Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

SCHOOL OF COMPUTER SCIENCE AND STATISTICS

TOWARD A FUTURISTIC EMERALD ISLE

CHRIS CASEY

BINH-SON HUA
DECEMBER 28, 2024

Adapted from a template created by Prof. Michael Brady

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
COMPUTER ENGINEERING

Declaration

I hereby declare that this CG Project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed: Chris Casey

Date: 27/12/24

Introduction

In this project OpenGL 3.3 and C++ were used to simulate an infinite scene with a theme: "Towards and Futuristic Emerald Isle." Features from all four labs were required to be implemented. They included: geometry rendering, texture mapping, lighting and shadows and finally animation.

Some other considerations for the scene were that the frame rate must not fall below 15 FPS, allow user control of camera and implement an advanced feature.

The deliverables for this project were:

- Source code and Git Repository

- Final report in LaTeX (here)

- Progress report showing screenshots (included here)

- MP4 video showing the rendering of the scene.

1 | Introduction - Chapter

1.1 Overview of Scene

1.1.1 Implementations

This paragraph denotes what was achieved in the final source code. I successfully rendered and textured buildings, roads and other objects for my scene. A SkyBox was put together and implemented correctly. Camera control and the infinite-effect were implemented. A point light source and lighting were correctly implemented. Animation is also included.]Several complex features were required to be implemented as detailed above. Unfortunately not all of these requirements were met. Below is a comprehensive list of what I did and what I struggled to do.

I successfully rendered and textured buildings, roads and other objects for my scene. A SkyBox was put together and implemented correctly. Camera control and the infinite-effect were implemented. A point light source and lighting were correctly implemented. Animation is also included.

1.1.2 Challenges

This paragraph denotes what I struggled to include in the final source code.

Several complex features were required to be implemented as detailed above. Unfortunately not all of these requirements were met. Below is a comprehensive list of what I did and what I struggled to do.

2 | Progress Report

GitHub was used for version control and mapping progress. Screenshots were taken along the way which show the scene through various stages of development.

A screenshot of the git history is shown:

Progress: Description

Screenshot 1: Geometry Rendering

Screenshot 2: Texturing and SkyBox

Screenshot 3: Custom SkyBox

Screenshot 4: Infinite Plane and Camera Control

Screenshot 5: Lighting and Animation

Screenshot 6: Final Render

Progress: Screenshots

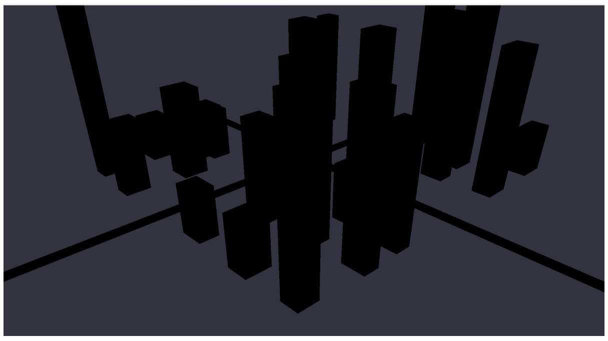

Figure 2.1: This screenshot shows successful rendering of buildings and roads in the scene.

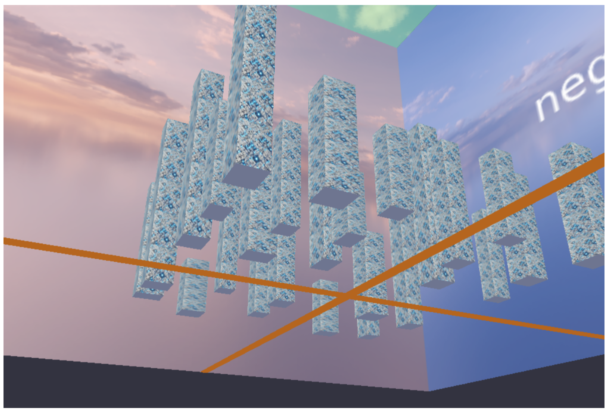

Figure 2.2: This screenshot shows texturing of objects (buildings and roads), and implementation of skybox debug from lab 2.

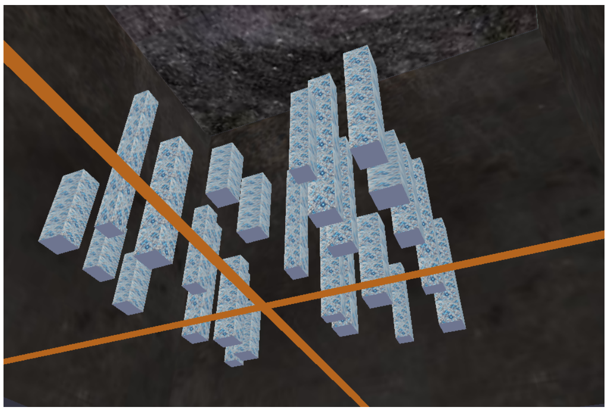

Figure 2.3: This screenshot shows a custom "cave themed" SkyBox. This was created using online resources like Adobe Image Editor and free-use google images.

3 | Discussion: Quality + Robustness

I primarily worked using a combination of different lab templates. I primarily used lab 2 and lab 4. However, this approach came with its limitations. Primarily, a lack of object-oriented code as well as being more application specific than I wanted. This lead to lengthy attempts and difficulty debugging.

As a result, though the code is easily understood upon visual inspection, additional elements required a significant amount of work in order to integrate with the existing scene. In this way, I was too dependent on the templates given and should have deviated my approach earlier.

I would focus on technical implementations at an early and revise my attempts to debug if I was to repeat the project. Inclusion of modular programming would likely make the project's completion far more effective. Also, I spent too much time 'gold-plating.'

The code includes error-handling much in the same way we did in class. File loading, shader implementation and null pointers are managed appropriately. All cleanup is done after rendering as required. Performance was consistently good, however was never properly challenged. The option to include more animation, more lighting, etc. is there. The scene consistently runs well over 15 FPS on my laptop.

4 | Conclusions and Future Work

Acknowledgements

You should acknowledge any help that you have received (for example from technical staff), or input provided by, for example, a company.

Contents