White Paper
Intel Software Solutions Group
Jeffrey M. Freeman

Procedural Terrain Generation With Fractional Brownian Motion

Information for Developers and ISVs
This white paper investigates techniques for leveraging procedurally generated terrain with fractional Brownian Motion on the CPU and multi-texture blending on the GPU.

December 2007

Intel Corporation
Contents
1  Introduction ................................................................................................... 3
2  Previous Work .............................................................................................. 5
3  Implementation .............................................................................................. 7
   3.1 Multitextured Pixel Shading ................................................................. 8
4  Future Work .................................................................................................... 9
5  References ..................................................................................................... 11
6  About the Author .......................................................................................... 13
A  Additional Images ........................................................................................ 15

Figures
3-1 Fractal Terrain, Simple fBm ................................................................. 7
1 Introduction

The computer graphics industry has a long history of attempting to model real world terrain. These efforts try to capture the seemingly limitless complexity of natural terrain through modeling and rendering techniques. As early as the late 1960’s Dr. Benoit Mandelbrot linked natural forms that maintain a level of self-similarity such as coastlines to mathematical constructs [1]. Notable achievements in this field since that time have utilized fractals to achieve approximations of terrain patches using stochastic processes such as fractional Brownian motion. In this article, we demonstrate several techniques of generating terrain patches as proposed by Dr. F Kenton Musgrave [2] along with texture blending and Shader Model 3.0 to create a synthetic scene on integrated graphics solutions such as the Intel® 965 Express Chipset and Mobile Intel® 965 Express Chipset family.

First, we describe a list of previous work in this field followed by the approach utilized by our implementation, which leverages both the CPU and GPU to render the scene. Source code is provided with the demonstration to be used in your terrain rendering extensions and implementation.
2 Previous Work

A number of researchers have investigated terrain generation using fractals to perturb surfaces in 2D and 3D space. B. Mandelbrot provided some of the earliest representations of terrain generation with fractals by comparing the self-similarity of mountainous terrain to Brownian motion, resulting in realistic skylines when charting a 2D random walk. Later work by Mandelbrot and Musgrave later showed increasingly compelling approximations of terrain utilizing fractional Brownian motion in 3D space with Perlin noise and the concept of multifractals both represented in [2].

Noise-based systems for generating fractal terrains as proposed by [2] and [4] are not exclusive to creating good approximations to real landscapes as several other calculations have been used to create aesthetically pleasing approximations to real terrain. Of interest in this group, include mid-point displacement calculations including the diamond-square and triangle-edge subdivision algorithms, Poisson faulting, and Fast Fourier Filtering.

While some of these systems can and do produce realistic looking scenes, the noise synthesis method proposed in [2] is utilized in this work as these calculations provide an interesting set of controls to the resulting terrain from a mathematical model. While these properties do not necessarily provide a mechanism to definitively control the shape of the rendered scene as indicated in [5] to constrain terrain to realistic properties, they do provide many interesting real and imaginative results. We present a CPU based set of algorithms demonstrating these controls balanced with smooth stepped texture blending in the pixel shader on the GPU using Microsoft DirectX 9 and Shader Model 3.0.
3 Implementation

Our implementation was inspired by Musgrave’s work in [2], showcasing three methods from that text: simple fBm, hybrid fBm, and the ridged multifractal algorithm, each based on Perlin’s noise algorithm. The output from these methods is used to perturb the Z direction of a fixed size polygon mesh.

Figure 3-1. Fractal Terrain, Simple fBm

In Figure 3-1, we present our implementation. On the right hand side, one can see the controls used to adjust properties of each fBm algorithm as selected from the combo box. Our demo is adapted from the BasicHLSL demo from [7] with default algorithm parameters adjust to demonstrate interesting terrain properties.

Method Parameters:

- (H) Hurst index – In mathematical literature, classifies the fBm and dictates fractal dimension.
- (Lacunarity) – Dictates the gap between successive frequencies.
- (Octaves) – Dictates the number of frequencies and scales Level of Detail in the scene.
- (Offset) – Offset from the lowest elevation and determines “multifractality” [2].
- (Gain) – Controls the amplitude of the frequency.
3.1 Multitextured Pixel Shading

Pixels are sampled from texture images and Hermite spline interpolated \( \{0..1\} \) over regions of existence based on elevation as passed to the pixel shader function. The multi-texturing blend operation, proposed by [3] was chosen over older fixed-function texture blending operations and implemented without a blend map. While a blend map would produce a finer and more realistic blend of each region’s texture in multi-textured surfaces as demonstrated in [3], the chosen method provides reasonable blending for higher altitude scenes which necessarily lack a finer level of detail due to the height of the viewpoint as may be experienced in a flight simulator. Each elevation zone is separated by constants indicating gradual shifts between regions of sand, grass, rock, and snow to create a blended effect over the range of elevations with regions nearing the next texture region taking on pixel hues of samples of that area, a method suggested by [9] in the shader effect file.
Future Work

There are a number of interesting problems remaining to be tackled with respect to fractal terrain generation. Applications of automatic landscape generation face decisions associated with conflicting game-play elements in the storyline or unrealistic features that present themselves from both fBm and other fractional models of terrain.

In addition, most terrain generation methods are calculation intensive and are not real-time. While some fractal algorithms lend themselves easily to multi-threading, the result is still time consuming as is the case with the Mandelbrot set [10] and may not apply well without significantly reducing the size of the perturbed surface greatly or reducing the number of iterations inspected. An interesting and difficult problem remains open for multi-threading a fractional Brownian motion implementation since it is a global operation, taking into account different domains of the variable. One could split processing by row or column, spanning a surface’s Z values or even by quadrants for processing with results that skew the controls of the algorithm’s parameters and render more noticeable regions of disjoint terrain.

It is this interdependence of previous values from the calculation that introduces the very properties we are seeking from fBm introduced by way of the Hurst index, Lacunarity, and Octave. As such, it becomes difficult to split calculations amongst physical processing units. Investigation of this concept with Dr. Mandelbrot [8] shows that there may be a solution to the threading problem through truncation of fBm. However, some level of introduced errors may be present in the calculation as a result. If the desired scene produced by the rendering process is aesthetically pleasing, some inaccuracy in the operation may be acceptable. We would like to determine what effect a multi-threaded fBm algorithm would have on a rendered terrain as well as analyze its performance for multi-core processors with optimized code.
5 References


6 About the Author

Jeff Freeman is a Software Engineer in the Software and Solutions Group, where he supports Intel integrated graphics solutions in the Client Scale Enabling team. He holds a B.S. in Computer Science from Rensselaer Polytechnic Institute. He can be reached at jeffrey.m.freeman@intel.com.

Credits:

- Steve Pitzel steve.pitzel@intel.com: Art/Textures
- Jeremy Hayes jeremy.hayes@intel.com: Guidance on terrain shading
A  Additional Images

[Image of FractalTerrain software interface showing two different fractal terrain models: Hybrid Multifractal and Rridged Multifractal.]
Procedural Terrain Generation With Fractional Brownian Motion

White Paper