Goldsmiths, University of London

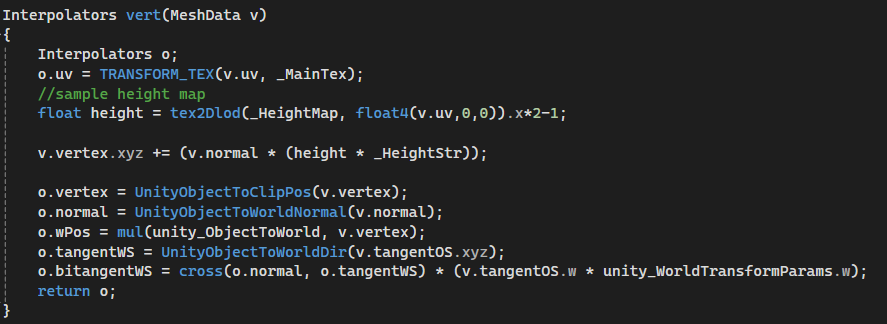
CM3045 – 3D Graphics & Animation

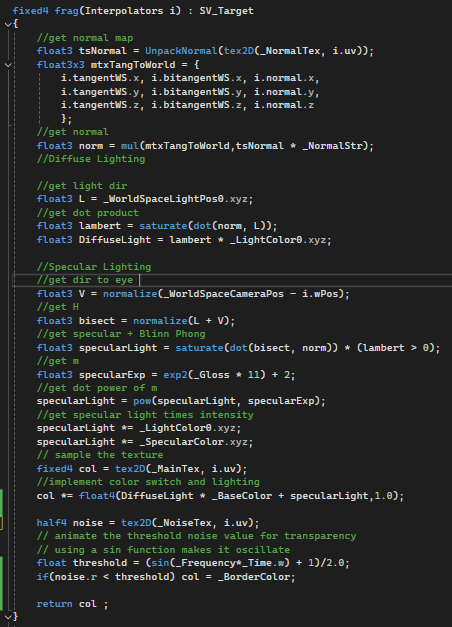
Peer-Graded Assignment: Texture Shader

Report by Hristo Stantchev

This report covers my submission for the “Texture Shader” Peer-Graded Assignment. This shader implements Normal Maps, Perlin Noise Effect, and Height Maps.

**Code:**

**Vertex-to-fragment: **

**Fragment:** 

**Task 1**: Normal Map

Normal maps begin in the vertex shader. As normal maps dictate varied lighting for object relief, they require the vertex surface normal, UV, a transformed tangent to world space, as well as the bitangent in world space – a cross product of the world space normal and tangent vectors multiplied by the product of the object space tangent and the unity world transform parameters (range (-1,1); includes where the normal is facing).

In this code, heavily borrowed from Freya Holmér [1], the fragment shader implements the normal maps by unpacking it with the input UVs in mind, then by introducing a tangent to world space 3x3 matrix it creates the normal directions used in the Lambert lighting model set up by multiplying the matrix by the normal map times its intensity.

**Task 2:** Perlin Noise

This is very much taken from the code provided in this course. It Implements Perlin Noise in the fragment shader by sampling the Perlin Noise texture from WikiPedia [2]. This is then multiplied by the sine of frequency times the w portion of the elapsed time, added by 1, and divided by 2, which gives a smoother speed of animation. Some form of scaled or unscaled time is always used for animation.

**Extension Task:** Height Map

Heavily sampled from Freya Holmér as well, displacement happens in the vertex shader where the texture is sampled through tex2DLod (which takes a vector 4 input, however only the x value is needed, it is written to sidestep tex2D as non-applicable to vertex shaders). It is multiplied by 2 and subtracted 1 in order to remap it from (0, 1) to (-1, 1) as it allows for negative vertex displacement. Only its X value is taken into the “height” variable. The product of the input normals, the height, and its intensity, is added to the X, Y, Z values of the input vertex, which displaces the pixel alongside the x values of the height map, leading to vertex displacement. This is all handled by the GPU and is surprisingly inexpensive performance-wise.

**References:**

1. **Holmér, Freya, “Normal Maps, Tangent Space & IBL • Shaders for Game Devs [Part 3]”, 26, 2021, YouTube:** <https://www.youtube.com/watch?v=E4PHFnvMzFc>
2. **WikiPedia Perlin Noise 1000x1000 texture, CC License:  
   By Lord Belbury - Own work, CC0,** [**https://commons.wikimedia.org/w/index.php?curid=121585558**](https://commons.wikimedia.org/w/index.php?curid=121585558)
3. **Sphere textures – Tuytel, Rob, PolyHaven, CC0:**[**https://polyhaven.com/a/patterned\_cobblestone\_02**](https://polyhaven.com/a/patterned_cobblestone_02)