Praktikum Echtzeit-Computergrafik
Assignment 3 – Terrain Shading

Technische Universität München
Institut für Informatik
Lehrstuhl für Computergrafik & Visualisierung

Christoph Neuhauser, Simon Niedermayr – SS 24

In the last assignment, you have generated a terrain field, converted this field into an .obj triangle mesh, and loaded the mesh into ShaderLabWeb (https://vmwestermann10.in.tum.de/). In this assignment, you should color the terrain surface using vertex and fragment shaders.

Tasks

Task 1

Write a vertex shader that assigns to each vertex a color depending on the height of the vertex over ground. For instance, let the terrain color go linearly from greenish to greyish to white, to indicate, e.g., the transition from gras to rock to snow. You can also implement other effects, for instance, by interpolating from green to grey over the first n height units, and then interpolating from grey to white over the remaining units. For this task, you need to add an `out` declaration in the base vertex shader and an `in` declaration in the base fragment shader, where you can pass the fragment color as a `vec4` object, storing the red, green and blue color channel (normalized to [0, 1]) and the alpha channel (always set to 1 for now). Alternatively, you can also pass only the RGB color channels as a `vec3` object and use $vec4(vertexColor, 1.0)$ in the fragment shader.

Hint: Using the built-in GLSL functions `mix`[1] and `smoothstep`[2] might help you write shorter code.

Task 2

Do the same you did in Task 1, but perform coloring in the fragment shader. Think about how to determine the terrain height at each fragment, and then use the same coloring as in Task 1. For this task, you need to add an `out` declaration in the base vertex shader and an `in` declaration in the base fragment shader, where you can pass either `vertexPosition` or directly the height (i.e., the shifted/scaled z coordinate) to the fragment shader.

[1] https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/mix.xhtml
[2] https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/smoothstep.xhtml

Task 3

For the .obj file you have generated in Assignment 2, now include per-vertex normals and texture coordinates. For the per-vertex normals, compute for each vertex the normals of the triangles sharing this vertex, and average these normals to form the per-vertex normal. Triangle normals are perpendicular to the surface of the triangle and can be computed as $n = (v_0 - v_1) \times (v_0 - v_2)$. Alternatively, you can also compute the normals directly from the height field using central differences as $n = (s_x \frac{h(x+1,y)-h(x-1,y)}{2}, s_y \frac{h(x,y+1)-h(x,y-1)}{2}, 1)^T$. s_x and s_y are in this case two scaling factors for the size of the terrain geometry and $h(x, y)$ is the height at the array position x, y. Don't forget to normalize the resulting normal vectors! s_x and s_y need to be chosen as the inverse of the geometry scale you used for the vertex positions.

For the texture coordinates you can write values in the range $[0, 1]$ by dividing the x and y vertex position by the size of the terrain in x and y direction. You will need the texture coordinates in a future assignment.

Task 4

Now you can use the newly computed normals in the vertex shader to additionally color the vertices according to the terrain slope ("Gouraud shading"). Create a nice mix between height- and slope-based shading.

As a measure of the terrain slope, you can for example use the function `dot`[3], which returns the cosine of the angle between two vectors of length 1. One input vector to this function would be the (not transformed) object space vertex normal, and one input the the vector pointing up in model coordinates (i.e., $(0, 0, 1)^T$). Take the cosine as an illumination value and use it to modulate (i.e., multiply with) the fragment color.

Note: The standard transform from object space to world space rotates the object along the x axis by 90°. In the next assignment we will have a closer look at transformation matrices. You can do all computations involving normals in object space in this assignment.

Task 5

Pass the normals to the pixel shader and illuminate the fragments again using the slope. What differences can you see when using per-pixel shading instead of per-vertex shading?

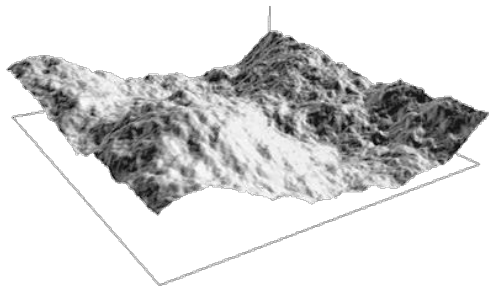

Figure 1: Triangle mesh generated in the last assignment from the height values.

[3]https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/dot.xhtml

Optional: Task 6

Find an algorithm that can help you smooth the surface of the terrain mesh.

Hint: You may consider using a mean filter or a Gaussian filter.

Optional: Task 7

The diamond-square algorithms has some drawbacks and shortcomings like the necessity to create the heightmap with a power of two resolution. Since there are many other algorithms out there to generate heightmaps, research one you find interesting and implement it as an alternative generation algorithm into your program.

Hint: You might want to search for terms like: Perlin Noise, Voronoi Noise and FBM.