Real-Time Massive Terrain Generation using Procedural Erosion on the GPU

Gabriel Costa Backes, Tiago Augusto Engel, Cesar Tadeu Pozzer
PPGCC - Programa de Pós-Graduação em Ciência da Computação
UFSM - Universidade Federal de Santa Maria
Santa Maria, Brazil
{*gbackes, tengel, pozzer*}*@inf.ufsm.br*

***Abstract*—Although terrain modeling is a widely explored field, problems such as scalability remain present when it comes to massive terrain sizes. While procedural methods have disadvantages regarding controllability and visual fidelity, it provides a great effectiveness handling with extremely large terrains due its computational performance. Our approach consists of generating the elevation data using a procedural function in real-time. We provide an intuitive set of parameters that allow control of several landscape characteristic. Real-world natural aspects were taken into account on the generation process. The terrain is represented by a dynamic quadtree structure and the elevation data is stored in a virtual texture. To address real-time performance, we adopt a GPU-based approach for the heightmap generation. Furthermore, a GPU instancing approach was used for rendering.**

***Keywords*-massive terrains, procedural generation, perlin noise, erosive noise, fractional brownian motion, real time rendering.**

I. Introduction

Procedural terrain generation is a widely explored field due to its significance in the entertainment industry, more specifically in movies and video games. It is imperative for such applications to generate visually appealing and detail rich landscapes. While physically plausible landscapes can be generated by hydraulic and wind erosion, and tectonic uplifting simulations, such approaches are not feasible for generating elevation data in real-time and often lack on controllability. On the other hand, the terrain can be manually sketched or entirely loaded by an input data. However, these approaches are not scalable for extremely large terrains, which may extent several kilometers at the resolution of centimeters.

Example-based approaches use real world datasets, so its resolution and size are limited by the input. Physically-based approaches demand lengthy preprocessing times and are accurate, being a suitable approach for movies and games with limited scenarios. Differently, open-world games often allow the user to interact with massive worlds. Fractal-based procedural approaches are efficient but lack realism and controllability. We therefore provide a process for generating visually appealing terrains using fractals without compromising performance.

In this paper we introduce an novel approach to address massively terrain generation in real-time. Our work is based on a fractional Brownian motion (fBm) that uses a continuous procedural Perlin function to define the elevation data [1]. Moreover, we provide different kinds of primitives to work around the terrain natural properties, providing the global control over the features.

The coarse features are defined by the base parameters, such as the terrain height scale, the highlight on mountains and valleys, and the terrain roughness and details on its composition. While these parameters can generate interesting terrains, its intrinsic features are far from real. In the real world, the terrain has many complex features of different sizes on its composition. Pinter and Brandon [2] shown that natural agents modify these features, causing irregularly distribution. While higher elevations are characterized by an irregular surfaces due to, among other factors, its average temperature and scrub vegetation, lower elevations tend to be more flat, once they concentrate more sedimentation. To replicate these properties, we propose a procedural erosion set, which controls the amount details over the terrain altitude, steepness and curvature.

Unlike physics-based approach, the procedural erosion allows us to generate credible terrains at real-time frame rates. An infinity of complex terrains can be easily defined through the intuitive input set. Our model is computationally efficient and provides a flexible control over the terrain features. As result, it is suitable for editing and rendering massive terrains.

II. Related Works

Procedural algorithms have been used in the terrain generation field for more than three decades. Usually, the terrain is discretized by a regular grid, which stores the elevation data in a texture, called as heightmap. Early methods were based on iterative subdivision of the grid. Fournier [3] presented a method that became known as midpoint displacement. Level of details are generated by displacing a new vertex by the parents average height of a coarse level combined by a random value. Noise based approaches, such as Perlin noise [4], were shown as promising due its computational costs. The details are created by summing several noise octaves on differentes frequencies and scaled amplitudes [1]. Despite these methods produce many particular features, its distribution are regular over the surface.

Physics-based approaches were incorporated to introduce natural aspects over the terrain features. Musgrave addresses geomorphic agents by two erosion algorithms to simulate hydraulic erosion and thermal weathering [5]. Nevertheless, its computational cost hinder the use on large terrains with high level of details. Vanek [6] introduced a GPU-based approach, in which large terrains are divided into tiles of different resolutions, the physics-based simulation use different level of details to speedup the computation. Unfortunately, due to the heights evaluation are locally dependent, GPU-based approaches can not deplete parallel programing potential and the scalability remains a problem.

Sketched-based methods address the low controllability of the procedural generation. Rusnell [7] describes a terrain synthesis method based on distances in a weighted graph created from user-specified generator nodes. Gain [8] introduces a height map generation using silhouette and bounds of mountains user sketched. Hnaidi [9] proposed a method based on diffusion of constraints, in which the user control the landforms by feature curves. Recently, Gnevaux [10] presents a hierarchical procedural model that combines feature-based primitives to create complex terrains. It has been extended by Gurin [11], in which proposed a procedural modeling that can automatically generate large scale terrain rather than authoring the terrain interactively by hand.

III. Overview

An overview of the proposed terrain generation is shown in the Fig. 1. The terrain base is defined by the 2D Perlin noise, in which, along with a fBm, introduce details to the terrain. The visual appealing relies on an intuitive parameters set, which provides the user a global controllability. The inputs manage the terrain scale, roughness and geometric characteristics. Moreover, an additional manageable erosion set improves the terrain natural aspect, damping the details on particular regions, as specific altitudes, slopes or concavities. The proposed approach is inspired by Carpentier [12], which present a brush editing using several noises primitives.

Our approach relies on discretizing the terrain into multi resolution tiles using a dynamic quadtree structure. In this way, once we adopted an continuous procedural function, we can handle an endless terrain on arbitrary resolution. The elevation data of each tile is stored on a virtual texture. A view frustum culling is performed on CPU to select the quadtree nodes to be rendered. However, since the Perlin noise is locally independent, i.e., the heights can be evaluated in parallel, a GPU-based approach was adopted to perform the heightmap generation. Furthermore, we used a GPU-instancing approach to render the terrain. In this step, the vertexes of a grid mesh are displaced by the elevation data on the vertex shader.

IV. Procedural Terrain Generation

The terrain base is provided by the fBm method based on 2D Perlin noise, denoted by F. Since perlin noise evaluates the noise value individually, the heights for a given patch can be computed in parallel.

The naive fBm algorithm calculates a weighted sum of scaled noises from several octaves. The amplitude and frequency of the noise are modified, respectively, by a gain and lacunarity on each octave, thus, originating features of manifold size. The equation is defined as follows:

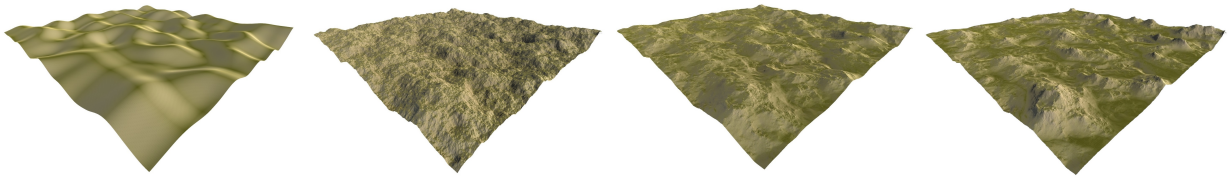
$$F(p) = \sum_{i=0}^{n-1} \xi_i N(\lambda_i p) \tag{1}$$

where n represents the number of octaves and i the current octave, $\xi \in [0, 1]$ represents the amplitude. $N(p)$ is the perlin noise function, in which given a position p returns a scalar $s \in [-1, 1]$. λ represents the frequency.

Although the fBm can create really interesting terrains,the noise distribution is uniform. Thus, there is no variations on the terrain features. To address this natural lack aspect, we presented bellow some intuitive parameters to be introduced on the fBm, modifying the geometric characteristics.

A. Terrain roughness, sharpness and scale

The terrain roughness is directly controlled by the amplitude ξ and the frequency λ. The amplitude in each octave ξ_i is given by the multiplication of an initial value I_a with a gain $g \in [0, 1]$ (Equation 2a). In the same way, the frequency λ_i is given by the multiplication of an initial value I_f with a lacunarity l (Equation 2b). Lower values to g will make the terrain be more flat, while higher values will generate



(a) (b) (c) (d)

Figure 1. Terrain generation pipeline. (a) First, the base terrain is generated using a 2D perlin noise. (b)The details are introduced using the fBm method. (c) Then, terrain features are sculpted using the procedural erosion set to give a more real aspect. (d) Lastly, an enhancement is applied over the terrain features.

protuberant features. The figures in this article use $I_a = 1$, $I_f = 0.0001$, $g = 0.5$, and $l = 2$.

$$\xi_i = I_a \times g^i \tag{2a}$$

$$\lambda_i = I_f \times l^i \tag{2b}$$

Two well known noises were used to define the terrain sharpness. The first one is the billow noise (N_B), given by the absolut value from the noise function (Equation 3a), providing waved features. The second one is the ridge noise (N_R), resulting from the billow noise complement (Equation 3b), setting up sharp patterns. A sharpness scalar $S_s \in [-1, 1]$ is used in a linear interpolation among the perlin noise, billow noise and ridge noise, as shown in Equation 3c.

$$N_B(p) = |N(p)| \tag{3a}$$

$$N_R(p) = 1 - N_B(p) \tag{3b}$$

$$N_C(p) = \begin{cases} lerp(N(p), N_B(p), |S_s|) & S_s >= 0 \\ lerp(N(p), N_R(p), S_s) & S_s < 0 \end{cases} \tag{3c}$$

Since we are working with fBm output values in the range [-1, 1], the mountains and valleys are enhanced by a scalar S_e and the terrain height is rescaled by and scalar S_h as the following equation:

$$H(p) = F(p) \times |F(p)|^{S_e} \times S_h \tag{4}$$

First, the noise output value is multiplied by its absolute value on the power of S_e. Higher values of S_e turn the height exponentially close to zero, causing abrupt height variations to be highlighted, such as valleys and mountains. Subsequently, the value is multiplied by the scalar S_h, which rescale the height range [-1, 1] into the range [-S_h, S_h].

B. Terrain Erosion

To satisfy the properties elected by [2], variables were aggregated on the fBm to damp the summed noise height, denoted by σ_i, and the noise amplitude, based on the terrain altitude, slope and concavity. Moreover, scalars were defined to provide the user control over the damp ratio.

The altitude erosion makes the finer details of the noise fade out quickly on lower heights.It can be introduced on the fBm to progressively damp the amplitude based on the summed height σ_i and an altitude factor (E_a), as shown as follows:

$$\xi_i = \begin{cases} I_a & i = 0 \\ \xi_{i-1} \times lerp(g_i, g_i \times max(0, \sigma_i), E_a) & i > 0 \end{cases} \tag{5}$$

On the first octave, the amplitude value is set to the initial value I_a. On the subsequents octaves, instead of multiply the value by the gain g, as shown on the Equation 2a, we multiply by the result of a linear interpolation between the current gain g_i and the current gain g_i damped by the current height σ_i. Since we are working on the height range $\in [-1, 1]$, where 1 is the highest point, lower values will decrease the gain value, and consequently, the amplitude. The amplitude erosion is controlled by the altitude factor E_a used on the linear interpolation.

A similar idea is used on the slope erosion. The details are directly dependent of the terrain steepness. The terrain slope can be retrieved by its derivative. Thus, we used the analytical derivative from the Perlin noise to get its gradient. The slope erosion ζ_s is given by the following formula:

$$\zeta_s = \frac{1}{1 + (G_i(p) \cdot G_i(p))} \tag{6}$$

$G(p)$ represents the gradient at p (point on XY plane), which is given by the summed partial derivative of $N(p)$, weighted by the slope factor E_s:

$$G_i(x, y) = \begin{cases} (0, 0) & i = -1 \\ G_{i-1}(x, y) + (\frac{\partial N(x,y)}{\partial x}, \frac{\partial N(x,y)}{\partial y}) \times E_s & i >= 0 \end{cases} \tag{7}$$

The idea is to decrease the amplitude on the steeper slopes, represented by larger derivative values. In this way, we perform the scalar product from the gradient vector with itself, giving its magnitude squared. The gradient G_i is initialized as (0,0), and stores the summed gradient until the current octave. The equation 7 gives values closer to 0 to greater gradients. Then, we can use this value to damp the amplitude. The damp ratio is controlled by the slope factor E_s which scale the point's gradient.

Considering that concaves surfaces tend to be more sedimented, the concavity erosion turns the terrain smoother based on its concavity. Therefore, the current gain g_i is damped by the surface concavity and the concavity factor E_c:

$$g_i = lerp(g, g \times (\frac{1}{1 + abs(min(C_i, 0))}), E_c) \tag{8}$$

The surface concavity, denoted by C_i, is a scalar that defines the concavity of the current height. Values greater than 0 represent a concave surface. On the other hand, values lower then 0 represent a convex surface. C_i is computed using the second partial derivatives, as follows:

$$C_i = (\frac{\partial^2 N(x, y)}{\partial x^2} + \frac{\partial^2 N(x, y)}{\partial y^2}) \times 0.5 \tag{9}$$

As we expect to decrease the details on the concaves areas, we compute a minimum between C_i and 0. We use the absolute value to damp the gain. The altitude erosion, we perform a linear interpolation between the gain and the

(a) (b) (c)

Figure 2. Example of terrains generated by our model.

damped gain. The concavity factor E_c is used to control the damp ratio.

V. Results and Discussion

Since it is known that procedural methods are extremely efficient, we have analyzed the performance variation of the heightmap generation using the native fBm with a 2D Perlin noise and the proposed solution. The tests were performed on a AMD Ryzen 5 1600 3.2GHz CPU, 16GB RAM, and GeForce 1070. The proposed algorithm was implemented in C# using the Unity engine.

In the tests, we have played with three different octaves for each noise. The times to generate a heightmap of 128x128 size for 2, 4 and 8 octaves were, respectively, 0.291, 0.298, 0.318 for the naive fBm and 0.310, 0.312, 0.327 for the proposed solution, requiring 1 second to generate a terrain with 7x7 kilometers at a resolution of 1 meter.

We aimed at a massive terrain generation with real-time performance. Hence, our model does not handle erosion correctness because these methods rely on the computation of neighboring values which is not suitable for parallel evaluation approach. Despite our implementation does not produce realistic results as physics-based approaches, it can generates more truthful features without losing significant performance when compared to the naive fBm, preserving the high scalability from the procedural methods.

VI. Conclusion

This paper presents a GPU-based procedural terrain generation, controlled by an intuitive parameters set as input, which supports massive terrains at real-time frame rates. The terrain generation is based on the fBm method. The proposed procedural erosion set gives more naturalness to the terrain, concomitantly, guides the user to create unique terrains with singular features. Moreover, the heightmap generation performance was highly increase due to the fBm parallel implementation on the GPU. Likewise, the real-time rendering also relies on an efficient terrain management due to (a) a multi LOD terrain provided by a dynamic quadtree, and a (b) GPU instancing approach.

VII. Acknowledgement

We thank the Brazilian Army for the financial support through the SIS-ASTROS project.

References

[1] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley, *Texturing and Modeling: A Procedural Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 3rd ed., 2002.

[2] N. Pinter and M. T. Brandon, "How erosion builds mountains," *Scientific American*, vol. 276, no. 4, pp. 74–79, 1997.

[3] A. Fournier, D. Fussell, and L. Carpenter, "Computer rendering of stochastic models," *Communications of the ACM*, vol. 25, no. 6, pp. 371–384, 1982.

[4] K. Perlin, "An image synthesizer," *ACM SIGGRAPH Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

[5] F. K. Musgrave, C. E. Kolb, and R. S. Mace, "The synthesis and rendering of eroded fractal terrains," *Proceedings of the 16th annual conference on Computer graphics and interactive techniques - SIGGRAPH '89*, vol. 23, no. 3, pp. 41–50, 1989.

[6] J. Vanek and A. Herout, "Large-Scale Physics-Based Terrain Editing Using Adaptive Tiles on the GPU," pp. 1–9, 2011.

[7] B. Rusnell, D. Mould, and M. Eramian, "Feature-rich distance-based terrain synthesis," pp. 573–579, 2009.

[8] J. Gain, P. Marais, and W. Straßer, "Terrain sketching," *Proceedings of the 2009 symposium on Interactive 3D graphics and games - I3D '09*, vol. 1, no. 212, p. 31, 2009.

[9] H. Hnaidi, E. Guérin, S. Akkouche, A. Peytavie, and E. Galin, "Feature based terrain generation using diffusion equation," *Computer Graphics Forum*, vol. 29, no. 7, pp. 2179–2186, 2010.

[10] J. D. Génevaux, E. Galin, A. Peytavie, E. Guérin, C. Briquet, F. Grosbellet, and B. Benes, "Terrain Modelling from Feature Primitives," *Computer Graphics Forum*, vol. 34, no. 6, pp. 198–210, 2015.

[11] E. Guérin, J. Digne, E. Galin, and A. Peytavie, "Sparse representation of terrains for procedural modeling," *Computer Graphics Forum*, vol. 35, no. 2, pp. 177–187, 2016.

[12] G. J. de Carpentier and R. Bidarra, "Interactive gpu-based procedural heightfield brushes," in *Proceedings of the 4th International Conference on Foundations of Digital Games*, pp. 55–62, ACM, 2009.