Final Project - Group 5

COMP371 – Computer Graphics

Procedural Forest

Mair Elbaz	40004558
Daniel Vigny-Pau	40034769
Noah Horowitz	40033784
Dania Kalomiris	40005674

Presented to

Kaustubha Mendhurwar

Sudhir Mudur

Winter 2020

Concordia University

TABLE OF CONTENTS

VIDEO DEMONSTRATION LINK:
https://youtu.be/mI86BRqBofA

GITHUB REPOSITORY LINK:
https://github.com/Nwitz/Procedural-Forest

1. Objective

The objective of this project was to create a graphics program using OpenGL for walking through a procedurally created virtual modeled world. In this case, we wanted to create a virtual forest terrain composed of multiple complex objects such as trees, bushes, grass and rocks. The terrain will be procedurally created. All repeating items, such as trees in the virtual forest, will be procedurally placed, using a few user specified parameters if necessary. The user can navigate through the world using the keyboard keys and mouse, as outlined in the README file.

Furthermore, the goal of this project is to enhance our understanding of computer graphics principles and programming skills in OpenGL, as well as developing an application as a team.

2. Interest

This project gave us the opportunity to learn many new aspects about computer graphics that we previously did not know about. Seeing as prior to this course and project, most of us had never used OpenGL or done 3D modeling before, this project was an opportunity to create our first virtual 3D modeled world. Learning OpenGL can be quite time consuming to learn, but working as a team made that process much easier and allowed us to divide up the workload and work more efficiently than we would have had we done the project solo. Some of our team members also had an interest in video game design and designing 3D models, so this project also offered an insight into that domain, notably building a 3D world and its models from the ground up, as well as the code and logic behind it.

3. Implementation

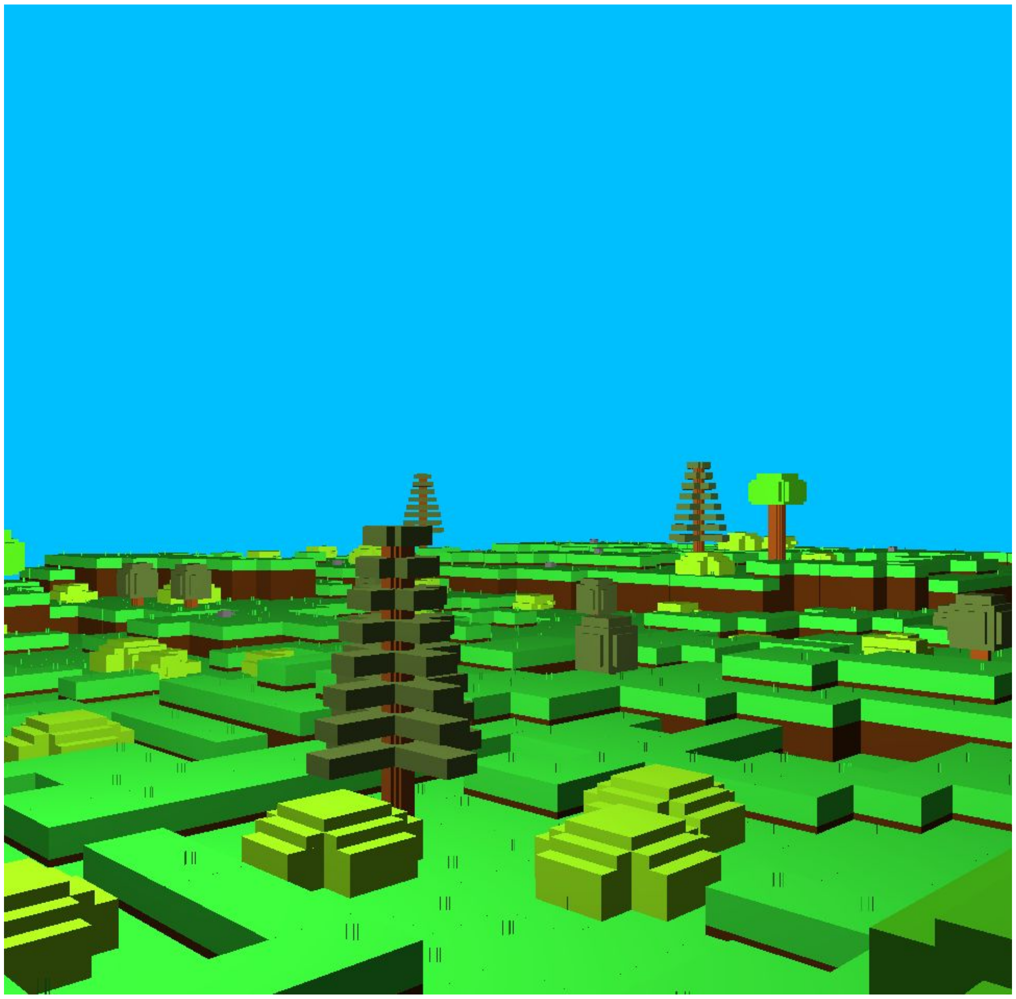
We created a world with randomly generated nature-themed terrain, grass and rock detailing, and repeated objects such as trees and bushes. The terrain varies in height, while the tree and bush objects vary in size, colour, and shape. The terrain is procedurally generated and the models are procedurally placed.

Terrain generation is done based on a simple algorithm. A user can define the desired length and width of the generated terrain as well as the number of possible levels which may exist. Since therrain is generated using cubes, elevation is expressed in levels rather than a gradual slope. Terrain is procedurally generated based on the following algorithm and rules.

First, the top most row, and left most column are generate, each block in the top row is created relative to the block to its left, (it may be at the same level, one level down, or one level up if allowed by the provided number of levels constraint), the same generates the left row except based on the block to its front. Once the left column and top row are generated the following rules generate each block in column major order starting at position (1, 1). The block to be generated checks the block to its left, and the block to its front, these are position (1, 0) and (0, 1) respectively for the starting position. If both left block and front block are the same height, the block to be generated may choose to be generated at that level, one above or one below, there is weighting here which favors staying at the same level. If the left block and front block are not at the same level, the block to be generated may only choose to be generated at either the left block's level, or the front block's level. This simple algorithm generates terrain with a specified number of levels but maintains large regions without changes in level to emulate a more realistic looking environment.

Along with the generate heightmap is a landscape object placement map. This map defines where landscape objects can be placed based on their size. The object map generator creates a 2D integer array the same size as the height map. Local neighborhoods in the heightmap are used to define flat terrain. A flat local neighborhood of 5 by 5 results in a valid location for a large object, and a flat local neighborhood of 3 by 3 results in a valid location for a medium object. The option is also available to disqualify any surrounding valid locations upon corresponding object placement to prevent crowding. These two maps are then used to create objects at the respective locations, further weighting can be used to tweak frequency of placement.

Generated Terrain



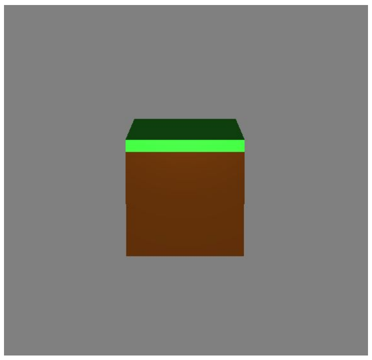
An instance of our procedurally generated terrain, including all of our models.

To create our various models, we initially created a *ComplexModel* object, which allowed us to create basic shaded and coloured cube objects. We then extended this object into separate child classes, such as *Tree1*, *Bush1*, *Grass*, *Rocks*, and so on, allowing us to hierarchically model each one and draw it in OpenGL on demand.

Each model is built using a model cube object, which is then repeated, translated, scaled and coloured to create different shapes and details.

Terrain

We created a basic *Terrain* cube as our base terrain model. It is composed of one cube for dirt and another for grass. This cube is repeated at randomly varying heights in order to create our terrain in our randomly generated landscape.

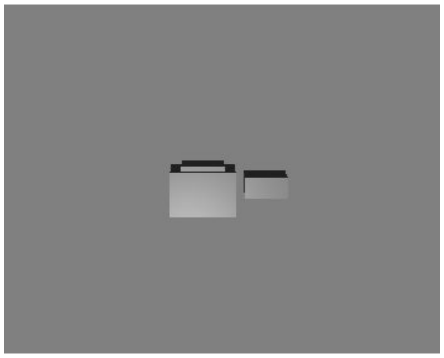


Terrain.cpp

Grass and rocks

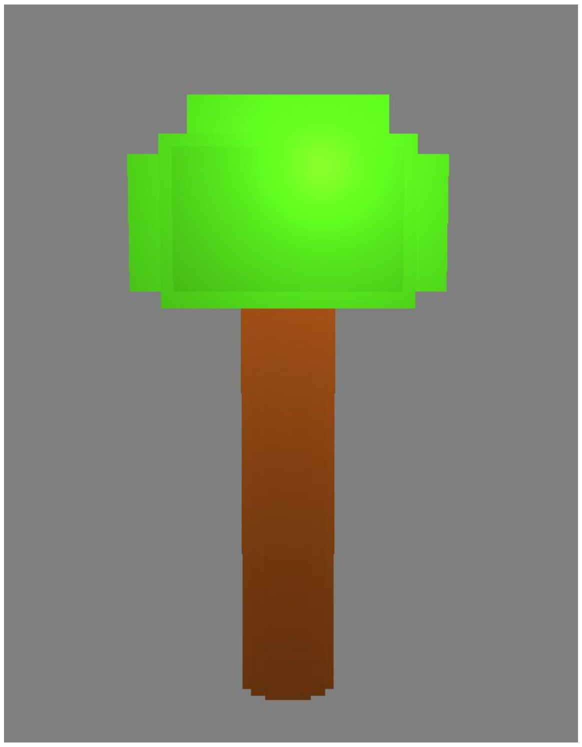
To add more variety to our terrain, we have also included smaller detail-oriented models, these being *Grass* and *Rocks*. These models are each made up of multiple instances of our initial cube. They are spread randomly throughout the landscape over the terrain to provide a bit more life and detail to the otherwise flat *Terrain* cube that we are generating throughout our nature landscape.

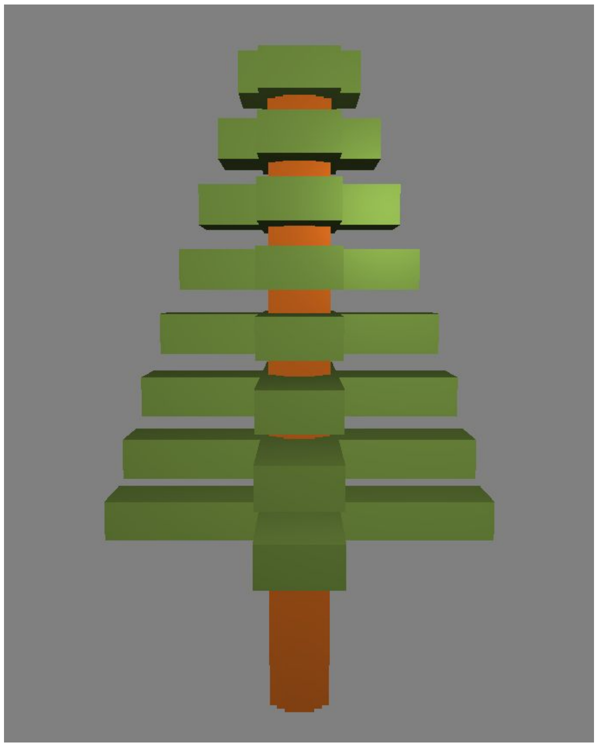
Grass.cpp


Rocks.cpp

Trees

An important part of our procedural forest landscape is, of course, the trees. We added two different tree models, *Tree1* and *Tree2*, each resembling a maple tree and a pine tree respectively. Each tree consists of 5 “trunk” cubes, the same initial cube as the other models, scaled and translated to simulate the trunk and additional cubes to simulate leaves. They are the largest models out of the ones we created.

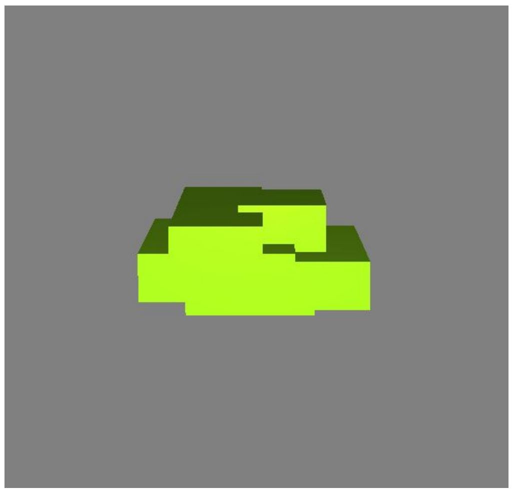

Tree1.cpp

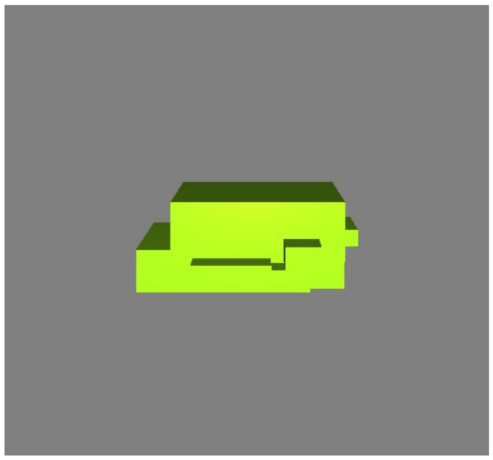

Tree2.cpp

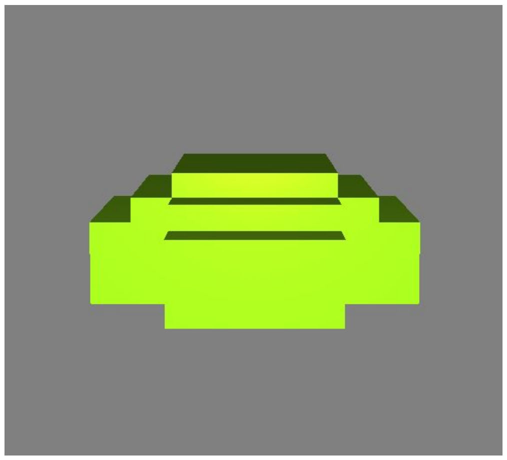
Bushes

We created 4 bush types, *Bush*, *Bush1*, *Bush2*, and *Bush3* respectively. These bushes are intended to be an in-between medium-sized model, bridging the gap between our small grass and rock models and our larger tree models. The different bush model types allowed us to create a terrain with more variety and achieve a more natural looking aesthetic.

Bush.cpp


Bush1.cpp


Bush2.cpp


Bush3.cpp

4. What We Learned

This project allowed us to dive deeper into the world of computer graphics and further understand OpenGL. We gained a better understanding of the inner workings of OpenGL by using our prior knowledge from the course and assignments to build an application with a more coherent design. We also learned about the process of creating and generating a procedural terrain. One concept in particular is the heightmap which was one of the crucial concepts needed to generate our terrain with different elevations.

Additionally, this experience gave us an opportunity to develop a computer graphics application by working as a team. Effective teamwork is needed in order to make any project succeed. Our excellent communication between every member of the group allowed us to give and receive feedback which improved our work. We also made use of version control by using Git which allowed us to identify different builds of the application and made it easier to track and maintain everyone’s code.

5. Resources & References

There are many tools we relied on throughout the project that have helped us.

1) The lab code provided to us by our TAs proved very useful in helping us further understand how OpenGL works. Every lab provides an example on how to do a specific function in OpenGL, with an additional explanation on what is happening throughout the code thanks to the generously commented code and slideshow guidelines . Our world movement, as well as the mouse movement particularly were based on the code given to us for labs 2 and 5. These labs gave us the resources on how to incorporate both a camera and an event manager to handle these different movements. Matrix transformations, as well as hierarchical modeling were learned in lectures and labs, which we then implemented into our project.

2) learnopengl.com was an incredibly helpful website that helped us understand how to use OpenGL properly. Their tutorials step by step tutorials proved to be very helpful in us understanding concepts we may not have fully understood in the labs, such as basic lighting, which we followed for this project.

“Basic Lighting,” *LearnOpenGL*. [Online]. Available: https://learnopengl.com/Lighting/Basic-Lighting. [Accessed: 19-Apr-2020].

3) We watched and followed some videos from “The Cherno”, a YouTube channel that uploads OpenGL tutorials regularly. The initial structure and organization of our project was inspired by these videos, notably the way we created models and tested them using a testing menu.

The Cherno, “OpenGL,” *YouTube*. [Online]. Available: https://www.youtube.com/playlist?list=PLlrATfBNZ98foTJPJ_Ev03o2oq3-GGOS2. [Accessed: 19-Apr-2020].