Procedural Terrain Detail Based on Patch-LOD Algorithm*

Xuexian Pi, Junqiang Song, Liang Zeng, and Sikun Li

School of Computer, National University of Defense Technology,
410073, Changsha, China
pixuexian@hotmail.com

Abstract. The navigation and rendering of very large-scale terrain are facing a difficult problem that the geometry data and texture data cannot be used directly due to the storage space, computation capacity, and I/O bandwidth. To provide more realistic detail of terrain scene, procedural detail is a good solution. Firstly, this paper introduces a method of procedural geometry based on the terrain tile quad-tree and the Patch-LOD algorithm. Then, the texture generation operator is described and the method of dynamic pre-computation of patch-texture is presented. Finally, the experimental system based on these above ideas and methods has been implemented. The experimental results show that these methods are effective and are appropriate to the development of 3D games and battlefield applications.

1 Introduction

Generation of photorealistic virtual natural scenery is the hot topic of Computer Graphics nowadays. Real-time large-scale terrain rendering has attracted more and more attention for many years, since it is widely used in many applications, such as simulation training and 3D games. There is the conflict of precision and scale in the Large-scale terrain rendering. In the applications of very large-scale terrain scene, the distance between neighbor samples has been restricted. For example, for the Digital Elevation Map(DEM) of a region, whose scope is $160km \times 160km$, its data size will be up to 100 Giga bytes if the distance between neighbor sample vertices is 1 meter. The corresponding texture will be much larger than the geometry data. The geometry data and texture data will go beyond the capacity of current personal computers. In this paper, an algorithm of procedural detail is proposed to deal with this difficulty. Procedural detail, including procedural geometry and procedural texture, can be generated to provide realistic scene without any additional pre-computed data. The rest of the paper is organized as follows. In the next section, we give an overview of related work. Section 3 describes the Patch-LOD Algorithm of Terrain. In

* This research is supported partially by National "973" Foundation Research Plan of China (No. 2002CB312105) and National "863" High Technology Plan Foundation of China (No. 2004AA115130).

Z. Pan et al. (Eds.): Edutainment 2006, LNCS 3942, pp. 913–920, 2006.

Section 4, we detail the procedural detail algorithms. Then, we describe the implementation of our technique and present the results in Section 5. Finally, Section 6 concludes the paper.

2 Relate Works

The most similar work to ours is Carsten's[1]. In Carsten's paper, a static height-map is defined as the global shape, and procedural detail is generated during rendering time. A new concept named "Importance Map" is introduced and the process of generation of procedural detail is driven and controlled by the Importance Map. There are some disadvantages in Carsten's method [1].The pre-computed, periodic turbulence textures are used and render-to-vertex-array must be supported by graphic hardware. In our work, all textures are procedural, not pre-computed and pre-loaded from disk. There are others papers about texture synthesis and procedural detail. A fast method to synthesize natural looking fractal terrain was introduced by Jacob Olsen[2], an algorithm for generating large static texture was presented in[4] and an algorithm of multi-band fractal terrain is initialed in [7], [8], [10], [9], and [11] have discussed the issue about synthesizing of large terrain texture. These papers provide many ideas about how to synthesis terrain texture to enhance the reality of 3D landscape. Our goal was to develop an algorithm that combines the advantages of previous approaches. We want it to be efficient, general, and able to produce high quality landscape for 3D game and other outdoor scene applications. The main advantages of our work are following: 1)It provides procedural geometry, procedural texture simultaneously. 2)The procedural texture is based on patch-LOD algorithm of terrain. 3)The procedural geometry and texture are generated and destroyed dynamically. Due to these advantages, our experimental system's frame rate is up to 50FPS.

3 Patch-LOD Algorithm of Terrain

Since our procedural algorithm is based on patch-LOD algorithm of terrain, the concept of Patch-LOD algorithm has to be introduced at first. With the development of GPU, the overhead of reducing triangle patches with continuous LOD in CPU is more the overhead of rendering patches directly in GPU. Moreover, regular mesh is apt to create display list and vertex array by GPU, with this way we can improve the rendering efficiency. Therefore, the algorithms based on Tiled Quad-tree([3],[4],[6]) have newly advantages in recent years. We organize the whole terrain data into Terrain Tile Pyramid(TTP)(see Fig.1) and isomorphic Terrain Summary Pyramid(TSP) with TTP. The each tile has $(2^n+1)\times(2^n+1)$ vertices and the neighbor tiles overlap a row or a column. Through constructing visible quad-tree and renderable quad-tree, visibility culling and determining LOD level can be implemented simultaneously. During being rendered, different tiles have different levels of detail. For take advantage of the graphics hardware's powerful capacity, the Index-Template is presented and the Joint Index-Template

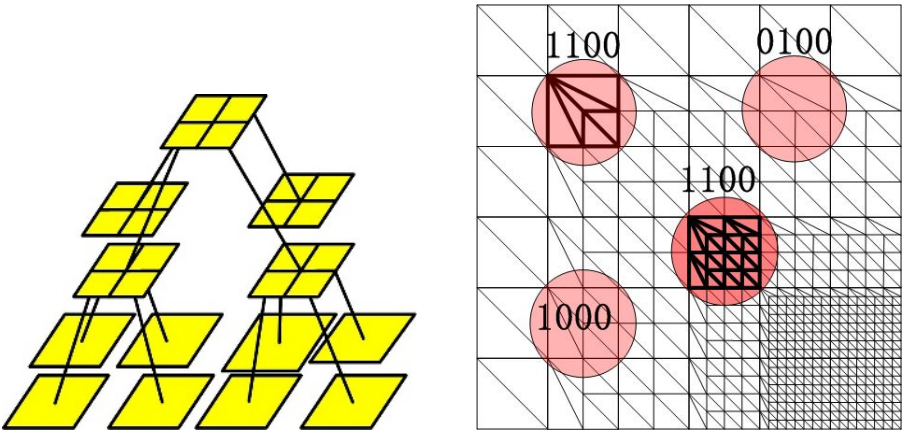

Fig. 1. Terrain Tile Pyramid and Patch-LOD scheme of terrain rendering

are used to stitching the boundary of neighbor tiles. For using Joint Index Template, the level of two neighbor tiles must not be greater than 1. The quad-tree that satisfied this condition is named as "restricted quad-tree". The detail information about Patch-LOD algorithm is illustrated in the paper of CCVRV05[13].

4 Procedural Detail

The procedural detail can be implemented through procedural geometry and procedural texture. In our paper, procedural geometry is provided with Perlin noise, while the procedural texture is based on texture generation operator.

4.1 Procedural Geometry

Perlin Fractal Surface. The Perlin noise is used to provide geometry detail. Perlin noise function was presented by Perlin in 1985[12]. Perlin noise is a smooth noise with point coordinates as input parameters and the noise value as output result. After producing the noise of some discrete points, the noise value of the arbitrary points can be computed through interpolating. Although the Perlin noise generation function is based on random function, it is used in simulation of continuous objects, since its transition between neighbor points is smooth. Fig.2 shows increasing harmonics of 2D Perlin noise. The sum of multiple octaves of Perlin noise results in a fractal noise. The character of the noise will depend on the amplitude relationship between successive octaves of the noise. Equation.1 demonstrates how a fractal noise can be created from multiple octaves of Perlin noise and how all the amplitudes can be represented by a single parameter α. Higher values for α will give a rougher looking noise whereas lower values will make the noise smoother.

$$fnoise(x) = \sum_{i=0}^{octaves-1} \alpha_i \cdot noise(\omega_i \cdot x) \quad (1)$$

Practically, we choose the different frequencies of Perlin noise that depend on terrain tiles LOD to synthesize the procedural geometry detail.

4.2 Mapping Perlin Fractal to Large-Scale Terrain

After generation of Perlin fractal surface of terrain, we will map the fractal surface into large scale terrain. As described above, the process of terrain navigation and rendering is based on the Terrain Tile Pyramid, so the procedural geometry must be expanded from original framework. As shown in Fig.2, the yellow tiles are preloaded tiles from outer storage, which are parts of Terrain Tile Pyramid. The green tiles are procedural terrain tiles, which are generated real-time. Fig.3 is comparison between landscape with and without procedural geometry.The left of Fig.4 has been added with procedural geometry, while the right is plat plain without procedural geometry.

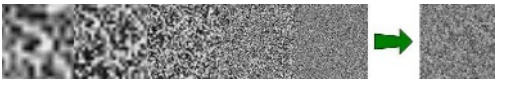

Fig. 2. A sequence of multiple octaves(left)of Perlin noise and fractal surface(right)

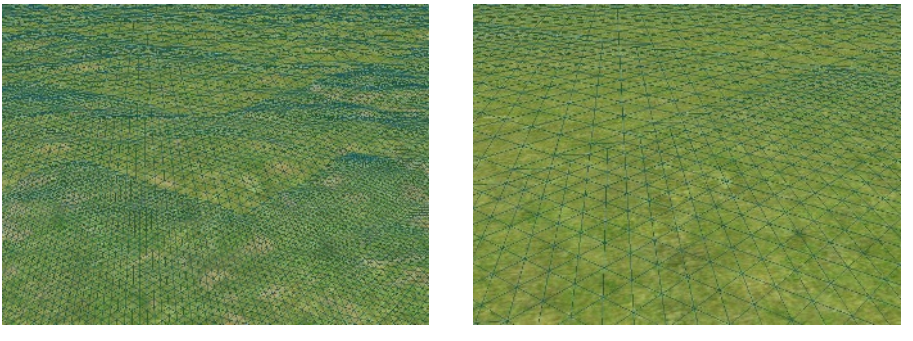

Fig. 3. Comparison between landscape with and without procedural geometry

4.3 Procedural Texture

The larger scale terrain may be up to several thousands kilometers and the terrain data is so huge, and the terrain texture data should take up more memory and disk space. For many virtual scene applications such as 3D games, the static storage of textures is not suitable. Our procedural texture algorithm can be a good solution for such applications. Our algorithm is described as follows.

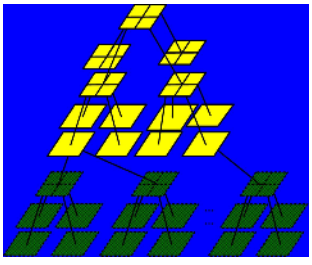

Fig. 4. Procedural geometry

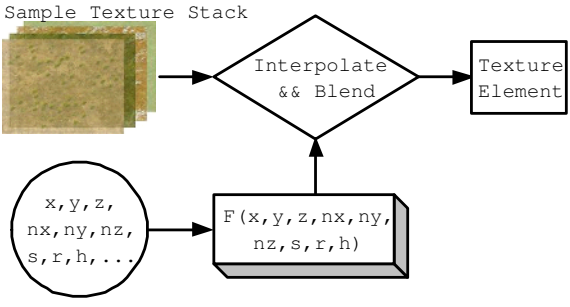

Fig. 5. The texture generation operator

1. Some sample textures have been prepared, and these sample textures, as a whole, are named as Sample Texture Stack(STS). The target texture can be regarded as the composition of some sample textures.
2. The terrain height-map has been added the needed information for any vertex of overall terrain region. Every vertex of the height-map, has the elevation, relative height, normal vector, and other geographic information, such as humidity and rainfall.
3. We construct a texture generation operator as described as Fig.5. The operator is a function as follows: $I = F($ x , y, z, nx, ny, nz, g, s, r, h ,$\cdots$),here (x, y, z) is 3D coordinates of the vertex, and (nx, ny, nz) is the normal vector of height-map in this point. Additionally, g is grads vector, s is a Boolean value which identify whether the vertex is sunny-side or not, and h is float which expresses the humidity. The return value of function F may be an index of the Sample texture stack(STS), if that value is an integer, while the target texture may be interpolation of the two textures of STS if the index may be a faction.

4.4 Dynamic Pre-computation of Patch-Texture

The algorithm of terrain rendering is based on patch-LOD scheme. Every terrain block can be loaded and unloaded by the scheduler. Our procedural texture algorithm can be implemented under the patch-LOD framework. Since the patch-LOD algorithm makes the patch (terrain block) be loaded and unload dynamically, the texture coupled with terrain block can be pre-computed when loading corresponding geometry block and destroyed when unloading the block. As shown in Fig.6, the whole terrain region can be divided into four small patches, and one of these patches can be divided into smaller ones. When a terrain patch is loaded into the terrain block pool, the corresponding texture should be computed. If the terrain patch need not be divided into smaller ones, the pre-generated texture patch is used. Due to the dynamic pre-computation algorithm of terrain texture, the frame rate is up to 50 FPS.

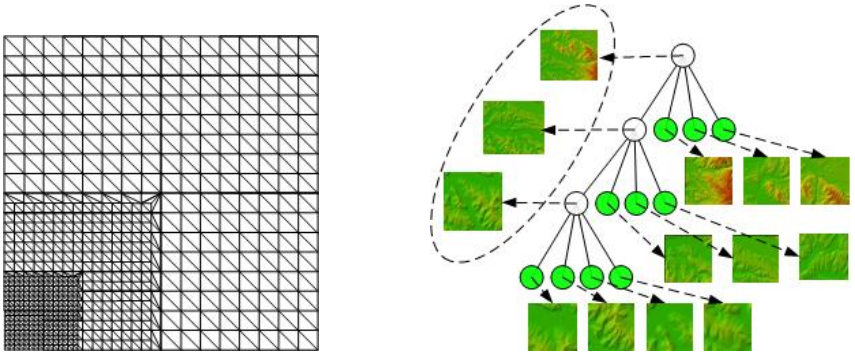

Fig. 6. Patch-LOD Scheme and Procedural Texture

5 Implementation and Analysis

We have carried out a series of experiments and accomplished a navigation system of demo that is based on the algorithms and concepts introduced above. The experiments are carried under the condition of Windows 2000, with Visual C++6.0 and DirectX9.0c. Our computer configure: 1.6G Intel P4,512M RAM, FX5600GPU. The size of terrain height-map is 16385×16385, the interval distance of two neighbor sample vertex is ten meters.

5.1 Implementation of Our Algorithm

The algorithm is based Terrain Tile Pyramid and the Patch-LOD scheme. The process can be divided into two stages: Preprocess and Real-time Process.

- Preprocess
  1. Organize the terrain data, and bind the data of geometry information (gradient and normaletc.) and geographic information.
  2. Construct and load the Terrain Summary Pyramid.
- Real-time process
  1. Construct the multi-resolution quad-tree.The data of tiles of quad-tree are loaded by I/O thread automatically. The terrain surface is composed of tiles of different level of detail, and the texture patch of each newly-loaded tile is synthesized.
  2. Visibility culling through searching the restricted quad-tree.
  3. Compute illumination of terrain scene, and produce the final realistic landscape.

5.2 Experimental Results and Comparison

Some experimental results will be introduced as follows. Fig. 7(left) is a path of navigation; Fig.7(right) is the graph of relation between frame number and frame per second. There are a series of snapshot pictures of our system in Fig.8. The realistic landscape demonstrates that the algorithms we presented in this paper are effective surely.

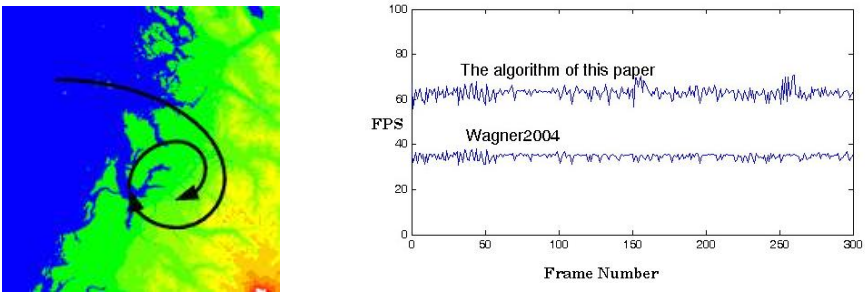

Fig. 7. The relation between the FPS and the frame number during navigation

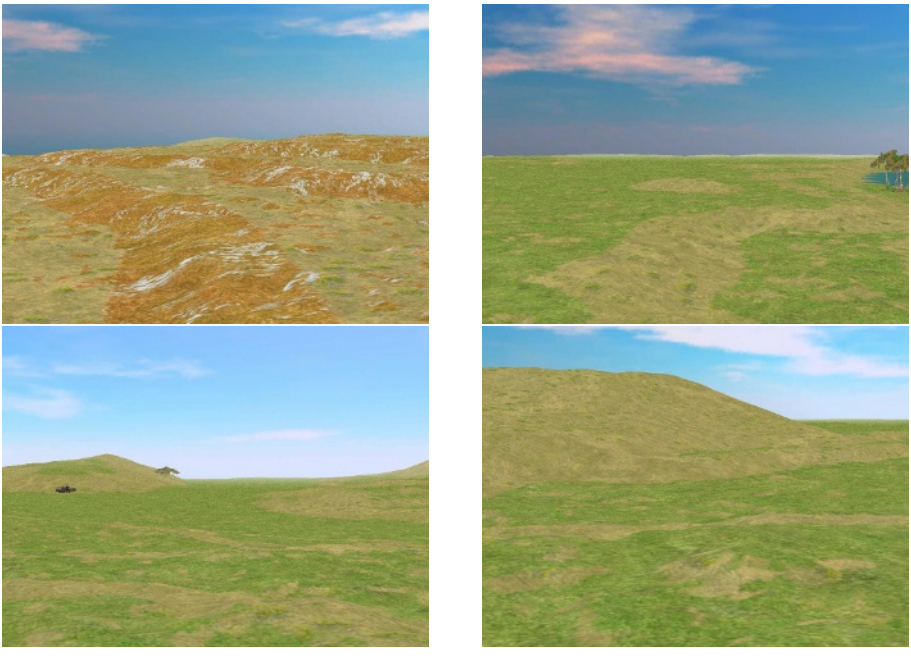

Fig. 8. The snapshots of our experimental system

6 Conclusion

We presented a novel method of procedural geometry and procedural texture. The procedural geometry can be generated through adding Perlin noise, and the procedural texture can be generated through constructing the synthesis operator of procedural texture. Our method is a good solution to the large-scale landscape applications. In the future, we will focus on following issues: improving the synthesis operator of procedural texture, providing various algorithms of procedural detail for various types of terrain.

References

1. Carsten Dachsbacher, Marc Stamminger. Rendering Procedural Terrain by Geometry Image Warping, Eurographics Symposium on Rendering (2004), 2004.
2. Jacob Olsen, Realtime Procedural Terrain Generation, Department of Mathematics And Computer Science (IMADA),University of Southern Denmark, October 31, 2004.
3. BISHOP, L.,EBERLY, D., WHITTED, T., FINCH, M., AND SHANTZ, M. 1998. Designing a PC game engine.IEEE CG&A18(1),46-53.
4. Wagner, D. Terrain geomorphing in the vertex shader. In ShaderX2: Shader Programming Tips & Tricks with DirectX 9. Wordware Publishing. 2004.
5. M. Platings and A. M. Day, Compression of Large-Scale Terrain Data for Real-Time Visualization Using a Tiled Quad Tree, Eurographics Forum, volume 23, 741-759 , December 2004.
6. Zhao Youbing, Zhou Ji, Shi Jiaoying and Pan Zhigeng.2001. A fast algorithm for large scale terrain walkthrough. In: Peng Qunsheng ed. Proceedings of CAD/Graphics'2001. Kunming: International Academic Publishers , 567 572
7. Frank Losasso, Hugues Hoppe, Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids, SIGGRAPH'2004, 2004.
8. DoLLNER, J., BAUMANN, K., AND HINRICHS, K. 2000. Texturing tech-niques for terrain visualization. IEEE Visualization 2000, 227-234.
9. D. Cline, P. Egbert: Interactive Display of Very Large Textures. Proceedings IEEE Visualization '98, 343-350, 1998.
10. T.Huttner,W. Strasser: FlyAway: a 3D terrain visualization system using multiresolution principles.Computers & Graphics, 23: 479-485, 1999.
11. C. C. Tanner, C. J. Migdal, M. T. Jones: The Clipmap: A Virtual Mipmap. Proceedings of SIGGRAPH '98, 151-159,1998
12. Perlin, K. 1985. An image synthesizer. In Computer Graphics (SIGGRAPH '85 Proceedings), B. A. Barsky, Ed., vol. 19(3), 287-296.
13. Pi Xuexian, Yang Xudong, Li Sikun, Song Junqiang, Patch-LOD Algorithm of Terrain Rendering Based on Index Template . CCVRV05, Beijing, PRC, 2005