University of Edinburgh

Programming Skills

Development Coursework

Programming Skills Report

Authors:
Jack Frankland

Denitsa Bankova

October 30, 2017

Abstract

This report forms part of the documentation for the coursework. It accompanies a reference manual which provides information on the class structures and methods, as well as the comments in the source code, which provide the implementation details. This document is designed to contain more general information about the code, how to run it and any important design decisions.

1 Introduction

In what follows the basic information about the project implementation is given and instructions are provided for building, running and testing the code. Finally, key design decisions are discussed and justified.

2 General Information

2.1 Programming Language

The project source code and tests are written in C++. Due to the fact that it uses move semantics; the source code must be compiled under the C++11 (or later) standard, which is when move semantics were introduced.

2.2 Version Control

The version control system used for this project was Git with a master repository hosted on Github. Both team members worked from the command line, using the git commands to pull, add, commit changes and push to the repository hosted on Github. The Github repository can be found here: `https://github.com/FranklandJack/2DPredPreyModel`.

2.3 Debuggers

The GNU GDB debugger was used for debugging from the command line and memory leak checks were performed with valgrind.

2.4 Build Tools

A makefile was used with the make command for automated building. The main functionality of the makefile compiles and links the source code, but it also includes commands to generate input files for the code from pnm files using the tool provided in the assignment, and various utility commands such as cleaning up all auto-generated files. There is a second configuration makefile which is included in the primary makefile to take care of any automated building for the converter tool that was provided in the assignment.

2.5 Test Tools

Unit tests for the program were written using the CppUnit test framework. For continuous integration Travis CI was used, meaning that each time a change was committed to the repository hosted on Github, the code was automatically built and tested by Travis CI using our unit tests. The Travis CI repository can be found here: `https://travis-ci.org/FranklandJack/2DPredPreyModel`

2.6 Documentation

Doxygen was used as a documentation tool for the source code. A doxyfile is present in the main directory which specifies preferences for the documentation. Using the doxygen commands it is then possible to generate a reference manual(in latex) and on-line documentation browser(in HTML) from the special comments in the source code header files.

3 Usage Instructions

3.1 Building

To build the executable run:

```
$ make
```

From the main directory. This will generate the object files and place them in the main directory, as well as linking them to create an executable called PredPrey.

3.2 Running

To run the code run:

```
$ ./PredPrey YOURINPUTLANDSCAPE.dat
```

where the first command line argument YOURINPUTLANDSCAPE.dat is the name of the .dat file containing the landscape you wish to simulate.

3.3 Testing

To run the unit tests on the code: !!!!!!!!!!!!!!!!!

3.4 Generating Documentation

To generate the documentation run:

```
$ doxygen Doxyfile
```

The generated files will then be outputted to the documentation directory, which in tern contains two directories; latex and html. To generate the reference manual, move into the latex directory

```
$ mv documentation/latex
```

and run:

```
$ make
```

which will generate a refman.pdf file that contains the class documentation. To open the on-line code browser move to the html directory:

```
$ mv documentation/html
```

and open the index.html file with a web browser such as Chrome. (Note: for ease of access the reference manual and on-line code browser files have been copied into the main directory, since they will not need regenerating after submission.)

3.5 Misc.

To generate .dat files from any pnm files, first place the pnm files in the landscape directory then run:

```
$ make dats
```

This will generate .dat files for all .pnm files in the landscape folder and place them in the main directory, ready to be run with the program.

To clean up all auto-generated files (this includes any output files from the program after it is run, object files and executables, as well as .dat files in the main directory) run:

```
$ make clean
```

For a full list of make commands and the function run:

```
$ make help
```

4 Summary of Key Design Decisions