

Genesis
Ratatosk

Software Requirements Specification Document

Richard Bailey / Troy Brant / Jonathan Clark / Chad Hansen

Table of Contents

1. Introduction	4
1.1 Purpose	4
1.2 Scope	4
1.3 Definitions, Acronyms, and Abbreviations.	4
1.4 References	5
1.5 Overview	5
2. The Overall Description	6
2.1 Product Perspective	6
2.1.1 System Interfaces	6
2.1.2 Interfaces	6
2.1.3 Hardware Interfaces	7
2.1.4 Software Interfaces	7
2.1.5 Communications Interfaces	7
2.1.6 Memory Constraints	7
2.1.7 Operations	7
2.1.8 Site Adaptation Requirements	8
2.2 Product Functions	8
2.2.1 Primary Functions	8
2.2.2 Secondary Functions	9
2.3 User Characteristics	9
2.4 Constraints	9
2.5 Assumptions and Dependencies	9
2.6 Apportioning of Requirements.	9
3. Specific Requirements	10
3.1 External Interfaces	10
3.2 Functions	11
3.2.1 Primary Functional Requirements	11
3.2.1.1 Create a new 3D map	11
3.2.1.3 Load a saved map	12
3.2.1.4 Import 3D models into libraries.	12
3.2.1.5 Save library	13
3.2.1.6 Load library	13
3.2.1.7 Create resource archive	14
3.2.1.8 Place 3D models in world	14
3.2.1.9 Filter Onscreen Objects	14
3.2.1.10 Changing the view perspective	15
3.2.1.11 Apply texture to tile(s)	15
3.2.1.12 Deform terrain	16
3.2.1.13 Create a Project	16
3.2.1.14 Save a Project	17
3.2.1.15 Load a Project	17
3.2.1.16 Moving 3D Models	17

Software Requirements Specifications Document

3.2.1.17	Scaling 3D Models	18
3.2.1.18	Rotating 3D Models	18
3.2.2	Secondary Functional Requirements	19
3.2.2.1	Overlay Grid	19
3.2.2.2	Resize terrain	19
3.3	<i>Performance Requirements</i>	20
3.4	<i>Logical Database Requirements</i>	20
3.5	<i>Design Constraints</i>	20
3.5.1	Standards Compliance	20
3.6	<i>Software System Attributes</i>	20
3.6.1	Reliability	20
3.6.2	Efficiency	20
3.6.3	Availability	20
3.6.4	Maintainability	20
3.6.5	Portability	21
3.6.6	Security	21
4.	Change Management Process	21
5.	Document Approvals	22

1. Introduction

1.1 Purpose

This SRS document is intended to provide a full listing of the product requirements for a fully operable 3D map editing program. This document will explicitly describe each function of the system, and it will identify any constraints, assumptions, or dependencies of the system. The primary target audience of this document is the developers of the system, Team Ratatosk. Others who may read this document include game developers using our system or anyone considering using our system.

1.2 Scope

The application under development is called Genesis. The name comes directly from its purpose which is to provide the user with a simple and intuitive way to build 3D virtual environments. Our product will provide basic viewing functionality but is not intended to be used as a 3D rendering application or game engine. Ideally the final product will have a rich file format that may be utilized in many different games and genres. The main benefit to using our application is that most (if not all) map editors on the market are highly specialized for a given game and are fairly complex to use. This means that a lot of time will be required to learn the editor as well as writing an importer for what is (often) a poorly documented file format. Our product will produce a content-rich file format that is well documented making it applicable for many different genres. It will be easy to write code to support our application's file format.

The ideal customer is one who has access to a game engine but does not want to spend the effort required to build a level editor. Our customer will benefit from using this tool over designing their own because of the time required to fully design, implement, and test an application of this type will be able to be spent creating content for their application. Further, we will provide a well thought out and documented file format that will allow those who wish to use this application for level design to select almost any game engine and will make the process of writing an environment importer for said engine fairly simple. This program will simply allow you to create the maps that a separate game engine will be able to use.

1.3 Definitions, Acronyms, and Abbreviations.

- SRS (Software Requirements Specification)
- 3D (3-dimensional)
- 3DS (File format for 3D Studio Max)
- GFF (Genesis File Format – file format for storing maps)
- GAR (Genesis ARchive – file format for storing resources sets for an environment)

- BVT (Build Verification Test)
- GUI (Graphical User Interface)
- Mb (Megabyte, ~1,000,000 bytes)
- RAM (Random Access Memory)
- BMP (Bitmap, an image file format)
- PPM (an image file format)
- Tile (A 1x1 unit length square that represents a portion of terrain in the 3D world)
- Event (An area of space in the 3D world that can trigger certain actions to occur in a game developer's engine)
- OpenGL (Programming language used to render and display objects in 3D)
- Project (A collection of related maps and libraries that make it easier to manage maps for a game)

1.4 References

- UnrealEd – level editor for popular video game Unreal 2004, extremely powerful and complex
 - <http://en.wikipedia.org/wiki/UnrealEd>
- Sim Golf – inspiration for our editor's interface
 - <http://www.simgaming.net/simgolf/screens/screen6.jpg>
 - <http://www.simgaming.net/simgolf/screens/screen8.jpg>
- StarEdit – simple, somewhat powerful level editor for StarCraft, another inspiration for our product
 - <http://www.staredit.net/>

1.5 Overview

Here is an outline of this document, including a brief description of each section:

- Section 1 – This section provides an overview of this SRS document.
- Section 2 - In section two, we look at the product from a user's perspective. Here we will attempt to make explicit the assumptions that the customer has about product functionality as well as what we expect of the end user. This will include assumptions about the user's technical knowledge and system requirements. Section two will also address what outside software and hardware our product will have to communicate with and will provide a rough outline of when specific functionality will be presented to the user for review. Most importantly we will provide product requirements in language that the user can understand and respond to.
- Section 3 - Section three is written in such a manner that a designer responsible for a portion of the system will be able to sit down with this portion of the SRS and design a system that will fully meet the requirements that the customers have set forth as well as any other requirements that are imposed upon the product. Within this section, we will list all inputs and outputs to the system as well as all requirements associated with them. This section provides requirements in enough

detail to implement as well as write test cases and gives the process of dealing with specific error cases.

- Section 4 - Section four addresses how the customer requests changes to the product. Additionally we specify the process by which we review customer requested changes and enumerate actions that may be taken on these requests.
- Section 5 - Section five is verification that the entire team agrees upon the current version of the SRS document.

2. The Overall Description

2.1 Product Perspective

Our product seeks to fill the need for a generic 3D map design tool that can be used to build levels for any game engine. As a level designer, this product will stand alone, needing no other program to reach full functionality. There are several other comparable products on the market now, including StarEdit for Starcraft and UnrealEd for Unreal Tournament. The difference between these editors and Genesis is that Genesis will be able to create maps that any 3D game can use instead of being tied to just one game.

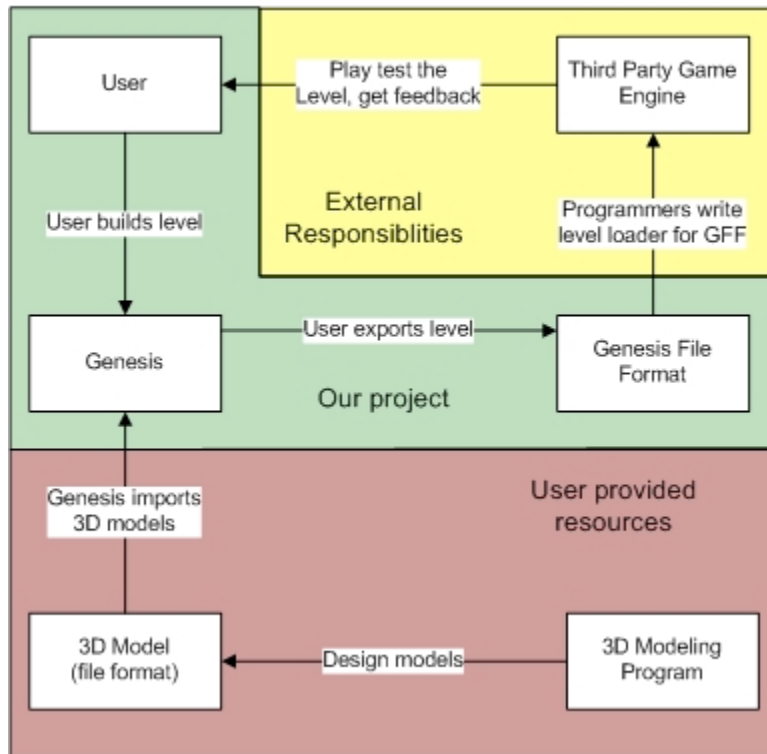
2.1.1 System Interfaces

- Game engine – A game developer will use the maps generated from our program in their game.
- 3D models – 3DS models will be loaded into the Genesis application.

2.1.2 Interfaces

The system will interface with the user through a GUI. The user will be able perform all the functions of the system by interacting with a visual representation of a 3D map. They will indicate what action they want to take while using the application through mouse clicks and keystrokes. This interface will be optimized by keeping the interface uncluttered and easy to understand.

Genesis Interaction Diagram



2.1.3 Hardware Interfaces

The user will use the mouse and keyboard to interface with the program.

2.1.4 Software Interfaces

The system has no software interface requirements.

2.1.5 Communications Interfaces

The system has no software interface requirements.

2.1.6 Memory Constraints

The minimum memory requirements include 32Mb of video RAM on the machine's video card, and 64Mb of system RAM on the machine. This memory constraint is due to the memory-intensive nature of the 3D modeling software, OpenGL.

2.1.7 Operations

The system has no operations requirements.

2.1.8 Site Adaptation Requirements

The system has no site adaptation requirements.

2.2 Product Functions

2.2.1 Primary Functions

The following is a list of primary functions the user will be able to perform:

1. Create a new map – A new 3D map can be created by specifying the number of tiles for the length and width of the map.
2. Save a map – A map can be saved to disk.
3. Load a map – A map can be loaded from disk and displayed.
4. Load 3D model into a library – A 3D model can be added to a set of 3D models.
5. Save libraries – A set of textures or models can be saved to disk.
6. Load libraries – A set of textures or models can be loaded from disk.
7. Create a resource archive - All the resources used for creating map can be saved into a single file.
8. Place models in the 3D world – A 3D model can be placed in the 3D world. After it's placed, it can be manipulated according to requirement 3.3.2.1.
9. Filter which type of objects to view – The view can be changed in order to view individual components of the 3D world. In particular, the view can be filtered to view only the terrain, only 3D models, or just a wireframe.
10. Change the view perspective – The system will have the ability to zoom in and zoom out. The system will also have the ability to pan (move laterally) across an area.
11. Apply textures to tiles – A texture can be selected and then applied to either a single tile or a set of tiles.
12. Raise or lower the terrain – A single tile or a set of tiles can be selected and then raised or lowered.
13. Create a project – A project of related maps and libraries can be created.
14. Save a project – A project can be saved to the disk.
15. Load a project – A project can be loaded from the disk.
16. Move a 3D model – A model in the 3D world can be moved around.

17. Scale a 3D model – A model in the 3D world can be resized.

18. Rotate a 3D model – A model in the 3D world can be rotated.

2.2.2 Secondary Functions

The following is a list of functions that may or may not be implemented:

1. View a grid over the tiles on the terrain – The terrain should have a grid placed over it to provide a finer detail of its slopes and deformations.
2. Resize the terrain – The terrain should have the ability to have tiles added or removed. This will in effect change the size of the terrain.

2.3 User Characteristics

The typical user of this program will be some type of game developer. However, one of the goals of the program is to make the program simple to use. With that in mind, the user of this program needs to have a basic knowledge of how to use a computer, which includes how to:

- find, open, and close applications and files
- navigate through a graphical user interface using mouse clicks and keystrokes

Though game developers may have experience in using 3D map generation software, this experience will be beneficial, but not necessary.

2.4 Constraints

- Response Time - The system needs to be fast enough that the user will not see any choppiness while editing or viewing the 3D map.
- Extensibility - The system needs to be created so that programmers in the future will be able to easily add functionality to the system.

2.5 Assumptions and Dependencies

There are no assumptions or dependencies.

2.6 Apportioning of Requirements.

These are the requirements that may be delayed until future releases of the system:

1. View a grid over the tiles on the terrain
2. Resize the terrain

3. Specific Requirements

3.1 External Interfaces

1. Inputs for Genesis include the user interface, saved files, external 3D model files, and texture files. More specifically:
 - User Interface
 - This is the primary source of input for the system. It is the medium through which the user will perform all terrain editing tasks and model placement.
 - This input will come from the user via keyboard and mouse.
 - We should be able to handle any possible input.
 - Saved Files
 - A saved file is a level design that is in some various stage of completion which Genesis will be able to load, allowing the user to resume work at the previous state.
 - These files will be generated by Genesis itself when a user chooses to save the current level being designed.
 - .gff files will be the file extension used to identify files created by Genesis and that contain an environment in progress. Within we will store
 - terrain mesh
 - trigger data (optional)
 - event data (optional)
 - .gar files are an optional requirement and, should they be implemented, will contain all data stored in the standard .gff but will additionally store
 - 3D model mesh data
 - texture information
 - External 3D model files
 - These are the sources of models that the user will place into the environment.

- These files will be provided by the user. Currently we plan only to support the 3DS file format.
- Texture files
 - These are simply image files that the user will provide. We will support the .bmp and .ppm image formats. These will provide the detail view of the environment.

3.2 Functions

The individual system requirements are listed below using the following format:

- *Requirement name*
 - *Description*
 - *Inputs (pre-conditions)*
 - *Outputs (post-conditions)*
 - *Success Cases*
 - *Failure Cases*

3.2.1 Primary Functional Requirements

3.2.1.1 Create a new 3D map

3.2.1.1.1) Description / Rationale

3.2.1.1.1.1) The user needs to be able to create a new map that he may then begin editing. Without this, there would be nothing to edit.

3.2.1.1.2) Inputs

3.2.1.1.2.1) A size (in tiles) given by the user

3.2.1.1.2.2) A default tile

3.2.1.1.3) Outputs

3.2.1.1.3.1) A new map of a chosen size. The size will be measured by the number of tiles in the width and length. All tiles will be standard size.

3.2.1.1.4) Success Cases:

3.2.1.1.4.1) The user decides to create a new map and chooses to do so. If a map is currently open and unsaved he will be offered the choice to save it. He then inputs an x by y size for the new map, an x by y size for the tile size, and is allowed to change the default tile if he wishes. If the provided limits are within the bounds, the new map is created entirely flat (i.e. constant elevation throughout) and with the given default texture.

3.2.1.1.5) Failure Cases:

3.2.1.1.5.1) The user tries to create a new map. When asked to enter the size, he enters a size that is out of bounds or is not a valid number. An error should be displayed and the user is allowed to re-enter the size.

3.2.1.1.5.2) The user tries to create a new map. When asked to change the default tile, he enters a tile that is invalid or is not the correct size as specified above. An error should be displayed and the user is allowed to select a different default tile.

3.2.1.2 Save a created map

- 3.2.1.2.1) Description / Rationale
 - 3.2.1.2.1.1) The user needs to be able to save any maps he has created. This includes saving all changes in elevation and textures, as well as any 3D objects that were placed.
- 3.2.1.2.2) Inputs
 - 3.2.1.2.2.1) A created map including any 3D objects that were placed
- 3.2.1.2.3) Outputs
 - 3.2.1.2.3.1) A map file defined by the genesis file format (.GFF)
- 3.2.1.2.4) Success Cases
 - 3.2.1.2.4.1) The user selects to save his map and is given the choice of a directory and file name. Once entered, the application writes all necessary data into a file on the disk.
- 3.2.1.2.5) Failure Cases
 - 3.2.1.2.5.1) The user selects to save his map and enters the correct information. When the save is attempted, a disk error occurs (such as a full disk). As this is out of our control, an error will be displayed.
- 3.2.1.3 **Load a saved map**
 - 3.2.1.3.1) Description / Rationale
 - 3.2.1.3.1.1) The user needs to be able to load any created maps into the program.
 - 3.2.1.3.2) Inputs
 - 3.2.1.3.2.1) A map file saved to disk in the genesis file format.
 - 3.2.1.3.3) Outputs
 - 3.2.1.3.3.1) The map in a 3D view ready for editing and/or viewing.
 - 3.2.1.3.4) Success Cases
 - 3.2.1.3.4.1) The user selects the map he would like to load. The file is then read and verified to be sure it is the correct format. After that, the information is read from the file and the map is displayed.
 - 3.2.1.3.5) Failure Cases
 - 3.2.1.3.5.1) The user chooses a map to load. The file chosen fails verification because it is either the wrong format or corrupt. An error will be displayed, and the user will be prompted to select another file.
- 3.2.1.4 **Import 3D models into libraries.**
 - 3.2.1.4.1) Description / Rationale
 - 3.2.1.4.1.1) This will be useful to the users as it will allow them to use imported 3D models that can later be added to the world. Without these 3D models, the world would be simple, dull terrain. Since our application only supports terrain editing within the program, we must allow the user to import models created in other programs so that they may be seamlessly added to the world.
 - 3.2.1.4.2) Inputs
 - 3.2.1.4.2.1) The inputs will be the 3D models that are being imported. These 3D models will be in the 3DS file format.
 - 3.2.1.4.3) Outputs

- 3.2.1.4.3.1) The output will be the model being added into a list (a library). This library can be saved to disk and will remember all models that were added to it upon being opened.
- 3.2.1.4.4) Success Cases
 - 3.2.1.4.4.1) The user uses a file chooser to select a model they would like to import. That file is then read and verified to be a working model. The name of the model is added to the library chosen and is available for placement at any time.
- 3.2.1.4.5) Failure Cases
 - 3.2.1.4.5.1) The user uses a file chooser to select a model they would like to import. The file is read and is found to be an incorrect format or corrupt. The application will report an error and the model will not be added.
- 3.2.1.5) **Save library**
 - 3.2.1.5.1) Description / Rationale
 - 3.2.1.5.1.1) The user should be able to save lists (aka libraries) of objects that he has imported for use in his world. These libraries will be saved as simple lists of paths that point to the various objects in the file system.
 - 3.2.1.5.2) Inputs
 - 3.2.1.5.2.1) A user created library of 3D models or Textures
 - 3.2.1.5.3) Outputs
 - 3.2.1.5.3.1) A file that contains the list of these objects
 - 3.2.1.5.4) Success Cases
 - 3.2.1.5.4.1) The user decides to save a library that he has created and selects to do so. The user is then asked for a name and directory to store the library under. Once chosen, a file is written.
 - 3.2.1.5.5) Failure Cases
 - 3.2.1.5.5.1) The user selects to save his library and enters the correct information. When the save is attempted, a disk error occurs (such as a full disk). As this is out of our control, an error will be displayed.
- 3.2.1.6) **Load library**
 - 3.2.1.6.1) Description / Rationale
 - 3.2.1.6.1.1) The user should be able to load the libraries created in 3.2.1.4 and saved in 3.2.1.5 of this document.
 - 3.2.1.6.2) Inputs
 - 3.2.1.6.2.1) The library file to load
 - 3.2.1.6.3) Outputs
 - 3.2.1.6.3.1) The application will load all models or textures associated with a given library.
 - 3.2.1.6.4) Success Cases
 - 3.2.1.6.4.1) The user specifies a library and it completes loading
 - 3.2.1.6.5) Failure Cases

3.2.1.6.5.1) The user specifies an incorrect library – they are notified of the error.

3.2.1.7 **Create resource archive**

3.2.1.7.1) Description / Rationale

3.2.1.7.1.1) This will result in a file containing all resources used in a given map. Included will be the terrain mesh, environment textures, and 3D models.

3.2.1.7.2) Inputs

3.2.1.7.2.1) The map to save

3.2.1.7.2.2) The textures associated with the map

3.2.1.7.2.3) The trigger data

3.2.1.7.2.4) The 3D models loaded on the map

3.2.1.7.3) Outputs

3.2.1.7.3.1) .gar file which contains all resources.

3.2.1.7.4) Success Cases

3.2.1.7.4.1) The user chooses to save a map as a Genesis Archive and Genesis produces a file that contains all relevant data.

3.2.1.7.5) Failure Cases

3.2.1.7.5.1) Unable to write file, disk is full. In this event an error will be displayed to the user.

3.2.1.7.5.2) Invalid file name. Display error to user.

3.2.1.8 **Place 3D models in world**

3.2.1.8.1) Description / Rationale

3.2.1.8.1.1) The user should be able to place 3D models that have been imported into their map.

3.2.1.8.2) Inputs

3.2.1.8.2.1) A 3D model from a model library.

3.2.1.8.3) Outputs

3.2.1.8.3.1) An instance of the 3D model placed into the map.

3.2.1.8.4) Success Cases

3.2.1.8.4.1) The user selects an object out of a currently open library. He then clicks on the map where he wants the object to be and it is placed.

3.2.1.8.5) Failure Cases

3.2.1.8.5.1) None.

3.2.1.9 **Filter Onscreen Objects**

3.2.1.9.1) Description / Rationale

3.2.1.9.1.1) The user should be able to view certain subsets of all of the items that can be displayed on a map. These items include textures and 3D models.

3.2.1.9.2) Inputs

3.2.1.9.2.1) A boolean to specify if the user wishes to view 3D models.

- 3.2.1.9.2.2) A boolean to specify if the user wishes to view textures.
- 3.2.1.9.3) Outputs
 - 3.2.1.9.3.1) The 3D view without the items that the user has specified to filter.
- 3.2.1.9.4) Success Cases
 - 3.2.1.9.4.1) The user selects to filter 3D models. The 3D view will update and no longer show any 3D models.
 - 3.2.1.9.4.2) The user selects to filter textures. The 3D view will update and show just a wireframe of the terrain.
 - 3.2.1.9.4.3) The user selects to not filter 3D models. The 3D view will update and now show 3D models.
 - 3.2.1.9.4.4) The user selects to not filter textures. The 3D view will update and show textures instead of a wireframe of the terrain.
- 3.2.1.9.5) Failure Cases
 - 3.2.1.9.5.1) None.
- 3.2.1.10 **Changing the view perspective**
 - 3.2.1.10.1) Description / Rationale
 - 3.2.1.10.1.1) The user should be able to move the camera's current position to view different portions of the map. This include both moving around the map and zooming in and out of the map. This will allow the map to be viewed as close as necessary to get the details of the map exactly how they want. The user will not be able to scroll completely off the map.
 - 3.2.1.10.2) Inputs
 - 3.2.1.10.2.1) User input on where to move or zoom
 - 3.2.1.10.3) Outputs
 - 3.2.1.10.3.1) The movement and zoom of the map
 - 3.2.1.10.4) Success Cases
 - 3.2.1.10.4.1) The user selects to move in a direction. The camera then moves in the way that was requested.
 - 3.2.1.10.4.2) The user selects to zoom in or out. The camera then adjusts its distance from the terrain as requested.
 - 3.2.1.10.5) Failure Cases
 - 3.2.1.10.5.1) The user attempts to zoom into the terrain. The camera will stop zooming in.
 - 3.2.1.10.5.2) The user attempts to pan such that no portion of the map is on the screen. The camera will stop moving in the requested direction .
- 3.2.1.11 **Apply texture to tile(s)**
 - 3.2.1.11.1) Description / Rationale
 - 3.2.1.11.1.1) The user should be able to select the texture of each tile to allow greater customization.
 - 3.2.1.11.2) Inputs
 - 3.2.1.11.2.1) A texture to be applied
 - 3.2.1.11.2.2) A tile to apply the texture to

- 3.2.1.11.3) **Outputs**
 - 3.2.1.11.3.1) The tile with the given texture applied
- 3.2.1.11.4) **Success Cases**
 - 3.2.1.11.4.1) The user selects a tile and chooses to change its texture. He is given a list of available textures that can be applied. After choosing one that texture is applied and the tile is changed on the map.
- 3.2.1.11.5) **Failure Cases**
 - 3.2.1.11.5.1) The user selects an invalid texture. An error will be displayed.

3.2.1.12 **Deform terrain**

- 3.2.1.12.1) **Description / Rationale**
 - 3.2.1.12.1.1) The user should be allowed to deform the terrain to create any manner of hills and valleys he wants.
- 3.2.1.12.2) **Inputs:**
 - 3.2.1.12.2.1) An area to raise or lower
 - 3.2.1.12.2.2) The user command to either raise or lower the terrain
- 3.2.1.12.3) **Outputs:**
 - 3.2.1.12.3.1) The area selected having been raised or lowered as requested
- 3.2.1.12.4) **Success Cases**
 - 3.2.1.12.4.1) The user selects an area of terrain of some size and chooses to raise or lower it. The whole area is raised or lowered with the slopes gradually rising to a peak in the center of the area. There is no reasonable limit on the ability to raise or lower terrain.
- 3.2.1.12.5) **Failure Cases**
 - 3.2.1.12.5.1) None.

3.2.1.13 **Create a Project**

- 3.2.1.13.1) **Description / Rationale**
 - 3.2.1.13.1.1) The user will want to be able to save collections of maps, 3D model libraries, and texture libraries that are related to each other. This can be used to create a project for a game or to create a collection of levels with similar tilesets and 3D models.
- 3.2.1.13.2) **Inputs**
 - 3.2.1.13.2.1) An initial map or new map.
 - 3.2.1.13.2.2) The name of the project.
 - 3.2.1.13.2.3) A project file location.
- 3.2.1.13.3) **Outputs**
 - 3.2.1.13.3.1) The 3D view of the initial map with the ability to view all of the information in the project.
- 3.2.1.13.4) **Success Cases**
 - 3.2.1.13.4.1) The user selects to create a new project, enters in information and selects to create a new map. The project is created.
 - 3.2.1.13.4.2) The user selects to create a new project, enters in the project information and selects a pre-existing map. The application validates the map successfully and creates the project.

3.2.1.13.5) Failure Cases

3.2.1.13.5.1) The user selects to create a new project, enters in the project information and the project location is invalid. An error window will be displayed and the user will be prompted to enter in a different project location.

3.2.1.13.5.2) The user selects to create a new project, enters in the project information, and selects a pre-existing map. The map fails validation

3.2.1.14 **Save a Project**

3.2.1.14.1) Description / Rationale

3.2.1.14.1.1) The user will wish to be able to save a project so that he can load it again the next time he runs the application.

3.2.1.14.2) Inputs

3.2.1.14.2.1) The project name.

3.2.1.14.2.2) The maps in the project.

3.2.1.14.2.3) The libraries in the project

3.2.1.14.3) Outputs

3.2.1.14.3.1) A project file containing the list of maps and the information for the libraries.

3.2.1.14.4) Success Cases

3.2.1.14.4.1) The user selects to save the project and it is saved successfully.

3.2.1.14.5) Failure Cases

3.2.1.14.5.1) The user selects to save the project and there is a disk error. Since this is beyond our control, an error will be displayed.

3.2.1.15 **Load a Project**

3.2.1.15.1) Description / Rationale

3.2.1.15.1.1) The user will wish to be able to load a project that he has previously saved.

3.2.1.15.2) Inputs

3.2.1.15.2.1) A project file.

3.2.1.15.3) Outputs

3.2.1.15.3.1) A 3D view of the first map in the project and all of the information and libraries are loaded.

3.2.1.15.4) Success Cases

3.2.1.15.4.1) The user selects a project file and it is validated.

3.2.1.15.5) Failure Cases

3.2.1.15.5.1) The user selects a project file and it is invalid or corrupt. An error window will be displayed and the user will be prompted to select a different file.

3.2.1.16 **Moving 3D Models**

3.2.1.16.1) Description / Rationale

- 3.2.1.16.1.1) The user will wish to be able to move a 3D model on the x, y, and z axes so that he can precisely place the object. The model can be moved anywhere the user wishes.
- 3.2.1.16.2) Inputs
 - 3.2.1.16.2.1) A 3D model.
 - 3.2.1.16.2.2) An amount to move the model.
 - 3.2.1.16.2.3) The axes to move the model on.
- 3.2.1.16.3) Outputs
 - 3.2.1.16.3.1) The 3D model moved in the direction(s) the user has selected.
- 3.2.1.16.4) Success Cases
 - 3.2.1.16.4.1) The user selects a model in the world and moves it in one of the three axes and the model moves in the intended direction.
- 3.2.1.16.5) Failure Cases
 - 3.2.1.16.5.1) The user inputs a non-numeric value for the move distance. The input will be ignored and the model will not be moved. The system will continue to function as it did before the input was entered.

3.2.1.17 **Scaling 3D Models**

- 3.2.1.17.1) Description / Rationale
 - 3.2.1.17.1.1) The user will wish to be able to scale the size of the models. This allows for a map to have differently sized trees or buildings. The model can be scaled to any size the user wishes.
- 3.2.1.17.2) Inputs
 - 3.2.1.17.2.1) A 3D model.
 - 3.2.1.17.2.2) Amount to scale the 3D model by.
- 3.2.1.17.3) Outputs
 - 3.2.1.17.3.1) The 3D model resized by the amount specified by the user.
- 3.2.1.17.4) Success Cases
 - 3.2.1.17.4.1) The user selects a model and scales it.
- 3.2.1.17.5) Failure Cases
 - 3.2.1.17.5.1) The user inputs a non-numeric value for the scaling amount. The input will be ignored and the model will not be scaled. The system will continue to function as it did before the input was entered.

3.2.1.18 **Rotating 3D Models**

- 3.2.1.18.1) Description / Rationale
 - 3.2.1.18.1.1) The user will wish to be able to rotate the models in the 3D world.
- 3.2.1.18.2) Inputs
 - 3.2.1.18.2.1) A 3D model.
 - 3.2.1.18.2.2) Amount to rotate the 3D model by.
- 3.2.1.18.3) Outputs
 - 3.2.1.18.3.1) The 3D model rotated by the amount specified by the user.
- 3.2.1.18.4) Success Cases
 - 3.2.1.18.4.1) The user selects a model and rotates it.
- 3.2.1.18.5) Failure Cases

- 3.2.1.18.5.1) The user inputs a non-numeric value for the rotating amount. The input will be ignored and the model will not be rotated. The system will continue to function as it did before the input was entered.

3.2.2 **Secondary Functional Requirements**

3.2.2.1 **Overlay Grid**

3.2.2.1.1) Description / Rationale

- 3.2.2.1.1.1) Provide the ability to place a grid on the map that allows the user to see division of the tiles easily.

3.2.2.1.2) Inputs

- 3.2.2.1.2.1) The map the user is editing.

3.2.2.1.3) Outputs

- 3.2.2.1.3.1) A grid overlay of the map

3.2.2.1.4) Success Cases

- 3.2.2.1.4.1) The user enables the grid and the overlay is displayed.
- 3.2.2.1.4.2) The user disables the grid and the overlay is removed.

3.2.2.1.5) Failure Cases

3.2.2.2 **Resize terrain**

3.2.2.2.1) Description / Rationale

- 3.2.2.2.1.1) The user should be able to resize a map. Initiated at user will, he will select to add or remove a number of columns and rows of tiles and it will be done from a standard area. This function will only affect the terrain; the 3D models will not be altered in any way by this function.

3.2.2.2.2) Inputs

- 3.2.2.2.2.1) How many rows to add to the map or which row to remove from the map

3.2.2.2.3) Outputs

- 3.2.2.2.3.1) The map with the rows added.
- 3.2.2.2.3.2) The map with the rows removed.

3.2.2.2.4) Success Cases

- 3.2.2.2.4.1) The user selects to resize the map. He selects to increase the size. He enters how many rows and columns should be added and they are added.
- 3.2.2.2.4.2) The user selects to resize the map. He chooses to remove a row. He selects which row is to be removed and it is removed.

3.2.2.2.5) Failure Cases

- 3.2.2.2.5.1) The user enters a non-numerical value. An error dialog is displayed and the user is prompted for another value.
- 3.2.2.2.5.2) The user enters a number less than 1. An error dialog is displayed and the user is prompted for another value.

3.3 Performance Requirements

Genesis should run at 20 frames per second with any user configurable options at the default levels on the minimum system described in section 2.1.6. There will be an option available that will print out the number of frames per second in order to test this requirement.

3.4 Logical Database Requirements

There will be no database used in our product. This section is not applicable.

3.5 Design Constraints

3.5.1 Standards Compliance

There are no standards compliance requirements to meet.

3.6 Software System Attributes

The following is a list of our non-functional requirements. They are ordered based on the importance:

3.6.1 Reliability

Genesis should meet the performance requirements described in section 3.3 95% of the time.

3.6.2 Efficiency

Genesis needs to be efficient enough that a system meeting the requirements described in section 2.1.6 doesn't experience a slowdown of more than 15% while Genesis is running.

3.6.3 Availability

Genesis is an application that will be run on-demand. As a result we can not rely on any internal or external interface that will limit the time frame of Genesis' functionality.

3.6.4 Maintainability

Genesis will not require much maintenance after it is developed since it will not be developed around any aspect that is constantly changing. For instance, the 3DS file format is set and more than likely will not change in the future, so the system will not have to be updated in that respect. However, the system should be written in a modular

way so that if a developer in future would like to extend the system, they could do it without have to learn the entire code base.

3.6.5 Portability

The product is being developed only for the Windows operating system. This is the only operating system supported, and there is no requirement for the system to be compatible with any other operating system.

3.6.6 Security

There are no security concerns regarding Genesis.

4. Change Management Process

The customer requests changes to the product by emailing their contact with our team. When the customer requests a new feature to be added or to alter a current feature, we will add the request to a queue of pending changes which will be reviewed at the end of every implementation phase. This list will be kept informally on the swiki. The list is modifiable by the all members of the team. Before an action is taken on any particular item, a consensus must be reached among the team. Possible actions are:

- Add as a requirement
- Modify current requirement
- Delete current requirement
- Request further clarification from customer
- Table discussion on the request until the next requirements meeting

5. Document Approvals

The SRS document requires approval from the following people

Name	Signature	Date
Richard Bailey	_____	_____
Chad Hansen	_____	_____
Troy Brant	_____	_____
Jonathon Clark	_____	_____