Programming Assignment 1

Date of announcement: 2^{nd} May 2018
Submission deadline: 14^{th} May 2018

Description

This OpenGL programming assignment will provide an introduction to OpenGL programming. In particular, you will learn geometry information can be stored in the form of a map, how to load this information and display it as wireframe mesh by drawing triangles, interact with it, and also how to set up and manipulate the virtual camera to view the scene from different angles and distances.

A **mesh** is a set of polygons that share vertices and edges which describe the shape of a geometric object. Meshes can be simple consisting only of a few primitives, or very complex as shown below. In this homework we will be using triangular meshes for the display of height data.

Figure 1: Images credit: Blender Foundation

A **heightmap or heightfield** is a 2D array represented as a raster image which can be used to store the height at each pixel. They are very common in computer graphics applications since they provide an easy and efficient way to represent terrain. A heightmap can be thought of as a function f which given as input the coordinates for x and y it returns the value z contained in the image at location (x, y) i.e. $f(x, y) \rightarrow z$. An example of a heightmap is shown in the image below.

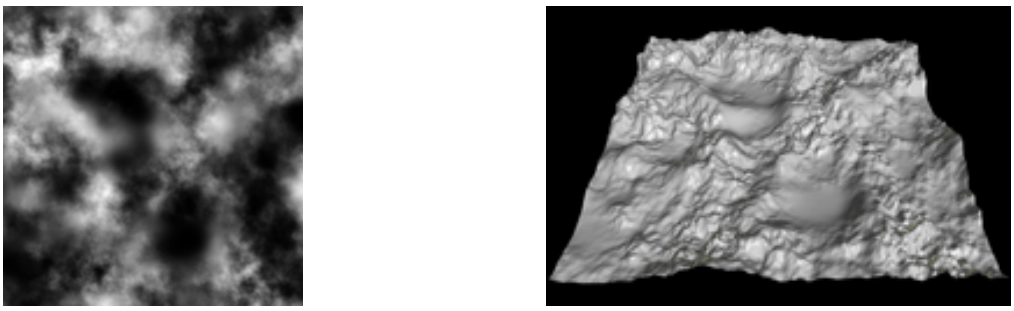

Figure 2: Images credit: `https://en.wikipedia.org/wiki/Heightmap`

Implementation Specifications - Grading Criteria

Develop an OpenGL application with the following functionalities and features:

- the minimum version of OpenGL should be 3.0 and up.
- OpenGL is used in retained mode.
- include comments explaining each step.

- load an image representing a heightmap. The application should handle images of minimum size 256×256 and maximum size of 1024×1024. You can assume that all input images will be gray-scale. A very simple and easy to use image API is CImg.
- create a mesh by forming triangles with each vertex in the heightmap. Each vertex position will be (x, y, z) where (x, y) are the image pixel coordinates and z the height value.
- create a GLFW window of size 800x800 with double buffering support.
- render the mesh on display.
- The application should use a perspective view to display the mesh and use the depth buffer for hidden surface removal.
- All vertices must be colored. You can select whatever color you like provided that it somehow relates to the elevation at that point. The simplest example would be to use the elevation value z to form the RGB color $(0, 0, z)$ at each point.
- handle the following input:
  - the user can rotate the mesh using keyboard input i.e. left arrow $\rightarrow R_z$, right arrow $\rightarrow R_{-z}$, up arrow $\rightarrow R_x$, down arrow $\rightarrow R_{-x}$
  - the user can change the rendering mode i.e. points, lines, triangles based on keyboard input i.e. key P for points, key W for lines, key T for triangles
  - the user can move the camera using the mouse i.e. moving forward/backward while left button is pressed $\rightarrow$ move into/out of the scene

Submission (electronic submission through Moodle only)

Please create a zip file containing your C/C++ code, vertex shader, fragment shader, a readme text file (.txt). In the readme file document the features and functionality of the application, and anything else you want the grader to know i.e. control keys, keyboard/mouse shortcuts, etc.

Additional Information

- You can use the skeleton code provided during the lab sessions to get started.
- A video demonstrating the functionality is posted on YouTube: `https://youtu.be/oZZeJ-27IOM`

Evaluation Procedure

You MUST demonstrate your solution program to the lab instructor during lab hours. You will be asked to download and run your submitted code, demonstrate its full functionality and answer questions about the OpenGL programming aspects of your solution. Major marking is done on the spot during demonstration. Your code will be further checked for structure, non-plagiarism, etc. However, ONLY demonstrated submissions will receive marks. Other submissions will not be marked.

Sample images