# Generating 3D Terrains from 2D Noise Maps

Shannon Williams

Computer and Information Sciences

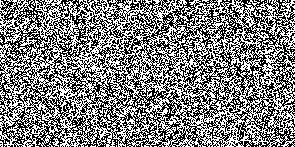
Harrisburg University of Science and Technology

***Abstract:*** *As part of the course requirements for CISC 400 - Computer Graphics, this paper will serve to aid my final project.*

*Keywords: noise, simplex, perlin*

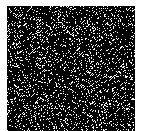
1. **Introduction**
2. **Background**
3. **Noise Height Maps**

Started with simple random generation of pixels.

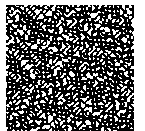


Noise generation ( Math.round( Math.random() % 32768) \* 255)

Then resorted to using simplex2 noise. Started with regular noise(x,y):



By giving a “zoom” appearance with factor of 8; I.e noise2(x/zoom, y/zoom):

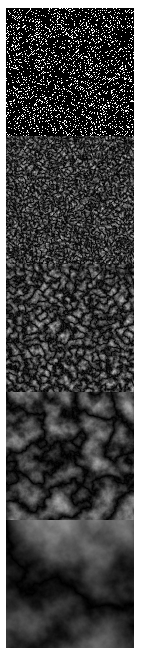


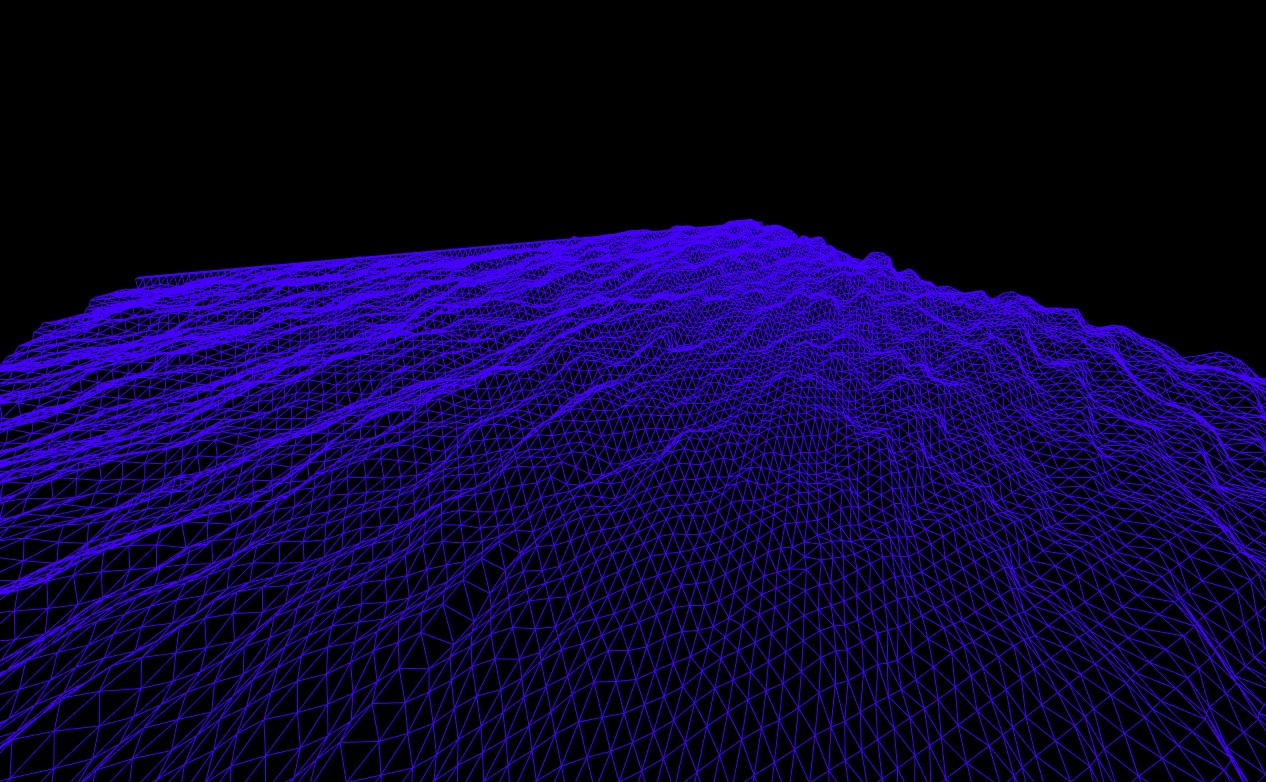
We can see we achieve a zooming effect. Lets try a zoom of 64.



What if we begin to combine these together? Now show concept of turbulence (achieving grayscale and not just black or white). See written notes for more details.

Now think of increasing the iterations. Again see notes for details. Below is a side-by-side comparison of images with turbulence with zoom factors of 1, 8, 16, 64, N=256, respectively.



Now using the same algorithm to set the height of vertices:

Applying texture in Three.JS

Creating thresholds for water, dark grass/mud, grass, dirty snow, and pure snow mountain tops. (Don’t forget to attempt adding gradients for smooth color transitions)

<http://lodev.org/cgtutor/randomnoise.html>

<https://www.redblobgames.com/maps/terrain-from-noise/>

<https://github.com/josephg/noisejs> for perlin.js