

Team Ratatosk

Genesis

Test Plan

November 27, 2005

History of Changes

Version	Date	Change
1.0	9-28-05	First Version
1.1	11-27-05	Revised version for resubmission

Table of Contents

History of Changes	1
Table of Contents	2
1. Introduction	3
1.1 Definitions and Abbreviations	3
1.2 References.....	3
1.3 Test Team.....	3
1.4 Items Not Covered by These Test Cases	3
1.5 Bug Tracking	4
1.6 Quality Control	4
1.7 Adequacy Criterion.....	4
2 Test Strategy	6
2.1 Testing Process	6
2.2 Technology	6
3 Test Cases	7
3.1 Functional Testing	7
3.1.1 Testing Form.....	7
3.2 Non-functional Testing.....	18
3.2.1 Test Descriptions	18
3.2.1.1 Reliability.....	18
3.2.1.2 Efficiency.....	18
3.2.1.3 Availability	18
3.2.1.4 Maintainability.....	18
3.2.2 Testing Form.....	19
4 Appendix	20
4.1 XML Doctype Definitions	20
4.1.1 Model Library DTD	20
4.1.2 Project DTD	20
4.1.3 Texture Library DTD.....	21
4.1.4 Map DTD	21

1. Introduction

Genesis is a project that will allow users of the program to build their own 3D world. Users will be able to modify their environment by raising or lowering areas of terrain, and by placing 3D models, such as trees or bushes, in the 3D world.

We will be using mainly a blackbox system of testing our components. We will test each component by sending the component inputs and analyzing the outputs without analyzing the code used to create the functionality. Each of our components we are testing correspond directly to a requirement in our requirements document (SRS – see section 1.2).

1.1 Definitions and Abbreviations

This is our table of definitions:

Term	Definition
Test #	Test Case Number / Identifier
Requirement	Requirement that the test cases are validating (number / identifier)
Action	Action to perform or input to produce
Expected Result	Result expected when action is complete
Actual Result	What was actually seen
P / F	Pass / Fail indicator. "P" = Pass. "F" = Fail
Notes	Additional notes, error messages, or other information about the test.
DTD	Doctype Definition
XML	Extensible Markup Language

1.2 References

- SRS – Software Requirements Document, version 1.1 (<http://swiki.cc.gatech.edu:8080/cs3300-fl05/uploads/9/1/5/BG%5D%20Requirements%20Final.pdf>)
- XML DTD's (Appendix 4)
- XML Schema Definition, version 1.0 (<http://swiki.cc.gatech.edu:8080/cs3300-fl05/107>)

1.3 Test Team

Troy Brant
Richard Bailey
Chad Hansen
Jonathan Clark

1.4 Items Not Covered by These Test Cases

All the functionality of our system is covered by these test cases.

1.5 Bug Tracking

We will track bugs using the class swiki. Our bugs will be recorded using the following format:

Bug #	Status	Description	Author	Assignee	Resolution

This is a description of what will be recorded in each column:

Bug #	A unique id for each bug
Status	I = initial (no attempt to resolve bug), W = working on it (attempting to locate and eradicate the bug), R = resolved (the bug has been fixed)
Description	A short description of the bug and its attributes
Author	The person who found the bug and submitted the bug report
Assignee	Who is responsible for fixing the bug
Resolution	A short description of the steps taken to fix the bug

1.6 Quality Control

We will maintain the quality of the testing process by everyone sharing the workload of developing and performing the test cases. This will ensure more thorough testing because each person on the team will be performing tests on code that they did not write, and they won't have any inhibitions of pointing out shortcomings of the code. Someone who didn't write the code is more likely to give quality feedback after testing because they don't have any attachment to the code and don't see it as infallible. This will help ensure that we have the highest quality of testing possible.

We will know there will be enough test cases when the test cases cover all of the requirements and any errors those requirements may produce.

The test cases will be of acceptable quality if they test small pieces of functionality of each requirement. A single test case should not cover several pieces of functionality if the requirement can be broken up into smaller testable chunks.

1.7 Adequacy Criterion

Testing will be completed when 100% of our test cases pass. However, this is after the system has been completed, and introducing any new functionality into the system will require that we resume the testing process. If we add more functionality, we will need to add new test cases for that functionality and we will need to perform regression testing to make sure none of the other functionality breaks by adding the new code.

All the test cases are of equal importance. If one of the test cases fail, the product will be considered incomplete.

For the non-functional requirements, we will have 5 people use the system, and then we will give them a series of tasks to complete. We will monitor them and then ask them several questions after they complete their tasks. If 4 out of the 5 people provide acceptable feedback for the questions that directly affect our non-functional requirements, then the application will be good enough.

2 Test Strategy

2.1 Testing Process

The majority of our testing will be done through Build Verification Test. A Build Verification Test is a well-defined set of actions to be performed to make sure that all of our functionality works. The QA Lead will sit down with the tests and perform each one as specified to ensure that the test works. The tests should be performed in order. The order of test case performance is unimportant as we have attempted to keep interdependence between them from arising.

The Build Verification Test will cover all functionality in Genesis as it is relatively straightforward. Most input is taken through buttons and various other components with a limited range (i.e. a slider or dropdown box). This means that there are not many ways to allow for invalid input from the user. However, anywhere that invalid input is a possibility, a test will be created to be sure that it is handled.

We will use two types of testing: informal and formal testing. Informal testing is testing each piece of functionality we add to the system, and it will be performed throughout the development process. Formal testing is testing that includes filling out test reports, and this will take place during the Integration and Testing Phase of our development process. The Integration and Testing Phase is explained in the Project Plan document.

Testing of our non-functional requirements can be performed at a different time than the testing of the functional requirements. The order of the testing of functional and non-functional requirements is not important.

In order to fully test the system, the following pre-loaded data is required:

- An image between the sizes of 10x10 pixels and 100x100 pixels. All image formats are acceptable.
- A 3ds model. Free 3ds models can be found at the following website:
<http://www.3dnuts.com/models.shtml>

2.2 Technology

Most of the system will not be using unit tests. This is due to the fact that most of our system is GUI-based and requires a person for visual inspection to make sure the application is running correctly.

There will be classes in the RenderingEngine that provide backend functionality and may use unit test functionality if sufficiently isolated from the GUI portion of the system. In the event that we need to write unit tests we will write our own testing framework. We are not utilizing an existing testing framework because the number of tests that we expect will be needed is not a great number and the overhead of learning a framework and incorporating this into our code will be unneeded complexity.

3 Test Cases

3.1 Functional Testing

3.1.1 Testing Form

Date test performed: _____

Tester: _____

Test #	Requirement or Purpose	Action / Input	Expected Result	Actual Result	P/F	Notes
1a	SRS 3.2.1.1.1	A new map will be created at the startup of the application when creating a new project. Input will be size, name, and default texture.	The main gui will open with a newly created flat map of the correct size where each tile has the default texture. The map will be displayed in the viewport.			
1b	SRS 3.2.1.1.1	A new map will be created after the main gui has already been loaded. Again all of the correct information will be entered.	The main gui will clear the currently open map and replace the map area with the new map.			
2a	SRS 3.2.1.1.2	A request will be made to save the currently open map. The request will contain the save path and the map information. The map file will then be	A set of files will be created containing all of the information about the map. The file should have XML that contains information about the terrain and			

Test #	Requirement or Purpose	Action / Input	Expected Result	Actual Result	P/F	Notes
		verified to make sure it's syntactically correct.	information about the 3D models.			
2b	SRS 3.2.1.1.2	A request will be made to save the map to a file, and an invalid file name, as the operating system reports (for instance, "(fo&.ya^") will be entered.	An error will be displayed that says "Invalid file name. Please choose another name." The user will only be given the choice of hitting OK. Also a new path will be requested.			
3a	SRS 3.2.1.1.3	A request will be made to load a given map. The input will be a path to the desired map.	The main gui will load and display the selected map.			
3b	SRS 3.2.1.1.3	A request will be made to load a map, but a path to a file that is not a valid map file will be specified (such as "leda.pdf").	An error will be displayed that says "Invalid file name. Please choose another name." The user will only be given the choice of hitting OK. Also a new path will be requested.			
4a	SRS 3.2.1.1.4	A path to a 3d model will be specified.	The selected model will be imported and placed in the selected			

Test #	Requirement or Purpose	Action / Input	Expected Result	Actual Result	P/F	Notes
			library.			
4b	SRS 3.2.1.1.4	An attempt will be made to import an invalid file as a 3d object.	An error message will be shown saying "Unrecognized file format. Cannot load <filename>. Please select another file." Another request will be made for a valid file.			
5a	SRS 3.2.1.1.5	A library will be selected, and a save path will be taken as input.	A file will be created that conforms to the XML DTD specified by 4.1.1 if the library is of type Model and 4.1.3 if the library is of type Texture.			
5b	SRS 3.2.1.1.5	A created library will be selected, and a save attempt will be made. The save path will contain invalid characters.	An error will be displayed that says "Invalid file name. Please choose another name." The user will only be given the choice of hitting OK. Also a new path will be requested.			
6a	SRS 3.2.1.1.6	A model library will be	The library will be loaded			

Genesis Test Cases

Test #	Requirement or Purpose	Action / Input	Expected Result	Actual Result	P/F	Notes
		imported into the program. A path will be taken as input.	into the application, and all of its elements will be displayed in the bottom panel.			
6b	SRS 3.2.1.1.6	A library import attempt will be made but a non-library file will be specified.	An error message will be shown that says "Unrecognized file format. Cannot load <filename>. Please select another file." Another request will be made for a valid file.			
7	SRS 3.2.1.7	The user will open a project, make a change, and then choose to save that project thus updating the appropriate resource archive.	The GAR file on disk at the project-specified path has its contents updated to reflect the changes made to the project. This file should conform to the project XML DTD specified in 4.1.2			
8	SRS 3.2.1.8	An imported 3d model will be placed onto the map. The model will then be moved,	The 3d model will appear on the map. This model will be moved, rotated, and			

Test #	Requirement or Purpose	Action / Input	Expected Result	Actual Result	P/F	Notes
		rotated, and scaled. Finally, the model will be removed from the map.	scaled as requested. It will then be completely removed.			
9	SRS 3.2.1.9	The user will load an existing project. Now, from the default mode we will toggle models, terrain, and models and terrain.	The project will load displaying both models and terrain. First we toggle models which will display only terrain. Next, when terrain is toggled only models will be shown. Finally, with models and terrain toggled, nothing will be shown in the viewport.			
10a	SRS 3.2.1.10	The user will load a project and then use the mouse middle-click to rotate the contents of the viewport. This is accomplished by middle-clicking anywhere in the viewport and, without releasing, dragging to another portion	As the user drags with the middle mouse button the camera view angle will alter itself in the following manner: When dragging up/down the scene should appear to rotate into / out of. When			

Test #	Requirement or Purpose	Action / Input	Expected Result	Actual Result	P/F	Notes
		of the viewport.	dragging left/right the scene should appear to rotate counterclockwise / clockwise.			
11	SRS 3.2.1.11	A texture will be selected and dragged over the map. This will include going over hills and through valleys to check that it wraps correctly. The input will be a texture and an area for it to be placed.	The selected texture will replace the area of texture that is currently on the map under the mouse cursor.			
12	SRS 3.2.1.12	The user will select an amount of terrain and will raise and lower the selected terrain using the mousewheel.	The selected terrain's elevation will change. If the user scrolls up, the terrain's elevation increases. If the user scrolls down, the terrain's elevation decreases.			
13a	SRS 3.2.1.13	The user will choose to create a new project from the project gallery window. He	The main gui will open with a newly created flat map of the correct size where each tile			

Test #	Requirement or Purpose	Action / Input	Expected Result	Actual Result	P/F	Notes
		will enter a valid location to save the project, valid name, and valid map information. The map information is a map name, valid file location, map size, and a default texture.	has the default texture. The map will be displayed in the viewport.			
13b	SRS 3.2.1.13	The user will choose to create a new project from the project gallery window. He will enter a valid location to save the project, valid name, and valid map file.	The user will choose to create a new project from the project gallery window. He will enter a valid location to save the project, valid name, and valid map information.			
13c	SRS 3.2.1.13	The user will choose to create a new project from the project gallery window. He will enter an invalid location to save the project, valid name, and valid map information. The map	An error will be displayed that says "The project directory that was specified is invalid. Please select another directory." The user will be taken back to the create project window.			

Test #	Requirement or Purpose	Action / Input	Expected Result	Actual Result	P/F	Notes
		information is a map name, valid file location, map size, and a default texture.				
13d	SRS 3.2.1.13	The user will choose to create a new project from the project gallery window. He will enter a valid location to save the project, valid name, and valid map information. The map information is a map name, an invalid file location, map size, and a default texture.	An error will be displayed that says “The map file name that was specified is invalid. Please select another file name.” The user will be taken back to the create project window.			
13e	SRS 3.2.1.13	The user will choose to create a new project from the project gallery window. He will enter a valid location to save the project, valid name, and an invalid map file.	An error will be displayed that says “The file <filename> is not a recognizable format. Please select another map file.” The user will be taken back to the create project window.			
14	SRS 3.2.1.14	The user will choose to	The user successfully			

Test #	Requirement or Purpose	Action / Input	Expected Result	Actual Result	P/F	Notes
		initiate a save project command.	saves and the project resource archive is updated.			
15a	SRS 3.2.1.15	The user chooses to load a project from the project gallery or from the project menu. He then selects a valid project file from the file dialog.	The currently open project (if any) is closed and the selected project is loaded and the first map in the project is displayed in the viewport.			
15b	SRS 3.2.1.15	The user chooses to load a project from the project menu. He then selects an invalid project file from the file dialog.	An error will be displayed saying "The selected file is not a recognizable format. Please select a different file." The file dialog will be shown again.			
16	SRS 3.2.1.16	The user selects a 3D model that has been placed on the map. He then clicks the button labeled "Move Model" to enable moving of the model. He then clicks on one of the handles coming out of the model and drags the	The model will be moved a distance proportional to the amount of movement of the mouse.			

Test #	Requirement or Purpose	Action / Input	Expected Result	Actual Result	P/F	Notes
		mouse while holding down the button.				
17	SRS 3.2.1.17	The user selects a 3D model that has been placed on the map. He then clicks the button labeled "Scale Model" to enable resizing of the model. He then clicks on one of the handles coming out of the model and drags the mouse while holding down the button.	The model will be uniformly resized either larger or smaller based on the direction of movement of the mouse.			
18	SRS 3.2.1.18	The user selects a 3D model that has been placed on the map. He then clicks the button labeled "Rotate Model" to enable rotating of the model. He then clicks on the model and drags the mouse while holding down the button.	The model will be rotated in a direction that is proportional to the movement of the mouse. The model will have full 360 degree rotational ability.			
19a	SRS 3.2.2.1	Select to show a grid on the terrain. Pan around the terrain to make	A grid overlay is displayed on the terrain. It should cover the entire			

Genesis Test Cases

Test #	Requirement or Purpose	Action / Input	Expected Result	Actual Result	P/F	Notes
		sure the grid was placed correctly.	terrain.			
19b	SRS 3.2.2.1	Toggle grid visibility to hide.	The grid overlay is no longer visible on the map.			

3.2 Non-functional Testing

3.2.1 Test Descriptions

3.2.1.1 Reliability

Reliability will be tracked and tested by maintaining a log of any unhandled exception or application failure. If the ration of failed executions to total executions is greater than 5% then we will consider the non-functional requirement of reliability to be in jeopardy.

An execution consists of performing the full test suite specified in section 3.1 of this document.

3.2.1.2 Efficiency

We will stress test the application by creating a new map, adding 25 3D models to the map and if the application continues to perform at 20 or more frames per second on a system that meets the minimum requirements, than we will consider non-functional requirement of efficiency to be a success. The minimum requirements are:

- Pentium 4 2 GHz processor
- 512 MB of ram
- 128 MB Direct 9 capable video card
- Keyboard
- Mouse
- Windows XP

3.2.1.3 Availability

The availability requirement will be met if and only if running the Genesis installer completely prepares the user's system to run our application.

3.2.1.4 Maintainability

The maintainability requirement will be met if the code for the application is understandable to a programmer that is not associated to this project. To this end, we will periodically ask associates of ours to review our code. We will also review other team members' code periodically and constantly update documentation.

3.2.2 Testing Form

Date test performed: _____

Tester: _____

Test #	Test Description	Results and Notes	P/F
1	3.2.1.1 Reliability		
2	3.2.1.2 Efficiency		
3	3.2.1.1 Availability		
4	3.2.1.1 Maintainability		

4 Appendix

4.1 XML Doctype Definitions

4.1.1 Model Library DTD

This is a DTD describing the format that the XML file that contains the serialization of the Model Library.

```
<?xml encoding="UTF-8"?>

<!ELEMENT ModelLibrary (ModelItems*)>
<!ATTLIST ModelLibrary
  ID PCDATA #REQUIRED
  name PCDATA #REQUIRED
  path PCDATA #REQUIRED
  versionID PCDATA #REQUIRED>

<!ELEMENT ModelItems EMPTY>
<!ATTLIST ModelItems
  TextureID PCDATA #REQUIRED
  instanceID PCDATA #REQUIRED
  name PCDATA #REQUIRED
  path PCDATA #REQUIRED
  typeID PCDATA #REQUIRED
  versionID PCDATA #REQUIRED>
```

4.1.2 Project DTD

This describes the project as a whole including the associated Libraries.

```
<?xml encoding="UTF-8"?>

<!ELEMENT Project (TextureLibraries*,ModelLibraries*,PlaceableID,LibraryID)>
<!ATTLIST Project
  date PCDATA #REQUIRED
  name PCDATA #REQUIRED
  path PCDATA #REQUIRED
  versionID PCDATA #REQUIRED>

<!ELEMENT TextureLibraries (#PCDATA)>
<!ELEMENT ModelLibraries (#PCDATA)>
<!ELEMENT PlaceableID (#PCDATA)>
<!ELEMENT LibraryID (#PCDATA)>
```

4.1.3 Texture Library DTD

The texture library and associated textures are described by this.

```
<?xml encoding="UTF-8"?>

<!ELEMENT TextureLibrary (TextureItems*)>
<!ATTLIST TextureLibrary
  ID PCDATA #REQUIRED
  name PCDATA #REQUIRED
  path PCDATA #REQUIRED
  versionID PCDATA #REQUIRED>

<!ELEMENT TextureItems EMPTY>
<!ATTLIST TextureItems
  instanceID PCDATA #REQUIRED
  name PCDATA #REQUIRED
  path PCDATA #REQUIRED
  typeID PCDATA #REQUIRED
  versionID PCDATA #REQUIRED>
```

4.1.4 Map DTD

A serialization of the map to XML is stored here.

```
<?xml encoding="UTF-8"?>

<!ELEMENT Map (Vertices*,Quads*)>
<!ATTLIST Map
  LastID PCDATA #REQUIRED
  MapID PCDATA #REQUIRED
  name PCDATA #REQUIRED
  path PCDATA #REQUIRED
  versionID PCDATA #REQUIRED
  xsize PCDATA #REQUIRED
  ysize PCDATA #REQUIRED>

<!ELEMENT Vertices EMPTY>
<!ATTLIST Vertices
  VertexOf PCDATA #REQUIRED
  X PCDATA #REQUIRED
  Y PCDATA #REQUIRED
  Z PCDATA #REQUIRED>

<!ELEMENT Quads EMPTY>
<!ATTLIST Quads
  PlaceableID PCDATA #REQUIRED
  TerrainX PCDATA #REQUIRED>
```

TerrainY PCDATA #REQUIRED
TextureID PCDATA #REQUIRED
VertexIds PCDATA #REQUIRED>