BROCK UNIVERSITY COMPUTER SCIENCE 3P98

Modern OpenGL and Procedural Terrain Generation Exploration

Tyler Kennedy Collins
tk11br@brocku.ca
4956637

Preston Engstrom
pe12nh@brocku.ca
5228549

1 INTRODUCTION

The main purpose of this project was to explore not only the use of modern OpenGL, (version 3.3) but to experiment with the process of developing a system which could procedurally generate terrain. An additional motivation was the ability to obtain experience working with third party libraries. Some examples from among these were glm, SDL, and SOIL. Included with our project is a small C# based application which we used to explore the boundaries of Perlin Noise in relation to terrain generation. An additional two avenues that we explored were model loading, and engine building. Luckily for us, our implementation of a physics engine was simply just collision with a surface. On the other hand, our model loading system allows for us to import complex meshed objects with different materials. This project thus takes on the form of a graphics application style game, with the ability to fly a model around some world that possesses completely procedurally generated terrain.

2 LIBRARIES

Below is an explanation of the function of the main libraries included in our project, their main uses, and why we chose them over the alternatives.

2.1 GLEW

GLEW stands for The OpenGL Extension Wrangler Library. The function of GLEW is to provide some sort of open source C++ extension loading dependency. This means that it would be GLEW's responsibility to determine if certain functions are available on the hardware running a project with GLEW. If a function were not supported - it is possible GLEW would throw an error. But in a better case, GLEW would be able to translate the impossible call into something the hardware could understand and then execute it properly. The reason we chose GLEW is its constant good reviews and simplicity as well as its ability to work on multiple operation systems.

2.2 SDL

The library we chose for creating windows and handling user input was the popular SDL framework. SDL stands for Simple DirectMedia Layer, and it aims to provide access to all sorts of low level input, and output. Again, keeping with theme, SDL was chosen for its ability to work natively with C, and as such work on multiple operating systems. Another factor in choosing SDL was the availability of documentation and code examples. When learning a new library, one should not have to go scouring the depth of forums to find your input output functions.[1]

2.3 GLM

In modern OpenGL a lot of functionality has been removed for the purpose of giving you more of an ability to customize behavior. One of the things lost was the ability to simply transform any sort of view space or other that you were using. GLM does not add this back in, instead it gives you all of the tools to compute vector and matrix math for rotations, translations, and scaling. Without GLM, you would have to implement all of these methods yourself. The reason we chose GLM was its availability as well as its functionality when creating new vector and matrix objects.

2.4 SOIL

SOIL stands for Simple OpenGL Image Library which suited our needs perfectly. The same way that FreeImage functioned in class - SOIL did for us. Meaning it allowed us to quickly and easily important images for our textures and skyboxes without any issues.

3 Architecture and Class Design

Early in our design process we decided to make the switch to modern OpenGL. Knowing that this would be much more work with having to learn things like GLSL, shader language, we designed a graphics application in the form of a game. This would allow us to explore different

[1] It should perhaps be noted that the GLFW library had its Oculus Rift guide on the front page - so at least it has that going for it.

portions of modern computer graphics in many directions. This section will highlight the most important classes and their functions, why they were designed the way they were, and how they interact with modern OpenGL.

3.1 SimulationGame

blah