FORNUTS

FORest NUTrients Simulation

Software Design Document

Version 1.3

Team Members:
Karl Ott
Matt Van Veldhuizen
Syler Clayton
03/23/2013

Table of Contents

1. Introduction

1.1 Goals and Objectives

The purpose of this document is to provide the design of the FORNUTS program. This document will be used as a blueprint for the implementation of the program. This document is intended for developers implementing the program.

1.2 Statement of Scope

The scope of the program is to provide the original functionality of the FORNUTS program in a C++ port. Functionality will also be added to the program as well as adding a GUI to make the user experience easier. C++ was chosen for a number of reasons. Our client requested an executable for the program, so this rules out most interpreted languages. C++ is also the language most familiar to the developers so this would reduce the time as new languages would not need to be learned. Lastly most desktop based cross platform freely available graphical toolkits are implemented in C or C++, this lets our code natively use these toolkits.

1.3 Software Context

This program will be used primarily by students in the NRM 380 Soils Lab. The program could also be used by anyone if it fits their needs.

1.4 Major Constraints

The program will

1.4.1 Support the original functionality of FORNUTS
1.4.2 Extend the functionality in the port of the code
1.4.3 Add a GUI to the program for usability

2. Data Design

The program will be hosted in binary form on the corresponding Google code page. The source code will also be available on the same page.

2.1 Internal Software Data Structure

The internals of the program only need to use primitives that are in C++, STL and any structures used in the Qt Framework.

2.2 Temporary Data Structure

The program will be able to use files for input and output. The files are stored in ini format. Ini was chosen because of the relative simplicity of it. Another deciding factor was that ini is more human readable compared to other formats that were looked at.

3. Architectural and Component-Level Design

3.1 Program Structure

The program is broken up into three main components. The core component is shared by both interfaces, command line and GUI. The core component is responsible for carrying out the simulations. With the GUI and command line supplying the interface to the core component.

3.1.1 Architecture Diagram

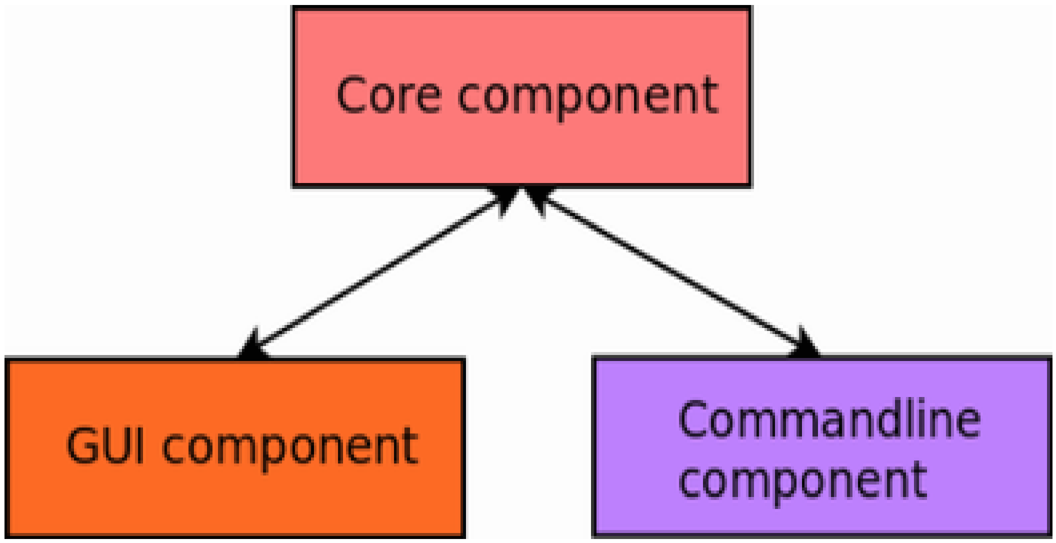


Figure 1: Component Diagram

3.2 Description of core component

The core component receives input from either interface and carries out the simulation. A detailed diagram of the simulation is pictured below:

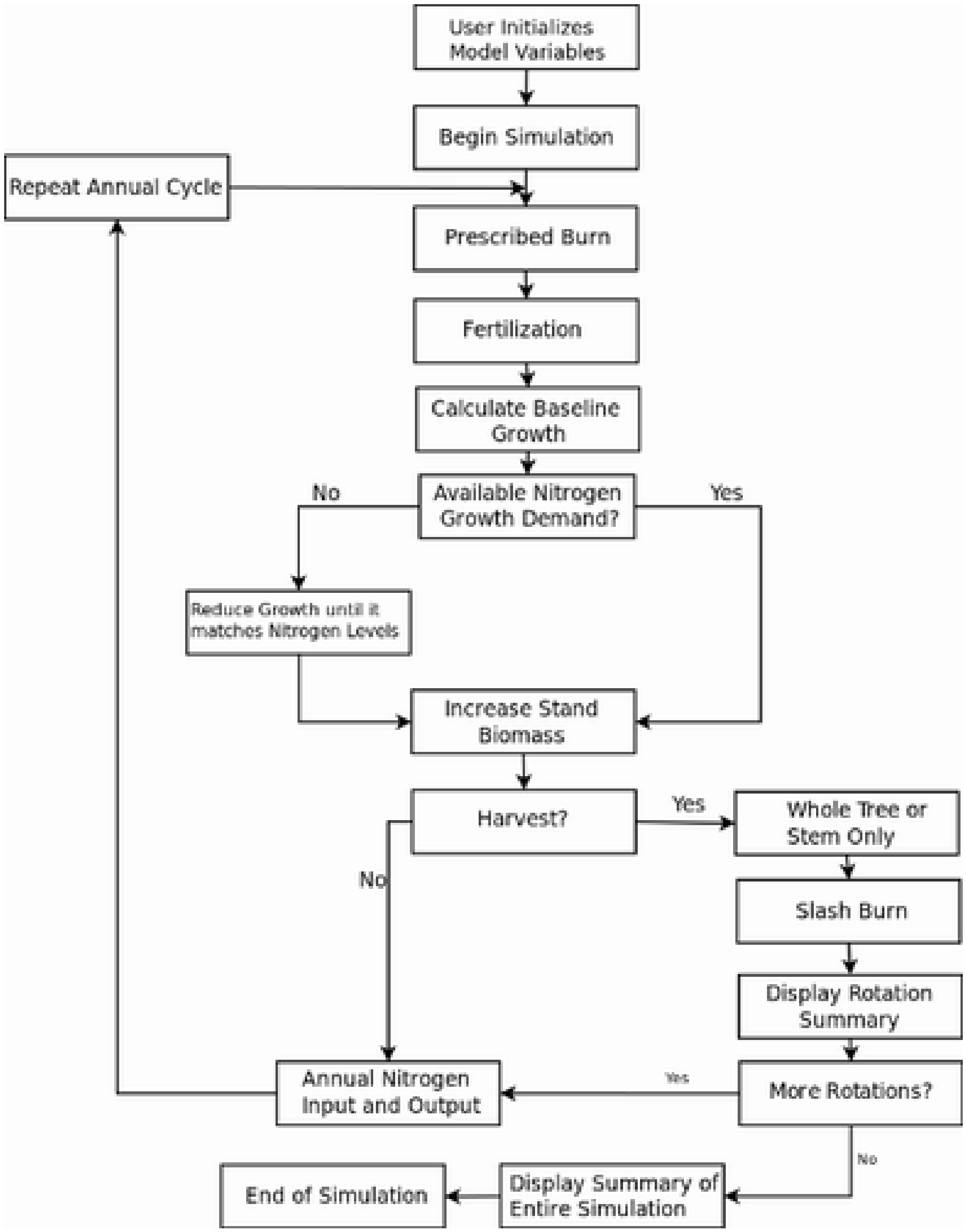
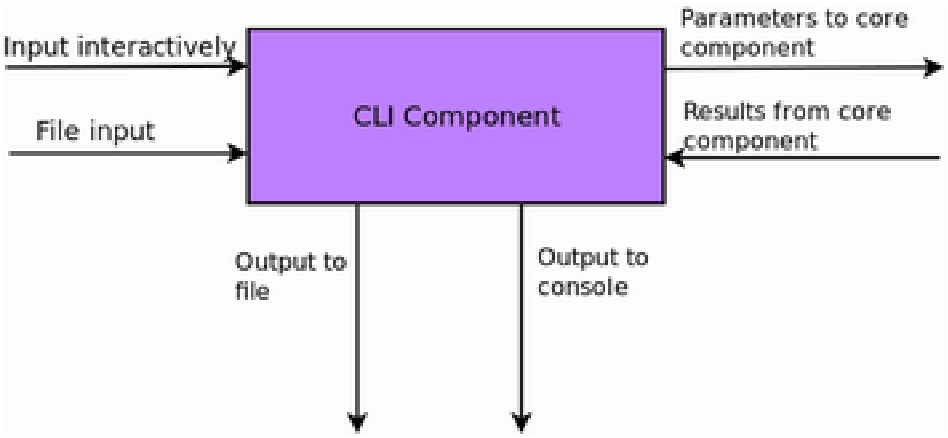


Figure 2: Simulation Flow

This component was spilt out from the other components in an effort to reduce redundant code. As the component is common between both interfaces this means that a change in the code here will result in changes in both interfaces.

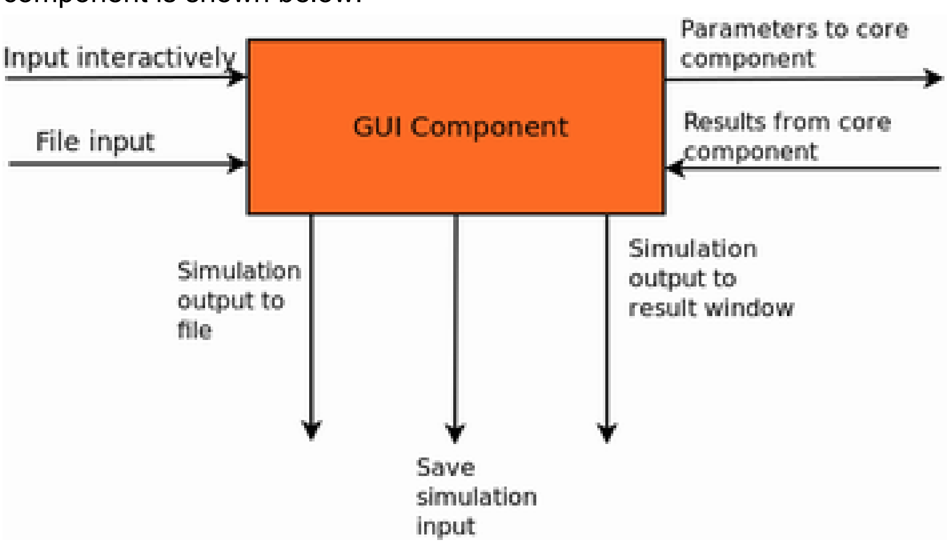
3.3 Description of Command Line Interface Component

The command line component aims to provide a similar interface to that of the original FORNUTS program. This will also have a number of improvements. These improvements are, but not limited to: file based input and output, refined interactive input, and statistical data from the simulation output in the form of a Comma Separated Value (CSV) file. A diagram showing the operation of the command line interface is shown below:



3.4 Description of GUI component

The GUI component of the program aims to provide an easier user experience than that of the command line interface. The GUI will get user input and then send it to the core component so the simulation can be run. The settings the user provides can be saved to a formatted ini file that can be used later in either command line interface, or loaded up later in the GUI interface. The GUI will also allow the user to save the output of the simulation into a text file that the user can then use for whatever purpose. The GUI will also display the statistical graphs that the simulation generates. A diagram of the GUI component is shown below:



The GUI will be made with Qt, which is an open source, cross platform, and freely available graphical framework. Qt was chosen because it is implemented in C++ so it can natively interact with our code. In addition it allows the program's GUI to be implemented once and run on any platform where Qt is available. This means that multiple GUI implements do not have to be written for each platform, ultimately leading to a lower integration time.

4. User Interface Design

4.1 Description of the User Interface

There is the command line interface is textual and provides an interactive mode and file mode. The file mode uses files are for input and output. The input files are the ini file format. The output in this mode can be specified to print to the console or saved to a file. The interactive mode will prompt the user for input in the form of questions. After the simulation has ran the user can choose to save the inputs entered to an ini file.The other half of the interface is the GUI. The interface will be split into two separate windows.

The first window will display all of the input variables the user can change. And the second window will have the results of the simulation. The user will set the inputs either by manually entering each input variable in or by loading the input variables from a file. Each time a variable is changed the results window will be updated and refreshed to show the simulation results. The GUI will also have a menu bar which will be located at the top of the input window on both the Windows and Linux version, where the Mac version will be located on the menu bar. There will be two menu's the user can use. The first menu called File will have the options to open an ini file, save the output, save the graphs, and exit program. The second menu is called Help, which will have options to view the about window, and load several examples to view. The about window will display information about the program, such as the version licence information, website and program name. The last part of the GUI interface are pop up warning windows. These are used when the user tries to input bad numbers, or if some of the numbers are in conflict. A warning pop up window will open and display a warning message to the user to fix the input numbers.

4.1.1 Screen Images

4.1.1.1 Windows 7

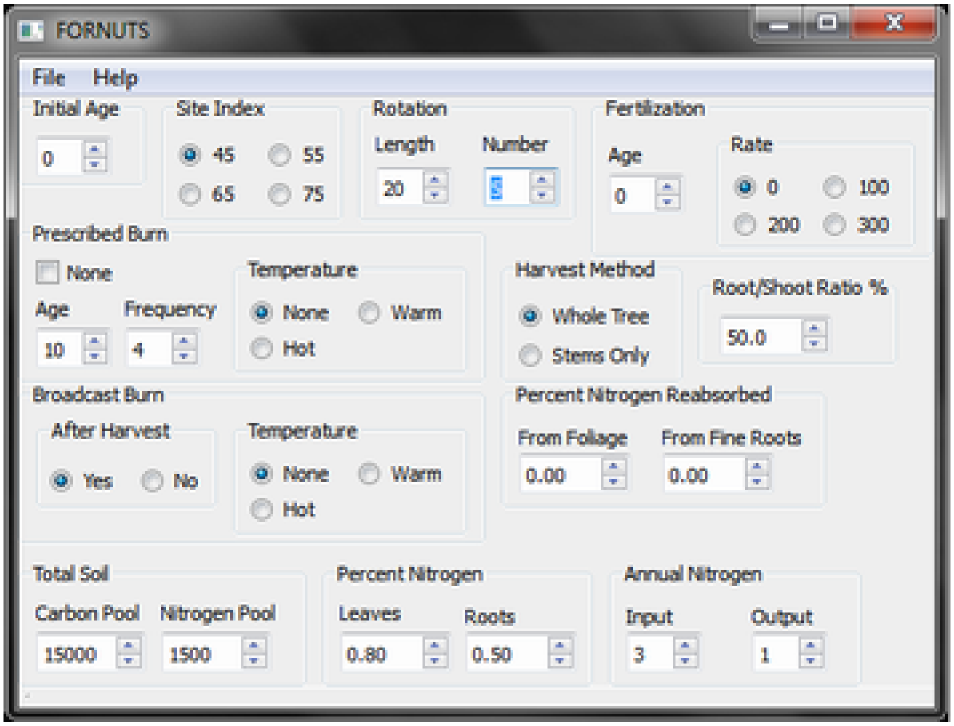


Figure 3: Windows Input Window

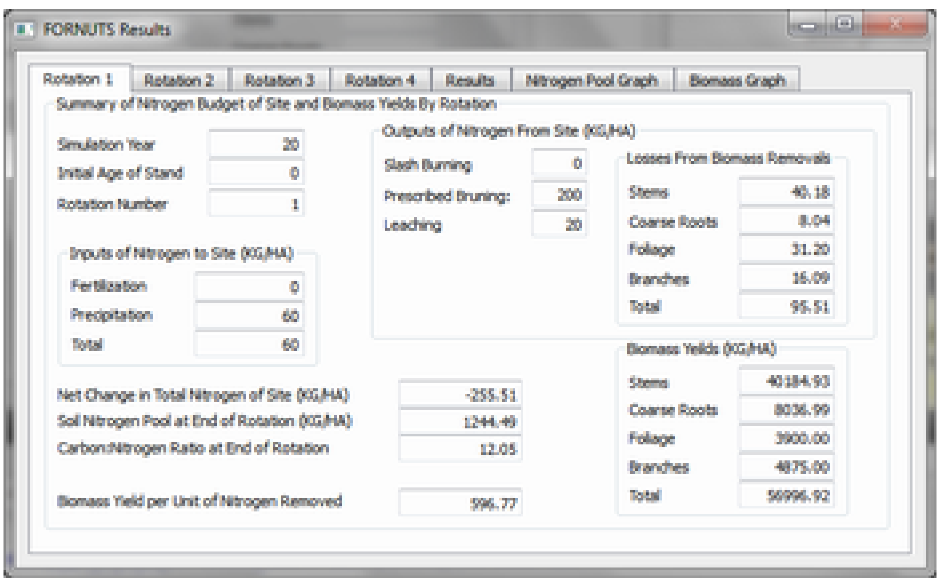


Figure 4: Windows Window

4.1.1.2 MAC OS 10.8

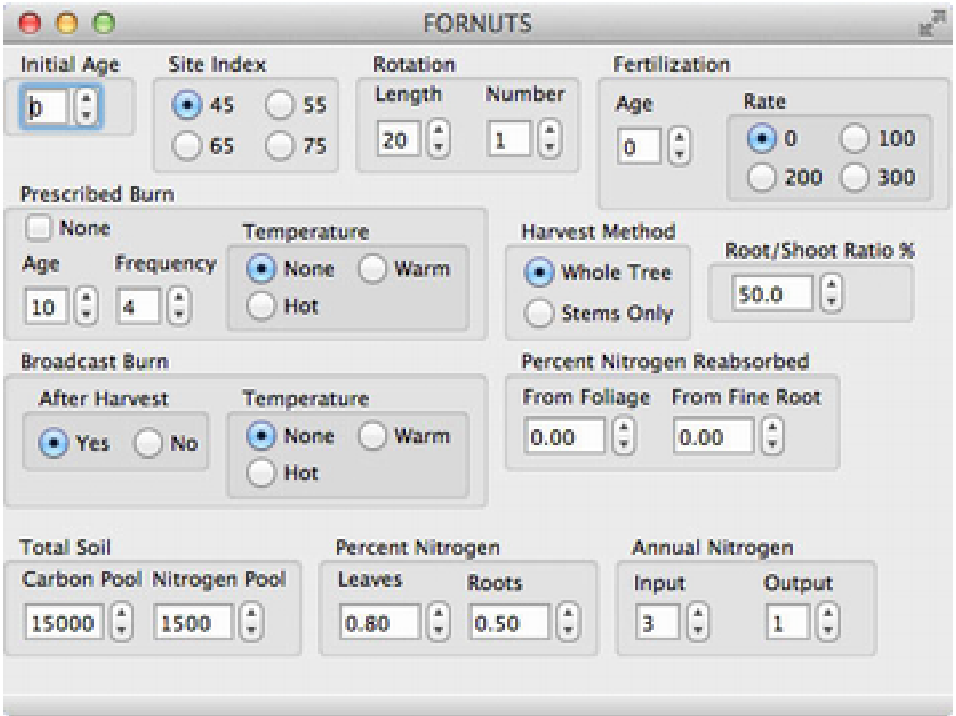


Figure 5: MAC Input Window

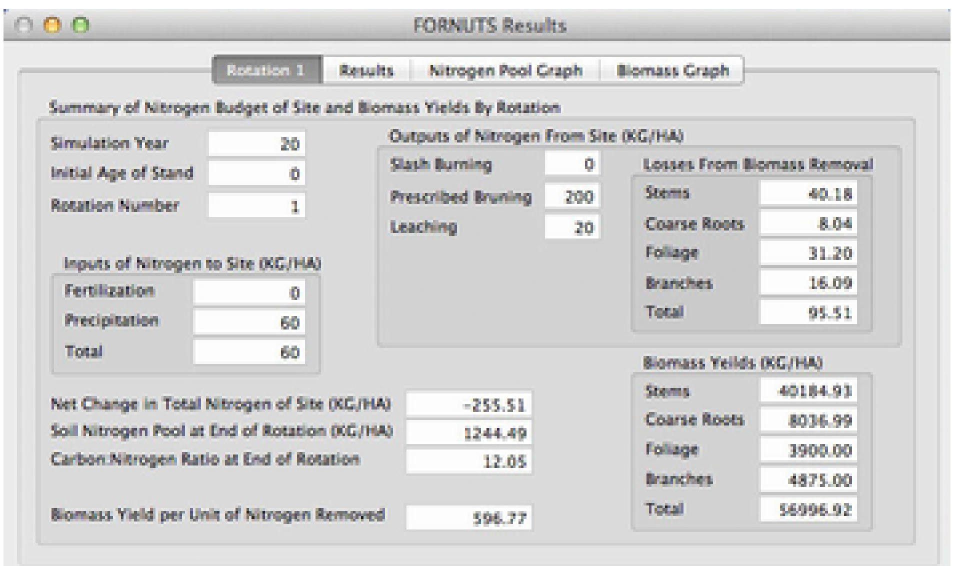


Figure 6: MAC Output Window

4.1.1.3 Linux (Ubuntu)

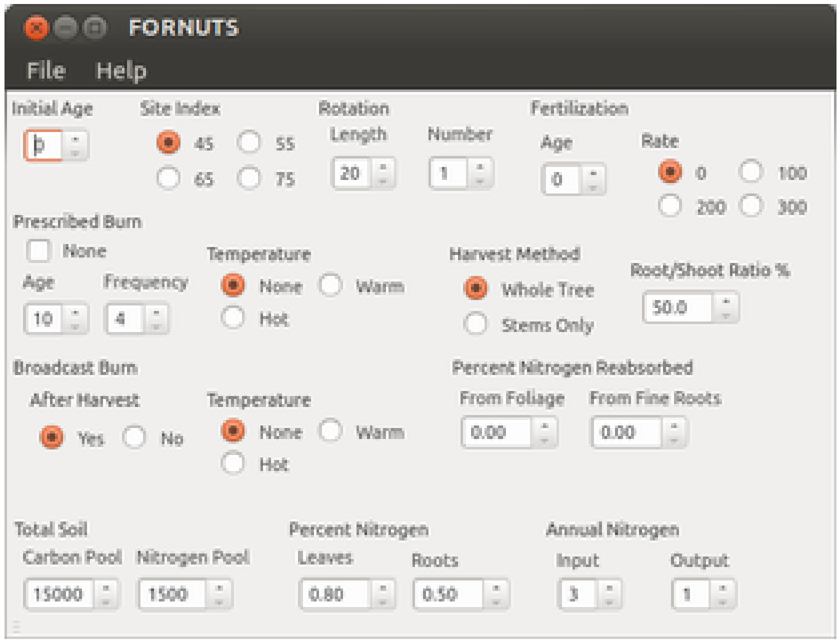


Figure 7: Linux Input Window

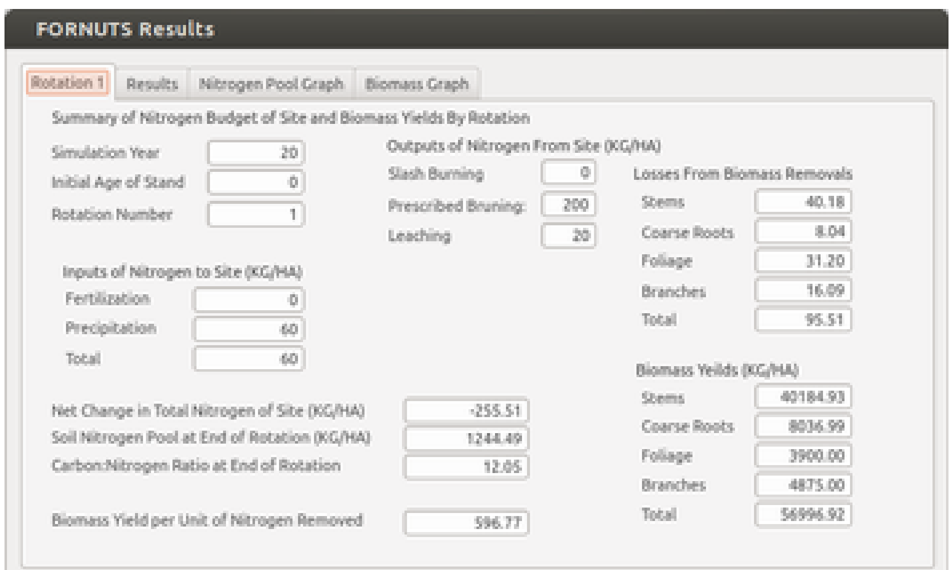


Figure 8: Linux Output Window

4.1.1.4 About Windows and Error Windows

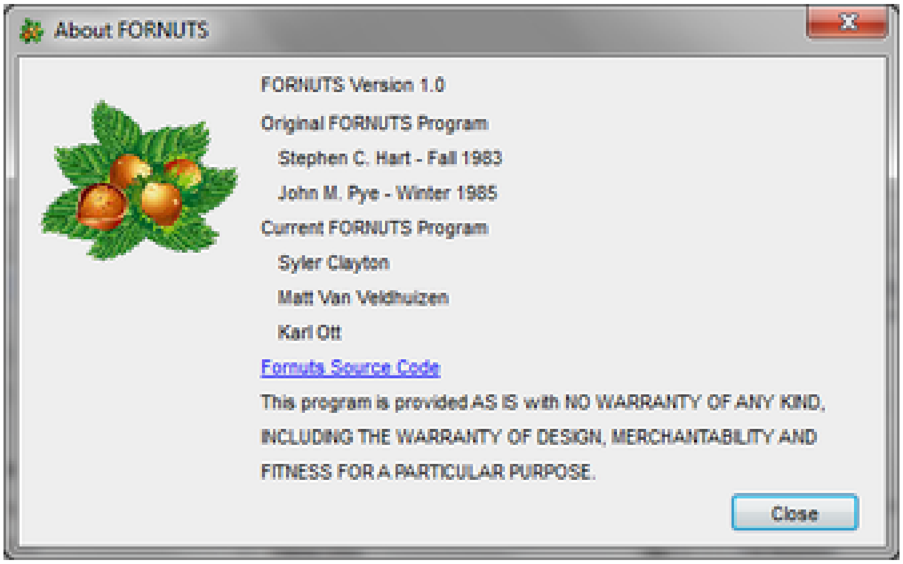


Figure 9: Sample About Windows

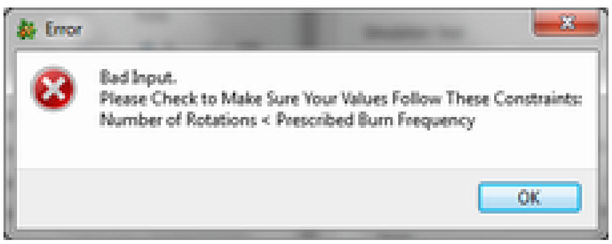


Figure 10: Sample Error Windows

4.2 Interface Design Rules

The font used was Droid Sans with a font size of 11 for all text used in the GUI component.

4.3 Components Available

There are three components available in the program. The command line component, the GUI component and the core component. The command line and the GUI component both use the core component to run the simulations.

5. Restrictions, Limitations, and Constraints

At this time there are no specific restrictions, limitations, or constraints.

6.0 Testing Issues

Testing will be split up for each component, the Command Line tests, and the GUI tests. Each test will verify that the core of the program is working as expected.

6.1 Testing Tools for Command Line

6.1.1 Scripts to Test Input and Output

A series of scripts will be made to test the extreme values of every input, checking to see if all values are within the acceptable ranges. With these input tests, the output can also be checked to see that it is correctly displaying the information. This test will also verify that the input from file and output to file options are working correctly.

6.2 Testing Tools for GUI

6.2.1 Testing Input Ranges and Input from File

This test will check that the ranges of inputs will correctly report errors, and test that the input is valid. The will be done manually by having pre-generated input files that will have the minimum and maximum numbers for each input. Because the input ranges will be checked in this manner the input from file will also be tested.

6.2.3 Test Save Output and Graphs

This test will check to make sure that saving the output to a file works as expected. And this test will check to make sure that saving the graph as an image works as expected.

6.3 Test Core Code

A series of scripts will be made to verify that the simulation was ran and that the output matches the given input. These scripts will use specific input that will generate specific output,

6.4 Testing Tools for Installation

Test installer on several machines, including different operating system versions.

A. Appendix

A.1 Revision History

Date	Version	Notes

02/07/13	1.0	Setup basic outline for document
3/23/13	1.1	Rough draft
4/10/13	1.2	Added images and Test Cases
4/16/13	1.3	Added diagrams