Geometry-based Mapping and Rendering of Vector Data over LOD Phototextured 3D Terrain Models

Anupam Agrawal
Indian Institute of Information Technology,
Deoghat, Jhalwa,
India (211011), Allahabad, U.P.
anupam@iiita.ac.in

M. Radhakrishna
Indian Institute of Information Technology,
Deoghat, Jhalwa,
India (211011), Allahabad, U.P.
mkrishna@iiita.ac.in

R.C. Joshi
Dept. of E& C Engineering,
Indian Institute of Technology,
India (247667), Roorkee, U.A.
joshifcc@iitr.ernet.in

ABSTRACT

Interactive three-dimensional (3D) visualization of very large-scale grid digital elevation models coupled with corresponding high-resolution remote-sensing phototexture images is a hard problem. The graphics load must be controlled by an adaptive view-dependent surface triangulation and by taking advantage of different levels of detail (LODs) using multiresolution modeling of terrain geometry. Furthermore, the display of vector data over the level of detail terrain models is a challenging task. In this case, rendering artifacts are likely to occur until vector data is mapped consistently and exactly to the current level-of-detail of terrain geometry. In our prior work, we have developed a view-dependent dynamic block-based LOD mesh simplification scheme and out-of-core management of large terrain data for real-time rendering on desktop PCs. In this paper, we have proposed a new rendering algorithm for the combined display of multiresolution 3D terrain and polyline vector data representing the geographical entities such as roads, state or country boundaries etc. Our algorithm for multiresolution modeling of vector data allows the system to adapt the visual mapping without rendering artifacts to the context and the user needs while maintaining interactive frame rates. The algorithms have been implemented using Visual C++ and OpenGL 3D API and successfully tested on different real-world terrain raster and vector data sets.

Keywords

Digital terrain models, Multiresolution Modeling, Level-of-Detail Rendering, Vector and Raster Data.

1. INTRODUCTION

In conventional printed topographic maps, the real three-dimensional (3D) world is projected vertically onto a two-dimensional plane with symbolization of ground objects. On these maps, topography of the terrain is represented by contours, which are digitized and converted into grid digital elevation model (height map). Apart from contour information, the map consists of variety of other information including point features (e.g. buildings, trees etc.), line features (e.g. road networks, rivers etc.) and polygon features (e.g. country boundaries, vegetation zone etc.). It is not easy to understand information on 2D topographic maps, as it demands some knowledge and skills in map reading.

Short Communications proceedings ISBN 80-86943-05-4
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.

3D rendering of the map provides information on geographical data about the shape of the terrain and location of other objects on a map quickly and easily. An interactive system with real-time rendering is useful in spatial support systems, virtual reality applications, real-time GIS and cartography.

Geographic data may be categorized as raster data and vector data. Raster data are analogous to a bit map or a regular 2D array where each array element contains a data value for a corresponding rectangular grid cell in the 2D plane. Common sizes of digital terrain raster data including height map and corresponding geo-referenced remote-sensing satellite imagery may consist of 1K*1K to 16K*16K or more grid cells. The interactive visualization of such large datasets has been a challenging problem. The main problem in real-time graphics is rendering efficiency [Ake02]. In order to get high rendering

performance, one of the approaches is to reduce the scene complexity without leading to an inferior visual representation. Multiresolution models provide different levels of detail (LODs) representation of the modeled objects. In LOD scheme, the close regions are approximated more accurately than regions that are far away such that the resulting image is without any noticeable visual difference. In our prior work, we have developed a view-dependent dynamic block-based level-of-detail mesh simplification scheme and out-of-core management of large terrain data for real-time rendering on desktop PCs [Agr04a], [Agr04b].

Vector data represents one major category of geographic information and defines geometry as lists of 2D coordinates that form points, lines and polygons. Narrow linear features such as roads, railway lines etc. are usually not visible on a satellite image used in creating 3D phototextured views of the terrain. Other vector features such as state or country boundaries, property lines etc., usually used for logical demarcation, are also required to be overlaid on top of the 3D views with appropriate display properties. These vector features are separately digitized from corresponding topographic map.

Multiresolution level-of-detail terrain models, where underlying mesh geometry is changing for each frame, pose significant challenges in visual mapping of vector data in 3D without any rendering artifacts. Section 2 discusses related work and complications in displaying vector data over multiresolution terrain models. Section 3 briefly explains our view-dependent adaptive multiresolution mesh simplification framework, which is supporting real-time frame rates on desktop PCs on arbitrarily large digital terrain raster data. The proposed geometry-based mapping approach of polyline vector data for its integration with above multiresolution geometry modeling framework is explained in section 4. The approach smoothly adapts with our tile-based out-of-core management of terrain raster data. In section 5, we have shown the results of the proposed algorithm and performance analysis on a real-world terrain raster and vector data set. Finally section 6 gives conclusions and scope for future work.

2. RELATED WORK AND COMPLIC-ATIONS IN VECTOR DATA DISPLAY

Display of 2D polyline vector data over 3D terrain geometry becomes relatively much simpler task if the underlying terrain geometry is static and is not changing with time i.e. the surface is being rendered at constant resolution [Agr98]. In this case, height values at points on vector data can be picked up from underlying geo-referenced DEM or height map. But this approach has the drawback that it very much restricts the size of the terrain data. It is not feasible to render large terrain raster data sets of size 16K*16K or more at interactive frame rates even on a very high-end graphics workstation. Szenberg et al. [Sze97] describe a method of terrain visualization with polyline vector data such as transmission lines. However, the visualization scheme for terrain height field is not based on multiresolution modeling but combines the Z-buffer with the floating Horizon algorithm. Also results are shown on limited sized terrain data (512*512 size) only. Xiaoping et al. [Xia04] describe a method to render vector data on static terrain geometry. The actual size of the terrain data has not been reported in the paper.

In general, multiresolution modeling is necessary for representing large size geo-referenced surfaces in order to reduce their geometric complexity and to achieve real-time rendering [Lue03]. Relatively very less work has been reported in literature on displaying vector data over multiresolution terrain.

There are two options to rendering polyline vector data on multiresolution 3D mesh. One option is to convert the polyline data to a texture image layer and combine this polyline image layer with the primary terrain texture image layer (e.g. from a satellite/aerial photograph). The second option is to render the polyline data as separate 3D geometric primitives. We may call these two approaches as polyline-as-texture solution and polyline-as-geometry solution respectively. Both the approaches present a number of complications.

A simple polyline-as-texture solution is to rasterize the polyline into the primary texture image at the image's highest resolution and then to render the terrain in the standard way using mipmaps or other suitable filtering. However, this is a poor solution because when zoomed out on the 3D terrain, the user will find much of the polyline vector information filtered away especially if single pixel lines were used for the rasterized vector data. This could cause district borders to be nearly or completely filtered away when zoomed out to view an entire state. In fact, the abstract line's accuracy should be independent of the resolution of the primary imagery and also its visual representation's accuracy should not be limited to the texel resolution of the primary imagery. More complex polyline-as-texture approach is required to deal with above problems. Kersting et al. [Ker02] describe a texture-based rendering of vector data onto the level-of-detail terrain geometry. The method uses OpenGL P-buffer for rendering which allows to rasterize vector data within real-time so that on-demand generation of textures becomes practical.

The polyline-as-geometry approach allows more flexibility in polyline vector rendering as it supports interactive enabling and disabling of the display of different subsets of polyline data and interactive adjustment of their display properties. Douglass et al. [Dou99] describe a bottom-up LOD height-map rendering scheme by placing building objects over the terrain. In contrast to a top-down LOD approach, a bottom-up approach necessitates the entire model being available at the first step and therefore has higher memory and computational demands. Zachary et al. [Zac03] extend the approach of Douglass et al. to overlay vector data over multiresolution 3D terrain. It is important to note that any multiresolution modeling of vector data approach will depend on the underlying multiresolution level-of-detail terrain geometry mesh simplification scheme.

In this paper, we have proposed a new polyline-as-geometry approach, which integrates well with our dynamic multiresolution level-of-detail mesh simplification algorithm for real-time rendering. Displaying 2D polyline data on top of 3D terrain becomes challenging in our terrain visualization system for several reasons. First, we have used raster data tiling approach to handle terrain's 3D geometry and image data, as the same are too large to fit into primary memory. It requires dynamic paging of both the data types based on the current 3D view of the terrain. At any instant of time, only nine tiles, each of size 256*256 pixels, are kept in main memory based on viewer's location. Accordingly, the display of vector layer should also be limited to currently active nine tiles at a time. Second, the 2D polyline data should be treated independently from the raster data and therefore should be rendered as separate geometry by the graphics pipeline. This presents a challenge because our multiresolution LOD algorithm renders a 3D mesh whose constituent triangles of different patches are changing at nearly every frame. In order for the polyline vector data to appear overlaid on the 3D mesh, the rendered polyline geometry (height values on polyline points) must therefore also change at each frame. Fig. 1 shows the need of multiresolution modeling of vector data for geometry-based mapping.

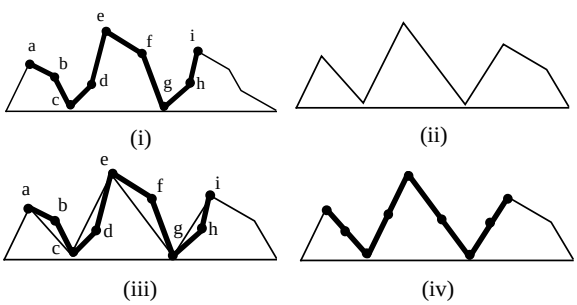


Figure 1. Geometry-based Mapping of Vector Data

Fig. 1(i) shows the 2D cross-sectional view of terrain where current geometry patch is rendered at full resolution and road polyline (bold line segments) tightly follows the terrain surface. Fig. 1(ii) shows the situation when the current patch is being rendered at lower resolution because of viewer position and screen resolution. If road polyline is displayed over the low-resolution terrain mesh without changing its geometry, then visual artifacts will be seen as shown in Fig. 1(iii). Hence, it is required to adjust height values on polyline points b, d, f, and h according to slopes of the triangles of lower resolution terrain patch (Fig. 1(iv)).

3. UNDERLYING MULTI-RESOLUTION FRAMEWORK FOR TERRAIN MODELS

We have developed a view-dependent dynamic block-based LOD modeling for mesh simplification and using tiled geospecific texture, to display the details of the high-resolution satellite imagery in real-time rendering [Agr04a]. The terrain geometry and texture data are organized in titles of size 257*257 and 256*256 respectively. One pixel overlap is kept between adjacent geometry tiles to ensure proper stitching of tiles. At any instance of time, only nine tiles are kept in main memory. The viewer position is always assumed to be inside the centre tile. The algorithm efficiently handles out-of-core data by dynamic paging of terrain tiles between secondary storage and main memory. Each geometry tile data is organized with a quadtree with leaves corresponding to patches or blocks of size 17*17 (the size decided after experimentation) to speed up the view-frustum culling. Fig. 2 shows quad-tree based decomposition of the terrain geometry up to second level where the gray area indicates the current view-frustum seen by the camera.

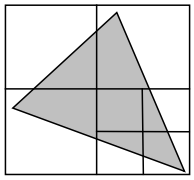


Figure 2. Quad-tree based Decomposition of Terrain Geometry

Multiresolution pyramid representation is used to define each terrain block. Fig. 3 shows the four pyramid levels of the height map block of size 17*17. Considering the multiresolution representation for each patch, the algorithm employs a variable screen-space threshold to limit the maximum error of the projected image considering the terrain complexity, viewer distance and viewing direction as the viewer navigates the terrain. The

algorithm pre-computes a look-up table at terrain tile load time to decide the tessellation level of each block within view-frustum based on position of the camera from the block [Agr04b]. In our approach, a group of vertices are considered instead of single vertex for deciding whether to remove them or not. Hence CPU requirements are many times lower as compared to the other LOD mesh simplification algorithms, which work on individual vertices of the height field.

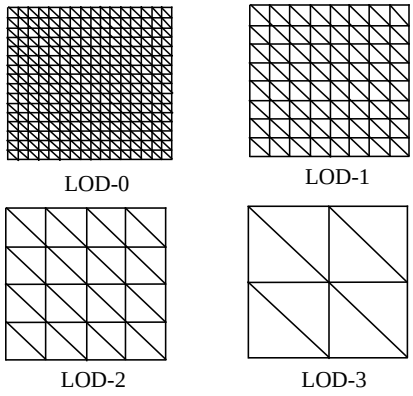
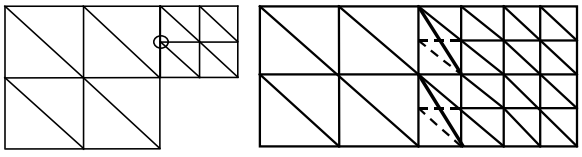


Figure 3. Multiresolution Modeling of Height Map

It is important to note that in a view-dependent framework, the resolution of adjacent patches might change at every frame. Hence, cracks occur on borders of adjacent patches of different levels of detail. In Fig. 4 (a), the circle shows the position of crack in tessellation with level difference one (right side patch is shown partially). Crack-filling methods usually involve creating additional triangles to fill in the gaps between patches, and/or modifying the geometry of one or other of the patches to produce a crack-free join. Fig. 4 (b) shows the modified geometry to remove the cracks where the dashed edges are excluded in triangulation and the bold edges are included. Similar procedure is followed to eliminate cracks when level difference between adjacent patches is two or three. Image draping over 3D mesh geometry is performed using texture mipmapping. The algorithm also handles the problem of texture seams between adjacent texture tiles [Agr04a].



(a) Before crack removal (b) After crack removal

Figure 4. Removing Cracks between Adjacent Patches

To exploit the full performance of current GPUs hardware, transmission of large data chunks is advantageous. Graphics rendering can be accelerated through compact representations of polygonal meshes using data structures such as triangle strips and triangle fans. Using triangle strip primitive, it is possible to form a longer length of connected triangles as compared to triangle fan. Generating long triangle strips efficiently solves the CPU-to-card bandwidth problem and avoids redundant 3D vertex transformation and lighting (T&L) calculations. However in view-dependent meshing methods the underlying mesh is in a constant state of flux between view positions. This poses a significant hurdle to construct long triangle strips. Our triangle strip generation scheme for view-dependent dynamic multiresolution terrain shows significant improvement in rendering speed as compared to individual triangle-based and triangle-fan based rendering schemes [Agr05]. The snapshot of the triangulated height map is given in Fig. 5 using triangle-strip primitive.

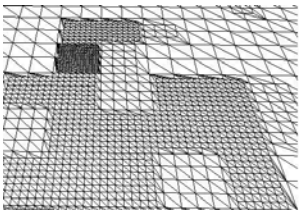


Figure 5. Wireframe View of Terrain Geometry

4. PROPOSED APPROACH TO RENDERING VECTOR DATA OVER MULTIRESOLUTION TERRAIN MODELS

The proposed geometry-based mapping approach to rendering polyline vector data over the multi-resolution terrain consists of following four steps:

4.1 Vector Data Capture

The software TREND (acronym for **T**errain **Rend**ering) has the provision for interactively digitizing the polyline vector data. The user may open a geo-referenced map or image in a 2D display window and select option for vector digitization. The selected points on a polyline may be stored in a new vector file or may be appended at the end of an existing vector file. There are ‘start’ and ‘end’ options for polyline digitization so that user may capture different polylines in one session and save the same in a file as shown in Fig. 6.

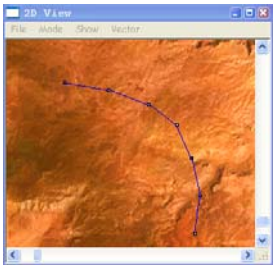


Figure 6. Digitized Polyline Vector Data

4.2 Vector Data Organization

As discussed in section 2, the display of vector data should be limited to current active nine tiles in memory at a given time instant. As the user moves over the terrain, these current nine tiles change through dynamic terrain paging scheme. Accordingly before storing the captured polyline vector data in a file, it is rasterized at highest geometry mesh resolution and has been divided into segments based on number of geometry tiles (each of size 257*257 pixels) it is passing through. A unique sequence number is also stored along with each point of the vector polyline. This helps in identifying multiple vector segments (part of same or different vector polylines) inside a tile and draws them appropriately.

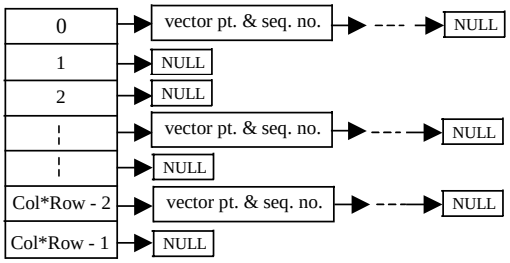
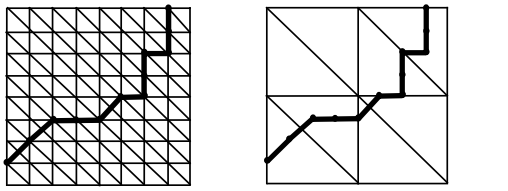


Figure 7. Tile-wise Organization of Vector Data

The software internally manages a dynamic data structure to store tile-wise vector segment details as shown in Fig. 7. When an existing vector file is loaded into memory, different vector segments are assigned to appropriate tile vector addresses. Based on the viewer position, vector data corresponding to current nine tiles is picked up for display.

4.3 Geometry-based Mapping and display

As discussed in section 3, the algorithm selects the resolution of a patch (size 17*17 pixels) using a lookup table based on its distance from the viewer. Fig. 5 shows the snapshot of a portion of displayed multiresolution mesh geometry. In our proposed approach, we map the vector data over different terrain geometry patches over which it is passing through. Height values of all the vector polyline points are picked up from the underlying geometry patch. Now let us consider a situation when a patch is rendered at lower resolution with an overlay of a vector segment. Fig. 8(a) shows $1/4^{th}$ portion of highest resolution patch (LOD-0) and Fig. 8(b) shows corresponding patch portion at lower resolution (LOD-2) both with an overlay of same vector segments.



(a) $1/4^{th}$ portion LOD-0 patch (b) $1/4^{th}$ portion LOD-2 patch

Figure 8. Geometry-based Mapping

When the patch is rendered at highest resolution, the vector polyline segments are displayed without any visual artifact (except in special case 1 discussed in subsection 4.4.1) because heights of all vector polyline points are matching with corresponding geometry patch heights (ref. Fig. 1(i)). However, when a patch is rendered at lower resolution, visual artifacts may occur in displaying vector data (ref. Fig. 1(iii)). In fact height values at all the vector points in Fig. 8(b) should be matched with the corresponding enclosing triangle slopes.

To compute correct height values, we first determine a plane passing through three vertices (x_i,y_i,z_i), i=1,2,3 of the enclosing triangle with following equation:

$$ax + by + cz + d = 0 \qquad (1)$$

The height value z at vector point (x, y) may be computed by solving following determinant:

$$\begin{vmatrix} x - x1 & y - y1 & z - z1 \\ x2 - x1 & y2 - y1 & z2 - z1 \\ x3 - x1 & y3 - y1 & z3 - z1 \end{vmatrix} = 0 \qquad (2)$$

The height values at points on triangle edges may be computed through linear interpolation of heights at the end points. To minimize computational overhead, this process is applied only for those geometry patches through which the vector segment is passing on. The polyline vector points with updated height values will now tightly follow the terrain geometry.

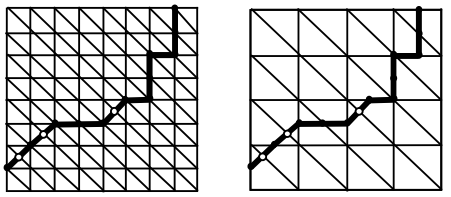
4.4 Dealing with Special Cases

The algorithm discussed in subsection 4.3 is required to be extended to deal with following two special cases otherwise the output of vector data rendering over multiresolution terrain will still suffer with some visual artifacts.

4.4.1 Case 1

When a vector segment crosses common diagonal of adjacent triangles considering patch resolution of different levels.

Fig. 9(a) shows a polyline vector drawn over 1/4th of the full resolution patch (LOD-0) whereas Fig. 9(b) shows the same polyline vector drawn over a lower resolution patch (LOD-1) with level difference 1.



(a) 1/4th portion LOD-0 patch (b) 1/4th portion LOD-1 patch

Figure 9. Segments Crossing Common Diagonals

By careful examination of Figures 9(a) and 9(b), we find that a vector segment, which crosses a common diagonal of two adjacent triangles, may not be visible in some situations. Consider the case in Fig. 10. Assume that heights of points **a** & **c** are 100 and 110 respectively whereas heights of points **b** & **d** are 80 and 90 respectively. In this situation, the vector segment **bd** hides beneath the two triangular faces **abc** and **acd** of terrain geometry.

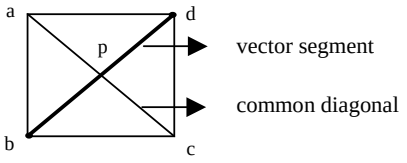


Figure 10. Vector Segment Crossing Common Diagonal

To deal with this situation, we should compute height 'h' at intersection point **p** considering heights at end points **a** and **c** and then render the vector segment **bd** as two sub-segments **bp** and **pd** with height 'h' at **p**. In Figures 9(a) and 9(b), the hollow circles show the positions where height values are to computed at the intersection points and corresponding vector segment is to be drawn in two parts. We deal in similar manner when rendering of vector segments over two other lower resolution patches (LOD-2 and LOD-3).

4.4.2 Case 2

When a vector segment is passing through the crack-joining triangle at the boundary of two different resolution patches.

Recall that as shown in Fig. 4, the geometry of boundary triangles of higher resolution patch is altered to avoid cracks between adjacent patches of different resolutions. The problem arises when LOD-1 patch is adjacent to LOD-2 (or LOD-3) patch as shown in Fig. 11. Here height at a vector point either inside the crack-filling triangle or on common boundary (shown as filled squares nodes) should be computed considering the modified vertices of the adapting or crack-filling triangle instead of vertices of enclosing triangle of current resolution patch.

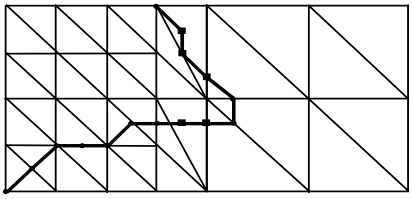


Figure 11. Vector Segments Passing through Crack-filling Triangles

We deal in similar manner when LOD-2 patch is adjacent to LOD-3 patch or when LOD-0 patch is adjacent to LOD-1 or LOD-2 or LOD-3 patch.

5. RESULTS AND PERFORMANCE ANALYSIS

The proposed algorithms have been implemented using Visual C++ and OpenGL 3D API for a Win32 environment. We have tested the software with 2K*4K terrain raster dataset of Grand Canyon and 16K*16K terrain data set of Puget Sound area obtained from Georgia Institute of Technology website. We have also generated the height map of Dehradun (India) area using digitized contours on Survey of India (SOI), India supplied topographic map. The corresponding geo-referenced IRS-1D FCC Satellite imagery has been used for image draping.

The software TREND provides a 2D window interface to digitize polyline vector data over the geo-referenced map or image in the background (Fig. 6). Figures 12(a) and 12(b) show the 3D wireframe display of terrain mesh geometry with overlay of polyline vector data and corresponding phototextured terrain view respectively using Grand Canyon dataset. Without multiresolution modeling of the polyline vector data, the visual artifacts are visible in vector data display.

Figures 13(a) and 13(b) show the views obtained after the proposed geometry-based mapping of polyline vector data over multiresolution 3D terrain as discussed in section 4.3. The visual appearance of the displayed vector data is now much improved. Fig. 14(a) shows the case when vector segments are crossing common diagonals of adjacent triangles. Fig. 14(b) shows visual artifact due to case 1 and Fig.

14(c) shows the result of algorithm proposed in subsection 4.4.1. The Fig. 15(a) shows the case when a vector segment crosses a crack-joining triangle. Fig. 15(b) shows visual artifact due to case 2 and Fig. 15(c) shows the result of algorithm proposed in subsection 4.4.2.

We have tested our software on a PC with Intel PIV 2.4 GHz CPU, 512 MB RAM, and Intel 82865G onboard Graphics Controller on 865GL motherboard. The performance of the algorithm on raster data is independent of size of terrain data as with the tiles indexing scheme, the algorithm only keeps 9 tiles active in main memory. The organization of the terrain data in tiles of defined size is required to be done only once on the same data set. For raster data, the number of frames rendered per second mainly depends on the complexity of the terrain (roughness) under the view-frustum and the user defined image quality metric (τ). Table 1 shows performance analysis of the algorithms without and with geometry-based mapping of polyline vector data overlay.

Terrain Rendering	**Avg. no. of Triangles**	**Avg. Frames per Second**
Full Resolution Raster Data (considering 9 tiles only)	1327104.00	1.72
View-frustum culled Surface	268120.70	7.18
Adaptive LOD algorithm for Raster Data (τ=4)		
(a) using triangle list	20368.45	57.23
(b) using triangle fan	20368.44	74.11
(c) using triangle strip (indexed vertex array)	23733.79	130.79
Multiresolution modeling of Vector Data (total points: 573) and rendering using Triangle Strip as above	23698.74	92.76

Table 1. Performance Analysis

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a new geometry-based mapping approach to rendering map vector data over multiresolution 3D terrain models. The proposed algorithm maps polyline vector data consistently and exactly to the current level-of-detail of terrain geometry and thus minimizes the rendering artifacts. Performance analysis results show that our combined multiresolution 3D terrain and polyline vector data display algorithm maintains real-time frame-rates on a desktop PC for large real-world terrain data sets.

Currently all vector polyline data is kept in a single file, but internally it is organized in the tile-wise order for all the terrain tiles. To reduce main memory overhead, the dynamic paging concept of tiled terrain raster data may be extended to vector data also. We may apply B-spline function to input points data to make smooth curved polyline features. Future work also includes extending the approach to display wider width polyline data (e.g. roads) with appropriate texture mapping.

7. ACKNOWLEDGMENTS

The research reported here has been partially supported by the Ministry of Human Resource Development (MHRD), Govt. of India under contract F.26-4/2002.TS.V (R&D Scheme).

8. REFERENCES

[Agr05] Agrawal, Anupam et al. An Approach to Improve Rendering Performance of Large Multiresolution Phototextured Terrain Models using Efficient Triangle Strip Generation. Paper presented at IEEE IGARSS-2005 (Proc. on DVD) held in Seoul, Korea, July 25-29, 2005.

[Agr04a] Agrawal, Anupam et al. Dynamic Multiresolution Level-of-Detail Mesh Simplification for Real-time Rendering of Large Digital Terrain Models. Proc. IEEE INDICON-2004, IIT Kharagpur, India, pp. 278-282, Dec 20-22, 2004.

[Agr04b] Agrawal, Anupam et al. TREND: Adaptive Real-time View-dependent Level-of-Detail-based Terrain Rendering. Proc. IT++: The Next Generation - the 39th Annual National Convention of Computer Society of India (CSI) held in Mumbai, pp. 146-157, Dec1-4, 2004.

[Agr98] Agrawal, Anupam et al. Delaunay Triangulation Based Surface Modeling and Three-Dimensional Visualization of Landforms. Journal of IETE Technical Review, pp. 425-433, 15(6), 1998.

[Ake02] Akenine-Moller, T., and Haines, E. Real-Time Rendering. Ed.2. A.K. Peters, 2002.

[Dou99] Douglass, D. et al. Real-Time Visualization of Scalably Large Collections of Heterogeneous Objects. Proc. IEEE Visualization'99, pp. 437-440, 1999.

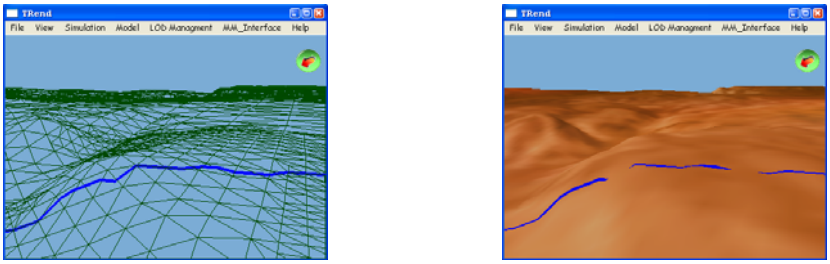
[Ker02] Kersting, O., and Dollner, J. Interactive 3D Visualization of Vector Data in GIS. Proc. of 10th ACM Int. Sym. on Advances in Geographic Information Systems, pp. 107-112, Nov. 2002.

[Lue03] Luebke, D. et al. Level-of-Detail for 3D Graphics. Morgan Kaufmann Pub., 2003.

[Sze97] Szenberg, Flavio et al. An Algorithm for the Visualization of a Terrain with Objects. URL: http://www.tecgraf.pucrio.br/~szenberg/ artigo_sib97/ artigo_sib97. html.

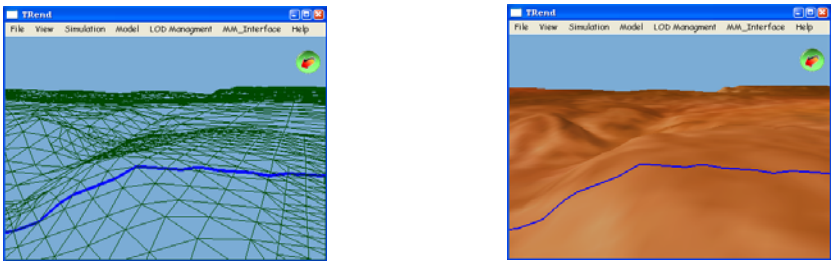
[Xia04] Xiaoping, R. and Yanmin, Z. Overlaying Vector Data On 3D Terrain. Proc. IEEE IGARSS 2004, Alaska, USA, pp. 4560-4563, Sept. 20-24, 2004.

[Zac03] Zachary, W. et al. Rendering Vector Data over Global, Multiresolution 3D Terrain. Joint EUROGRAPHICS – IEEE TCV Symposium on Visualization, pp. 213-222, 2003.



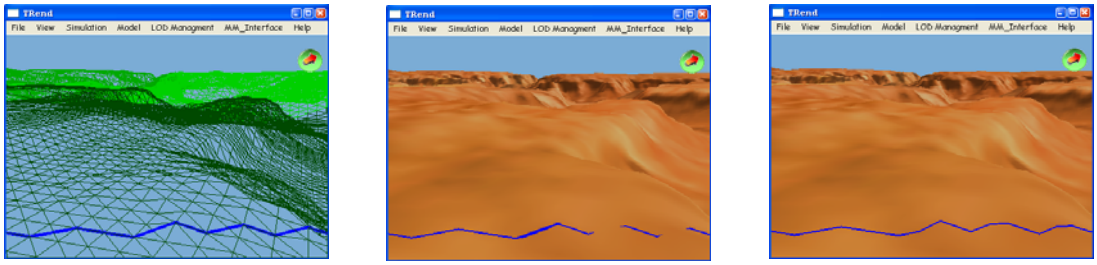
(a) **(b)**

Figure 12. Display of Vector Data over LOD 3D Terrain (without Multiresolution Modeling)



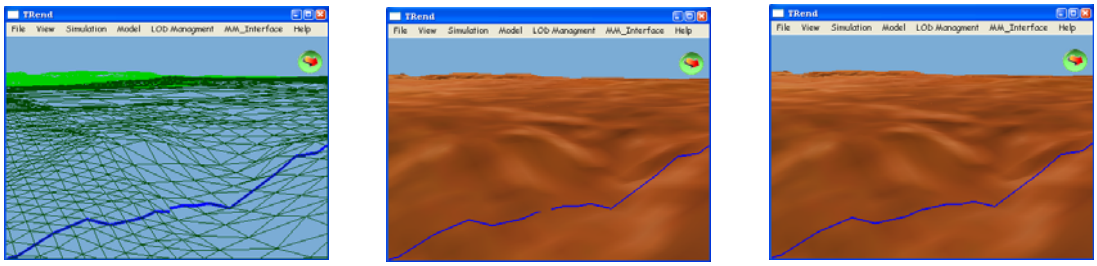
(a) **(b)**

Figure 13. Display of Vector Data over LOD 3D Terrain (with Multiresolution Modeling)



(a) **(b)** **(c)**

Figure 14. Handling Special Case 1



(a) **(b)** **(c)**

Figure 15. Handling Special Case 2