Programming Assignment 2

Announcement:	**Session # 5**
Submission Deadline:	**Session # 7**

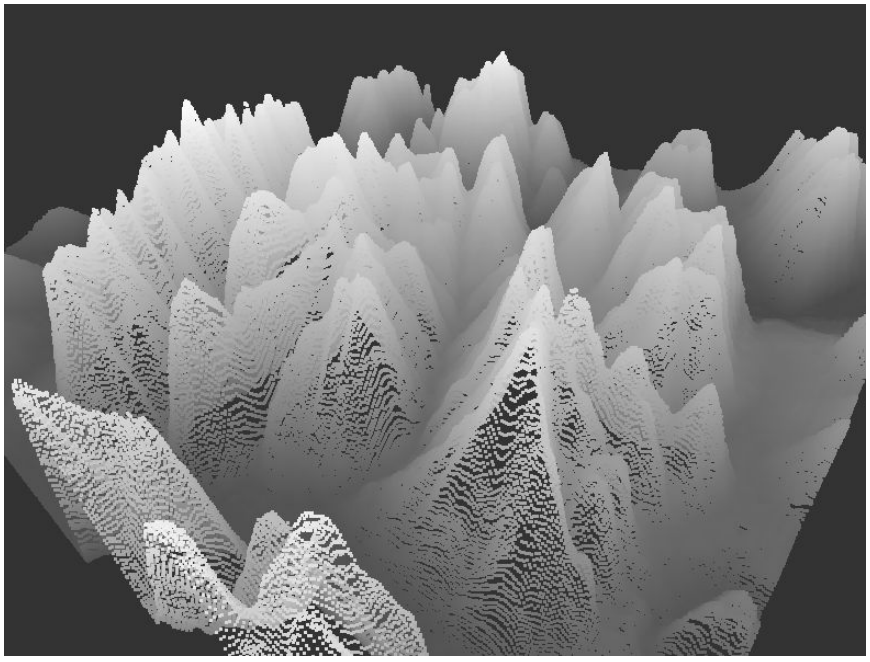
Description

This OpenGL programming assignment provides implementation experience of a few newly covered graphical concepts in class (triangle meshes and splines). In this assignment you will explore image interpretation as well as point interpolation using splines.

You have to first import an image and create a heightmap (3D mesh) using colour as height. Next, you have to create a 'smooth' version of the heightmap mesh.

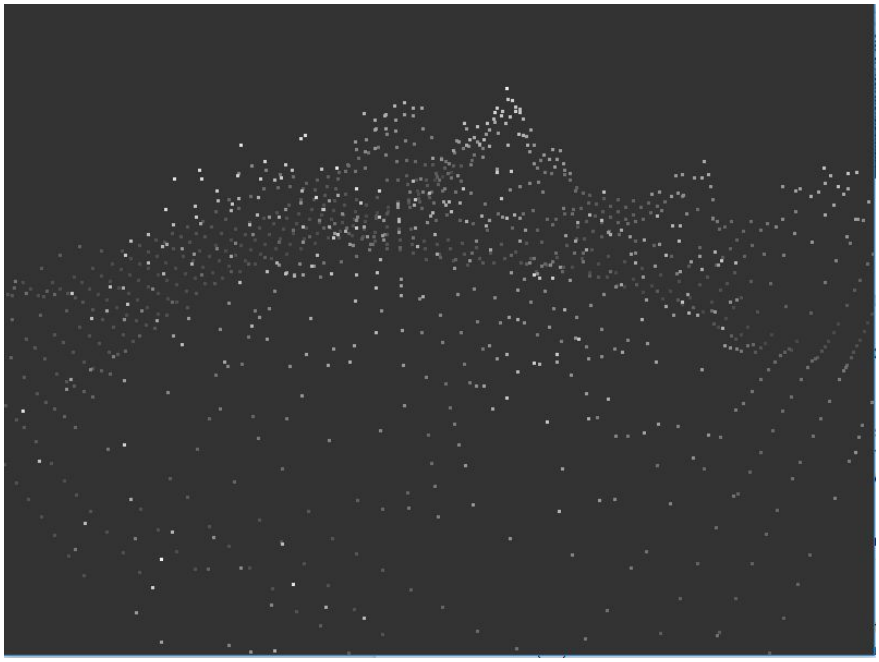
Implementation Specifications

1. The first step of this assignment is to import the given image into your OpenGL program. This can be done using CImg.
2. The next step is to interpret the image such that each pixel of the image represents a point in a 3D mesh. Each pixel's center position provides x and z coordinates of the corresponding point. The point's y coordinate is the actual value (colour) of the pixel.

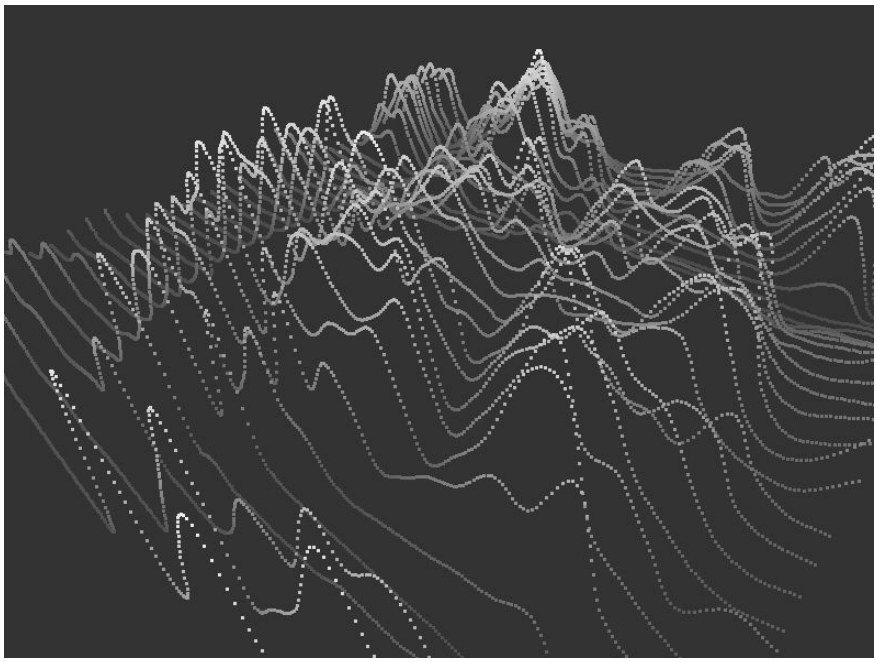


3. Next you will have to reduce the number of points in the 3D mesh. This you do by selecting some points and removing other points as follows. You use a skip-size integer for this. The skip-size should be given as input at the beginning of the program. For

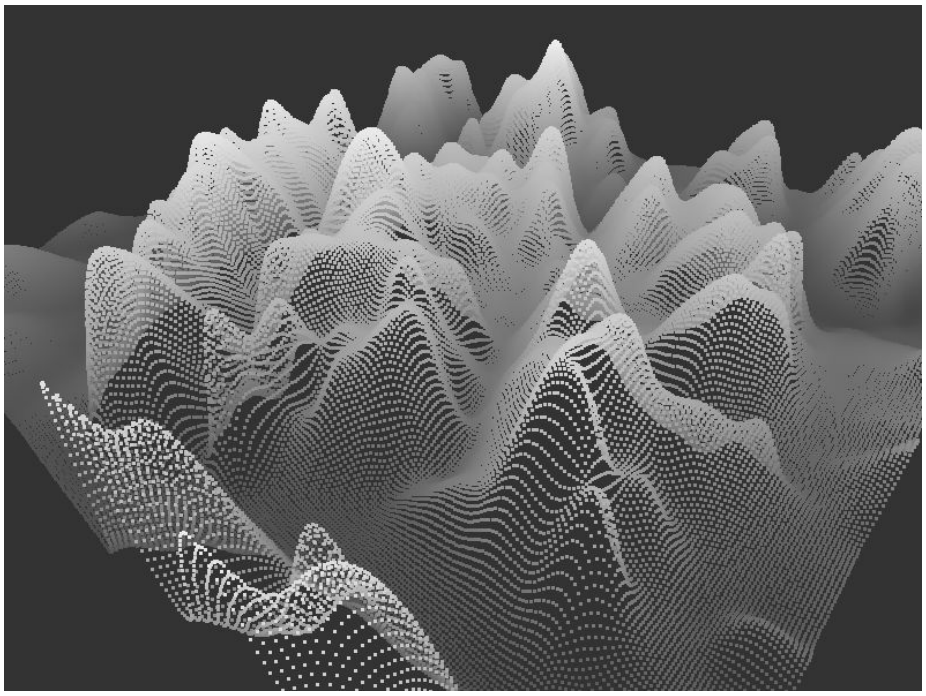
example, if the skip-size is 20, then points with index, 0, 40, 60, etc. along the x axes are selected and others removed.



4. This step consists of interpolating a smooth curve through points for each row, along x axis. This must be done using the ITERATIVE catmull rom technique, using equal curve parameter spacing between control points along the selected axis. The parameter step-size is a fraction that is given as input at the start of the program. For example, a step-size of 0.1 would mean that you compute curve points for parameter values of 0.0, 0.1, 0.2, 0.3, etc.



5. Finally you must interpolate the points in the other axis, namely z axis, for EVERY column of points in that axis; INCLUDING the curve points added in the previous step.



Requirements Specifications

- Each step of the implementation must be executed independently at runtime upon the press of a button of your choice.

- The user must be able to seamlessly move around the world to view the 3D mesh from any angle. This must be executed as follows:
    - Mouse to move the camera direction
    - Keys to move the camera position
- You must be able to render the mesh with triangles at ALL steps of the implementation. The use of an element buffer is required and will help.
    - You must be able to switch between points and triangles at any point with the press of a/two button/s.
- The BACKSPACE key should:
    - Reset the camera
    - Reset the 3D mesh
    - Ask the user for a skip-size for step 3.
- Window resize events must be handled.
- The color of the 3D mesh must be dynamic allowing us to see differences in height.

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:

Extra Credit (6%)
Keep the original height map interpretation from step 2 in a separate buffer. While the program is running the user can press a key to swap between the original 3D mesh and the modified one at ANY step of the implementation.

Evaluation Procedure
You MUST demonstrate your solution program to the lab instructor during lab hours. You must 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.