Chapter 1

TEST PLAN (TP) for Game Rules and Play

Version 1.1, April 2014

TEST PLAN (TP)

FOR

Swords and Sorcery

Version 1
April 30, 2014

Prepared for:
Clinton Jeffery

Prepared by:
Tao Zhang, Cameron Simon, Matthrew Brown,
Tyler Jaszkowiak, Ian Westrope, ChiHsiang Wang
(Rule Team)
University of Idaho
Moscow, ID 83844-1010
CS383 TPD

RECORD OF CHANGES (Change History)

Change Number	Date completed	Location of change (e.g., page or figure #)	A M D	Brief description of change	Approved by (initials)	Date approved
1	4/30/2014	Document	A	Original Document Created	MB	4/30/2014

A - ADDED M - MODIFIED D – DELETED

TABLE OF CONTENTS

Section Page

1 TEST PLAN IDENTIFIER

This test plan has no unique identifier other than Spring 2014 CS 383 sworsorc. This particular test plan is in conjunction to the rules and game play group test plans. There are many test plans for this group due to the size and span of the group's contents. This test plan, for example, will be known as the Units test plan.

2 REFERENCES

Use Cases and State diagrams are available at https://github.com/cjeffery/sworsorc/tree/master/doc

3 INTRODUCTION

The purpose of this test plan is to state the processes used by Game Rules and Play Team in testing the Spells and Characters for the Swords & Sorcery project.(Tao, Cameron)

This test plan covers the creation and implementation of the Moveableunit class and its sub class Armyunits. (Matt)

This test plan is also to provide as much coverage as possible to the methods and data of the Scenario class by verifying the results of execution against expectations through a series of automated unit tests and a manual GUI test. (Tyler)

4 TEST ITEMS

The army units will be tested for the proper member variable values as well as proper results from the member function of the Moveableunit and Armyunit classes. Variables will tested at creation and members variable will be tested for proper results when applicable. Some example tests will be that the location member variable is properly set during movement. Also another type of test that will be preformed is that conjured units are properly created and and placed in the proper place on the game board.

4.1 Character

- Class Interfaces
- Class Interactions
- Spells Implementation
- Character

4.2 Scenario

These are things you intend to test within the scope of this test plan. Essentially, something you will test, a list of what is to be tested. This can be developed from the software application inventories as well as other sources of documentation and information.

The data read by the Scenario test will be read from one of the simple scenario configuration files and then verified against expectations through the class's accessor methods. These data items include

- The scenario's name
- Number of players
- Game length
- The blue sun's initial position

- The names of armies in the scenario
- The controlling players of these armies
- The setup order of these armies
- The movement order of these armies
- Names of nations within these armies
- Names of the neutrals in the scenario
- The provinces controlled by a nation
- The characters within a nation
- The units within a player nation
- The units within a neutral nation
- The races of both neutrals and player nations
- The reinforcement and replacement description strings
- Data related to where a neutral is leaning toward
- Whether or not a neutral accepts human sacrifice

Unfortunately, the most complex functionality of the Scenario class cannot be tested by automated unit tests. The unit pool populator requires a manual check.

5 SOFTWARE RISK ISSUES

- Improper casting of units
- Unit ID's interpreted incorrectly
- Depends on Java's JSON reader and the programmer's understanding of it
- Complex data structures such as a map of maps
- Complex loops to iterate over data structures
- Poor documentation surrounding some of these iterators

6 FEATURES TO BE TESTED

6.1 Scenario

From the user's perspective, much of this data reading occurs under the hood. In all cases except for populating the unit pool, the Scenario class does not manipulate any pieces of the rest of the project. Other components read the data from the Scenario class. Therefore, while the solar configuration relies on a properly-working Scenario class, this is not apparent to the user because solar configuration is also handled by SolarConfig and HUDInitializer classes.

Therefore, the only visible component being tested by this plan is the placement of units and characters into the correct provinces of the map.

6.2 Movement

- conjured unit appear when casted
- units are properly represented on HUD
- movement location is correct on map
- moral status is properly updated after combat

6.3 Character, Spells

- Select character on GUI
- Select Spell
- Effects of Partial Spells
- Manna Costing

7 FEATURES NOT TO BE TESTED

This is a listing of what is NOT to be tested from both the Users viewpoint of what the system does and a configuration management/version control view. This is not a technical description of the software, but a USERS view of the functions.

All data is verified, but testing the Scenario class alone cannot ensure that it reaches the HUD successfully. Integration tests are required outside of this plan for information such as solar configuration, move order, setup order, game length, and diplomacy.

What is not tested is the count of the units on the map that correspond to the scenario loaded. Also the type of unit is not test, this is assumed correct.

8 APPROACH

We will use JUnit tests as well as manual tests to determine the effectiveness and accuracy of the methods and variables tested.

9 ITEM PASS/FAIL CRITERIA

Items will receive a PASS once the variables/methods have been test multiple times and under different circumstances and the results of the test were within expected parameters. A FAIL will be received if the test results are not within expected parameters.

9.1 Unit Testing Pass/Fail Criteria

9.2 Intergration Testing Pass/Fail Criteria

9.2.1 Scenario

The automated tests must all pass and result in 100each of the class's accessor methods. One of the nations in the scenario must have the correct number and types of units loaded into the provinces it controls.

9.2.2 Character

Rule Description	Test Description	Expected Result
Create new character object with its information.	createCharacter function is called with specific character name in CharacterMaker.java.	A new character object is created and returned that contains all of the specified characters information.
Potential spells for selected character displayed.	Select character in GUI, then click cast spell button on sidebar panel.	List of spells that can be cast by selected character are displayed.

9.2.3 Spells

Rule Description	Test Description	Expected Result
Character with Power Level should have a spell book.	Generate a spell book for the character.	A frame with a list of spells should be shown on the screen.
Show spell description.	Click on the Spell button.	A frame will be displayed with all information about the spell.
A target need to be selected for most of the spells.	Click on cast button on the frame of displaying information about the spell. Then select a target by right click the target on the map.	A target is selected to cast the spell.

10 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

Tests will be suspended if a bigger piece of code that directly or indirectly effects the code is compromised. Tests will also be suspended once a FAIL has be received for that test. The tests will be resumed once the effected code fixed.

11 TEST DELIVERABLES

- Test plan document
- Test cases
- Relevant error logs or problem reports
- JUnit test cases
- A report of passes/failures from JUnit
- A coverage report from JaCoCo

12 REMAINING TEST TASKS

This is a multi-group project in which many test plans are created for the respective group. Also since the full project as not been completed there will be more tests developed in the future.

All other classes in the project require some level of testing as well. In particular, the HUD and SolarConfig related classes directly affect the performance of Scenario.

13 ENVIRONMENTAL NEEDS

There are no special requirements for this test plan.

14 STAFFING AND TRAINING NEEDS

Training in JUnit test is required to understand sections of the code.

Understading of NetBeans, Java, JUnit, and JaCoCo.

15 RESPONSIBILITIES

This is a co-operative, no one person is in charge. Tests are divided by group type developers.

16 SCHEDULE

Realistically most tests will not be completed due to the rapid approach of the end of the semester.

17 PLANNING RISKS AND CONTINGENCIES

Too many tests will cause a halt in development and since the project wont be completed by the end of the semester might as well get as much working functionality as possible.

18 APPROVALS

Dr. J is the evil genius behind the project and is the only one capable of approving this process. (Matt)

Dr. Jeffery has the final word. However, I imagine that the team generating the test documentation will decide which tests can be approved and whether to continue.(Tyler)

19 GLOSSARY

JaCoCo: The Java Code Coverage plugin for NetBeans

20 APPENDIX A. [insert name here]

Include copies of test examples, etc. supplied or derived from the customer. Appendices are labeled A, B, . . . n. Reference each appendix as appropriate in the text of the document.

[insert appendix A here]

21 APPENDIX B. [insert name here]

[insert appendix B here]