The Key Technology of Realistic Terrain Construction

Xiao SONG¹,*, Rui-ying PU²,c, Guo-qiang SHI³,d, Shuang-shuang ZHOU⁴,a and Hong-yan QUAN⁴ b

¹School of Automation Science and Electrical Engineering, Beihang University, Beijing, 100191, China
²Beijing Complex Product Advanced Manufacturing Engineering Research Center, Beijing, Simulation Center, Beijing, 100854, China
³Key Laboratory of Intelligent Manufacturing System Technology, Science and Technology on Space System Simulation Laboratory, Beijing, 100854, China
⁴College of Computer Science and Software Engineering, East China Normal University, Shanghai, 200062, China

E-mail: *songxiao@buaa.edu.cn, http://www.buaa.edu.cn/  
c windflowerpray@sohu.com, d sunnyqiang737@163.com  
a zhou_ss@foxmail.com, b hyquan@sei.ecnu.edu.cn

*Corresponding author

Keywords: Terrain, Simulation, Normal, Realistic effect.

Abstract. One of the key technologies in virtual scene generation is how to make a terrain with realistic appearance. In order to overcome the seams from large terrain construction using Virtual Planet Builder (VPB), we acquire digital evaluation data and geospatial imagery data, and make large scale terrain databases. Secondly, we calculate the normal of local particle within every tile’s boundary region from Digital Elevation Model (DEM) from the Earth geometry database. Thirdly, the smoothed normal of every boundary particle is obtained from its neighbors. At last, the height of particle within the boundary region is updated from the smoothed normal. We demonstrate our results with several challenging terrains and provide qualitative evaluation to our method. Some applications based on our provided approach are also demonstrated in the implementation.

Introduction

How to build realistic scenes is very meaningful in wide range of applications such as geographic exploration, 3D film technology and many other fields. And the terrain rendering is the foundation in building such scenes. In early studies of terrain rendering [1] [2], models people constructed were rough, simple and lack of reality due to poor computer hardware processing power. With the development of hardware technology, now people can just use some evaluation data to realize the terrain rendering with high performance [2]. The existing technologies of generate terrains can be divided into several categories [3][4]: surface generation method [5] [6], fractal [7], regular grid method, TIN-grid method[6] [8], etc. When creating compatible large scale paged terrains from geospatial imagery and digital elevation maps using VPB provided by Open Scene Graph, it always brings about a problem that the terrain created by VPB looks like unsmoothed and discontinuous. In this paper, we present an efficient approach to construct realistic terrain with rich details to overcome the seams problem existing in the traditional mechanism used in OSG.

Major Headings Realistic Terrain Construction Method

Our approach has been proved to be convenient and with better time performance.
Obtaining Digital Evaluation and Texture Data

In the pre-process, we adopt the strategy of stitching geospatial terrain tiles for large scale 3D geometry terrain. Our provided method adopts following steps in the preprocessing [9]:

**Step 1:** Acquiring some geospatial tiles file according to customizing requirements for terrain.

**Step 2:** Using the following command of GDAL to stitch all acquired tiles for large geospatial terrain file: `gdalwarp.exe tile1.tif tile2.tif tile3.tif terrain.tif`. Where tile1.tif tile2.tif tile3.tif are the customized geospatial tiles, terrain.tif is the target stitched result.

**Step 3:** Doing grid processing to digital evaluation data in GeoTiff format with following command: `gdaladdo.exe –r average terrainfile.tif 2 4 8 16 32`. Where 2 4 8 16 32 denote numbers mean the zoom levels, -r means resampling.

**Step 4:** Getting the geospatial imagery data according to the latitude and longitude coordinates of the specified range. Then, use GDAL to modify the resolution of the acquired geospatial imagery data according to the coordinate information of the DEM data.

**Step 5:** Using osgdem tool to build large scale pagedlod terrains with texture from the following command: `osgdem.exe --xx 10 --yy 10 -t terrain_texture.tif --xx 10 --yy 10 -d terrain.tif -l 8 -v 0.05 -o terrain.ive`. Where --xx and --yy indicate the size of the graph contained in each pixel, and -l means the levels of the pagedlod database. -v represents the scale factor. terrain.ive is terrain result.

**Calculate the Normal of Boundary Particle**

To overcome seams from tiles in stitching using OpenSceneGraph, we provide a new algorithm based on the tile normal-controlling. We select the particles in the boundary region (Shown in Fig. 1). \( C \) is the particle in the boundary region, and \( A, B, X, Y \) are the 4-neighbour particles of \( A \), and \( N_A, N_B, N_X, \) and \( N_Y \) are their normals.

![Figure 1. Example of boundary region.](image)

We recalculate the normal of particle \( C \) in the boundary region instead of its original normal using formula 1. Here we use \((x_n, y_n, z_n)\) stands for the component of the particle \( C \) normal \( N_c \). Any boundary regions of a tile, including left, right, up and down, have four different neighbor, the instance of left neighbor is shown in Fig. 2.

\[
N_c = \frac{1}{4}(N_A + N_B + N_X + N_Y)
\]  

(1)

![Figure 2. Left neighbour of a tile.](image)
In Fig. 2, the shaded part in both tiles is the boundary region of the tile and the width of the boundary is basically one fifth of the corresponding tile.

**Recalculate the Height of Local Particle**

For each particle in every boundary region, after having calculated its normal, we update the particle height. As shown in Figure 3, for particle C, particle A and particle B are its left and up neighbours.

![Figure 3. Particle normal.](image)

In Fig.3, \( n \) stands for the normal of particle \( C \), and we denote the coordinates of \( A \), \( B \) and \( C \) as \((x-1, y, h_A), (x, y-1, h_B)\), and \((x, y, h_C)\), respectively. Then the vectors \( V_{AC} \) and \( V_{BC} \) are calculated as

\[
V_{AC} = (1,0, h_C - h_A)
\]

(2)

\[
V_{BC} = (0,1, h_C - h_B)
\]

(3)

Then, we can obtain the height value of particle \( C \) from the dot product of vector \( V_{AC} \) and vector \( V_{BC} \) as

\[
h_C = \frac{z_A(h_C + h_B) - (x_A + y_A)}{2z_A}.
\]

**Realistic Terrain Construction Algorithm Based on Tiles Based Normal-control**

Our terrain construction algorithm can be summarized as follow:

**Step 1:** Acquire digital evaluation data and geospatial imagery data, and make large scale terrain databases.

**Step 2:** Recalculate the normal of local particle in every tile boundary region.

**Step 3:** Smooth the normal of every particle in the boundary region using its neighbors.

**Step 4:** Update the height of every particle in the boundary region from the smoothed normal.

**Step 5:** Finally reconstruct the whole terrain scene to ensure every tile is updated with the new height value to acquire realistic effect.

**Experiments**

Quantitative evaluations of our approach performance and several applications are provided. The data we use is from the SRTM 90m Digital Evaluation DataBase [10]. Our hardware platform is PC with Intel (R) Core (R) 2.7GHz CPU, 4 GB memory. The reconstruction result is rendered with Open Scene Graph [11].

**Experiment Results**

We apply our implementation on three groups of large scale terrains. The results of three groups are shown in Fig.4. The number of tiles in every terrain is unity of 21845 in every group. The first and the third group terrain have been mapped texture, while the second has not. Fig. 4 shows the results of the constructed results. In each group, the top one of each group is result form OSG engine without smoothing. The artifact with seams can be seen clearly, while the bottom is our improved result. It can be seen that most seams are removed from the whole scene after using our improved approach.
Time Performances

In addition, we use three groups of terrain to test the time performance of our improved approach. Group 1, 2 and 3 all have 21845 tiles in their databases, while group 2 is not mapped with texture. As shown in Table 1, for every group, the average cost time of our approach is less than our expectation while the final results remain realistic and keep rich details.

Table 1. Cost time of the improved approach.

<table>
<thead>
<tr>
<th>Group</th>
<th>Texture</th>
<th>Covering area(m²)</th>
<th>Average Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Y</td>
<td>0.605*10⁴</td>
<td>21ms</td>
</tr>
<tr>
<td>2</td>
<td>N</td>
<td>0.605*10⁴</td>
<td>14ms</td>
</tr>
<tr>
<td>3</td>
<td>Y</td>
<td>2.38*10⁴</td>
<td>22ms</td>
</tr>
</tbody>
</table>

Conclusions and Future Work

In this work, we provide a solution of realistic terrain construction. The highlight of our work includes both realistic effects for overcoming the seams from tiles stitching and time performance. 2D-3D consistence study is the considered problem in the further improving work.

References


