tiles reachable from the player character. Fortunately, thanks to the various tests that were implemented, these bugs were fixed.

FR.39: Working with C++ was extremely beneficial in some respects, but also possessed some drawbacks. An anomaly in the way C++ handles integers revealed a very serious bug in the code, in which the player character’s armor could reach ridiculous high values and essentially render them invincible. By simulating every possible [Armor](#) that can be instantiated, this bug was caught and patched. To elaborate, the bug existed because a particular integer had a possibility of never being assigned a value.

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 ~ 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 to 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 [RIP Screen](#); 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 and displayed on the [RUP Screen](#); 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: the original *Rogue* was released in 1980 (almost 40 years ago to date) when the discipline of software engineering was still in its infancy. Since then, the standards of software development have evolved into the rigorous forms they take today.

Given that no prior tests exist, there is nothing to compare. However, if such tests were to exist, the Rogue++ team would consider the following:

- How does the coverage of the existing test cases compare to Rogue Reborn's coverage?
- For cases where the modules are similar across both versions, how do the tests compare? Are there any significant benefits or drawbacks between the approaches?
- Do either of the implementations completely neglect an aspect of the other implementation? Is Rogue Reborn missing a critical functionality test that the original included?
- 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 dependencies and 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 all tests to extend an abstract class `Testable` with the interface described in [Figure 1](#). This restriction enabled the test driver `test.main.cpp` to invoke all of the concrete unit test classes in a uniform manner. Finally, 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);
};
```

Regarding coverage, the tests encompassed all item types and abilities, various player actions and monster mechanics, virtually all level generation and processing algorithms, and even several UI functions. As depicted in [Figure 1](#), each test 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 to guarantee a minimal level of functionality for each documented feature across the builds. Naturally, the suite did not consider the non-functional requirements of the project since these qualities lend themselves better to manual testing.

6 Changes Due to Testing

Since the Rogue++ developers did not embrace a TDD (Test-Driven Development) methodology, all of the unit tests were designed to catch flaws in existing code rather than guide the development of new code. As such, the unit testing phase of the Rogue Reborn application did not spawn any new features but instead served to minimize the quantity of programming errors.

One area that the greatly benefited from the unit tests were the [Item](#) modules, as numerous bugs were caught with respect to the initialization of particular variables (or lack thereof). For example, there was an error where a disenchanted [Armor](#) object would claim a completely nonsensical effective armor value since the enchantment variable was never initialized. Another portion of the tests detected several logic flaws (e.g., the negation of a condition was accidentally checked) present in the effects of certain items. For instance, the *Wand of Cold* was revealed to deal critical damage to ice monsters and almost no damage to fire monsters as a consequence of a negated condition.

With respect to the non-functional testing, the most significant changes arose in the modules that interacted with the file system. In particular, the [RipScreen](#) module was drastically improved by implementing input validation checks to detect anomalies in the high score record file. Additionally, although the actual testing itself did not compel any changes, various GUI elements in Rogue Reborn were enhanced (such as the help screen and splash screen) in preparation for future usability tests.

Aside from catching implementation mistakes, the testing process also influenced the architecture and design of the software system. To facilitate unit testing, various modules such as [PlayState](#) were revamped to improve modularization. Other modules introduced blocks of code that were activated by a special `#define DEBUG` directive; these sections either displayed debug information to standard output or modified conditionals to guarantee a particular trace through a function. Finally, as mentioned in the [Unit Testing](#) section of this document, the CI pipeline of the project was also modified to accommodate the execution of unit tests after each commit to the repository.

7 Automated Testing

7.1 Automated Testing Strategy

With respect to the [Unit Testing](#) section of this document, a custom testing framework was developed for the Rogue Reborn project. A series of files following the naming convention `test.<Class Name>.cpp` were created and contained the unit tests specific to the `<Class Name>` class. Whenever a change was pushed to the Rogue Reborn repository, all of these tests were run and their corresponding output was analyzed by a Python script to determine if any failures occurred.

7.2 Specific System Tests

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

| | |
|---|--|
| Name: Initial State: Input: Expected Output: | Amulet Construction None Coordinate, context value Amulet object in valid initial state |
| Name: Initial State: Input: Expected Output: | Armor Construction 1 None Coordinate Armor object in valid initial state |
| Name: Initial State: Input: Expected Output: | Armor Construction 2 None Coordinate, Context value, type value Armor object in valid initial state |
| Name: Initial State: Input: Expected Output: | Armor Identification Cursed Armor None Verification that Armor is identified |
| Name: | Armor Curse |

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

| | |
|---|--|
| Expected Output: | (1,1) coordinate |
| Name: Initial State: Input: Expected Output: | Feature Construction None Symbol, Coordinate, visibility, TCOColor Feature object in valid initial state |
| Name: Initial State: Input: Expected Output: | Feature Symbol Check Feature with given symbol Symbol Verification that Feature's symbol matches given symbol |
| Name: Initial State: Input: Expected Output: | Feature Invisibility Check Invisible Feature None Verification that Feature is invisible |
| Name: Initial State: Input: Expected Output: | Feature Visibility Check Visible Feature None Verification that Feature is visible |
| Name: Initial State: Input: Expected Output: | Feature Location Check Feature with given location Coordinate Verification that Feature's location matches given Coordinate |
| Name: Initial State: Input: Expected Output: | Food Construction None Coordinate and Context value Food object in valid initial state |
| Name: Initial State: Input: | Food Eating Food and PlayerChar objects None |

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

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

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

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

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

| | |
|-------------------------|--|
| Name: | Level Path Generation |
| Initial State: | PlayerChar object and generated Level |
| Input: | Series of path requests between random Coordinates |
| Expected Output: | Valid paths between locations |
| Name: | Level Connectedness |
| Initial State: | PlayerChar object and generated Level |
| Input: | Series of path requests between all Rooms in the Level |
| Expected Output: | Valid paths between each Room |
| Name: | Level Staircase Check |
| Initial State: | PlayerChar object and generated Level |
| Input: | None |
| Expected Output: | Verification that Level contains a reachable Stairs object |
| Name: | Level GoldPile Check |
| Initial State: | PlayerChar object and generated Level |
| Input: | None |
| Expected Output: | Verification that Level contains at least one GoldPile |
| Name: | Monster Construction |
| Initial State: | None |
| Input: | Symbol, Coordinate, armor value, HP value, exp value, level value, maxHP value, name value |
| Expected Output: | Monster object in valid initial state |
| Name: | Dice-Math 1 |
| Initial State: | None |
| Input: | 1 1-sided die |
| Expected Output: | Sum of values of 1 |
| Name: | Dice-Math 2 |
| Initial State: | None |
| Input: | 2 1-sided die |
| Expected Output: | Sum of values of 2 |

| | |
|-------------------------|---|
| Name: | Dice-Math 3 |
| Initial State: | None |
| Input: | 1 2-sided die |
| Expected Output: | $1 \leq \text{Sum of values} \leq 2$ |
| Name: | Dice-Math 4 |
| Initial State: | None |
| Input: | 3 4-sided die |
| Expected Output: | $3 \leq \text{Sum of values} \leq 12$ |
| Name: | Mob Armor Check |
| Initial State: | Mob object |
| Input: | None |
| Expected Output: | Verification that Mob armor is in valid range |
| Name: | Mob HP Check 1 |
| Initial State: | Mob with given HP value |
| Input: | HP value |
| Expected Output: | Verification tat Mob has correct HP value |
| Name: | Mob MaxHP Check |
| Initial State: | Mob with given MaxHP value |
| Input: | MaxHP value |
| Expected Output: | Verification that Mob has correct MaxHP value |
| Name: | Mob Level Check |
| Initial State: | Mob with given level value |
| Input: | Level value |
| Expected Output: | Verification that Mob has correct level value |
| Name: | Mob Location Check |
| Initial State: | Mob with given location |
| Input: | Coordinate |
| Expected Output: | Verification that Mob has correct location |
| Name: | Mob Name Check |

| | |
|-------------------------|--|
| Initial State: | Mob with given name |
| Input: | Name value |
| Expected Output: | Verification that Mob has correct name |
| Name: | Mob Set Max HP |
| Initial State: | Mob with default MaxHP |
| Input: | <code>setMaxHP</code> command with MaxHP value |
| Expected Output: | Verification that Mob has given MaxHP value |
| Name: | Mob Set Current HP |
| Initial State: | Mob with default currentHP |
| Input: | <code>setCurrentHP</code> command with currentHP value |
| Expected Output: | Verification that Mob has given currentHP value |
| Name: | Mob Dead Check 1 |
| Initial State: | Living Mob object |
| Input: | None |
| Expected Output: | Verification that Mob is alive |
| Name: | Mob HP Check 2 |
| Initial State: | Living Mob object |
| Input: | Hit command for \gg Mob's current HP |
| Expected Output: | Verification that Mob has $HP \leq 0$ |
| Name: | Mob Dead Check 2 |
| Initial State: | Dead Mob object |
| Input: | None |
| Expected Output: | Verification that Mob is dead |
| Name: | Monster Construction |
| Initial State: | None |
| Input: | Symbol, Coordinate |
| Expected Output: | Monster object in valid initial state |
| Name: | Monster Flag/Invisibility |
| Initial State: | Visible Monster object |
| Input: | Set Flag command to make Monster invisible |

| | |
|---|--|
| Expected Output: | Invisible Monster object |
| Name: Initial State: Input: Expected Output: | Monster Aggravate Idling, sleeping Monster object Aggravate command Awake, chasing Monster object |
| Name: Initial State: Input: Expected Output: | Monster Damage Calculation Monster object Calculate Damage command Correct amount of damage |
| Name: Initial State: Input: Expected Output: | Monster Hit Chance Monster and PlayerChar objects Calculate Hit Chance command Hit chance in valid range |
| Name: Initial State: Input: Expected Output: | Monster Armor Check Monster object None Verification that Monster armor is in valid range |
| Name: Initial State: Input: Expected Output: | Invisible Monster Name Check Invisible Monster object None Verification that Monster has hidden name |
| Name: Initial State: Input: Expected Output: | Visible Monster Name Check Invisible Monster object Remove Flag command to make Monster invisible Verification that Monster has real name |
| Name: Initial State: Input: | Monster Symbol/Level Association None Depth value |

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

| | |
|-------------------------|---|
| Initial State: | Unarmed PlayerChar object |
| Input: | Calculate Damage command |
| Expected Output: | 0 damage value |
| Name: | PlayerChar Armed |
| Initial State: | PlayerChar object armed with Weapon |
| Input: | Calculate Damage command |
| Expected Output: | Damage value i 0 |
| Name: | PlayerChar Stow Weapon |
| Initial State: | PlayerChar object armed with uncursed weapon |
| Input: | Remove Weapon command |
| Expected Output: | PlayerChar object unarmed |
| Name: | PlayerChar Unarmed 2 |
| Initial State: | Armed PlayerChar object |
| Input: | Remove Weapon command, then Calculate Damage |
| Expected Output: | 0 damage value |
| Name: | PlayerChar Remove Non-Armor |
| Initial State: | PlayerChar object with no Armor |
| Input: | Remove Armor command |
| Expected Output: | Boolean indicating failure to remove Armor |
| Name: | PlayerChar Remove Armor |
| Initial State: | PlayerChar object with uncursed Armor |
| Input: | Remove Armor command |
| Expected Output: | PlayerChar object without Armor |
| Name: | Potion Construction 1 |
| Initial State: | None |
| Input: | Coordinate |
| Expected Output: | Potion object in valid initial state |
| Name: | Potion Construction 2 |
| Initial State: | None |
| Input: | Coordinate, Item context value, Item type specifier |

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

| | |
|-------------------------|---|
| Name: | Potion of Hallucination |
| Initial State: | PlayerChar object without the hallucination condition |
| Input: | Potion of hallucination |
| Expected Output: | PlayerChar object with the hallucination condition |
| Name: | Potion of Detect Monster |
| Initial State: | PlayerChar object without the detect-monsters condition |
| Input: | Potion of detect monsters |
| Expected Output: | PlayerChar object with the detect-monsters condition |
| Name: | Potion of Detect Object |
| Initial State: | PlayerChar object without the detect-objects condition |
| Input: | Potion of detect objects |
| Expected Output: | PlayerChar object with the detect-objects condition |
| Name: | Potion of Confusion |
| Initial State: | PlayerChar object without the confusion condition |
| Input: | Potion of confusion |
| Expected Output: | PlayerChar object with the confusion condition |
| Name: | Potion of Confusion |
| Initial State: | PlayerChar object without the confusion condition |
| Input: | Potion of confusion |
| Expected Output: | PlayerChar object with the confusion condition |
| Name: | Potion of Levitation |
| Initial State: | PlayerChar object without the levitation condition |
| Input: | Potion of levitation |
| Expected Output: | PlayerChar object with the levitation condition |
| Name: | Potion of Haste |
| Initial State: | PlayerChar object without the haste condition |
| Input: | Potion of haste |
| Expected Output: | PlayerChar object with the haste condition |

| | |
|-------------------------|---|
| Name: | Potion of See-Invisible |
| Initial State: | PlayerChar object without the invisible-sight condition |
| Input: | Potion of invisible |
| Expected Output: | PlayerChar object with the invisible-sight condition |
| Name: | Random Range 1 |
| Initial State: | None |
| Input: | Upper and lower bounds 0,0 |
| Expected Output: | 0 |
| Name: | Random Range 2 |
| Initial State: | None |
| Input: | Upper and lower bounds 5,5 |
| Expected Output: | 5 |
| Name: | Random Range 3 |
| Initial State: | None |
| Input: | Upper and lower bounds 0,60, repeated 40 times |
| Expected Output: | $0 \leq \text{result} \leq 60$ |
| Name: | Random Float |
| Initial State: | None |
| Input: | 40 repeats |
| Expected Output: | $0 \leq \text{result} \leq 1$ |
| Name: | Random Boolean |
| Initial State: | None |
| Input: | 10 repeats |
| Expected Output: | Both true and false are generated |
| Name: | Random Percent |
| Initial State: | None |
| Input: | 40 repeats |
| Expected Output: | $0 \leq \text{result} \leq 100$ |
| Name: | Random Position |

| | |
|-------------------------|--|
| Initial State: | None |
| Input: | Two Coordinates (top-left and bottom-right of rectangle) |
| Expected Output: | Random Coordinate within bounds |
| Name: | Ring Construction 1 |
| Initial State: | None |
| Input: | Coordinate |
| Expected Output: | Ring object with valid initial state |
| Name: | Ring Construction 2 |
| Initial State: | None |
| Input: | Coordinate, Item context value, type identifier |
| Expected Output: | Ring object with valid initial state |
| Name: | Ring of Stealth |
| Initial State: | PlayerChar object without stealth condition |
| Input: | Ring of stealth |
| Expected Output: | PlayerChar object with the stealth condition |
| Name: | Ring of Stealth Deactivate |
| Initial State: | PlayerChar object with ring of stealth |
| Input: | Remove ring |
| Expected Output: | PlayerChar object without the stealth condition |
| Name: | Ring of Teleportation |
| Initial State: | PlayerChar object without random teleportation condition |
| Input: | Ring of teleportation |
| Expected Output: | PlayerChar object with the random teleportation condition |
| Name: | Ring of Teleportation Deactivate |
| Initial State: | PlayerChar object with ring of teleportation |
| Input: | Remove ring |
| Expected Output: | PlayerChar object without the random teleportation condition |

| | |
|-------------------------|--|
| Name: | Ring of Regeneration |
| Initial State: | PlayerChar object without regeneration condition |
| Input: | Ring of regeneration |
| Expected Output: | PlayerChar object with the regeneration condition |
| Name: | Ring of Regeneration Deactivate |
| Initial State: | PlayerChar object with ring of regeneration |
| Input: | Remove Ring |
| Expected Output: | PlayerChar object without the regeneration condition |
| Name: | Ring of Digestion |
| Initial State: | PlayerChar object without digestion condition |
| Input: | Ring of digestion |
| Expected Output: | PlayerChar object with the digestion condition |
| Name: | Ring of Digestion Deactivate |
| Initial State: | PlayerChar object with ring of digestion |
| Input: | Remove Ring |
| Expected Output: | PlayerChar object without the digestion condition |
| Name: | Ring of Dexterity |
| Initial State: | PlayerChar object |
| Input: | Ring of dexterity |
| Expected Output: | PlayerChar object with dexterity increased by the appropriate amount |
| Name: | Ring of Dexterity Deactivate |
| Initial State: | PlayerChar object with ring of dexterity |
| Input: | Remove ring |
| Expected Output: | PlayerChar object with normal dexterity |
| Name: | Ring of Adornment |
| Initial State: | PlayerChar object |
| Input: | Ring of adornment |
| Expected Output: | Identical PlayerChar object |

| | |
|-------------------------|--|
| Name: | Ring of Adornment |
| Initial State: | PlayerChar object with ring of adornment |
| Input: | Remove Ring |
| Expected Output: | Identical PlayerChar object |
| Name: | Ring of See-Invisible |
| Initial State: | PlayerChar object without the see-invisible condition |
| Input: | Ring of See-Invisible |
| Expected Output: | PlayerChar object with the see-invisible condition |
| Name: | Ring of See-Invisible Deactivate |
| Initial State: | PlayerChar object with ring of see-invisible |
| Input: | Remove Ring |
| Expected Output: | PlayerChar object without the see-invisible condition |
| Name: | Ring of Maintain-Armor |
| Initial State: | PlayerChar object without the maintain-armor condition |
| Input: | Ring of Maintain-Armor |
| Expected Output: | PlayerChar object with the maintain-armor condition |
| Name: | Ring of Maintain-Armor Deactivate |
| Initial State: | PlayerChar object with ring of maintain-armor |
| Input: | Remove Ring |
| Expected Output: | PlayerChar object without the maintain-armor condition |
| Name: | Ring of Searching |
| Initial State: | PlayerChar object without the auto-search condition |
| Input: | Ring of Searching |
| Expected Output: | PlayerChar object with the auto-search condition |
| Name: | Ring of Searching Deactivate |
| Initial State: | PlayerChar object with ring of searching |
| Input: | Remove Ring |

| | |
|---|--|
| Expected Output: | PlayerChar object without the auto-search condition |
| Name: Initial State: Input: Expected Output: | Room Construction Check 1 Randomly generated Room None Verification that Room's size is in valid range |
| Name: Initial State: Input: Expected Output: | Room Construction Check 2 Randomly generated Room None Verification that Room edges are within valid bounds |
| Name: Initial State: Input: Expected Output: | Scroll Construction 1 None Coordinate Scroll object in valid initial state |
| Name: Initial State: Input: Expected Output: | Scroll Construction 2 None Coordinate, Item context value, type identifier Scroll object in valid initial state |
| Name: Initial State: Input: Expected Output: | Scroll PseudoNames Scrolls are uninitialized Initialize Scroll Names command Vector of valid scroll psuedonames |
| Name: Initial State: Input: Expected Output: | Scroll of Protect Armor PlayerChar with cursed armor Scroll of Protect Armor PlayerChar with uncursed Armor with protect-armor effect |
| Name: Initial State: Input: | Scroll of Hold Monster Monster without the held flag Scroll of Hold Monster |

| | |
|---|---|
| Expected Output: | Monster with the held flag |
| Name: Initial State: Input: Expected Output: | Scroll of Enchant Weapon PlayerChar with weapon Scroll of Enchant Weapon PlayerChar with uncursed Weapon with higher enchant level |
| Name: Initial State: Input: Expected Output: | Scroll of Enchant Armor PlayerChar with armor Scroll of Enchant Armor PlayerChar with uncursed Armor with higher enchant level |
| Name: Initial State: Input: Expected Output: | Scroll of Identity None Scroll identity No exceptions |
| Name: Initial State: Input: Expected Output: | Scroll of Teleportation PlayerChar at coordinate (0,0) Scroll of Teleportation PlayerChar at coordinate \neq (0,0) |
| Name: Initial State: Input: Expected Output: | Scroll of Sleep PlayerChar without the sleep condition Scroll of Sleep PlayerChar with the sleep condition |
| Name: Initial State: Input: Expected Output: | Scroll of Scare Monster None Scroll of Scare Monster No exceptions |
| Name: Initial State: Input: | Scroll of Remove Curse PlayerChar with cursed Weapon Scroll of Remove Curse |

| | |
|---|---|
| Expected Output: | PlayerChar with uncursed Weapon |
| Name: Initial State: Input: Expected Output: | Scroll of Create Monster Level object Scroll of create Monster Level with 1 additional Monster |
| Name: Initial State: Input: Expected Output: | Scroll of Aggravate Monster Level with sleeping Monsters Scroll of Aggravate Monster Level with no sleeping Monsters |
| Name: Initial State: Input: Expected Output: | Scroll of Magic Mapping Unrevealed Level Scroll of Magic Mapping Level where all Tiles have been revealed |
| Name: Initial State: Input: Expected Output: | Scroll of Confuse Monster PlayerChar without the confuse-monster condition Scroll of Confuse Monster PlayerChar with the confuse-monster condition |
| Name: Initial State: Input: Expected Output: | Stairs Construction None Coordinate, direction value Stairs object in valid initial state |
| Name: Initial State: Input: Expected Output: | Stairs Direction Check Stairs constructed with direction Direction value Verification that Stairs has given direction value |
| Name: Initial State: Input: Expected Output: | Floor Passability Check Floor object None Verification that Floor is passable |

| | |
|-------------------------|---|
| Name: | Floor Symbol Check |
| Initial State: | Floor object |
| Input: | None |
| Expected Output: | Verification that Floor has correct symbol |
| Name: | Floor Transparency Check |
| Initial State: | Floor object |
| Input: | None |
| Expected Output: | Verification that Floor is transparent |
| Name: | Wall Passability Check |
| Initial State: | Wall object |
| Input: | None |
| Expected Output: | Verification that Wall is not passable |
| Name: | Wall Symbol Check |
| Initial State: | Wall object |
| Input: | None |
| Expected Output: | Verification that Wall has correct symbol |
| Name: | Wall Opacity Check |
| Initial State: | Wall object |
| Input: | None |
| Expected Output: | Verification that Wall is transparent |
| Name: | Corridor Passability Check |
| Initial State: | Corridor object |
| Input: | None |
| Expected Output: | Verification that Corridor is passable |
| Name: | Corridor Symbol Check |
| Initial State: | Corridor object |
| Input: | None |
| Expected Output: | Verification that Corridor has correct symbol |
| Name: | Corridor Transparency Check |

| | |
|-------------------------|--|
| Initial State: | Corridor object |
| Input: | None |
| Expected Output: | Verification that Corridor has special corridor transparency |
| Name: | Door Passability Check |
| Initial State: | Door object |
| Input: | None |
| Expected Output: | Verification that Door is not passable |
| Name: | Door Symbol Check |
| Initial State: | Door object |
| Input: | None |
| Expected Output: | Verification that Door has correct symbol |
| Name: | Door Transparency Check |
| Initial State: | Door object |
| Input: | None |
| Expected Output: | Verification that Door has special Corridor transparency |
| Name: | Door Trap |
| Initial State: | PlayerChar and Level |
| Input: | Door trap |
| Expected Output: | PlayerChar at a Level with depth + 1 |
| Name: | Rust Trap |
| Initial State: | PlayerChar with enchanted Weapon |
| Input: | Rust trap |
| Expected Output: | PlayerChar with unenchanted Weapon |
| Name: | Sleep Trap |
| Initial State: | PlayerChar without the sleep condition |
| Input: | Sleep trap |
| Expected Output: | PlayerChar with the sleep condition |
| Name: | Bear Trap |
| Initial State: | PlayerChar without the immobilized condition |

| | |
|-------------------------|--|
| Input: | Bear trap |
| Expected Output: | PlayerChar with the immobilized condition |
| Name: | Teleport Trap |
| Initial State: | PlayerChar |
| Input: | Teleport trap |
| Expected Output: | PlayerChar at a different location |
| Name: | Dart Trap |
| Initial State: | PlayerChar |
| Input: | Dart trap |
| Expected Output: | PlayerChar with less HP |
| Name: | Tunnel Digging |
| Initial State: | Level and pair of unconnected Rooms |
| Input: | Dig command |
| Expected Output: | Valid path between the two Rooms |
| Name: | Open Inventory Screen |
| Initial State: | Playstate, PlayerChar, empty Level |
| Input: | Inventory key |
| Expected Output: | Inventory screen |
| Name: | Close Inventory Screen |
| Initial State: | Inventory screen, PlayerChar, empty Level |
| Input: | Exit key |
| Expected Output: | PlayState |
| Name: | Movement |
| Initial State: | PlayState, player, empty Level |
| Input: | Movement key |
| Expected Output: | PlayerChar should be in expected location in the Level |
| Name: | Open Status Screen |
| Initial State: | PlayState, PlayerChar, empty Level |
| Input: | Status key |

| | |
|---|--|
| Expected Output: | Status screen |
| Name: Initial State: Input: Expected Output: | Exit Status Screen Status Screen, PlayerChar, empty Level Exit key PlayState |
| Name: Initial State: Input: Expected Output: | No Wand Zap PlayState, PlayerChar with no Wand Zap key Unchanged playState |
| Name: Initial State: Input: Expected Output: | Zap Wand Select PlayState, PlayerChar with wand, empty Level Zap key, then direction key Inventory screen |
| Name: Initial State: Input: Expected Output: | Zap Wand Fire 1 Inventory Wand select Item select hotkey PlayState |
| Name: Initial State: Input: Expected Output: | Zap Wand Fire 2 Inventory wand select Item select hotkey Wand with charges - 1 |
| Name: Initial State: Input: Expected Output: | Game Quit PlayState Quit key and confirmation key RipScreen |
| Name: Initial State: Input: Expected Output: | Wand Construction 1 None Coordinate Wand in valid initial state |

| | |
|-------------------------|---|
| Name: | Wand Construction 2 |
| Initial State: | None |
| Input: | Coordinate, Item context value, type specifier |
| Expected Output: | Wand in valid initial state |
| Name: | Wand of Teleport Away |
| Initial State: | PlayerChar, nearby Monster |
| Input: | Wand of Teleport Away |
| Expected Output: | Monster has distance to PlayerChar ≥ 20 |
| Name: | Wand of Slow Monster |
| Initial State: | PlayerChar, Monster without slowed flag |
| Input: | Wand of Slow Monster |
| Expected Output: | Monster has slowed flag |
| Name: | Wand of Invisibility |
| Initial State: | PlayerChar, Monster without invisible flag |
| Input: | Wand of Invisibility |
| Expected Output: | Monster with invisible flag |
| Name: | Wand of Polymorph |
| Initial State: | PlayerChar, Monster |
| Input: | Wand of Polymorph |
| Expected Output: | Different Monster at previous Monster's locations |
| Name: | Wand of Haste Monster |
| Initial State: | PlayerChar, Monster without haste flag |
| Input: | Wand of Haste Monster |
| Expected Output: | Monster with haste flag |
| Name: | Wand of Magic Missile |
| Initial State: | PlayerChar, Monster |
| Input: | Wand of Magic Missile |
| Expected Output: | Monster with reduced HP |
| Name: | Wand of Cancellation |

| | |
|-------------------------|---|
| Initial State: | PlayerChar, Monster without cancelled flag |
| Input: | Wand of Cancellation |
| Expected Output: | Monster with canceled flag |
| Name: | Wand of Do Nothing |
| Initial State: | PlayerChar, Monster |
| Input: | Wand of Do Nothing |
| Expected Output: | No exceptions |
| Name: | Wand of Drain Life |
| Initial State: | PlayerChar with reduced HP, Monster |
| Input: | Wand of Drain Life |
| Expected Output: | PlayerChar with increased HP, Monster with reduced HP |
| Name: | Wand of Cold |
| Initial State: | PlayerChar, Monster |
| Input: | Wand of Cold |
| Expected Output: | No exceptions |
| Name: | Wand of Fire |
| Initial State: | PlayerChar, Monster |
| Input: | Wand of Fire |
| Expected Output: | No exceptions |
| Name: | Weapon Construction 1 |
| Initial State: | None |
| Input: | Coordinate |
| Expected Output: | Weapon in valid initial state |
| Name: | Weapon Construction 2 |
| Initial State: | None |
| Input: | Coordinate, Item context value, type specifier |
| Expected Output: | Weapon in valid initial state |
| Name: | Weapon Identification Check |
| Initial State: | Identified Weapon |

| | |
|-------------------------|--|
| Input: | None |
| Expected Output: | Verification that Weapon is identified |
| Name: | Weapon Curse Check |
| Initial State: | Cursed Weapon |
| Input: | None |
| Expected Output: | Verification that Weapon is cursed |
| Name: | Weapon Name Check |
| Initial State: | Weapon |
| Input: | None |
| Expected Output: | Verification that Weapon has valid name |
| Name: | Weapon Enchantment Check |
| Initial State: | Cursed Weapon |
| Input: | None |
| Expected Output: | Verification that Weapon has expected enchantment values |

8 Trace to Requirements

The following table maps each implemented test file to its corresponding 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 | Runs Tests |
| 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 Structure |
| test.testable.h | Defines Test Structure |
| 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 table below re-iterates the modules of the project (for the convenience of the reader), along with their respective domain and module ID; the module IDs are used to refer to modules in the trace. More information about the modules can be found in the [Module Guide](#).

Table 4: **Module Hierarchy**

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

The following table maps test files, which implement unit tests, to their specific modules (denoted by their IDs).

Table 5: **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 (Runs Tests) |
| test.mob.cpp | M9, M10, M11, M13, M15, M16 |
| test.monster.cpp | M9, M10, M16 |
| test.playerchar.cpp | M5, M6, M9, M11, M12, M13, \leftarrow 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 Structure |
| test.testable.h | Defines Test Structure |
| 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

Given the [Test-Requirements](#) table, the [Test-Module](#) table, and the Module-Requirements trace given in the [Module Guide](#), the referential transitivity property can be applied to determine exactly which functional and non-functional requirements were satisfied by the created test cases.

With the exception of a few non-functional requirements, the implemented tests offered almost **complete coverage** of all product functional and non-functional requirements. As the Rogue++ team discovered, several of the non-functional requirements were impractical to test in an automated fashion because computing the prevalence of certain subjective qualities in a product is extremely complicated (or at least well beyond the scope of Rogue Reborn). For example, NFR.2 states, "The Rogue Reborn game shall be fun and entertaining." While it is infeasible to test such characteristics with code, it can be trivially accomplished with human playtesters. Along with NFR.2, various other non-functional requirements that were too difficult to automatically assert in software were verified by other means, namely manual human labor.

Interestingly, the Rogue++ team also encountered a few requirements where the achievable target was difficult to materialize but still algorithmic and computational in nature. A prime example includes the luminosity constraint which ruled that no two consecutive frames may have a change in contrast greater than [LUMINOSITY_DELTA](#). To measure this properly, the developer team wrote a Python script to analyze two static screenshots of the game. This program calculated the pixel-accurate luminosity of each screenshot and used the calculation proposed in the non-functional requirement to arrive at a final answer.

Static code coverage metrics were not utilized in Rogue Reborn due to the sheer volume of the code base and the state complexity of various modules.

References

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