Improving Terrain Visualization Through Procedural Generation and Hardware Tessellation

Alex Frasson * Tiago Augusto Engel Cesar Tadeu Pozzer

Universidade Federal de Santa Maria, Brazil

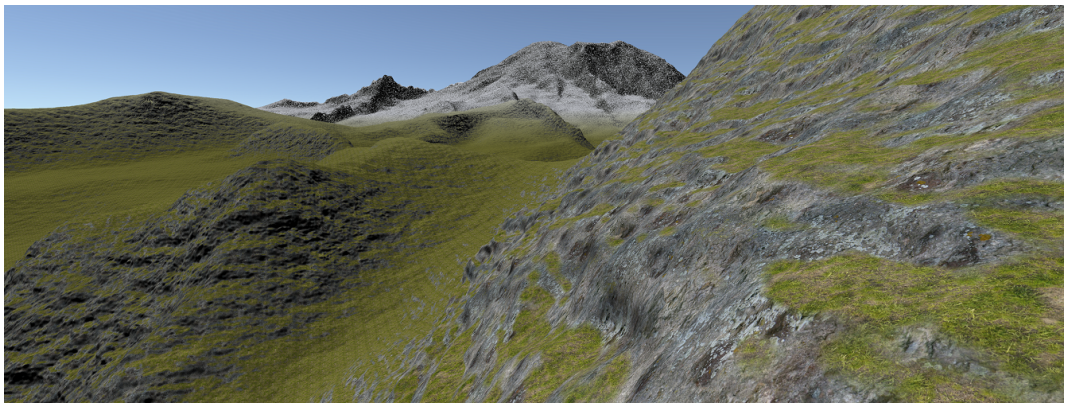

Figure 1: Puget Sound 40 meter dataset resolution showing great amounts of detail generated by our approach.

ABSTRACT

Terrain visualization is a well discussed and documented field of research. However, many issues can be faced with terrain rendering. Applications that must cover vast areas as well as applications that require high detailed visualization can be memory-bound since the required dataset for meeting such demands can become extremely large. Moreover, it is still challenging to find high-resolution real world digital elevation data. That can be an issue when applications such as games or simulations demand a ground-level visualization were the lack of high frequency details have a undesirable visual impact. There are many techniques to overcome those issues and one of them would be to use procedural generation techniques, not to generate the whole terrain, but to increase details by generating high frequency data to fill the gaps. In this paper we present an approach for augmenting and improving terrain visualization by applying procedural generation techniques to low resolution elevation data that is further subdivided by modern tessellation hardware at runtime.

Keywords: terrain, rendering, tessellation, procedural generation, perlin noise, fractional brownian motion.

1 INTRODUCTION

Interactive terrain visualization has been a very active computer graphics research field due to it's great importance in many applications such as games, simulations, film industry, computer-aided design, etc. The goal has always been to render vaster, more accurate and detailed terrain.

Terrain data can be hand made by an artist, procedurally generated or acquired from digital elevation models (DEMs) from real world elevation data. The most common DEM representation is as heightmaps, one of the most simple and used terrain representation formats. Independently from where the data came from, in order to produce detailed visualization, one must have a high-resolution terrain which can become source of storage and performance issues.

In this paper we describe an approach that addresses highly detailed terrain visualization. In order to have more details, one could naively increase input elevation data resolution, what may be an issue for DEMs, where, different from procedurally generated terrain, there is a maximum resolution[1] [2]. Instead, we use hardware tessellation to dynamically subdivide lower resolution input data. Therefore, this approach is suited for minimizing storage requirements, improve visualization of datasets only available at low resolutions and render more realistic scenarios for applications such as games and simulations.

On its own, tessellation adds more primitives, but not more details. We address that by using fractional Brownian motion (fBm) in 3D space with Perlin noise [9], as described by Musgrave [4], to displace the newly created primitives. We also use phong tessellation [2], a popular technique for geometry smoothing, to smooth mountain tops and minimize some common artifacts encountered on low triangle count meshes.

Applying noise throughout the whole terrain does not produce realistic results. Since this work has no intent of correctly quantifying terrain roughness, we used two simple assumptions to approximate it. We assume that flat areas tend to accumulate more sediments and for that reason are less noisy. Based on Brandon and Pinter's article [10], we also assume that elevation has an impact on terrain roughness since average temperature decreases with altitude, making higher peaks less likely to be protected by vegetation.

The paper is organized as follows: section 2 provides background and related works on the subject, in sections 3, 4, 5 and

*e-mail: afrasson@inf.ufsm.br

[1] ASTER GDEM is available at 30 meter resolution at *http://asterweb.jpl.nasa.gov/*

[2] SRTM DEM is available at 30 meter resolution at *http://eros.usgs.gov/*

6 we propose an approach and describe its implementation. Finally, in sections 7 and 8 we evaluate and discuss current results.

2 Related Works

Most works regarding terrain visualization try to either improve rendering times or to generate more realistic landscapes. Some cover both areas [8] [3] and, while presenting new level-of-detail approaches, also address terrain synthesis. Our work focus on procedural generation, but instead of generating the whole terrain as Genevaux et al. [5] and Hnaidi et al. [6], we target to add high frequency details to an existing terrain, similar to Losasso and Hoppe [8] and Dachsbacher and Stamminger [3]. The key difference is that we increase details at run time independently of level-of-detail through the use of tessellation hardware.

Losasso and Hoppe mostly cover terrain level-of-detail in their paper [8], but they also have addressed terrain synthesis by mixing their lod technique with procedural generation, keeping only the coarsest level and adding fractal noise displacement to subsequent levels.

Dachsbacher and Stamminger [3] focused on procedural detail generation at runtime. In their terrain rendering approach, procedural displacement is partly represented in geometry and partly accounted for during shading, similar to bump-mapping technique [1].

Genevaux et al. [5] present a framework for terrain modelling where rivers and rules from hydrology are used to generate a continuous terrain. Their work applies Perlin noise to modulate terrain's shape. They use distance to rivers and elevation data as input for the noise procedure.

Hnaidi et al. presents a method for generating terrains from a set of curves which represent terrain features [6]. They use noise function controlled by the diffused amplitude and roughness attributes to generate details.

3 Displacement Process

We propose an approach where all detail generation process is done on the GPU at runtime. For that we use modern graphics pipeline features, where the backbone of our approach are the tessellation stages. The core algorithm is implemented in the Domain Shader (OpenGL's Evaluation Shader), where vertex displacement is calculated. Figure 2 shows the process we apply for each Domain Shader's input vertex and Figure 4 shows an illustration of this process on a single primitive. Moreover, all texturing and lighting is done on the Pixel Shader stage.

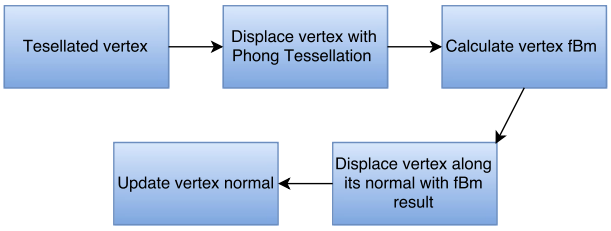

Figure 2: Domain shader displacement process.

4 Terrain Smoothing

The most distinguishable visual artifacts one can notice when rendering terrains are sharp transitions between samples in the shape of sharp peaks and large flat triangles. These issues are usually caused by low resolution input datasets. Some datasets even though available at high resolutions are still not detailed enough for some applications since they cover vast areas resulting in low sample density e.g., Puget Sound, the dataset used in this work, is available at 16385x16385 samples but covers an area of approximately 163kmx163km.

In order to reduce these undesired visual artifacts, we used phong tessellation [2], which is the geometric version of phong normal interpolation, to successfully reduce sharp transitions between samples. Figure 3 shows how phong tessellation can improve visual quality.

Figure 3: Terrain rendered without (top) and with (bottom) phong tessellation. Notice how sharp peaks are sucessfully minimized.

5 Adding Details

Our implementation is based on Musgraves simple fBm and multifractal algorithm [4], both based on Perlin's noise algorithm [9]. Algorithm 1 shows the fBm procedure used by our system. The output from these methods is used to perturb the tessellated terrain mesh as shown on Figures 4 and 5.

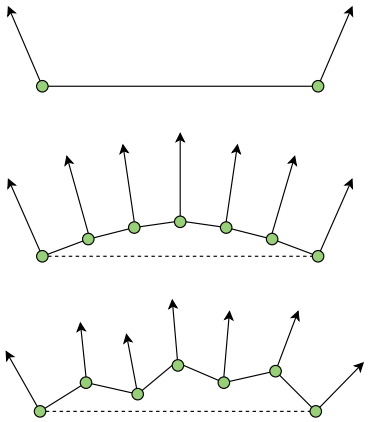

Figure 4: Iterative displacement process. We start with the original surface (top) wich is tessellated and displaced by phong tessellation (middle). Noise displacement is applied at the end on top of phong tessellation (bottom). Green circles represent vertices and arrows its normals.

Regarding the noise procedure, initially we were using Perlin2D noise considering only X and Z components. However, it was resulting in stretched noise whenever applied to steep surfaces (see Figure 6), since the Y component has a slower rate of change on these surfaces. One simple solution to this problem is the use of Perlin3D noise instead. With the three dimensional Perlin noise we can use all vertex components, X, Y and Z, ensuring that the value will not be distorted by steep slopes. Since we want to add details and not fully generate a new terrain, fBm's parameter must be adjusted so that the base terrain shape won't be excessively modified.

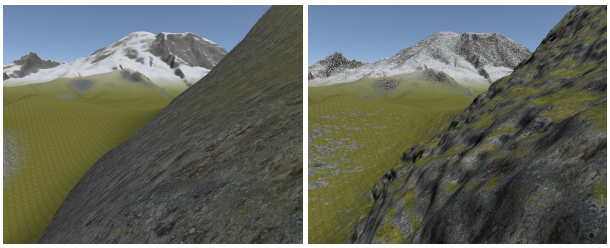

Figure 5: Terrain rendered without (left) and with (right) fbm displacement modulated by slope and elevation.

Input: p ← world space vertex
Input: frequency ← initial frequency
Input: octaves ← number of octaves
Input: lacunarity ← frequency multiplier
Input: gain ← amplitude multiplier
$sum \leftarrow noise(p, frequency)$
$amplitude \leftarrow 1.0$
$range \leftarrow 1.0$
$o \leftarrow 1$
while $o < octaves$ **do**
  $frequency \leftarrow frequency \times lacunarity$
  $amplitude \leftarrow amplitude \times gain$
  $range \leftarrow range + amplitude$
  $sum \leftarrow sum + noise(p, frequency) \times amplitude$
  $o \leftarrow o + 1$
end
$sum \leftarrow sum \div range$
return *sum*

Algorithm 1: FBM procedure.

Finally, we textured the terrain, also at runtime, according to two parameters: slope and elevation. The slope determines if the surface should be textured with rock texture (steep slopes), grass or snow texture, the two available low-slope textures. Furthermore, these low-slope textures are chosen according to the elevation and a user defined threshold. Surfaces below the threshold are textured with grass texture and surfaces above with snow texture. Results can be seen on Figures 5, 6 and 7. Even though terrain is initially correctly mapped with texture coordinates, steep slopes distort textures since there is a considerable amount of displacement. An solution to update texture coordinates according to the displacement applied on each vertex should be further investigated.

6 Results and Discussion

Throughout the development of this project the main test bench was the Puget Sound dataset[3] which covers an area of approximately 163kmx163km and is available at several resolutions. In our tests we used the 40 meter resolution dataset. We chose this dataset because it was previously used in other terrain rendering researches [7] [8], and includes both smooth and rough topology. Most tests were done around Mount Rainier location (Figure 7) and compared to real pictures[4].

Since this work aimed at adding details by tessellating an existing terrain, it was natural to use an already established terrain rendering tool for our tests. The chosen tool was the game engine Unity, for its popularity among game developers and for its native terrain rendering support with a built-in level-of-detail.

[3]We obtained the freely available data from *http://www.cc.gatech.edu/projects/large_models/ps.html*.

[4]Ralph Arvesen's picture available at *https://www.flickr.com/photos/rarvesen*.

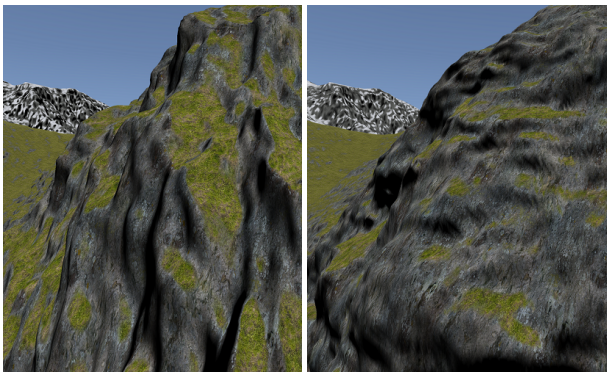

Figure 6: Terrain rendered with Perlin2D (left) and Perlin3D (right). Notice how details are vertically stretched with Perlin2D.

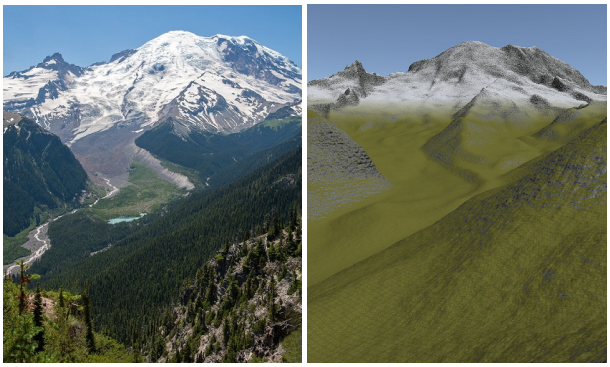

Figure 7: Mount Rainier (left) and it's 40m resolution dataset rendered with our approach (right).

We set-up three different scenes for testing purposes. Figure 7 shows one of these scenes and Figure 8 its rendering performance. We played with two different parameters: tessellation's edge length and the number of octaves. Edge length is used to calculate the amount of tessellation necessary to maintain triangle's screen size relatively constant. Thus, lower values will demand more subdivisions, generating more triangles and, as expected, decreasing performance. Octaves determines the number of octaves of noise that are added together to get the final result. The more octaves, more details can be generated. Therefore, as shown in Figure 8, more octaves will negatively impact performance. As reference, Figures 1, 5, 6 and 7 show details generated by 6 octaves.

7 Conclusion

The association of procedural generation and hardware tessellation allows the use of low resolution datasets while still delivering highly detailed terrain visualization. Our approach can improve visualization whithout any storage overhead, which is an important feature when working with large datasets. Ideally this solution should be used in association with an state-of-the-art level-of-detail technique. Moreover, rendering quality can be adjusted for low-end systems by controlling the amount of tessellation.

8 Future Works

Slope and elevation were used to calculate the amount of displacement generated by the fBm procedure. This approach ensures that high elevations will get rougher details, which resembles rocky

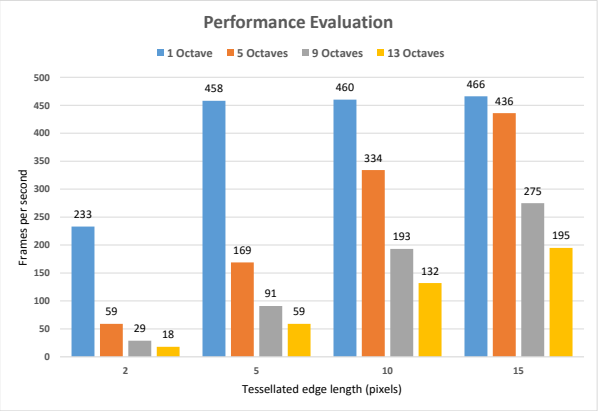

Figure 8: Performance evaluation of our detail generation approach. Tested on a Intel Core i7-4950HQ processor along with a Nvidia GeForce GTX 980 Ti at 1920x1080 screen resolution.

mountains. The same effect appears on steep slopes. One possible improvement would be to also modulate the noise according to surface concavity. Valleys, which are concave, tend to accumulate more sediments and should receive less high frequency noise. In the other hand, mountaintops, which are convex, tend to be rougher and thus high frequencies details should be more frequent. Textures are also dictated by slope and elevation. One could use the same fBm procedure to randomly distribute textures e.g., plains could display several regions with different types of grass.

Acknowledgements

We thank the Brazilian Army for the financial support through the SIS-ASTROS project, developed in the context of the ASTROS 2020 strategic project.

References

[1] J. F. Blinn. Simulation of wrinkled surfaces. *ACM SIGGRAPH Computer Graphics*, 12:286–292, 1978.

[2] T. Boubekeur and M. Alexa. Phong Tessellation. *ACM Transactions on Graphics*, 27(December 2008):1, 2008.

[3] C. Dachsbacher and M. Stamminger. Rendering procedural terrain by geometry image warping. *EGSR'04 Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, pages 103–110, 2004.

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

[5] J.-D. Génevaux, É. Galin, E. Guérin, A. Peytavie, and B. Beneš. Terrain generation using procedural models based on hydrology. *ACM Transactions on Graphics*, 32(4):1, 2013.

[6] H. Hnaidi, E. Guérin, S. Akkouche, A. Peytavie, and E. Galin. Feature based terrain generation using diffusion equation. *Computer Graphics Forum*, 29(7):2179–2186, 2010.

[7] P. Lindstrom and V. Pascucci. Visualization of large terrains made easy. *Proceedings Visualization 2001 VIS 01*, 2001(3):363–574, 2001.

[8] F. Losasso and H. Hoppe. Geometry clipmaps: terrain rendering using nested regular grids. *ACM Transaction on Graphics*, 1(212):769–776, 2004.

[9] K. Perlin. An image synthesizer. *ACM SIGGRAPH Computer Graphics*, 19(3):287–296, 1985.

[10] N. Pinter and M. T. Brandon. How erosion builds mountains. *Scientific American*, 15:74–81, 2005.