Real-Time Terrain Rendering Framework for GIS Applications

Dong-Soo Kang* Eun-Seok Lee** Byeong-Seok Shin***

Abstract Real-time 3D visualization of terrain data is one of the important issues in GIS(Geographic Information System) field. We present a real-time terrain rendering engine that can use several types of GIS data source such as DEM(Digital Elevation Map), DTED(Digital Terrain Elevation Data) and LIDAR(Light Detection And Ranging). Our rendering engine is a quadtree-based terrain rendering framework with several acceleration modules. This can generate an ocular and binocular image. Also it can be applied to the flight simulation, walk-through simulation and a variety of GIS applications.

Keywords : Terrain Rendering, GIS Application, Quadtree

1. Introduction

Terrain visualization is essential to present real environment in GIS, flight simulation[1] and 3D computer game. It is necessary to generate a more realistic scene in traditional application field such as virtual reality. In order to visualize a massive spatial data, we need a terrain rendering engine including massive data processing and rendering module with high-performance. The engine has to provide integrated I/O processing function since the spatial data might be presented as various types. And it has to become an integrated system with flexible interface which can cooperate with the spatial database, various devices and applications.

We present a real-time terrain rendering engine that guarantees a real-time visualization and be applied to a variety of applications. It is composed of a modeler, a renderer, a acceleration module and application interface. Firstly, the modeler takes charge of input and transformation of regular dataset as well as irregular dataset. It performs a compression and modification of data for acceleration. Second, the renderer is equipped with a quadtree-based terrain rendering core[2, 3, 4]. It is extended to GPU-based rendering from conventional CPU-based quadtree rendering method and is designed to perform a load balance between CPU and GPU. The acceleration module includes a uniform grid based propagation of quadtree with geometry reconstruction on GPU. Base on this feature, it can perform CLOD(Continuous Level of Detail) and block-based view-frustum culling using *ef*-buffer[5, 6]. Finally, the interface module is composed of sub-functions to support various display devices[7]. It has modules to present a terrain image on stereo display device[8, 9, 10, 11] and hemisphere display device[12].

In Sect. 2, we explain the real terrain visualization engine framework in detail. In Sect. 3, we show several applications using our framework. Lastly, we conclude our work.

2. Terrain Rendering Engine

The proposed integrated framework is designed to render a real terrain, which is composed of several functions shown in figure 1. The renderer contains an essential algorithm to visualize a terrain data and is designed to offer usability to an applications. The modeler is designed to allow rapid access to an input data source. Also, the acceleration module has a variety of acceleration techniques for terrain visualization.

† This research was supported by a grant (07KLSGC05) from Cutting-edge Urban Development - Korean Land Spatialization Research Project funded by Ministry of Construction & Transportation of Korean government

* Ph. D candidate, Department of Computer Science and Information Engineering, Inha University, gagalchi@msn.com
** Master student, Department of Computer Science and Information Engineering, Inha University, elflee77@hanmail.com
*** Associate Professor, Department of Computer Science and Information Engineering, Inha University, bsshin@inha.ac.kr(corresponding author)

Manuscript received : 2009.05.12 Revised : 2009.06.19 Accepted : 2009.06.20

Lastly, the application interface includes several modules that enable to cooperate with a variety display devices.

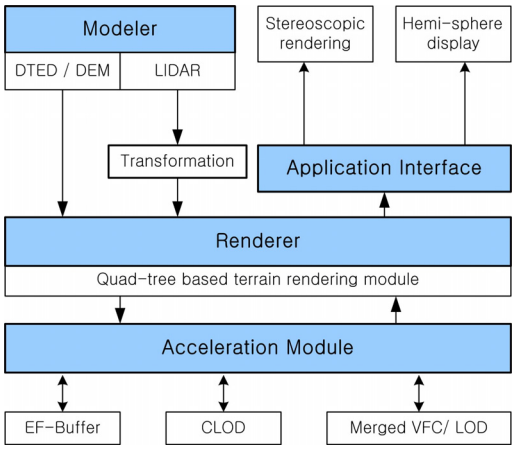

Fig. 1. Block Diagram of Our Terrain Visualization Engine

2.1 Modeler

There are two typical data sources in terrain modeling; regular sample data (DEM and DTED) and irregular sample data (LIDAR). We can achieve real-time rendering of regular sample data by using the spatial data structure such as quadtree and ROAM(Real-time Optimally Adaptive Meshes)[13, 14]. LIDAR scanner mounted on an airplane or a helicopter can produce a high-definition height field data[15, 16, 17]. There are several methods to generate original geometry from scattered point set such as the Delaunay triangulation[18, 19, 20, 21]. Even though it shows the surface detail with high resolution, it takes a long time to reconstruct geometric information from irregular samples. It is hard to apply the dataset to real-time application even in the case of exploiting highly optimized method. So, we design the modeler that can manipulate both a regular and irregular dataset.

In order to build up the integrated modeler for arbitrary input data, we convert an irregular data into a regular data using pseudo-grid[22, 23] in figure 2. Followings are converting process. Firstly, we eliminate a noise from input dataset since they do not contribute to the final results. We perform interpolation and noise removal of resampled data to generate a set of uniform grids containing height information. Finally, the modeler transmits a result dataset to renderer. Since resampling rate leads up the number of vertices and control points a rendering performance, we set the value as user-define parameter. If the rendering engine requires fast rendering, we set the resampling rate as a small value. Also, when requiring high quality rendering image, we set it as a large value. However, the coarser resamling may produce holes in generated grid because of a specific character of irregular data. We can solve this problem using real-time bilinear interpolation.

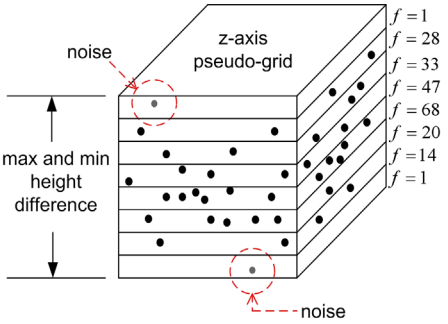

Fig. 2. An Example of a Pseudo-grid in z-axis for an Irregular Dataset

2.2 Renderer

We apply an acceleration technique to our engine using advantages of GPU. We present a quadtree-based terrain visualization method working on the GPU with specially designed data structure; the error buffer and flag buffer named *ef*-buffers. Figure 3 shows a diagram of our terrain visualization pipeline. In pre-processing step, error metrics are computed in world space and they are stored into the error buffer. In rendering step, LOD selection and view-frustum culling are performed by evaluating the error metrics. The result is stored into the flag buffer. Also, to remove cracks, the flag buffer should be refined. To generate the terrain mesh, we traverse the refined flag buffer hierarchically using the block-based method. However, since the updated flag buffer was generated without relationship of neighboring nodes, cracks might occur in a result of triangulation. So it is not suitable for a fine terrain mesh. To solve this problem, we refine the flag buffer to make a restricted quadtree by two criteria. First, flag of node is enabling when a least on edge is enabled among the four adjacent edges to the node. Second, flag of the edge is enabling when at least on node is enabled among the two adjacent nodes to edge of upper level. And we generate a triangle-fan for each node according to visibilities of

vertices in the flag buffer. Figure 4 is a resulting image of terrain using our method.

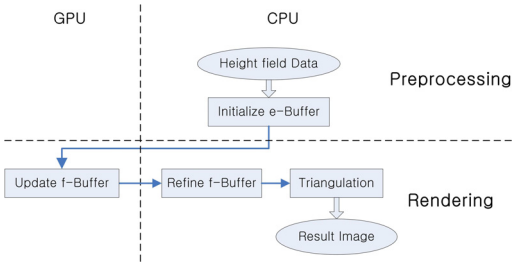

Fig. 3. A Procedure of Our Terrain Visualization Method using Quadtree

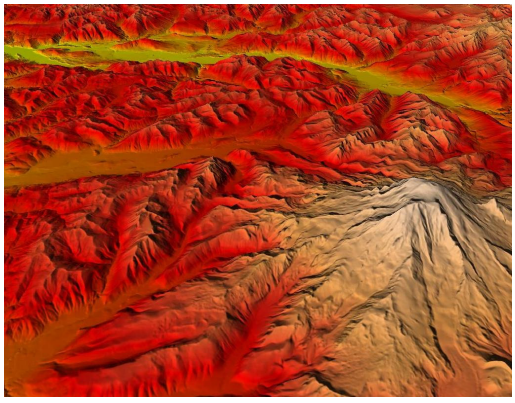

Fig. 4. Result Image using ef-buffer

In general, terrain rendering engine requires a simplification method since the visualization of real terrain uses a huge amount of dataset. Therefore, we exploit a quadtree-based CLOD method. Firstly, we calculate error metric recursively for all leaf nodes while traversing height field data. Secondly, we propagate the computed error metric of leaf nodes to their parent nodes. Then, we determine whether current node can have child nodes or not by applying accuracy level determination function to each node using a set of error metric calculated in previous step. We generate triangle mesh while recursively traversing the quadtree made from height field. Finally, we generate the triangle mesh to display. Determining whether subdivision is required in leaf nodes, we have to consider the distance from viewer to a specific point on a terrain. As show in figure 5, node size(d) should be enlarged as the distance(l) increases. We can control the size of node with regard to the distance to satisfy $l/d<C$. The C is a minimal desired resolution of image. The number of triangles is determined by the value of C. When we set a large value, more number of triangles will be produced from quadtree since it should be subdivided to smaller node. With second criterion we want to increase the resolution for regions of high surface roughness. When dropping one level of the hierarchy, new error is introduced at exactly adjacent surrounding points. In here, the adjacent points are a center point of the quadtree node and four midpoints of its edges. An upper bound to the approximation error can be given by tasking the maximum of the absolute values of the elevation difference. The error can now be computed by pre-calculating the maximum of absolute values of these elevation differences, which is called $d2$. Base on these, we make an equation for LOD selection of terrain as equation 1.

$$f = \frac{l}{d \times C \times \max(c \times d2, 1)} \qquad \text{[equation 1]}$$

In equation, c-value is a pre-defined global resolution and $d2$-value is an error that is computed by pre-calculating the maximum of the absolute values of these elevation differences. If the f-value is smaller than 1, we subdivide a node.

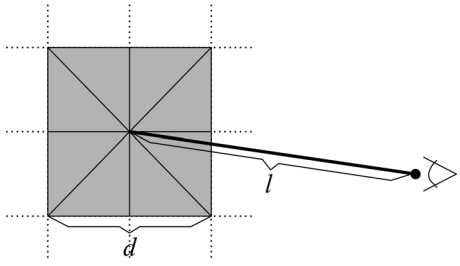

Fig. 5. Global Resolution Criterion: Distance(l) versus Size of Quadtree Nodes(d)

2.3 Acceleration Module

2.3.1 *ef*-Buffer Module

We apply an acceleration technique to our engine. In conventional quadtree-based terrain rendering method, most process is performed on the CPU. However, the *ef*-buffer is a method working most operation on the GPU. The *ef*-buffer is composed of an error buffer and a flag buffer. The error buffer is stored geometry error value for each block used in previous quadtree method on CPU. The *ef*-buffer is textures that have the equal to the size of input height field data. The flag buffer is a texture that

stores an error value of viewing condition. The error value is computed by projection of the image space with the value which is stored into error buffer. At first, the error buffer stores error values for nodes and edges of entire level on quadtree. The view-frustum culling is performed by refining flag buffer. We can implement as parallelized process in vertex shader since the flag buffer is not a hierarchical data structure. Finally, the flag buffer calculated by above criterion is used as height field data to generate a terrain mesh.

2.4 Application Interface

2.4.1 Stereoscopic Rendering Module

Even though general 3D terrain rendering offers depth cue of an object, it cannot provide immersive three-dimensional effect to user. So, recently developed simulators often use a stereoscopic binocular disparity[24]. For the visualization of stereoscopic terrain, we apply an on-axis based side-by-side stereoscopy method. However, the stereoscopy is difficult to guarantee a real-time rendering since it generates a pair of images for binocular of human in a frame. In order to solve this problem, we use an acceleration technique using merged view-frustum culling and merged LOD selection. It is possible to render a stereo terrain scene in real-time.

2.4.1 Hemisphere Display Module

We implement the warping algorithm for terrain image using quadtree-based engine for hemisphere display device. This device is widely used in FPS(First Person Shooting) game or virtual experience simulation. This generates images with mathematical model such as fisheye mapping and sphere mapping[25]. This model is designed to fit a hemisphere display devices. However, in warping processing, the complexity of computation and reconstruction of mesh cause performance degradation. To solve this problem, we implement modules that are possible to deal with warping operation for each vertex as parallelization using vertex shader in real-time. In additional, we can control a curvature which is calculated to fit an arbitrary display device.

3. Results

All applications are performed on a PC equipped with Intel Core2Duo 6400 2.13GHz CPU and 2GB main memory and NVIDIA Geforce™ 8800Ultra. The viewport size is 1024×768 for single image and 2048×768 for stereo image. We use the Puget Sound and the Grand Canyon dataset of which the resolution is 512×512 as regular dataset, as well as the LIDAR dataset of Inha University with 3,408,187 points.

Figure 6 illustrates the result image using stereo module of our engine. Average rendering speed is 78*fps* while applying acceleration technique.

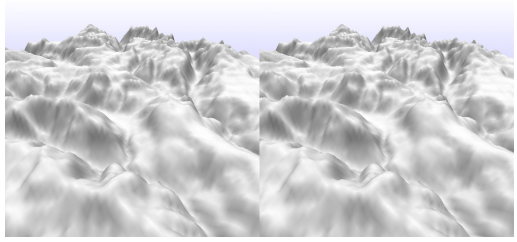

Fig. 6. Stereoscopic Terrain Visualization (Grand Canyon data - 78*fps*)

Figure 7 shows a result image of irregular spatial dataset. We use the LIDAR dataset of Inha University and each building is generated by the modeling of real data. This application can be visualized as walk-through simulation and is guaranteed speed of 60*fps*.

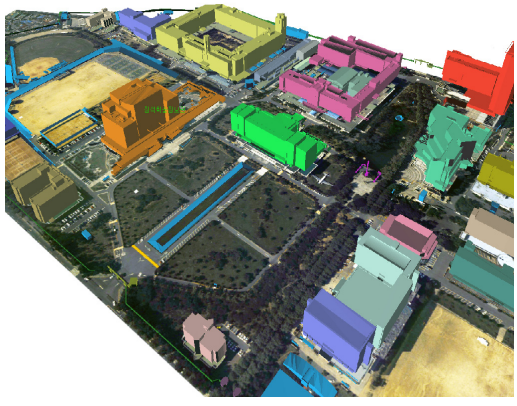

Fig. 7. Result Image of Representation using Spatial Data(Inha University data - 60*fps*)

Figure 8 is a flight simulation using our terrain engine. A cockpit is generated by GUI(Graphic User Interface) module in engine. This application is used of a control interface for simulation such as Logitech ™ WingMan Force3 joystick. Average rendering speed of simulation is over 100*fps*.

Fig. 8. Flight Simulation
(Grand Canyon -100*fps*)

Figure 9 shows a warped image of terrain for hemisphere display. Interface of this module is designed for hemisphere screen device with diameter of 1.5*m* and quarter reflecting mirror with radius of 30*cm*. For real-time fisheye mapping, we use a vertex transformation by vertex shader on GPU. Average rendering speed of this application is 45*fps*.

Fig. 9. Terrain Visualization for Hemisphere Display Device(Puget Sound data - 45*fps*)

4. Conclusion

Recently, the real terrain representation is important factor in the GIS applications. The most important issue is to produce realistic and high resolution image in real-time. To render various terrain dataset such as LIDAR, DTED and DEM in real-time, we propose the terrain rendering framework. It is an integrated terrain visualization engine that is included in the modeler, renderer, acceleration and application interface module. This engine can be performed the flight simulation and walk-through simulation with real-time correlation of spatial DB, interface to hemisphere display device by image warping and real-time visualization of stereo image through the acceleration module. The rendering core in engine guarantees real-time processing for GIS applications through the acceleration module and load balancing both CPU and GPU. Experimental results show that our engine can be accomplished a flexible interface with various applications.

Reference

[1] Zhang, Y., Cong, D., and Han, J., "Real-time Rendering of Large-scale Terrain in the Flight Simulator," Proceeding of the 2007 IEEE International Conference on Integration Technology, 2007, pp. 426-431.

[2] Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L., Faust, N., and Turner, G., "Real-time, Continuous Level of Detail Rendering of Height Fields," Proceedings of ACM SIGGRAPH, 1996, pp. 109-118.

[3] Röttger, S., Heidrich, W., Slusallek, P., and Seidel, H., "Real-Time Generation of Continuous Levels of Detail for Height Fields," Proceedings of 6th International Conference in Central Europe on Computer Graphics and Visualization, 1998, pp. 315-322.

[4] Dick, C., Schneider, J., and Westermann, R., "Efficient Geometry Compression for GPU-Based Decoding in Realtime Terrain Rendering," Computer Graphics & Visualization Forum, 2009.

[5] Shin, B., and Choi, E., "An Efficient CLOD Method for Large-Scale Terrain Visualization," Proceedings of the Entertainment Computing - ICEC, Vol.3166, 2004, pp. 592-597.

[6] Chang, H., and Shin, B., "Hardware Acceleration of Terrain Visualization Using *ef*-Buffers," International Symposium on Computer and Information Sciences, 2006, pp. 316-324.

[7] Hodges, L., "Tutorial: Time-Multiplexed Stereoscopic Computer Graphics," IEEE Computer Graphics and Applications, Vol.12 No.2, 1992, pp. 20-30.

[8] Patoor, S., and Wopking, M., "3-D Display: a Review of Current Technologies," Displays, Vol.17, 1997, pp. 100-110.

[9] Woods, A.J., "Optimal Usage of LCD Projectors for Polarised Stereoscopic Projection Stereoscopic Projection," Proceedings of Stereoscopic Displays and Virtual Reality System VIII, Vol.4297, 2001, pp. 5-7.

[10] Melzer, J.E., and Moffitt, K., Head Mounted Display: Designing for the User, McGraw-Hillm, New York, 1997.

[11] Halle, M., "Autostereoscopic Displays and Computer Graphics," Computer Graphics on Proceeding of SIGGRAPH, 1997, pp. 58-62.

[12] http:// www.geodomas.com

[13] Duchaineau, M., Wolinsky, M., Sigeti, D., Miller, M., Alrich, C., and Mineev-Weinstein, M., "ROAMing terrain: Real-time Optimally Adapting Meshes," IEEE Visualization, 1997, pp. 81-88.

[14] White, M., "Real-Time Optimally Adapting Meshes: Terrain Visualization in Games," International Journal of Computer Games Technology, Vol.2008, 2008, pp. 7.

[15] Wehr, A., and Lohr, U., "Airborn Laser Scanning-an Introduction and Overview," ISPRS Journal of Photogrammetry and Remote Sensing, Vol.54, 1999, pp. 68-82.

[16] Burman, H., "Adjustment of Laser Scanner Data for Correction of Orientation Errors," Proceedings of XIX™ congress of ISPRS, 2000, pp. 125-132.

[17] Casella, V., and Spalla, A., "Estimation of Planimetric Accuracy of Laser Scanning Data," Proceeding of XIX™ congress of ISPRS, 2000, pp. 157-163.

[18] Cohen-Steiner, D., Colin de Verdiere, E., and Yvinec, M., "Conforming Delaunay triangulations in 3D," Proceedings of the 18th Annual Symposium on Computational Geometry, 2002, pp. 199-208.

[19] Aggarwal, A., Guibas, L., Saxe, J., and Shor, P., "A Linear-time Algorithm for Computing the Voronoi Diagram of a Convex Polygon," Proceedings of the 19th Annual ACM conference on Theory of Computing, 1987, pp. 39-45.

[20] Attali, D., Boissonnat, J.D., and Lieuter, A., "Complexity of the Delaunay Triangulation of points on surfaces the smooth case," Proceedings of the 19th Annual Symposium on Computational Geometry, 2003, pp. 201-210.

[21] Akenine-Moller, T., and Haines, E., Real-Time Rendering, Second Edition, AK PETERS Ltd., Natick, 2002.

[22] Cho, W.S., Jwa, Y.S., Chang, H.J., and Lee, S.H., "Pseudo-Grid Based Building Extraction Using Airborne LIDAR Data," Proceedings of ISPRS, Vol.XXXV, 2004, pp. 298-301.

[23] Soo Hee Han, Joon Heo, Lkhagva, Enkhbaatar, "Efficient Processing of Huge Airborne Laser Scanned Data Utilizing Parallel Computing and Virtual Grid," Journal of Korea Spatial Information System Society, Vol.10, No.4, 2008.

[24] Güdükbay, U., and Yilmaz, T., "Stereoscopic View-Dependent Visualization of Terrain Height Fields," IEEE Transaction on visualization and Computer Graphics, Vol.8 Isssue.4, 2002, pp. 330-245.

[25] Feng, L., Micheal, G., "Automatic image retargeting with fisheye-view warping," Symposium on User Interface Software and Technology, 2005, pp. 153-162.

Dong-Soo Kang

2005 Dept. of Computer and Information Engineering, Inha University(B.S.)
2007 Dept. of Computer and Information Engineering, Inha University(M.S.)
Present Ph.D. Candidate, Dept. of Computer and Information Engineering, Inha University.
Research Interests : GPU-based Volume Graphics, Medical Imaging and Parallel Processing.

Eun-Seok Lee

2008 Dept. of Computer and Information Engineering, Inha University(B.S.)
Present Master Student, Dept. of Computer and Information Engineering, Inha University
Research Interests : Interactive Rendering of 3D Graphic Models

Byeong-Seok Shin

1990 Dept. of Computer Sicence and Engineering, Seoul National University (B.S.)
1992 Dept. of Computer Sicence and Engineering, Seoul National University (M.S.)
1997 Dept. of Computer Sicence and Engineering, Seoul National University(Ph.D.)
Present Associate Professor, Dept. of Computer and Information Engineering, Inha University
Research Interests : Realtime Rendering, Volume Graphics and Medical Imaging.