

Ratatosk

Genesis – 3D World Builder

Version 1.1

October 28, 2005

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Richard Bailey Chad Hansen Jonathan Clark Troy Brant	This is the initial version of the document	09/07/05
1.1	Richard Bailey Chad Hansen Jonathan Clark Troy Brant	This revision updates the changes that have been made to the project	10/28/05

Review & Approval

If appropriate, this section contains formal sign-off for both review and approval of the project plans. Normally the projects authority and agent should formally sign off the plan to launch a project. If you do not need all this approval, you can delete this section.

Project Plan Approval History

Approving Party	Version Approved	Signature	Date
Richard Bailey			
Chad Hansen			
Jonathan Clark			
Troy Brant			

Contents

<u>REVIEW & APPROVAL</u>	<u>II</u>
<u>1 INTRODUCTION</u>	<u>1</u>
1.1 OVERVIEW	1
1.2 DELIVERABLES	1
1.3 ASSUMPTIONS AND CONSTRAINTS	1
1.4 REFERENCE MATERIALS	2
1.5 DEFINITIONS AND ACRONYMS	2
<u>2 MANAGEMENT STRUCTURE</u>	<u>3</u>
2.1 PROJECT LIFECYCLE	3
2.2 PROJECT ORGANIZATION	3
2.2.1 External Interfaces	3
2.2.2 Internal Structure	4
2.2.3 Roles and Responsibilities	4
2.2.4 Staffing	4
2.3 COMMUNICATION	4
2.4 RISK AND ASSET MANAGEMENT	5
2.5 STARTUP	5
2.6 CLOSEOUT	5
<u>3 PLANNING AND CONTROL.....</u>	<u>7</u>
3.1 ESTIMATE.....	7
3.1.1 Estimation Process.....	7
3.2 RESOURCE IDENTIFICATION	7
3.2.1 Staff.....	7
3.2.2 Time.....	7
3.2.3 Cost.....	7
3.2.4 Materials	8
3.3 RESOURCE ALLOCATION	8
3.3.1 Work Breakdown Structure.....	10
3.3.2 Schedule	15
3.3.3 Budget	<i>Error! Bookmark not defined.</i>
3.4 TRACKING AND CONTROL.....	16
<u>4 TECHNICAL PROCESS.....</u>	<u>17</u>
4.1 ENGINEERING.....	17
4.1.1 Environment.....	17
4.1.2 Methods, Tools and Techniques	17

[4.2 TECHNOLOGY..... 17](#)
 [4.2.1 Environment..... 17](#)
 [4.2.2 Methods, Tools, and Techniques 17](#)
[4.3 INFRASTRUCTURE..... 17](#)
[4.4 PROJECT ARTIFACTS..... 18](#)

[5 SUPPORTING PLANS 19](#)
[5.1 CONFIGURATION MANAGEMENT..... 19](#)
[5.2 QUALITY ASSURANCE..... 19](#)
[5.3 TESTING..... 20](#)
[5.4 DEPLOYMENT..... 20](#)
[5.5 INTEGRATION..... 20](#)
[5.6 PROCUREMENT..... 20](#)
[5.7 OPERATIONS..... 20](#)
[5.8 MAINTENANCE..... 20](#)
[5.9 STAFF DEVELOPMENT..... 20](#)
[5.10 PRODUCT ACCEPTANCE..... 20](#)

1 Introduction

1.1 Overview (Executive Summary)

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

Our objective is to build a tool that will be intuitive and easy to use. The program will save the maps in a format that developers will easily be able to integrate into their game. We will always keep the customer in mind, and we'll create the highest quality product we possibly can.

1.2 Deliverables

Source Code

- All source code in a zip file: A zip file containing all of the source files needed to compile this application.

Documentation

- Tutorial in PDF format: A quick tutorial on how to use the application.
- Documentation of the source code in HTML format: The developer comments and explanations of how the code is intended to function.
- This project plan document in PDF format: This document.
- A Software Requirements document in PDF format: A document outlining the list of requirements for the application.
- A Software Design document in PDF format: A document that describes the high and low level designs of the product.
- A Test Plan document in PDF format: A document outlining the procedures that we will follow to insure that all functional and non-functional requirements are met.

Software

- Product installer: An executable that will install the 3D map creator program
- Libraries required to run our product:
 - Microsoft .NET framework install program

- The CSGL library install program
- SharpZip library install program

1.3 Assumptions and Constraints

- Project schedule must be 14 weeks or less
- OpenGL must be used
- Bug tracking mechanism will be used once formal testing can take place.
- An Object Oriented language must be used

1.4 Reference Materials

- Test Plan Document: <http://swiki.cc.gatech.edu:8080/cs3300-fl05/uploads/9/%5BG%5D%20TestPlan.pdf>

1.5 Definitions and Acronyms

- XML (Extended Markup Language)
- UML (Unified Markup Language)
- BVT (Build Verification Test)
- CVS (Concurrent Versioning System)
- COC (College of Computing)
- MSDNAA (Microsoft Developer's Network Academic Alliance)
- GUI (Graphical User Interface)
- SRS (Software Requirements Specification)

2 Management Structure

2.1 Project Lifecycle

We will use a modified waterfall lifecycle using feedback after each phase is complete to refine the previous phases. There will be two full iterations. The first will result in a prototype on October 3rd. The second iteration will be the fully completed and functional product, and will be finished on December 5th. The phases are as follows:

- Requirements: During this phase we brainstorm and gather requirements. On the second iteration, we reassess our requirements based on what we learned from the first iteration.
- Design: During this phase we progress from a high to a low level design while also creating GUI mockups.
- Implementation: During this phase we write the application.
- Integration and Testing: During this phase we integrate all of the various components and perform the test cases and generate the test report.
- Reflection: During this phase we will analyze the success and failures of the previous iteration and decide, by group consensus, what changes should be made for the next iteration.

The rationale for using this is that it allows for feedback from the first iteration so that the requirements and design can be of much higher quality. A second rationale for using this method is that we are not experienced in designing OpenGL components in an object oriented language and the first iteration will help us to make an improved design on the second iteration. The third rationale is that the customer demands it.

2.2 Project Organization

Since our organization/group is small at only four people, everyone will likely have at least a small hand in nearly every role. However, the person that is assigned a roll will be delegating responsibility to ensure completion of the tasks associated with that role in a timely manner.

2.2.1 External Interfaces

As we have no set customer, our primary external contacts are Ada and Daniel. Contact will be done through email or during office hours. Any group member may do so and will brief the others once the meeting is over.

2.2.2 Internal Structure

2.2.2.1 Roles and Responsibilities

Role	Responsibility
Project Manager	Managing meetings and various other team lead activities. He will ensure that milestones are delivered on time.
Requirements Lead	Creating the program requirements and making sure functionality is up to par based on our timeline. He will also be updating the requirements in the future based on knowledge gained.
Quality Assurance Lead	Running bug tests and maintaining a list of bugs. He will also be responsible for maintaining the documentation.
Application Design Lead	Designing the application. This covers both the front-end GUI design and the backend UML.
OpenGL Design Lead	Creating and documenting the design associated with object oriented OpenGL code.
Development Engineer	Develop the application.
QA Engineer	Running bug tests and fixing bugs that are found.

2.2.2.2 Staffing

Role	Staff Member	Start Date	End Date
Project Manager	Richard	9/6	12/5
Requirements Lead	Jonathan	9/6	12/5
Quality Assurance Lead	Chad	9/6	12/5
Application Design Lead	Chad	9/6	12/5
OpenGL Design Lead	Troy	9/6	12/5
Development Engineers	Jonathan, Troy, Chad, Richard	9/6	12/5
QA Engineers	Jonathan, Troy, Chad, Richard	9/6	12/5

2.3 Communication

Our team will maintain nearly constant availability through the use of our cell phone and email if there is a problem. The swiki will also be used if needed. We will also be holding short progress meetings at least once per week to keep tabs on how everyone is doing, and we will work together on the code as is necessary. The team members already have each other's contact information.

2.4 Risk and Asset Management

Risk	Manifestation	Likelihood	Damage	Contingency
Underestimate difficulty of the project	This occurs when we are consistently late completing milestones	Moderate	Very bad	If we have underestimated the difficulty of a certain portion of the application that causes delay in the completion of milestones then we have assigned levels of necessity to each functional requirement and will start by dropping those of the lowest priority.
Inability to integrate OpenGL into the C# environment	Unable to find a library simulating GLUT in C#	Low	Very bad	Switch to a language with a known good implementation of OpenGL (ie C++)
Sickness	Sickness of team member to the point of inability to work	Very High	Bad	Someone will pick up the slack for a short amount of time. We will have at least two people with extensive knowledge on any given portion of the application's development.
Scheduling conflicts	Inability to agree upon good meeting times.	Very high	Bad	If it happens, we may be forced to alter the requirements, move milestones, increase time spent doing school work.
Inserting an OpenGL panel into a C# form.	The library we are using to integrate into C# does not support using a C# Panel.	Moderate	Mild	Use a separate window to display the OpenGL portion of the application.

2.5 Startup

Before we start to actually design the program we will have to research some OpenGL and understand the interface as it is used in C#. Before we can begin to code we will have to all learn C#, set up our CVS repository, and get our weekly progress meetings started.

2.6 Closeout

We will use a well defined, well documented file format so that any files created in our program can be picked up by another 3d application (for example, a video game) and used fairly easily. We will also be heavily documenting the program itself so that it will be easy to learn for a new user. We are allowing 3d models to be imported into the created world and placed

so as to extend our functionality even further. The project will end when a user can reliably create, save, and load maps.

3 Planning and Control

3.1 Estimate

- 8,000 to 10,000 lines of code
- 8-10 hours per week per person (on average)
- About 500 man-hours

3.1.1 Estimation Process

We looked at some OpenGL code that part of the team had written and looked at past Java GUI code that the team. We used this knowledge and previous experience integrating applications, we arrived at 8-10,000 lines of code.

The time estimates, much like the project size estimate, was based on team experience with other projects of similar size.

We will be keeping a page on the swiki to keep track of how much time each person worked per week. This will help us keep track of how good our current estimates are. We will also be holding weekly status meetings to help keep up to date on our current status. We will refine our estimates after each iteration in our project lifecycle and update this document.

3.2 Resource Identification

- College of Computing computers
- Articles/Tutorials on the web
- Books on C# and OpenGL
- Ada and Daniel

3.2.1 Staff

This does not really apply to us as our staff is constant and set.

3.2.2 Time

September 6th to December 5th is the amount of time we have to complete the project.

3.2.3 Cost

As a four person group, we expect to have a total of 720 hours of work put in to this project by the end of the semester.

3.2.4 Materials

We do not foresee the need for any special materials.

3.3 Resource Allocation

Tasks come directly from the Work Breakdown Structure in section 3.3.2.

Task	Staff
Determining Milestones	Richard, Chad, Troy, Jonathan
Estimating Time	Richard, Chad, Troy, Jonathan
Writing Project Plan	Richard, Chad, Troy, Jonathan
Brainstorming	Richard, Chad, Troy, Jonathan
Writing SRS Document	Richard, Chad, Troy, Jonathan
UML Design	Richard, Chad, Troy, Jonathan
GUI Flow Diagrams	Chad, Jonathan
Writing Design Document	Richard, Chad, Troy, Jonathan
Viewable Terrain	Richard, Troy
New Project Window and functionality	Chad, Jonathan
Project Gallery functionality	Chad, Jonathan
Initial project XML serialization	Chad, Jonathan
Initial Main Window interface layout	Chad, Jonathan
Update project plan	Richard, Chad, Troy, Jonathan
Update requirements document	Richard, Chad, Troy, Jonathan
Redesign UML	Richard, Chad, Troy, Jonathan
Redesign GUI	Chad, Jonathan
Update Design Document	Richard, Chad, Troy, Jonathan
Ability to add a model to a library	Richard, Chad, Troy, Jonathan
Ability to place a model in the world	Richard, Troy
Ability to move model in the world	Richard, Troy
Ability to scale a model	Richard, Troy
Ability to change a texture	Richard, Troy
Set multiple tile's texture	Richard, Troy
Ability to raise and lower terrain	Richard, Troy
Save Maps	Chad, Jonathan

Task	Staff
Load Maps	Chad, Jonathan
Save Projects	Chad, Jonathan
Load Projects	Chad, Jonathan
Implement revised GUI	Chad, Jonathan
Update Test Plan for new design	Richard, Chad, Troy, Jonathan
Rerun Tests and fix bugs	Richard, Chad, Troy, Jonathan
Create Testing Report	Richard, Chad, Troy, Jonathan
Tutorial	Troy
User Manual	Richard, Chad, Troy, Jonathan
Quick Readme file	Chad
Create Installer	Richard
Package installer	Richard
Deliver product	Richard, Chad, Troy, Jonathan

3.3.1 Milestones

Milestone Number	Milestone (MS)	Phase	Date Due	Days Between Each MS
M1	Project Plan	Requirements	9-07	7
M2	Requirements	Requirements	9-14	7
M3	Design	Design	9-21	7
M4	Test Plan	Design	9-28	7
M5	Initial Prototype (limited graphics)	Implementation	10-03	5
M6	Refine Project Plan including Re-requirements	Reflection / Requirements	10-10	7
M7	Redesign Application	Design	10-17	7
M8	Placing and Manipulating 3D Models	Implementation	10-24	7
M9	Terrain Manipulation and Textures	Implementation	11-02	9
M10	Save and load our maps	Implementation	11-07	5
M11	Final front-end	Integration and Testing	11-14	7
M12	Test Report	Integration and Testing	11-21	7
M13	Documentation	Reflection	11-28	7
M14	Finished Product Delivery	Complete	12-05	7

3.3.2 Work Breakdown Structure

Breakdown areas:

- M1 – Project Plan
 - T1 - Determining Milestones: We will determine what we want to have for milestones.
 - Duration: 9-1 → 9-2
 - Dependencies: None
 - T2 - Estimating Time: We will estimate how long this project will take.
 - Duration: 9-3 → 9-5
 - Dependencies: T1
 - T3 - Writing Project Plan: Writing this document.
 - Duration: 9-5 → 9-6
 - T1, T2

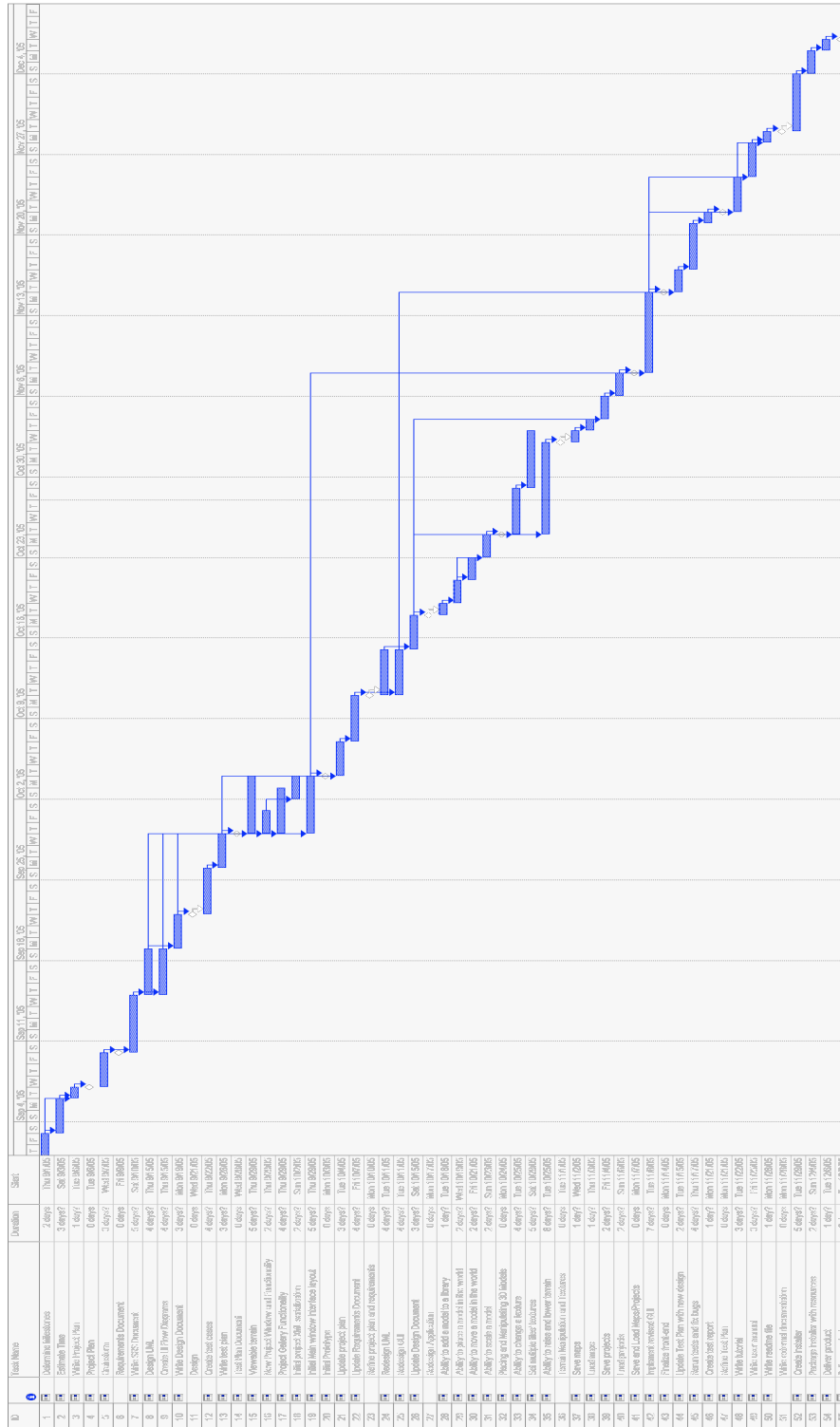
- M2 - Requirements Document
 - T4 - Brainstorming: We will brainstorm on what requirements are needed for this project.
 - Duration: 9-7 → 9-10
 - Dependencies: None
 - T5 - Writing SRS document: The actual writing of the requirements document.
 - Duration: 9-10 → 9-14
 - Dependencies: T4
- M3 - Design
 - T6 - UML design: Creation of the high and low level UML diagrams.
 - Duration: 9-15 → 9-18
 - Dependencies: T5
 - T7 - GUI flow diagrams: Creation of the GUI mockup images.
 - Duration: 9-15 → 9-18
 - Dependencies: T5
 - T8 - Writing Design Document: The writing of the design document.
 - Duration: 9-19 → 9-21
 - Dependencies: T6, T7
- M4 - Test Plan
 - T9 - Create Test Cases: The listing of the various test cases that will be used to create the test plan.
 - Duration: 9-22 → 9-25
 - Dependencies: T8
 - T10 - Write Test Plan: The writing of the test plan.
 - Duration: 9-26 → 9-28
 - Dependencies: T9
- M5 - Initial Prototype
 - T11 - Viewable terrain: Can view a 3d terrain
 - Duration: 9-29 → 10-3
 - Dependencies: T6
 - T12 - New Project Window and functionality: The new project window is laid out and can actually create a new project.
 - Duration: 9-29 → 9-30
 - Dependencies: T7, T8
 - T13 - Project Gallery functionality: A window that has the functionality for a project gallery
 - Duration: 9-29 → 10-2
 - Dependencies: T7, T8
 - T14 - Initial project XML serialization: The ability to serialize small portions of the project and maps.
 - Duration: 10-2 → 10-3
 - Dependencies: T12

- T15 - Initial Main window interface layout
 - Duration: 9-29 → 10-3
 - Dependencies: T7
- M6 - Refining the project plan and requirements document:
 - T16 - update project plan: Refines the project plan to reflect changes to the schedule and estimations.
 - Duration: 10-4 → 10-6
 - Dependencies: M4
 - T17 - update requirements document: Updates the requirements document to reflect changes to the project.
 - Duration: 10-7 → 10-10
 - Dependencies: T16
- M7 - Redesign Application
 - T18 - Redesign UML: Updates the UML diagrams to reflect changes to the high level and low level designs.
 - Duration: 10-11 → 10-14
 - Dependencies: T17
 - T19 - Redesign GUI: Updates the GUI mockups to reflect new design decisions.
 - Duration: 10-11 → 10-14
 - Dependencies: T17
 - T20 - Update design document: Updates the design document to reflect the changes made since the first iteration.
 - Duration: 10-15 → 10-17
 - Dependencies: T18, T19
- M8 - Placing and Manipulating 3D Models
 - T21 - Ability to add a model to a library: Allows for the functionality to select a model and add it to the model library.
 - Duration: 10-18
 - Dependencies: T20
 - T22 - Ability to place a model in the world
 - Duration: 10-19 → 10-20
 - Dependencies: T21
 - T23 - Ability to move a model in world
 - Duration: 10-21 → 10-22
 - Dependencies: T22
 - T24 - Ability to scale model
 - Duration: 10-23 → 10-24
 - Dependencies: T22
- M9 - Terrain Manipulation and Textures
 - T25 - Ability to change a texture: This allows for a tile's texture to be changed.
 - Duration: 10-25 → 10-28

- Dependencies: T20
 - T26 - Set multiple tile's texture: This allows for setting many tiles to have the same texture.
 - Duration: 10-29 → 11-02
 - Dependencies: T25
 - T27 - Ability to raise and lower terrain: This allows for the ability to raise and lower terrain.
 - Duration: 10-25 → 11-02
 - Dependencies: T20
- M10 - Save and Load Maps/Projects
 - T28 - Save Maps: Ability to save maps to an XML format.
 - Duration: 11-02
 - Dependencies: T27
 - T29 - Load Maps: Ability to load a map.
 - Duration: 11-03
 - Dependencies: T28
 - T30 - Save Projects: Ability to save a project to an XML format.
 - Duration: 11-04 → 11-05
 - Dependencies: T20
 - T31 - Load Projects: Ability to load a saved project.
 - Duration: 11-06 → 11-07
 - Dependencies: T30
- M11 - Finalize front-end
 - T32 - Implement revised GUI: Implements the new GUI design done during the redesign application milestone.
 - Duration: 11-08 → 11-14
 - Dependencies: T15
- M12 - Refine Test Plan
 - T33 - Update Test Plan for new design
 - Duration: 11-15 → 11-16
 - Dependencies: T19
 - T34 - Rerun tests and fix bugs: Runs the updated test plans and fixes the bugs found in the tests.
 - Duration: 11-17 → 11-20
 - Dependencies: T33
 - T35 - Create Testing Report: Create a test plan from the running of the tests.
 - Duration: 11-21
 - Dependencies: T34
- M13 - Write External Documentation
 - T36 - Tutorial: The writing of a tutorial on how to use the application.
 - Duration: 11-22 → 11-24
 - Dependencies: T32

- T37 - User Manual: The user manual describing the application and how to use it in detail.
 - Duration: 11-25 → 11-27
 - Dependencies: T32
- T38 - Quick Readme file: The writing of a quick readme file containing installation instructions and system requirements.
 - Duration: 11-28
 - Dependencies: T36, T37
- M14 - Deliver Product
 - T39 - Create installer: The creation of an installer that will install the application.
 - Duration: 11-29 → 12-3
 - Dependencies: T38
 - T40 - Package installer and resources into a self extractable file
 - Duration: 12-4 → 12-5
 - Dependencies: T39
 - T41 - Deliver product
 - Duration: 12-5
 - Dependencies: T40

3.3.3 Schedule



3.4 Tracking and Control

We will use a time tracker on the swiki to track the number of hours we have expended on the project so we can make sure we won't go over our budget. To make sure we stay on schedule, each team member will give a status report at the beginning of each meeting, and we will hold each other accountable for completing his work.

4 Technical Process

4.1 Engineering

4.1.1 Environment

We will be using pair programming and peer review on the code.

4.1.2 Methods, Tools and Techniques

We will be using Poseidon 3.1 Community Edition to create the following UML 2.0 diagrams to design our application:

- Class Diagrams
- Use Cases
- State Diagrams
- UI Flow Diagrams

All of our text documents will be written in Microsoft Word.

4.2 Technology

4.2.1 Environment

We will be using various computers running the Windows operating system. These computers include our team members' personal computers and the computers located in the College of Computing.

4.2.2 Methods, Tools, and Techniques

We will be developing and testing on Windows 2000 and Windows XP computers. The code will be developed using an open source C# IDE called SharpDevelop. Additionally, we will maintain multiple versions of the code using CVS, which will be located on helsinki.

4.3 Infrastructure

Microsoft Visual Studio .net will need to be installed to use the compiler. We will need to be able to install our IDE on the computers we are using. Finally, we will need to install the OpenGL library of choice.

4.4 Project Artifacts

- Class diagrams
- Use cases
- State diagrams
- UI Flow diagrams
- Source code
- Source documentation
- Application documentation
- User manual
- Software Requirements Specification
- Software Design Specification
- Installer
- Tutorial
- Test Plan

5 Supporting Plans

5.1 Configuration Management

We will be using CVS to manage versioning of our code. At the end of every week, milestone, turnin, and as needed we will tag the code in the format [module_name]_[YYYY]_[MM]_[DD]_[weekly|milestone|turnin|personal]. The tag name will indicate what was tagged (module name), when it was done (date), and what reason it was done (trailing qualifier). All project design artifacts will be stored in a separate cvs module (called design_docs) and will be managed in a similar manner as the code. Project documentation such as user manuals will be stored in a third cvs module (called documentation) and will be managed in a similar manner as the code.

5.2 Quality Assurance

In order to ensure that our team is developing a high quality project we will have several mechanisms in place to prevent errors from entering our code. Each design we create for our project will be done by at least two group members working together. Once a design has been finished, it will be written up and presented to the members of the group who were not involved in creating it. If all four group members were involved in design then we will take a one or two day break from working on it and reassess the design. This will maximize the amount of people involved in project design and should help us design the bugs out of our code. The code review process will be as follows: Those who wrote the code will present the design, and those who did not code will provide a new perspective on the problem and raise any questions they have. Changes will be made to the document and the original design group will rewrite it. This altered design will be the one that we will work by.

Before implementation begins on a module of our project, we will document certain critical functionality that the finished portion must be able to do as well. As implementation progresses, the subset of our team responsible for implementing a portion will update as the state of the code changes.

Before each check-in, we will follow these steps to keep bugs to a minimum:

1. cvs update
2. Run all relevant test code (If the test code does not pass then you fix the problem or don't commit)
3. cvs commit

In the sad event that we do find an error we will track bugs in one of two ways. If we have a server available to run bugzilla then we will be using bugzilla to track bugs and all information regarding them. Otherwise, we will use a wiki to track bugs. More formally: Each bug created will be given an id, this id will serve as a link to the bug's page. On a bug page we will track comments, changes related to the bug, and when the bug becomes resolved.

Each week we will have a 15 minute "stand-up" meeting to update each other on any problems that were not foreseen and if we are running on schedule. Ten days prior to each turnin we will gather and demo the product for ourselves paying special attention to the critical functionality we decided upon in the pre-implementation stages.

5.3 Testing

See Test Plan Document for testing guidelines.

5.4 Deployment

Our deployment process will consist of distribution of a .exe and (if needed) an archive of program resources. This is subject to change.

5.5 Integration

Integration will be handled by assigning milestones to the union of subprojects. Minimum time allocated will be two days for integration of two subprojects. Since our app will not be interacting with any outside devices all integration will be within the project.

5.6 Procurement

All materials needed for development are available to us via MSDNAA or are open source.

5.7 Operations

Not applicable.

5.8 Maintenance

Not applicable.

5.9 Staff Development

During the milestones of requirement gathering and design our team will be responsible for learning various portions of C# relevant to their portion of the project. Key areas are: OpenGL, UI design, and basic C#.

5.10 Product Acceptance

Not applicable.