Developing a Predator Prey Simulation

Programming Skills

Diamantis Dakaris, Ewen Lawson Gillies, Michal Kawalec, Claude Schmidt
University of Edinburgh, UK

Something here?

7th of November 2013

Abstract

The aim of this project is to develop the essential programming skills through a group project

Contents

1 Introduction **1**

2 Group Coordination **1**
2.1 Assignment of tasks to group members 1
2.2 Coordination 2

3 Design **2**
3.1 Programming Language 2
3.2 Revision Control 2
3.3 Build tools 2
3.4 Testing 2
3.5 Debugging 2
3.6 Testing 2

4 Conclusion **2**
4.1 Further Works 3

1 Introduction

The focus of this project was to apply the skills acquired in class to develop a Predator-Prey simulation in a group setting. In order to do so, the project members started with a rough outline of the program. From here, development tasks were assigned to each member. `git` was used to track the progress of the project. Once a working simulation was written in `C++`, it was thoroughly tested, debugged, and improved. The resulting simulation includes several output types for different simulation programs.

2 Group Coordination

As with any software development, the planning played a big role in the final product. Through regular meetings, emails, task management websites, and revision control repositories, plans were set and efforts were coordinated to ensure an efficient and productive group effort.

2.1 Assignment of tasks to group members

In order to increase efficiency and productivity, the division of labour was determined very early on. This division was based primarily on the existing skills set of each member. Each member was mandated to some of the coding. The more experienced coders edited the work of the less experienced once in a hierarchal order. The most experienced then explained his reasoning to the rest of the group, giving them an opportunity for further input into the final code. This ensured both an optimal simulation, coding experience, and supportive feedback for each member. The breakdown of the tasks and responsibilities for each member in order of coding previous experience in `C++` are as follows.

Michal Kawalec: Due to his experience in revision control and software development, Michal set up the structure of the simulation and submitted this structure to a repository on `https://www.github.com/?`. This structure included the header files and source files appropriately formatted with empty functions. Michal also conducted all final edits, adapting the other members ideas to the optimal syntax. (OTHER STUFF)

Claude Schmidt: WIth a strong working understanding of `C++`, Claude was tasked with writing the first draft of the constructor, as well as (OTHER STUFF). He also provided the group with the structure and planning needed for optimal time management.

Ewen Gillies: Given his experience in numerical solutions to partial differential equations, Ewen was tasked with writing the integrating function and dealing with the boundary cells. Since he lacked experience in `C++`, any additional structural ideas ideas were relayed to the more experience coders for optimal syntax. He then reviewed this code to ensure all ideas were properly expressed. Additionally, he was tasked with compiling the group report.

Diamantis Dakaris: Using his strong mathematical background, Diamantis conducted the preliminary analysis of the equation at hand, as well as the intermediate and final analysis of the output. He also wrote the initial output function, which was then extrapolated to the final one. As a relatively inexperienced coder, Diamantis was able to offer effective continual feedback on readability and the logical structure of the code, ensuring

that all levels of coders would be able to read code and reproduce the results. He also generated several of the input land maps for the simulation.

2.2 Coordination

The main method of group coordination and planning was through regular meetings. Full group meetings were held at least once per week, where new ideas and current progress could be discussed. During these meetings, smaller meetings between two or three members were organized where code would be peer edited or written by several members at once.

The members of the group agreed to use Trello[1] in order to ensure flexibility in task assignment and adaptability in the testing and debugging phases. This tool consists of four communal lists that each group can edit. These lists include To Do, Review, Doing, and Done. By adding items to the to do list, shifting them to doing, and placing them in done, group efforts were easily and effective coordinated without any overlap.

3 Design

Description of your design.

3.1 Programming Language

Description of the programming language you used and how useful it was.

3.2 Revision Control

Revision control Description of the revision control you used, how you used it and how useful it was.

3.3 Build tools

Description of the build tools you used and your views on the strengths and weaknesses of these.

3.4 Testing

Description of what testing you did and any test frameworks you used.

3.5 Debugging

Description of how you did any debugging and any tools you used.

3.6 Testing

Performance tests and analysis Description on the performance experiments you did and an analysis of the results of these.

4 Conclusion

Conclusions Some brief conclusions.

[1]This tool is found at: `https://www.trello.com`

4.1 Further Works

Ideas for further work, or what you would have done if you had had more time.