ADAPTIVE LEVEL OF DETAIL FOR LARGE TERRAIN VISUALIZATION

Fuan Tsai[12*] and Huan-Chih Chiu[2]
[1]Center for Space and Remote Sensing Research
[2]Department of Civil Engineering
National Central University
300 Zhong-Da Rd.,Zhongli,320 Taoyuan Taiwan,China -
ftsai@csrsr.ncu.edu.tw; 953202068@cc.ncu.edu.tw

KEY WORDS: Level of Detail, Mesh Refinement, Digital Terrain Model, Visualization, 3D GIS

ABSTRACT:

This paper presents a sophisticated tile-based terrain rendering system for large-area terrain visualization. The developed system employs quadtree algorithm to create multiple levels of details (LOD) of terrain tiles. The visualization determines visible tiles by view frustum culling and renders the scene efficiently. A key issue in quadtree-based LOD generation is the thresholding for different levels. A novel thresholding scheme based on calculated ground sample distances is proposed and provides better LODs with screen resolution. In addition, a nested LOD system is used to improve performance, especially during the initial stage of a visualization of large datasets. T-junctions caused by discontinuities among adjacent tiles are also effectively eliminated with the developed mesh refinement algorithm. Combined with texture LODs, the developed system can produce seamless visualization of large terrain datasets with high rendering performance. It provides near real-time terrain visualization capability for large-area demonstration and applications.

1. INTRODUCTION

Three-Dimensional (3D) terrain rendering is one of the most important components in the visualization of cyber city and other 3D GIS applications. When dealing with large-area terrain visualization, the vast amount of data may exceed the rendering capability of graphic hardware and cause poor performance of the system. Most importantly, in real-time visualization applications, the data resolution is much higher than screen, thus result in data redundancy and lower the efficiency. Furthermore, it may produce aliasing artefacts when rendering dense meshes. In order to reduce the number of polygons, a few researches have proposed different algorithms based on level of detail (LOD) for general 3D triangulated meshes (Cignoni et al., 1998; Luebke et al., 2003). Losasso and Hoppe (2004) categorized them as the following four categories:

1. Irregular meshes (triangulated irregular networks, TIN) (e.g. Cignoni et al., 1997).
2. Bin-tree hierarchies (longest-edge bisection, restricted quadtree, hierarchies of right triangles) (e.g. Blow, 2000).
3. Bin-tree regions (coarser than Bin-tree hierarchies) (e.g. Cignoni et al., 2003).
4. Tiled blocks (square patches that are tessellated at different resolutions) (e.g. Tsai et al., 2006).

Tile-based approaches have become popular for large-area terrain visualization because the original DEM data can be pre-processed by tiles and only visible tiles need to be rendered in runtime. When rendering, the data will be loaded and rendered quickly without further effort for triangulation. In a previous study (Tsai et al., 2006), a set of LODs was generated for each tile using a dynamic quadtree algorithm. When rendering the terrain, view frustum culling was used to decide visible tiles and computed view importance to assign suitable tile LOD. This study further improves the developed tile-based terrain visualization system. Firstly, the original thresholds for quadtree were determined by the height difference in each tile, but it is difficult to connect the LODs with view importance. A new thresholding scheme based on view-dependent image-space error metric (Lindstrom et al., 1996) is proposed to achieve more reasonable LOD generation.

Secondly, conventional LOD systems often divide the data set into small tiles geographically. This may result in poor performance or abrupt LOD changes when dealing with large-area visualization projects, especially during the initial stage. To address this issue, a nested LOD system is developed to create a finer set of LOD tiles and a coarser LOD set for quick representation of large areas. The switch between the two LOD sets is established according to viewer altitude, distance and resolution dependency etc. Thirdly, when visualizing a terrain by mesh tiles, there are usually discontinuities along tile edges, causing so-called T-junctions among different tiles. A mesh-merging algorithm is also proposed to refine the determined LOD meshes in order to eliminate T-junctions. Augmented with these improvements, the developed system will produce seamless landscape scenes consisting of multiple tiles of different LOD layers more efficiently.

2. DATA PRE-PROCESSING

In tiled-based visualization, each terrain tile should be processed before runtime. Fig. 1 explains the workflow of data pre-processing, including DEM pre-processing and processing of texture images. Two nested LOD sets will be generated for the DEM data. The first is called Core-LOD-Set (Nested LOD Sets 1) and is created using quadtree algorithm as demonstrated in Tsai et al. (2006). Based on the coarsest level of the Core-LOD-Set, an Outer-LOD-Set (Nested LOD Sets 2) is generated by resampling. Differences between adjacent LOD levels are identified and stored as the “difference vectors”, so when a tile’s LOD changes, the meshes can be refined using difference vectors instead of reconstructing a new mesh from the new LOD vertices. Furthermore, in order to remove T-junctions

* Corresponding author.

between different tiles, the edge information of all levels of all tiles will be obtained from the Core-LOD-Set, and stored in the server side database. As for the (texture) image pre-processing, the LODs are created using image pyramid or wavelet-based resampling schemes, if image streaming is considered.

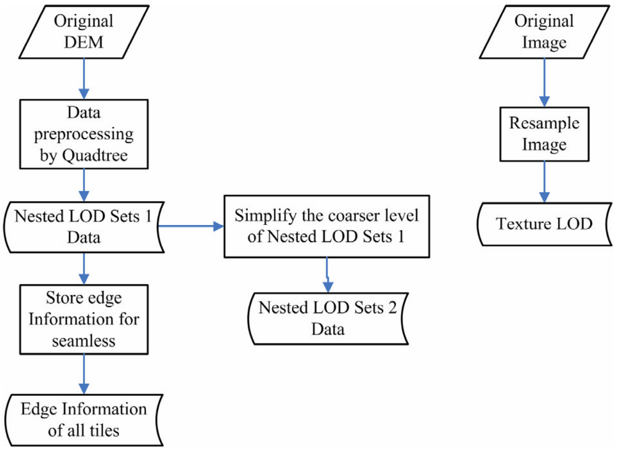


Figure 1. Procedure of data pre-processing.

2.1 Core-LOD-Set

In tile-based terrain visualization, original DEM must be tiled into regular blocks as illustrated in Fig. 2. The dimension of each tile must be 2^n+1 for quadtree processing. Zero data (hollow points) may be added to complete boundary tiles.

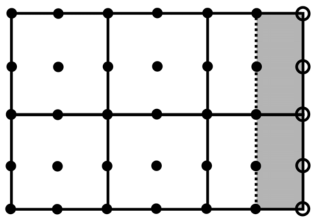


Figure 2. Tiled DEM.

For quadtree operations, threshold is the key factor. In dynamic thresholding (Tsai et al., 2006), the threshold is based on the maximum difference of height in each tile. They are computed from (1).

$$t_j = \log(H\_Max - H\_Min)$$
$$Th_n = (H\_Max - H\_Min) \times \frac{m-n}{m} \times t_j \quad (1)$$

where H_Max = maximum height
H_Min = minimum height
m = total number of levels
n = target level

A disadvantage of the dynamic thresholding is difficult to find the relationship between height difference and view importance. Therefore, this study develops a new thresholding scheme based on the view-dependent image-space error metric. The concept is to set thresholds by screen resolution and distances from view points to the centres of tiles. Taking Fig. 3 as an example, the Ground Sampling Distance (GSD) of the tile's centre is calculated using Equation (2). The GSD is used as the 1-pixel-error threshold for LOD generation.

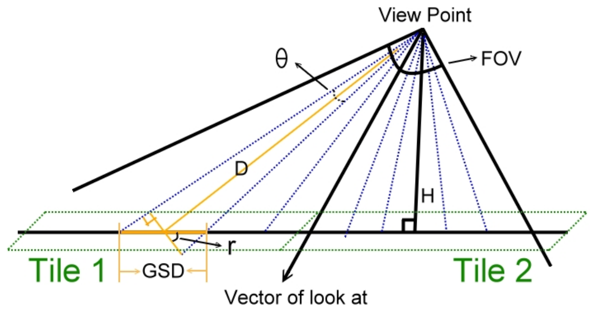


Figure 3. GSD determination.

$$\theta = \frac{\text{FOV}}{\text{pixels per scanline}}$$
$$GSD = \frac{D \times \tan(\theta)}{\cos(r)} \quad (2)$$

When thresholds of all LOD levels are decided, quadtree algorithm can be applied to generate all LOD levels of the tile according to the following procedure:

1. Search all vertices located in the tile (sub-tile) and get the total number of vertices, ***N***.
2. Calculate distances from vertices to sub-tile, $\boldsymbol{d_i}$
3. Compare $\boldsymbol{d_i}$ with the threshold and count how many $\boldsymbol{d_i}$ are smaller than the threshold, $\boldsymbol{N_d}$.
4. If $\boldsymbol{N_d}$ /***N*** is less than 95%, the sub-tile can be considered a plane, and the subdivision is done; otherwise, it should be divided and quartered continuously.

2.2 Outer-LOD-Set

Figure 4 displays the relationship of the nested LOD sets. The coarsest level of the Core-LOD-Set is the basis of the Outer-LOD-Set. By continuously consolidating meshes of the Outer-LOD-Set base, a new LOD set is created as shown in Fig. 5.

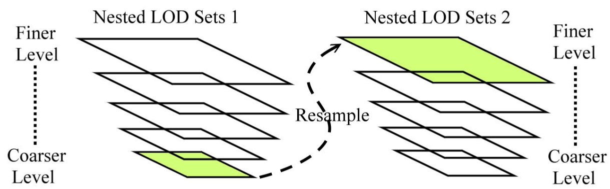


Figure 4. Nested LOD sets

The idea is to set threshold resolutions for different levels in the Outer-LOD-Set. When the level of detail is coarser, the threshold should be larger. If there are meshes whose resolutions are less than the threshold, the triangles within the mesh should be merged as the shaded meshes in Fig. 5.

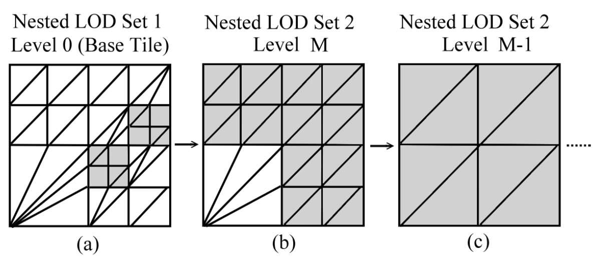


Figure 5. Generation of the Outer-LOD-Set.

2.3 Texture LOD

In addition to the height field meshes, terrain texture can be draped onto the generated 3D terrain surfaces in order to create a more realistic look and feel of the visualization. Similar to the DEM LOD process, the original texture image must divide into several tiles. Each image title will also be resampled to create an image pyramid. Or, if high performance data transmission on the internet is a concern, wavelet-based resampling schemes can be used in lieu of image pyramid, so it will be easy to utilize image streaming for data transmission. However, no matter what image resampling scheme is used, the dimension of a texture image (of each level) must be power of 2 for OpenGL implementation, which will then enable hardware rendering available with most modern graphical display cards.

3. TERRAIN RENDERING

Figure 6 illustrates the complete procedure for interactive (view-dependent) terrain rendering. Four pre-processed data sets are required. Before rendering, the system transmits the coarser level (base tile) of the Core-LOD-Set. At the same time, viewer orientation and position (view parameters) must be set. The view parameters are used for view-frustum culling to determine the visibility of tiles.

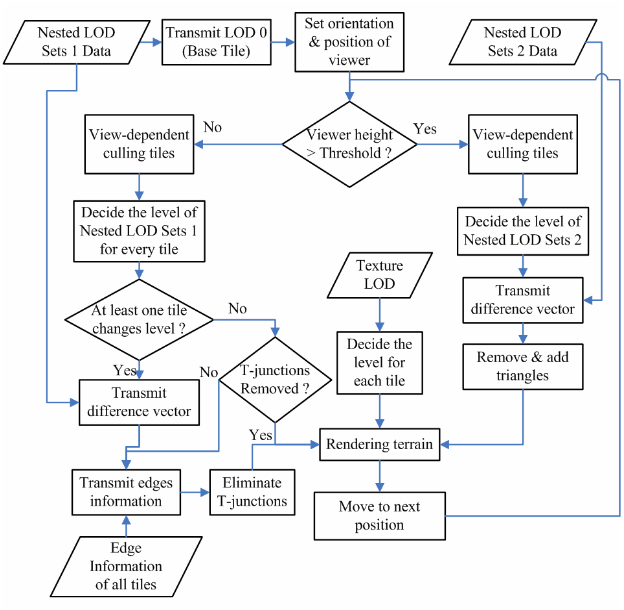


Figure 6. Terrain rendering procedure.

Next, a decision box will compare the viewer altitude with a pre-defined threshold for the switch of the two nested LOD sets. If the viewer height is larger than the threshold, the Outer-LOD-Set will be loaded and rendered. This usually happens in the initial stage of a high altitude fly-through simulation or similar visualization. By switching to the Outer-LOD-Set, the system can provide fast rendering and reduce the latency. Otherwise, users will have to wait a long time but gain little visual details because the DEM resolution is too high and the distance is too far away to show fine terrain details. When the viewer altitude is lower than the threshold, the rendering target will be switched to the Core-LOD-Set. Once the mesh is prepared, textures will be loaded and mapped onto terrain surfaces. The process will be repeated for each frame. When the view parameters change and cause the LOD of any tile changes, difference vectors are provided to refine the terrain meshes.

3.1 Removing T-junctions between Tiles

Because there are discontinuities between adjacent tiles, there may be so-called T-junctions along tile edges, producing cracks in the rendered scene. In order to deal with this problem, an algorithm is developed to remove T-junctions. Fig. 7 explains the procedure of T-junctions removal. In this figure, it shows a target tile and an adjacent tile. In the target tile, Ta~Tc are three boundary triangles, and T1~T7 are their vertices. The adjacent tile has a similar boundary meshes and vertices but they are noted as Aa~Ac and A1~A7. In this case, the procedure of T-junction removal is to compares Y coordinates of boundary vertices with the following steps:

1. The comparison starts from T1 and A1. Because their Y coordinates are equal, the two vertices remain unchanged.
2. Moving to next pair of vertices, A2 is larger than T2. This means a T-junction exists at A2, and it will be recorded. Now, replace A2 with the next position (A3) to see if A3 is equal to T2. If they are, then this round of comparison is done.
3. Because a crack has been found, the procedure will remove the original triangle (Ta) and add two new triangles (ΔT1 T5 A2, ΔA2 T5 T2). By this way, the number and position of triangles are identical on the edge of the two tiles and the T-junction is removed.
4. Starting from T2 and A3 and repeat steps 1~3. A crack can be found at T3, and it will be fixed by replacing Ac with two new triangles (ΔA3 T3 A7, ΔA7 T3 A4).

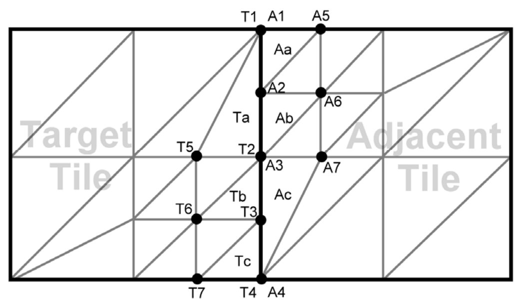


Figure 7. T-junctions removal.

3.2 Texture LOD

Texture mapping on terrain meshes is important for visualization because it provides a more realistic look and feel. In addition, it creates better visual effects when the terrain meshes are coarse. Texture mapping is also an effective way for information conveyance. For terrain visualization, satellite images or aerial photographs are common sources of terrain texture mapping. However, the original images are usually too large to handle when mapping in real-time. Using LOD techniques described in 2.3, the developed system is capable of applying appropriate levels of terrain texture mapping based on different viewing parameters.

Basically, the decision of suitable terrain texture LOD utilizes the same concept of GSD (Fig. 3). The GSD of a tile's centre will be calculated and compared with image resolutions. If GSD is larger than the resolution of selected level and less than the next level, this level of texture image will be mapped onto the corresponding tile meshes.

4. RESULTS AND DISCUSSIONS

The developed terrain rendering system has been tested on an AMD Athlon 3500 with 2GB memory, and an nVidia GeForce 7300 GT graphics card with 128M texture memory. The rendering application has been implemented using wxDev C++ with OpenGL for rendering. This section will describe and discuss selected experimental results.

4.1 Test Data Sets

The first data set used for testing is the DEM of entire Taiwan. The original DEM dataset is 5022 × 9555 grids with a 40m by 40m spatial resolution and 1m resolution in elevation (Z-axis). The maximum height in the DEM is 3941m above sea-level, and the minimum height is 0m. The entire DEM dataset is partitioned into 10 × 19 tiles and the size of each tile has been set at 513 × 513. A mosaic image generated from SPOT-5 satellite images acquired in 2005 is used as the source of terrain texture. The resolution of the mosaic image is resampled to 2.5m, and the image size is 88000×160000 pixels. Firstly, this image has been tiled into the same numbers and positions as the DEM tiles, and then resampled each sub-image into 8 levels from 4096×4096 to 32×32. All texture images are stored in PNG format.

The second data set, Puget Sound--located in Washington, USA--is also used for testing the developed algorithms. The DEM is about 1500 km^2 at 9 m ground resolution and 0.3 m altitude resolution. The maximum height is 3665 m, and the minimum height is -290 m. The dataset is divided into 23 × 29 tiles with 513 × 513 tile size. A synthesized image is used as the texture image. The dimension of the texture image is 2000 × 2500 pixels, and therefore no texture LOD is used in this dataset since the image size is relatively small.

Table 1 and 2 list 6 distances to compute GSDs from the Taiwan dataset and the Puget Sound dataset, respectively. This indicates that there are 6 levels of the Core-LOD-Set in both datasets. There are 6 GSDs in each dataset, and they can be considered as thresholds (of 1-pixel error). Thresholds of 2~4 pixels errors are calculated accordingly.

Table 1. Thresholds for the Taiwan DEM.

Distance	24000	32000	40000	48000	56000	64000
GSD	17	23	29	34	40	46
2 pixels error	34	46	58	68	80	92
3 pixels error	51	69	87	102	120	138
4 pixels error	68	92	116	136	160	184

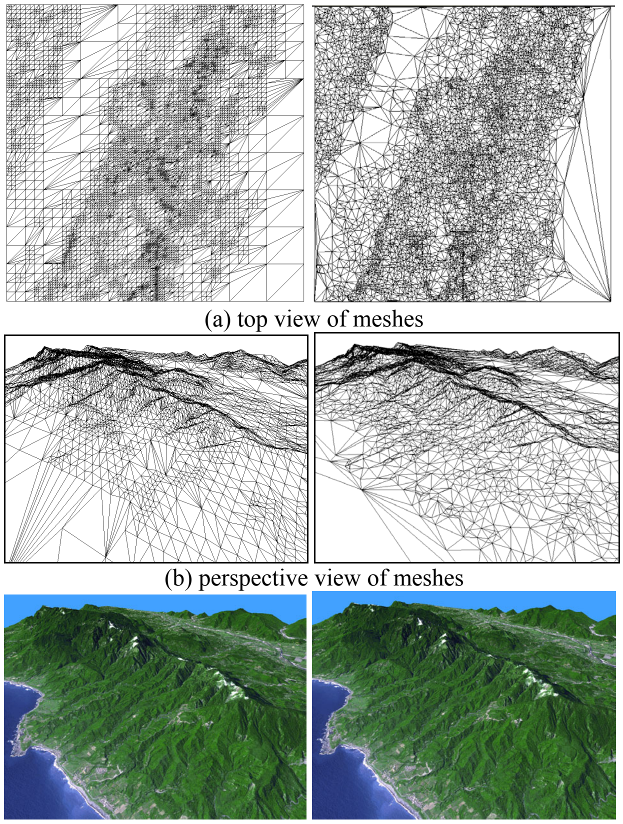
Table 2. Thresholds for the Puget Sound DEM.

Distance	5000	10000	15000	20000	25000	30000
GSD	4	8	11	15	19	22
2 pixels error	8	16	22	30	38	44
3 pixels error	12	24	33	45	57	66
4 pixels error	16	32	44	60	76	88

Outer-LOD-Sets are also created for both DEM datasets. Because the size of each tile is $(2^9+1)\times(2^9+1)$, the thresholds must be less than that size. Accordingly, four thresholds, from $(2^5+1)\times(2^5+1)$ to $(2^8+1)\times(2^8+1)$, are used to create the Outer-LOD-Sets.

4.2 Comparing with Delaunay Triangulation

The applied quadtree LOD generation is compared with Delaunay triangulations. Fig. 9 displays the visual effects of a tile with 3-pixels error LOD processing. It is a level 3 tile with 5808 vertices in both cases. The numbers of triangles are very close, too. There are 11407 triangles in quadtree and 11524 in Delaunay. Fig. 8(a) is a top view of Quadtree (left) and Delaunay (right), and fig. 8(b) is a perspective view of the meshes. It appears that Delaunay triangulation shows a better construction of terrain features than quadtree. However, when texture mapping is applied to meshes, it is almost impossible to differentiate one from the other visually.



(a) top view of meshes

(b) perspective view of meshes

(c) perspective view with texture mapping

Figure 8. Comparsion between Quadtree (left) and Delaunay (right) LODs.

Although Delaunay triangulation seems to produce better meshes in describing detail terrain features, the data structure of quadtree results is more organized than Delaunay and makes the rendering more efficient. Taking T-junctions removal for example, Delaunay will take significantly more efforts to remove T-junctions because the triangles on tile edges are irregular. In addition, for Delaunay triangulations, it will be inefficient to use the "difference vectors" scheme because unlike quadtree-based LOD, vertices in different levels of Delaunay-based LOD results do not have an "add-on" property. Therefore, it will be difficult to achieve progressive transmission and adaptive rendering of Delaunay-based LOD tiles and thus inadequate for real-time visualization applications.

4.3 Flythrough Simulation

To demonstrate the performance of the developed visualization system, two routes have been planed in both data sets for complete fly-through simulations. One of the plotted routes passes mountainous areas of the data set, and the other passes plain regions. In the Taiwan data set (as displayed in fig. 9), the mountain route starts from M1 and ends at M5. The total flight path is 513 km in ground track and exhibited with 960 rendered frames. The plain route starts from P1 and ends at P5. The length of flythrough is about 623 km and with 1521 rendered frames.

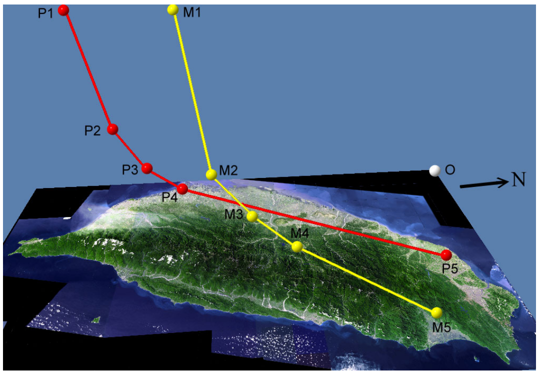


Figure 9. Flythrough routes (Taiwan).

In the Puget Sound case (as displayed in fig. 10), M1 ~ M4 and P1 ~ P4 represent the Puget Sound mountain route and plain route, respectively. The total distance of the mountain fly-through is 144 km in ground track with 1202 rendered frames. The flight path of the plain route is 160 km on the ground and with 1480 rendered frames. In addition, M2 and P2 are also the switching points from the Outer-LOD-Set to the Core-LOD-Set in both cases.

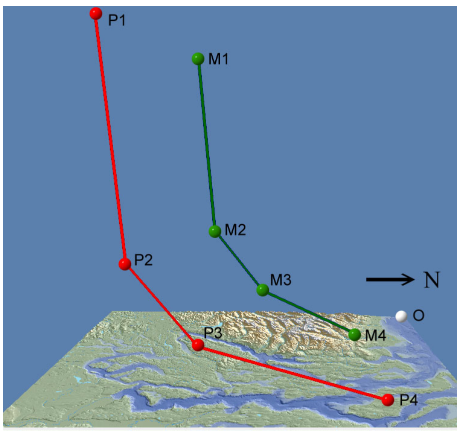


Figure 10. Flythrough routes (Puget Sound).

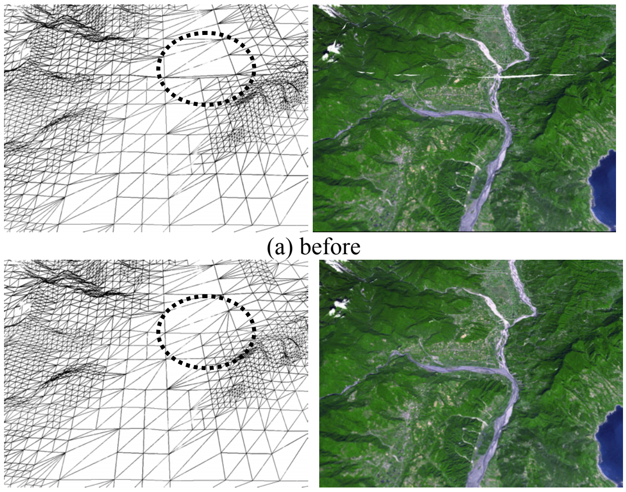
Table 3 demonstrates the rendering efficiency, in frames per second (FPS) and rendered number of triangles per second, of three (2~4 pixels errors) thresholding schemes. The lowest FPS acceptable for human eyes is about 25. The mountain route of Taiwan with 2-pixels error data shows the lowest frame rate (24.51 FPS), but is still good enough for a real-time visualization. For other cases, the frame rates are all higher than 25 FPS. These results prove that the developed terrain rendering system is very efficient.

Table 3. Efficiency of flythrough simulations.

Threshold Scheme	Taiwan Mountain		Taiwan Plain		Puget Sound Mountain		Puget Sound Plain	
	FPS	MΔ/s	FPS	MΔ/s	FPS	MΔ/s	FPS	MΔ/s
2 Pixels Error	24.51	5.00	42.71	5.07	48.84	4.47	125.49	2.90
3 Pixels Error	48.73	4.54	73.96	3.75	64.77	3.15	220.95	2.25
4 Pixels Error	61.26	3.22	91.01	2.46	97.37	3.00	307.20	1.81

4.4 T-junction Removal

Fig. 11(a) shows the T-junctions and the cracks in the rendered scene before removing them. The circle on the top left figure identifies T-junctions in meshes, which result in a noticeable crack after applying texture mapping. As displayed in Fig. 12(b), after removing the T-junctions with the algorithms described in 3.1, the crack artefact has been effectively corrected, resulting a seamless rendering of the scene.



(a) before

(b) after

Figure 11. Visual effect of T-junctions removal.

Fig. 12 plots the accumulated rendering time of the Taiwan dataset with and without T-junction removal. As displayed in the figure, curves of cumulative rendering time are almost identical in both cases, indicating that removing T-junctions does not take too much effort.

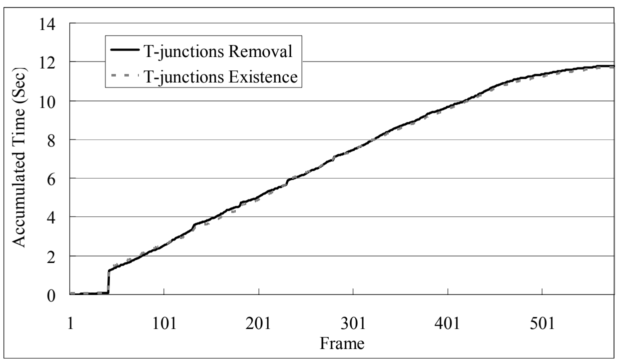


Figure 12. Accumulated rendering time (Taiwan).

Fig . 13 shows the same test using Puget Sound data. It exhibits a similar pattern as fig. 12. As a result, the developed T-junctions removal algorithm provides a significant improvement in visual effects, but causes insubstantial impact to the rendering performance.

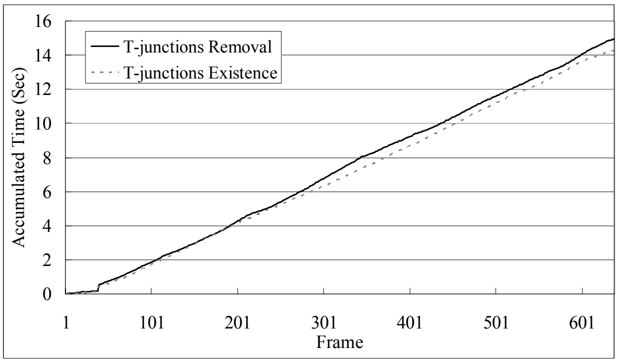


Figure 13. Accumulated rendering time (Puget Sound).

4.5 Effects of Nested LOD

Fig. 14 displays the accumulated rendering time and number of triangles with and without utilizing nested LOD for Taiwan data. When using nested LOD, a jump is observed at #230 frame approximately. It is a switching point of the two LOD sets. As shown in the figure, it takes about 3.5 seconds to initialize the visualization without using nested LOD, because it must load more than 200 thousand triangles in the first frame. On the other hand, using nested LOD, the system needs only milli-seconds to load about one thousand triangles. Similar result can be observed in the Puget Sound data set (fig. 15). Consequently, nested LOD makes a substantial improvement on rendering efficiency during the initial stage of visualization.

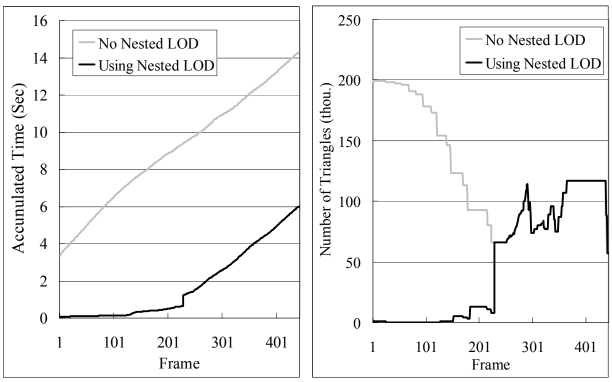


Figure 14. Performance of Nested LOD (Taiwan).

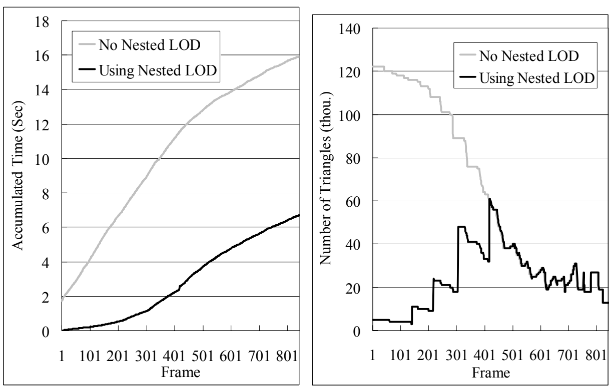


Figure 15. Performance of Nested LOD (Puget Sound).

5. CONCLUSIONS

The developed tile-based terrain rendering system allows users to obtain near real-time visualization of large terrain data sets. The proposed new thresholding scheme based on Ground Sample Distance enables more reasonable quadtree-based LOD generation for data pre-processing. It also provides better relationship of the LOD generation and view-importance for determining appropriate LOD levels of visible tiles. The developed T-junction removal algorithm can eliminate discontinuities between adjacent tile meshes effectively and have little impact to the overall rendering performance. Using nested LOD improves the performance significantly, especially during the initial stage of visualization. Test examples with two large DEM datasets demonstrated in this paper indicates that the developed system can produce seamless rendered scenes with high performance in near real-time visualization applications.

REFERENCES

Blow, J., 2000. “Terrain Rendering at High Levels of Detail”. In Proc. 2000 Game Developers Conference, pp. 903-912.

Cignoni, P., E. Puppo, R. Scopigno, 1997. “Representation and Visualization of Terrain Surfaces at Variable Resolution”. The Visual Computer, 13:5, pp. 199-217.

Cignoni, P., C. Montani, R. Scopigno, 1998. “A comparison of mesh simplification algorithms”. Computers & Graphics, 22:1, pp. 37–54.

Cignoni, P., F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, R. Scopigno, 2003. ”BDAM – Batched Dynamic Adaptive Meshes for High Performance Terrain Visualization”. Computer Graphics Forum, 22:3, pp. 505–514.

Losasso, F., H. Hoppe, 2004. “Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids”. ACM Transactions on Graphics (TOG) Archive, 23:3, pp. 769-776.

Lindstrom, P., D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, G. A. Turner, 1996. “Real-time continuous level of detail rendering of height fields”. In Proc. of ACM SIGGRAPH96, pp. 109–118.

Luebke, D., M. Reddy, J. D. Cohen, A. Varshney, B. Watson, R. Huebner, 2003. “Level of Detail for 3D Graphics”. Morgan Kaufmann Publishers, San Francisco, California.

Tsai, F., H-S Liu, J-K Liu, K-H Hsiao, 2006. “Progressive Streaming and Rendering of 3D Terrain for Cyber City Visualization”. In Proc. of ACRS2006, Oct. 2006, Ulaanbatar Mongolia.

ACKNOWLEDGEMENTS

This study was supported by the National Science Council and the Ministry of Interior of Taiwan (ROC) under project numbers NSC-95-2221-E-008-104-MY2 and H950925. The Puget Sound DEM dataset was from University of Washington: (http://www.ocean.washington.edu/data/pugetsound/).