Terrain Texturing

The multi-coloured terrain was good but not very realistic. Instead of defining colours, a texture needed to be mapped on top of the terrain. As explained in the background reading section, texture mapping in computer graphics is done by assigning vertices UV coordinates (also known texture coordinates).

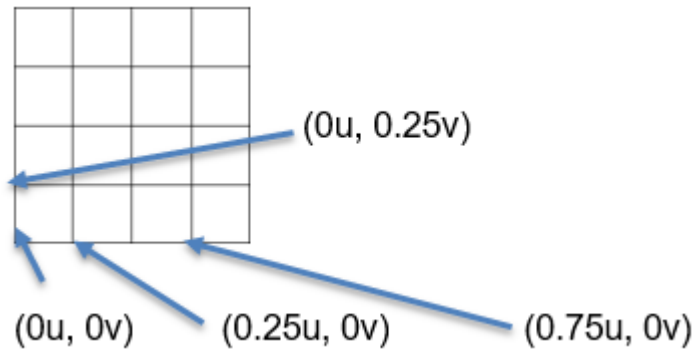
The terrain here is not just a simple square like in the background reading texture mapping example. Instead, it essentially a 2D grid of vertices. So to generate the texture coordinates for the vertices, a slightly more complex algorithm needs to be used. The below code from the Terrain.js createQuadrantUvs function shows the UV generation for a singular quadrant (quadrants are discussed later on). For now, just understand that each quadrant has 128 rows and 128 columns.

```
var incrementSize = 1 / quadrantRowSize;

// Set the start position for the UV coordinates
var xUV = 0;
var yUV = 0;

// Loop, (128 * 128) number of times
for(var x=0; x<quadrantColumnSize; x++){
    for(var y=0; y<quadrantRowSize; y++){
        quadrantUvs.push(xUV);
        quadrantUvs.push(yUV);
        xUV += incrementSize;
    }
    xUV = 0;
    yUV += incrementSize;
}
```

Since texture coordinates go from 0 to 1, the terrain size also needs to be mapped from 0 to 1. This can be done by dividing 1, by the quadrant row size (which is always 128).

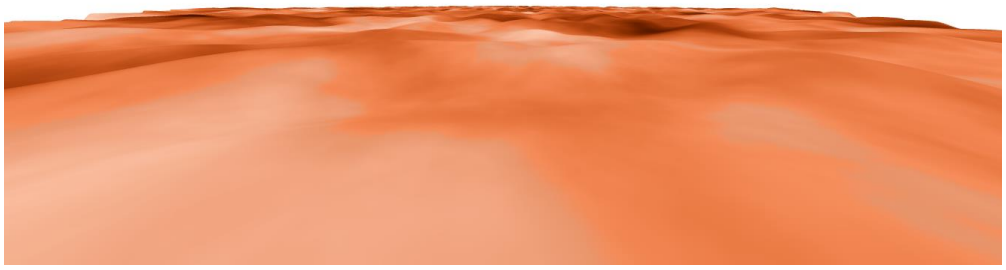
Given a quadrant size of 128, the texture coordinates increment value becomes 1/128 = 0.0078125. So for each new row, the vertices on that row have their respective X texture coordinates incremented by that value. Likewise for each new column, the vertices have their respective Y texture coordinates incremented by the value.

The example below illustrates this. It uses a 4x4 grid, instead of a 128x128 grid, for obvious reasons. The texture coordinate increment size for this grid would be 1 / 4 (row size) = 0.25. The double for loop would start at the bottom left of the grid, with initial values set to 0.



The same principle above is applied with the 128x128 terrain quadrants. The UV generation double for loop goes over each vertex in the quadrant, incrementing the UV values.

The below image is the result of the terrain being textured.



Each vertex now has its own 3D position and 2D texture coordinate. For a sprint or two, a procedural perlin noise texture was used on the terrain. The code to generate it can be seen here: Major Project/code/old code snippets/perlin.html

Open this page several times to view different texture results.

If you open the html page it will generate a new noise texture every time. Although technical, it did not look much like mars so it was eventually removed. The below images are examples of the noise textures generated by the code: