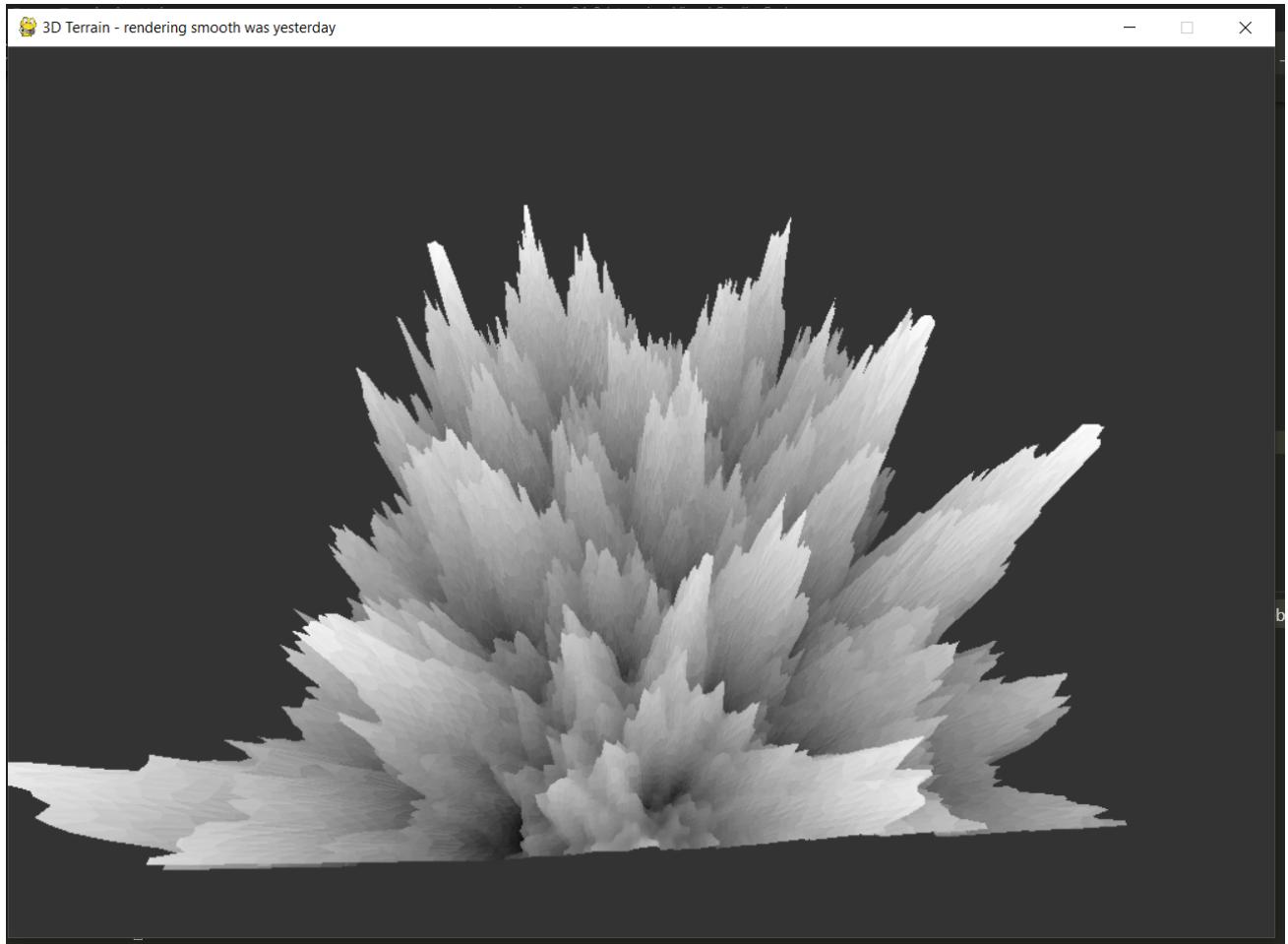


Gallery

Visual programming is fairly enjoyable due to some intermediate results and maybe some funny glitches. These usually don't make it into the final game. I've taken screenshots of my favorite or most confusing things I've seen during development. This gallery should give you a small insight into the development. Please enjoy!

The first terrain

This is the first terrain that was completely rendered using shaders. I used the height map as a texture. This results in an interesting depth effect. There is no other lighting in this scene



An error in the geometry shader

Roses are red
Violets are blue
I got an error on line 102
But the file ends at 32

I thought this was a meme but here it is. A legendary error on line 61 :).

The screenshot shows a code editor with a dark theme. On the left is the code editor pane displaying a C++ file with line numbers from 56 to 69. The code implements a lighting shader. Line 61 contains a syntax error: `f_color = color + u_diffuse * u_global_ambient;` The word `f_color` is highlighted in orange, indicating a potential error. Below the code editor is a terminal window showing the command-line interface. The terminal output includes the path to the project directory, the execution of a Python script, and the resulting pygame application. At the end of the application's output, there is a red box highlighting the error message: "b'ERROR: 0:61: 'constructor' : not enough data provided for construction \n\n'".

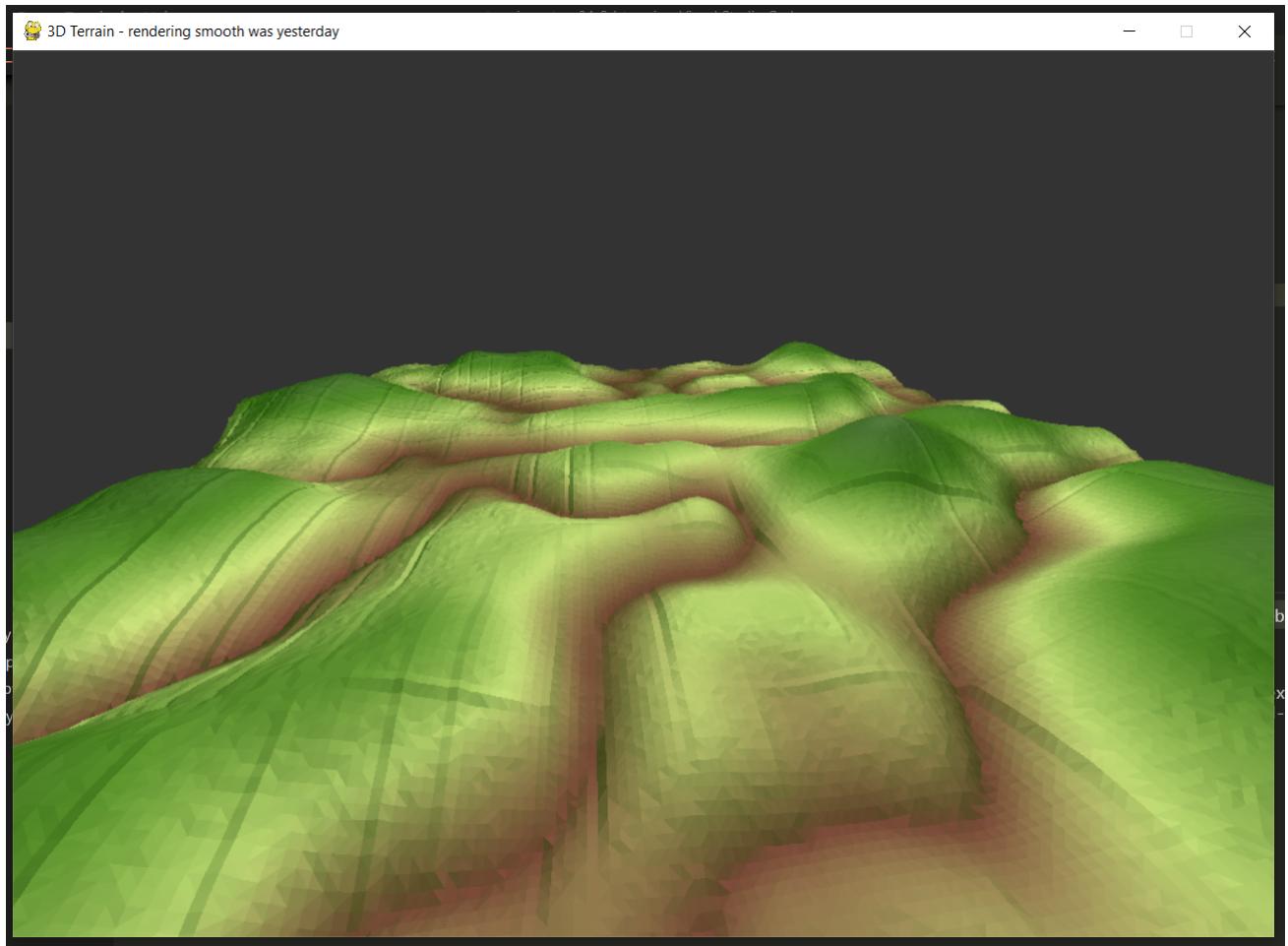
```
56 void main() {
57     vec3 normal = calculate_normal();
58     vec3 world_position = AVG_INPUT(v_world_position);
59     vec3 to_camera = u_camera_position - world_position;
60     vec3 color = vec3();
61     ...
62     for (int index = 0; index < MAX_LIGHT_COUNT; index++) {
63         if (index >= u_light_count) {
64             break;
65         }
66         color += calculate_light_effect(index, normal, world_position, to_camera);
67     }
68
69     f_color = color + u_diffuse * u_global_ambient;
```

PROBLEMS 19 OUTPUT TERMINAL

```
frist@LAPHH854 MINGW64 /c/workspace/ru/ru_tgra-graphics/projects/a04_3d-terrain (xfrednet/000-low-poly-
$ /usr/bin/env C:\\\\Users\\\\frist\\\\AppData\\\\Local\\\\Programs\\\\Python\\\\Python38-32\\\\python.exe c:\\\\Users\\\\020.9.114305\\\\pythonFiles\\\\lib\\\\python\\\\debugpy\\\\launcher 59132 -- c:\\\\workspace\\\\ru\\\\ru_tgra-graphics\\\\
pygame 2.0.0.dev12 (SDL 2.0.12, python 3.8.5)
Hello from the pygame community. https://www.pygame.org/contribute.html
Compiling: terrain.vert
Unable to load numpy_formathandler accelerator from OpenGL_accelerate
Compiling: terrain.geom
b'ERROR: 0:61: 'constructor' : not enough data provided for construction \n\n'"
```

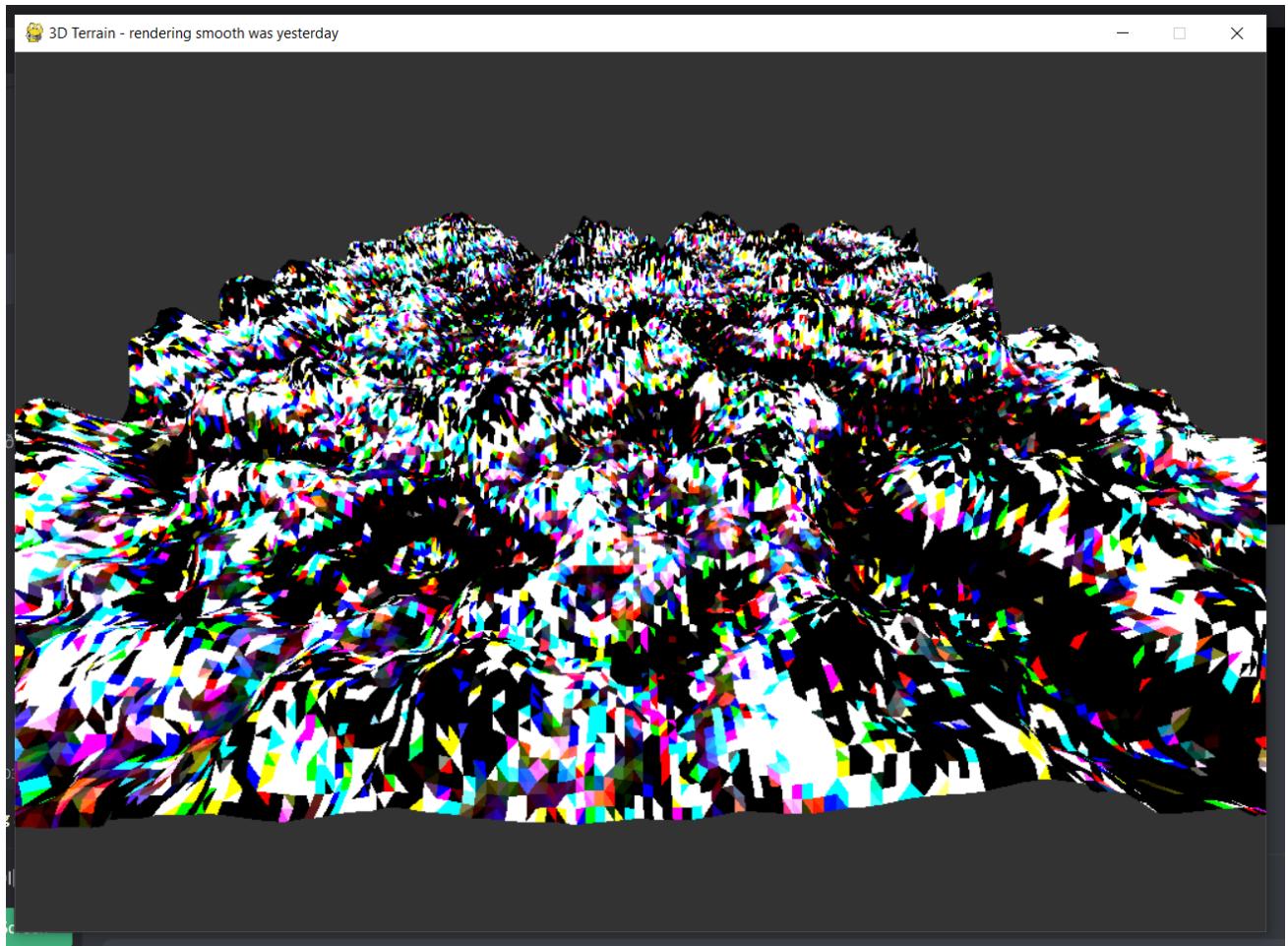
The first colored

This is an image of the first colored terrain. The height variation was still fairly small



Disco color flashing

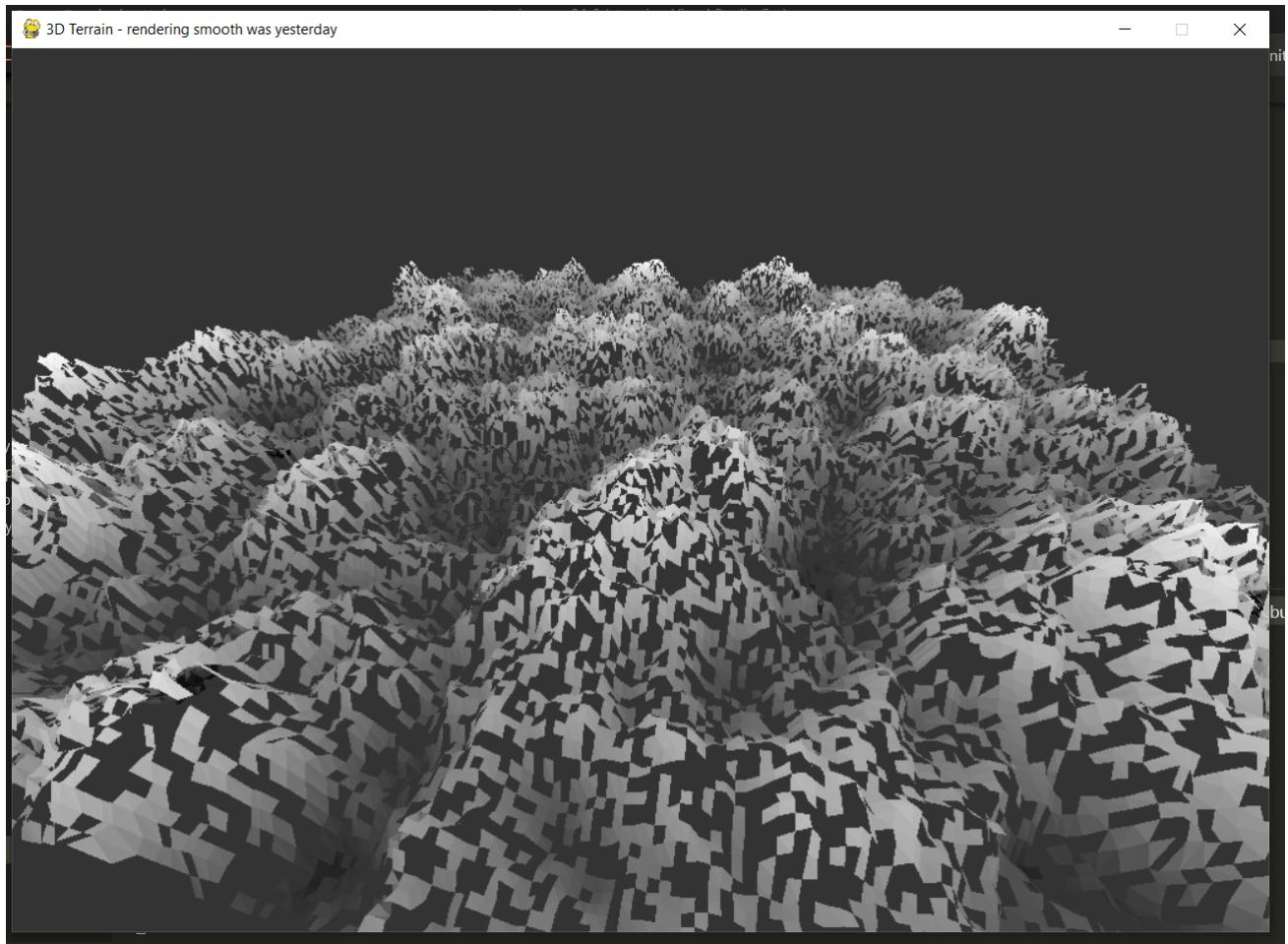
This *awesome* effect was a result of lighting in combination with over saturation. The entire scene was randomly flashing due to float point calculation errors and ever changing camera angles. It was still a but funny.



Face culling

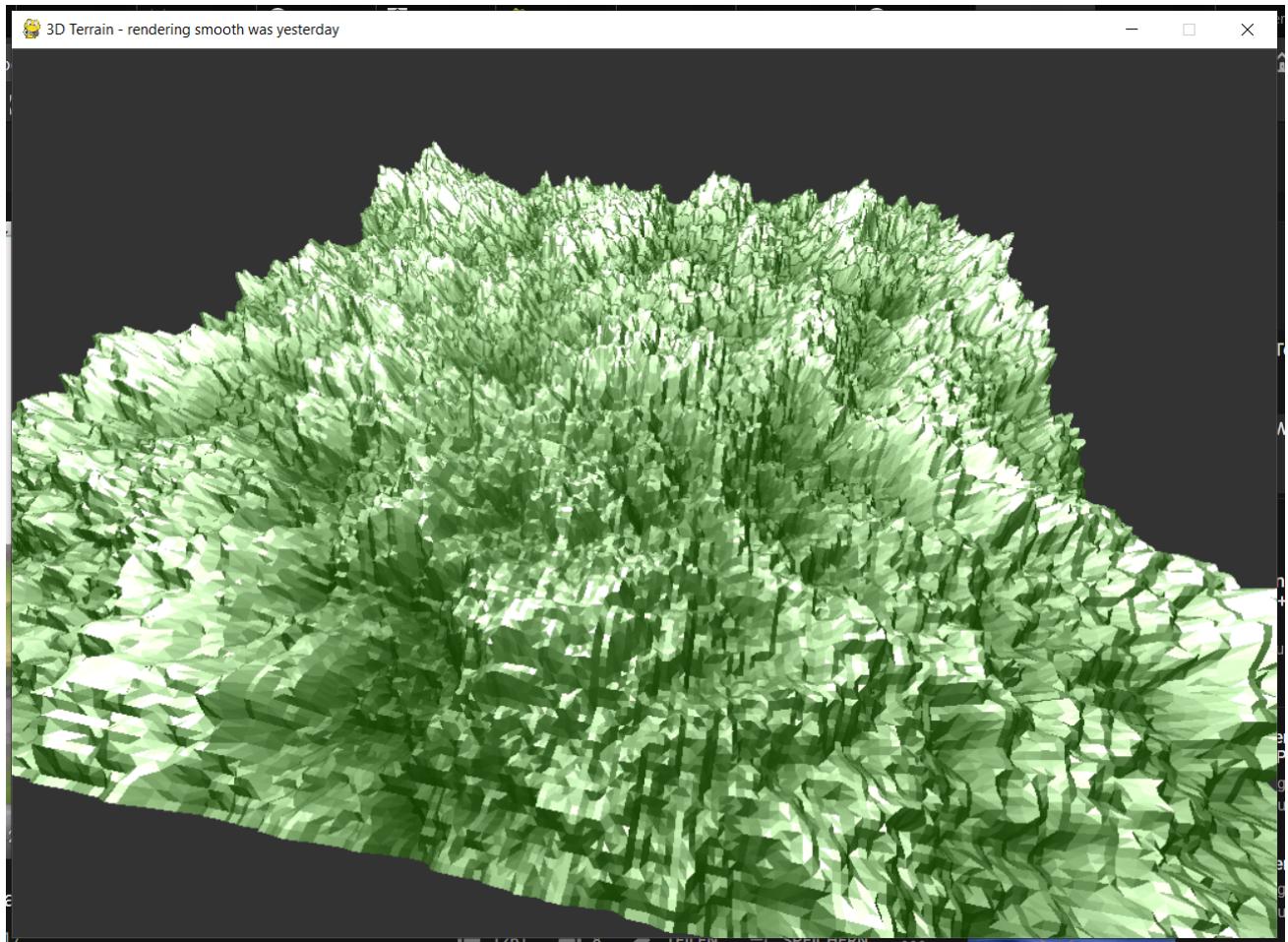
The project has enabled face culling to save some performance. This means that triangles are only drawn from one side. I wanted to add an effect where the triangle orientation was random. This is the result of the wrong vertex order ^^.

This piece of art is fittingly called: *almost cheese*



A working implementation

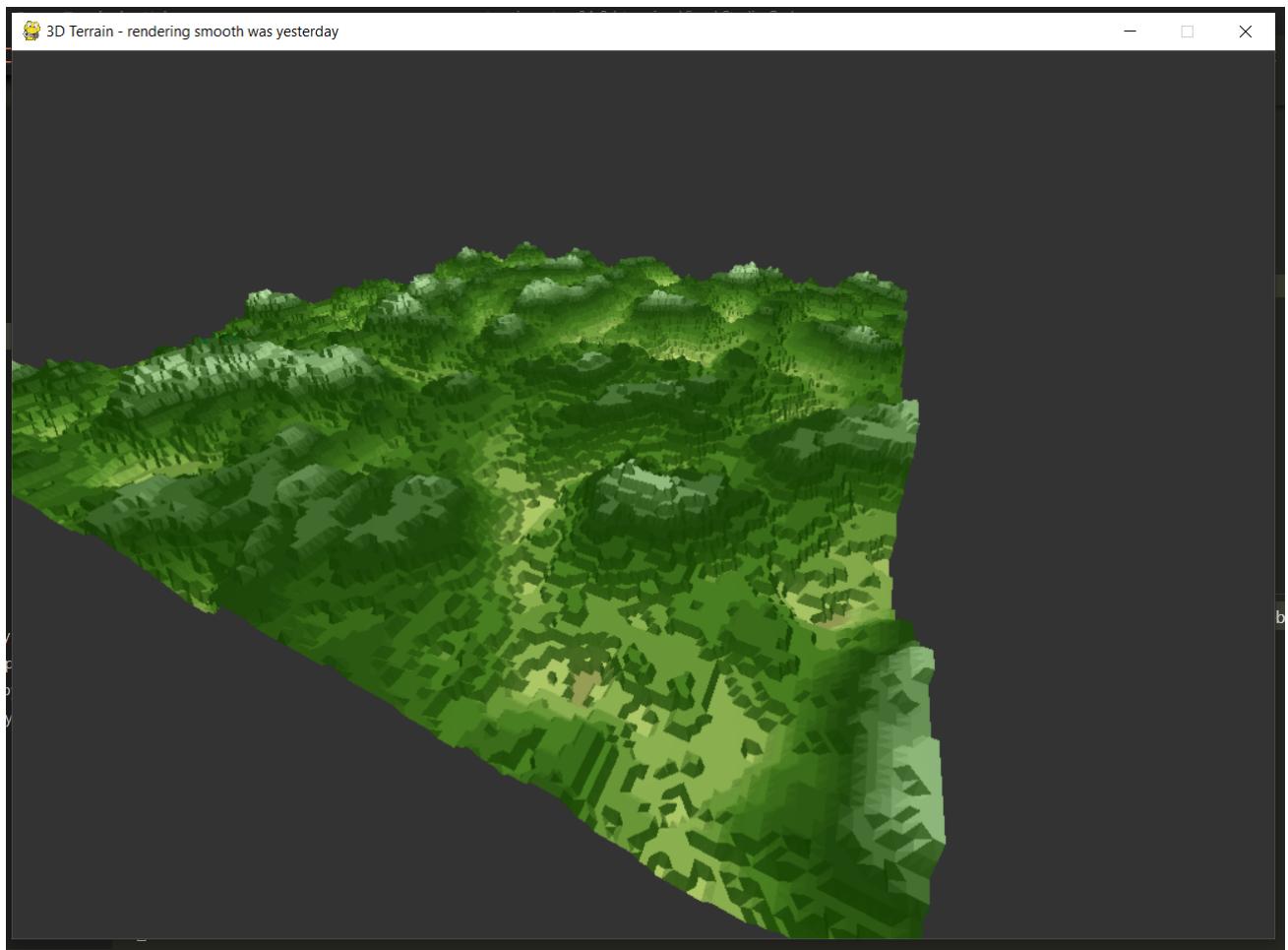
This was one of the first screenshots where everything was working together. There are still some rendering bugs as you can see. The reflectiveness is also way too high. It really looks like plastic or some other type of graph.



A different height map

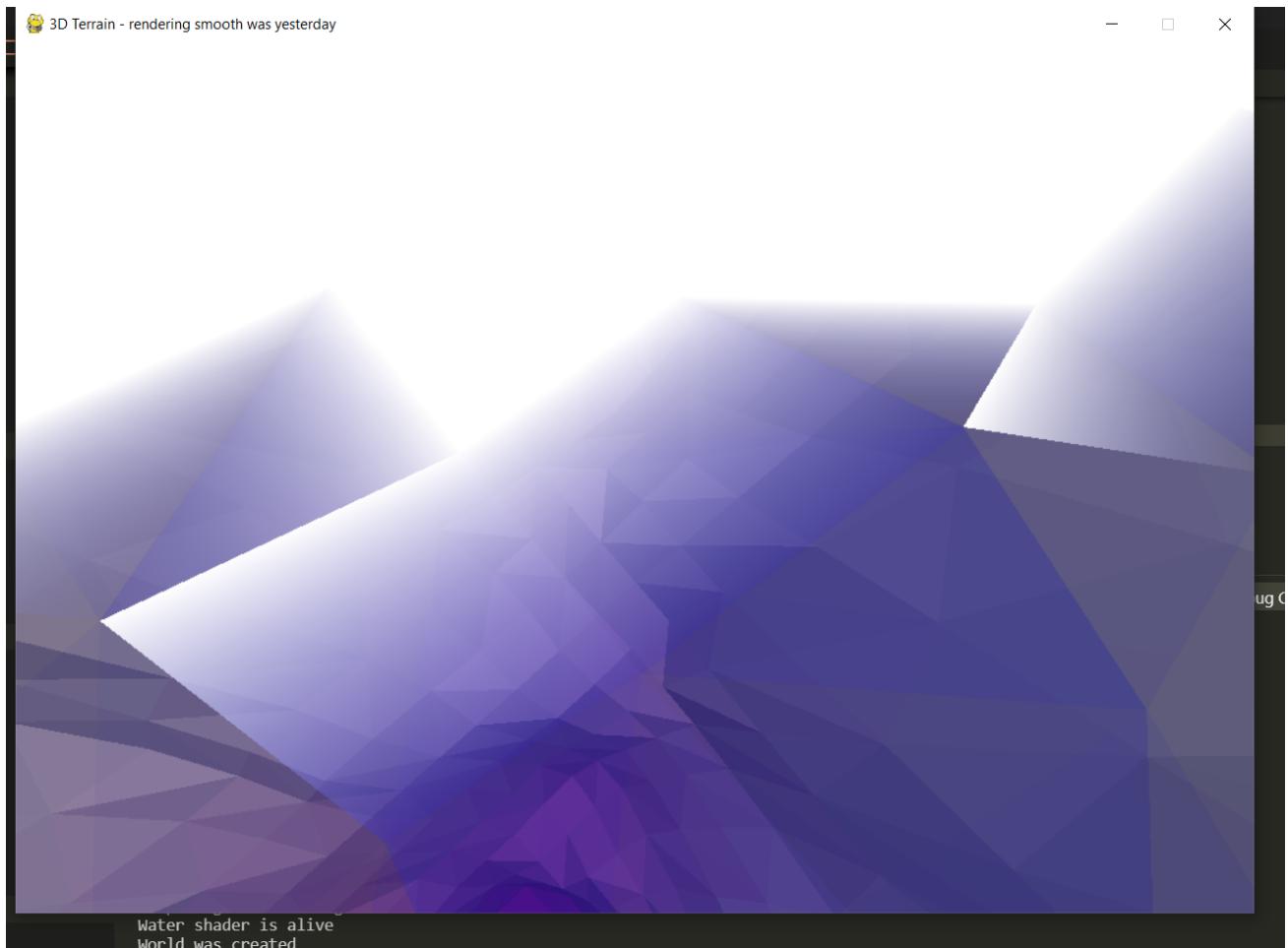
I had quite some fun playing with these height maps. I mostly used a *perlin noise* texture. This next piece of our exhibition shows such a texture with a flattening filter applied.

This is not the effect I was going for but it might definitely be used in another game :)



Water clipping test

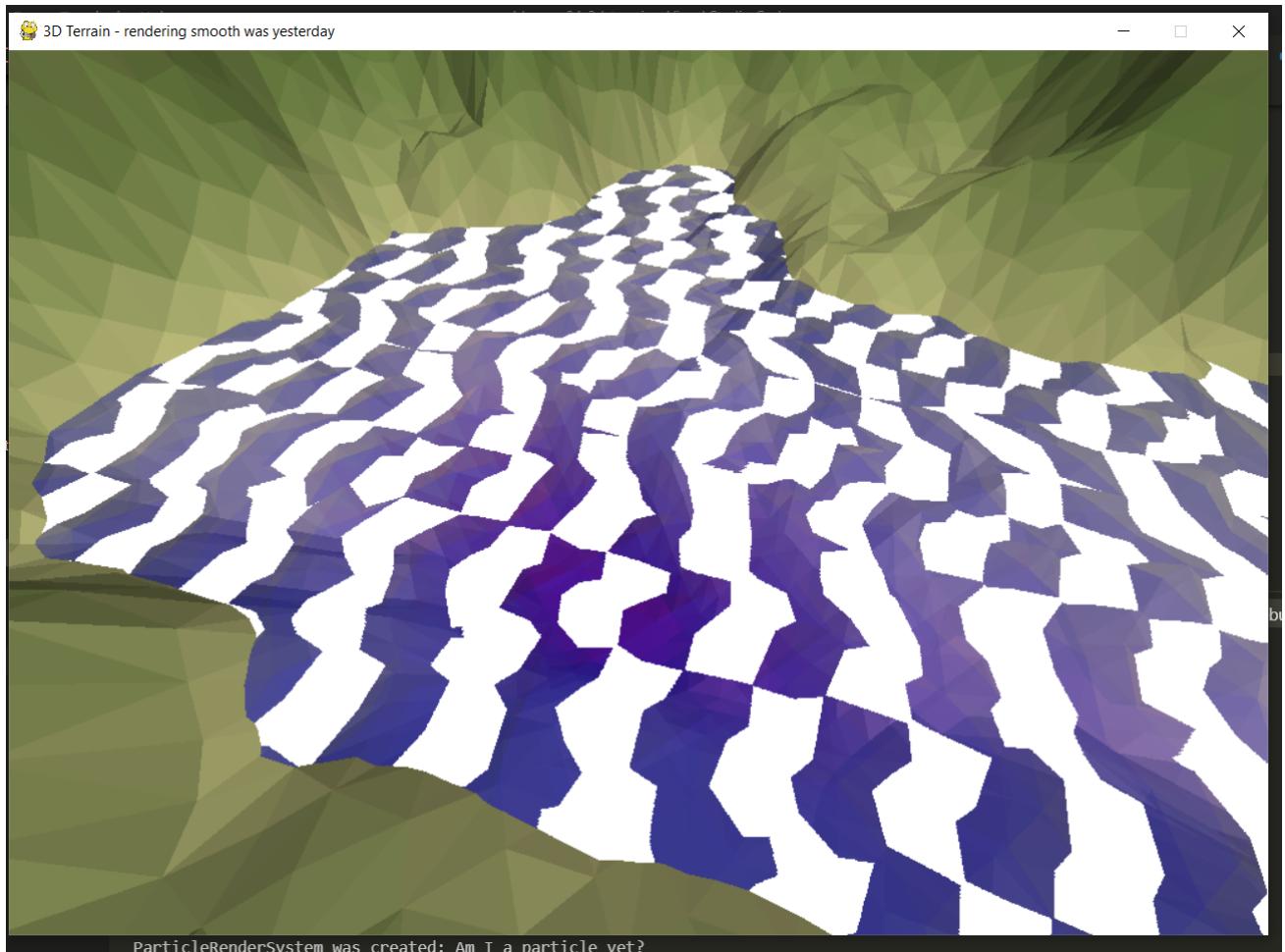
I was testing some stuff to create an underwater filter. I tried to apply a special effect to the triangles that where being clipped by the *projection matrix*. It sadly didn't work out the way I wanted it to but it still looks super interesting ^^.



(Extra points if you can guess how this effect was archived.)

Adding a plane at $y=0$

This was the result of a test where I added a plane at $y=0$. This actually enables us to see a pattern in the waves themselves:



The first particles

These were the first particle *rectangles* that were created using the geometric shader. It isn't perfect but it worked out at the end :).

