6CCS3PRJ Final Year Individual Project Report Title

Final Project Report

Author: Your Name

Supervisor: Name

Student ID: 000000

February 13, 2023

Abstract

Procedural generation refers to content in a medium that is produced algorithmically in lieu of by hand. Most notably, procedural generation algorithms are implemented in video games, for generating levels, terrain and other game contents programmatically. This project takes some of the more prominent algorithms for procedural generation- Lindenmeyer Systems, Voronoi Points, Poisson Disk Generation and Simplex Noise- and implements them in a 3D walking simulator in the open-source Godot game engine, and compares their workings and performance. My aim with this project is to (1) increase my knowledge of procedural generation in games beyond the surface level, by going in-depth into some of the algorithms that are used, and (2) use this knowledge to implement said algorithms in a 3D walking simulator scenario in Godot, then compare how each algorithm works and performs.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Your Name

February 13, 2023

Acknowledgements

It is usual to thank those individuals who have provided particularly useful assistance, technical or otherwise, during
your project. Your supervisor will obviously be pleased to be acknowledged as he or she will have invested quite a lot of time overseeing your progress.
Thanks to my supervisor Senir Dinar, for providing me the guidance I so badly needed
to make this project not only the best for my mark, but also one that I enjoy.

Contents

1 Introduction **2**
1.1 Report Structure 2

2 Background **3**
2.1 Section Heading 3

3 Report Body **4**
3.1 Section Heading 4

4 Design & Specification **5**
4.1 Section Heading 5

5 Implementation **6**
5.1 Section Heading 6

6 Legal, Social, Ethical and Professional Issues **7**
6.1 Section Heading 7

7 Results/Evaluation **8**
7.1 Software Testing 8
7.2 Section Heading 8

8 Conclusion and Future Work **9**
Bibliography 10

A Extra Information **11**
A.1 Tables, proofs, graphs, test cases, ... 11

B User Guide **12**
B.1 Instructions 12

C Source Code **13**
C.1 Instructions 13

Chapter 1

Introduction

This is one of the most important components of the report. It should begin with a clear statement of what the project is about so that the nature and scope of the project can be understood by a lay reader. It should summarise everything that you set out to achieve, provide a clear summary of the project's background and relevance to other work, and give pointers to the remaining sections of the report, which will contain the bulk of the technical material.

1.1 Report Structure

Chapter 2

Background

The background should set the project into context by motivating the subject matter and relating it to existing published work. The background will include a critical evaluation of the existing literature in the area in which your project work is based and should lead the reader to understand how your work is motivated by and related to existing work.

2.1 Section Heading

Chapter 3

Report Body

The central part of the report usually consists of three or four chapters detailing the technical work undertaken during the project. **The structure of these chapters is highly project dependent**. They can reflect the chronological development of the project, e.g. design, implementation, experimentation, optimisation, evaluation, etc (although this is not always the best approach). However you choose to structure this part of the report, you should make it clear how you arrived at your chosen approach in preference to other alternatives. In terms of the software that you produce, you should describe and justify the design of your programs at some high level, e.g. using OMT, Z, VDL, etc., and you should document any interesting problems with, or features of, your implementation. Integration and testing are also important to discuss in some cases. You may include fragments of your source code in the main body of the report to illustrate points; the full source code is included in an appendix to your written report.

3.1 Section Heading

3.1.1 Subsection Heading

Chapter 4

Design & Specification

4.1 Section Heading

Chapter 5

Implementation

5.1 Section Heading

Chapter 6

Legal, Social, Ethical and Professional Issues

Your report should include a chapter with a reasoned discussion about legal, social ethical and professional issues within the context of your project problem. You should also demonstrate that you are aware of the regulations governing your project area and the Code of Conduct & Code of Good Practice issued by the British Computer Society, and that you have applied their principles, where appropriate, as you carried out your project.

6.1 Section Heading

Chapter 7

Results/Evaluation

7.1 Software Testing

7.2 Section Heading

Chapter 8

Conclusion and Future Work

The project's conclusions should list the key things that have been learnt as a consequence of engaging in your project work. For example, "The use of overloading in C++ provides a very elegant mechanism for transparent parallelisation of sequential programs", or "The overheads of linear-time n-body algorithms makes them computationally less efficient than $O(n \log n)$ algorithms for systems with less than 100000 particles". Avoid tedious personal reflections like "I learned a lot about C++ programming...", or "Simulating colliding galaxies can be real fun...". It is common to finish the report by listing ways in which the project can be taken further. This might, for example, be a plan for turning a piece of software or hardware into a marketable product, or a set of ideas for possibly turning your project into an MPhil or PhD.

References

Appendix A

Extra Information

A.1 Tables, proofs, graphs, test cases, ...

The appendices contain information that is peripheral to the main body of the report. Information typically included in the Appendix are things like tables, proofs, graphs, test cases or any other material that would break up the theme of the text if it appeared in the body of the report. It is necessary to include your source code listings in an appendix that is separate from the body of your written report (see the information on Program Listings below).

Appendix B

User Guide

B.1 Instructions

You must provide an adequate user guide for your software. The guide should provide easily understood instructions on how to use your software. A particularly useful approach is to treat the user guide as a walk-through of a typical session, or set of sessions, which collectively display all of the features of your package. Technical details of how the package works are rarely required. Keep the guide concise and simple. The extensive use of diagrams, illustrating the package in action, can often be particularly helpful. The user guide is sometimes included as a chapter in the main body of the report, but is often better included in an appendix to the main report.

Appendix C

Source Code

C.1 Instructions

Complete source code listings must be submitted as an appendix to the report. The project source codes are usually spread out over several files/units. You should try to help the reader to navigate through your source code by providing a "table of contents" (titles of these files/units and one line descriptions). The first page of the program listings folder must contain the following statement certifying the work as your own: "I verify that I am the sole author of the programs contained in this folder, except where explicitly stated to the contrary". Your (typed) signature and the date should follow this statement.

All work on programs must stop once the code is submitted to KEATS. You are required to keep safely several copies of this version of the program and you must use one of these copies in the project examination. Your examiners may ask to see the last-modified dates of your program files, and may ask you to demonstrate that the program files you use in the project examination are identical to the program files you have uploaded to KEATS. Any attempt to demonstrate code that is not included in your submitted source listings is an attempt to cheat; any such attempt will be reported to the KCL Misconduct Committee.

You may find it easier to firstly generate a PDF of your source code using a text editor and then merge it to the end of your report. There are many free tools available that allow you to merge PDF files.