SE 3XA3: Test Plan
Rogue Reborn

Group #6, Team Rogue++

Ian Prins	prinsij
Mikhail Andrenkov	andrem5
Or Almog	almogo

Due Wednesday, Dec 7st, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
Dec 6	0.1	Initial draft

This document...

1 Functional Requirements Evaluation

Ori

2 Nonfunctional Requirements Evaluation

Mikhail

2.1 Usability

Mikhail

2.2 Performance

Mikhail

2.3 etc.

Mikhail

3 Comparison to Existing Implementation

Ori

4 Unit Testing

Mikhail

5 Changes Due to Testing

Mikhail

6 Automated Testing

6.1 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.

6.2 Specific System Tests

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

Name:	Amulet Construction
Initial State:	None
Input:	Coordinate, context value
Expected Output:	Amulet object in valid initial state
Name:	Armor Construction 1
Initial State:	None
Input:	Coordinate
Expected Output:	Armor object in valid initial state
Name:	Armor Construction 2
Initial State:	None
Input:	Coordinate, context value, type value
Expected Output:	Armor object in valid initial state
Name:	Armor Identification
Initial State:	Cursed Armor
Input:	None
Expected Output:	Verification that armor is identified
Name:	Armor Identification
Initial State:	Cursed Armor
Input:	None
Expected Output:	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:	Feature Construction
Initial State:	None
Input:	Symbol, coordinate, visibility, color
Expected Output:	Feature object in valid initial state
Name:	Feature Symbol Check

Initial State:	Feature with given symbol
Input:	Symbol
Expected Output:	Verification that feature's symbol matches given
Name:	Feature Invisibility Check
Initial State:	Invisible feature
Input:	None
Expected Output:	Verification that feature is invisible
Name:	Feature Visibility Check
Initial State:	Visible feature
Input:	None
Expected Output:	Verification that feature is visible
Name:	Feature Location Check
Initial State:	Feature with given location
Input:	Coordinate
Expected Output:	Verification that feature's location matches given coordinate
Name:	Food Construction
Initial State:	None
Input:	Coordinate and context value
Expected Output:	Food object in valid initial state
Name:	Food Eating
Initial State:	Food and player objects
Input:	None
Expected Output:	Verification that food has increased the player's food life by an appropria
Name:	GoldPile Construction
Initial State:	None
Input:	Coordinate, gold amount value
Expected Output:	GoldPile object in valid initial state
Name:	GoldPile Quantity Check
Initial State:	GoldPile with given amount of gold
Input:	Amount of gold value
Expected Output:	Verification that gold's amount matches given amount
Name:	Item Construction 1
Initial State:	None
Input:	Symbol, coordinate, context value, item class specifier, name value, psued
Expected Output:	Item object in valid initial state
Name:	Item Construction 2
Initial State:	None

Input:	Symbol, coordinate, context value, item class specifier, name value, psued
Expected Output:	Item object in valid initial state
Name:	Name Vector Check
Initial State:	None
Input:	Vector of item names
Expected Output:	Shuffled vector of item names
Name:	Item Curse Check
Initial State:	Uncursed item
Input:	None
Expected Output:	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 Unindentified 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:	ItemZone Containment Check 2
Initial State:	ItemZone with 2 items
Input:	None
Expected Output:	Verification that ItemZone contains the second item
Name:	ItemZone Empty Check
Initial State:	ItemZone with 2 items
Input:	None
Expected Output:	Verification that ItemZone is not empty
Name:	ItemZone Size Check
Initial State:	ItemZone with 2 items
Input:	None
Expected Output:	Verification that ItemZone's size is 2
Name:	ItemZone Keybind Check 1
Initial State:	ItemZone with 2 items
Input:	None
Expected Output:	Verification that first item is bound to 'a' key
Name:	ItemZone Keybind Check 2
Initial State:	ItemZone with 2 items
Input:	None
Expected Output:	Verification that second item is bound to 'b' key
Name:	ItemZone Contents Retrieval 1
Initial State:	ItemZone with 2 items
Input:	None
Expected Output:	Item map with exactly 1 copy of first item
Name:	ItemZone Contents Retrieval 2
Initial State:	ItemZone with 2 items
Input:	None
Expected Output:	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:	ItemZone with 1 item
Input:	Attempt to add 500 pieces of armor to ItemZone
Expected Output:	ItemZone with max-weight worth of armor
Name:	Level Construction
Initial State:	None
Input:	Depth, player 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 moveme
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 moveme
Name:	Level BFSPerp Vertical
Initial State:	Empty level object
Input:	Pair of coordinates with equal x-values
Expected Output:	Path between coordinates with expected length, utilizing taxicab moveme
Name:	Level BFSDiag Horizontal
Initial State:	Empty level object
Input:	Pair of coordinates with equal y-values
Expected Output:	Path between coordinates with expected length, utilizing orthogonal mov
Name:	Level BFSDiag Vertical
Initial State:	Empty level object
Input:	Pair of coordinates with equal x-values
Expected Output:	Path between coordinates with expected length, utilizing orthogonal mov
Name:	Level BFSPerp Diagonal
Initial State:	Empty level object
Input:	Pair of coordinates on diagonal line
Expected Output:	Path between coordinates with expected length, utilizing taxicab moveme
Name:	Level Starting Position

Initial State:	Empty level object
Input:	None
Expected Output:	Valid starting position coordinate
Name:	Level getAdjPassable
Initial State:	Empty level object
Input:	Coordinate
Expected Output:	List of coordinates orthogonally adjacent to given coordinate
Name:	
Initial State:	
Input:	
Expected Output:	
Name:	
Initial State:	
Input:	
Expected Output:	

7 Trace to Requirements

Ori

8 Trace to Modules

Ori

9 Code Coverage Metrics

Ori