SE 3XA3: Test Plan
Rogue Reborn

Team #6, Team Rogue++

Ian Prins	prinsij
Mikhail Andrenkov	andrem5
Or Almog	almogo

Due Monday, October 31th, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
10/21/16	0.1	Initial Setup
10/24/16	0.2	Add Unit Testing and Usability Survey

This document ...

1 General Information

1.1 Purpose

1.2 Scope

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
Abbreviation1	Definition1
Abbreviation2	Definition2

Table 3: **Table of Definitions**

Term	**Definition**
Term1	Definition1
Term2	Definition2

1.4 Overview of Document

2 Plan

2.1 Software Description

2.2 Test Team

2.3 Automated Testing Approach

2.4 Testing Tools

2.5 Testing Schedule

See Gantt Chart at the following url ...

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

Output:

How test will be performed:

3.1.2 Area of Testing2

...

3.2 Tests for Nonfunctional Requirements

3.2.1 Area of Testing1

Title for Test

1. test-id1

   Type:

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

3.2.2 Area of Testing2

...

4 Tests for Proof of Concept

4.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

4.2 Area of Testing2

...

5 Comparison to Existing Implementation

6 Unit Testing Plan

After examining the boost library's utilities for unit testing, we have decided we will not use a unit testing framework for testing the product. We concluded that adding a framework would not make the work significantly easier, while reducing our flexibility and adding installation difficulties.

6.1 Unit testing of internal functions

Internal functions in the product will be unit tested. This will be reserved for more complex functions so as to not waste development time unnecessarily. The following are examples of internal functions that are good candidates for unit testing:

- The dungeon generation functions. The work of generating the dungeon is complex, but it is also easy to automate verification of dungeon properties such as a correct number of rooms, connectness, compliance with formulas for item generation, presence or absence of certain key features such as the stairs connecting levels or the Amulet of Yendor in the final level.

- The keyboard input functions. As libtcod provides a Key struct which models keyboard input, we can mock/automate these functions. They are fairly complex, and since they return a pointer to the next desired state (similar to a finite state machine) we can easily verify their behavior.

- Some of the item activation functions. For example it could be verified that when the player drank a potion of healing their health increased (if it was not at its maximum), when a scroll of magic-mapping is read the level was revealed, or that a scroll of identification reveals the nature of an item.

6.2 Unit testing of output files

There is only one output file for the product, the high score file, which stores the scores in a csv format. The production and reading of this file can be unit-tested by verifying its contents after writing to it, and by providing a testing version of the file with known contents and verifying the function reads them correctly.

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

Table 4: **Symbolic Parameter Table**

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

7.2 Usability Survey Questions?

- Is there any game feature you were unable to figure out how to utilize?
- How helpful was the help screen for you?
- Was there anything going on in the game that the interface failed to make clear to you or deceived you about?
- What common UI interactions did you find particularly lengthy?
- What aspects of the interface did you find unintuitive?

- How responsive was the interface?
- How easy was it to see everything that was going on?
- How effective are the graphics/symbols?
- Would an alternative input device such as a mouse make interacting with the interface easier for you?