Vyatkin S. I.,
Ph.D., senior researcher, Institute of Automation and Electrometry SB RAS

Pavlov S. V.,
Doctor of Science, Professor, Vice-Rector, Vinnytsia National Technical University

Romanyuk S. A.
Graduate student, Vinnytsia National Technical University

REAL-TIME HYBRID TERRAIN RENDERING

This paper reports a three-dimensional modeling of the hybrid terrain. A terrain model is coded as multilevel height map. To visualize the terrain, tessellation is not required. During the recursive voxel subdivision on each level, voxels are projected onto the base surface (plane). The altitude corresponding to this address and a level of details is calculated, and use it to modify coefficients of the plane equation. As a result was obtained a terrain surface modulated with the values from the altitude map.

Keywords: *Shape Texture, Regular Grid, Multilevel Altitude Map, Hybrid-Based Terrain.*

1. Introduction

Visualize the topography is quite difficult, especially if it has a large elevation changes. An example is the mountainous terrain. Many systems use regular terrain elevation grid with square cells. It leads to algorithmic simplicity of computations, database uniformity, and strict definition of relationship between adjacent levels of details (LODs), which results in database generation simplification. On the other hand regular grid obviously involves information excess when considering the number of grid posts. Most novel real-time visual systems have terrain skinning processors (TSP), which guarantee continuous LODs change with respect to surface roughness and viewpoint distance. One of the main reasons to incorporate TSP in the visual system is its ability to generate terrain skin with low depth complexity (near 1) and hence reduce the image generator (IG) load when compared to traditional including terrain polygons in an environment database and LOD switching. TSP could be either an installable hardware device, or software process executed by geometry processor (GP). If TSP is an application specific device having local processor then it might be able to generate in real-time over 1000.000 terrain triangles when using regular grid and which is a considerable part of the IG system performance. This simulates search for a terrain skinning methods of IG unloading at the cost of more heavy TSP load, and particularly non-girded terrain. Regular grid could be square, or hexagonal, or triangular and so on. For Cartesian coordinate system most natural is square grid with axis collinear cell sides and grid aligned origin. Let us call this grid regular and all other – irregular. "Most" an irregular is grid with randomly spaced nodes. Others could have other kind of irregularity: square grid rotated, or shifted relatively to coordinate system, or with independently shifted nodes. Some properties of these grids are "regular", and we can use them for irregular grids evaluation. Generally, to achieve higher compression it takes more processing resources (time, memory), including decompression. In our case compression factor is a number of IG input triangles under irregular grid model with respect to that of regular grid terrain model.

Regular grid has fixed sampling rate for each LOD. In this case LODi is a set of triangles which approximates terrain so as the maximum error is not higher than appropriate constant Ei:

$$Emax < Ei \tag{1}$$

Irregular grid has fixed bandwidth, and LOD with the same maximum error could have fewer nodes in this case. Here LODi is represented by set of triangles, organized in clusters so as for each of them maximum approximation error is between current LOD maximum and next LOD maximum:

$$Ei+1<Emax<Ei \tag{2}$$

Irregular grid bandwidth wideness has two consequences. Each LOD can have "built-in" (implicit) surface roughness [1] and therefore database volume could be reduced.

For regular grid the maximum amount of memory also could be specified, because of limited mountain's height. If this limit is 300m for 10m space frequency then regular and irregular grids will introduce the same error at regular cell size 8 times less than that of irregular. The later gives us a difference in triangles number about 64 times. This is in accordance with [8].

Most of the methods use a polygonal task of relief, because there is hardware support [3-11].

Numerous methods for rendering height-based terrain surfaces have been developed [12]. Databases for terrain use DEM (digital elevation model) models. This standard is designed by U.S. Geological Survey and, on essences, is a table of heights terrain with counting out through 7.5 or 15 minutes. DEM model consists of two files, binary file of data in which recorded heights in the manner of 16- bit fixed numbers, and head file which describes a format of record of numbers used in the file of data (BigEndian or SmallEndian), but in the same way area on terrestrial surface which describe heights in the file of data. The continuous level of detail algorithm takes a two-part approach in which terrain is first divided into blocks for which a detail level can be selected at a coarse granularity [13]. The real-time optimally adapting meshes algorithm builds upon the algorithm [13] by organizing terrain meshes into a triangle bintree structure [14]. Geomorphing to the continuous level of detail algorithms described in [15]. The progressive mesh technique was extended to height-based terrain, and it enables smooth view-dependent terrain rendering with geomorphs [16].

There are methods in which no tessellation [17-21]. But these methods are slow. In order to render voxel-based terrain, proposed method must be able to convert a 3D scalar field representing the terrain into a set of vertices and triangles that can be rendered by the graphics hardware.

A method for constructing a triangle mesh whose vertices coincide with the zero-valued isosurface is the Marching Cubes algorithm [22]. Although it provides many greater capabilities, the use of voxel-based terrain without tessellation is slow for visualization. The algorithms used to extract the terrain surface from a voxel map produce far greater numbers of vertices and triangles when compared to conventional 2D terrain. The development of a seamless LOD algorithm for voxel-based terrain is vastly more complex than the analogous problem for height-based terrain. Texturing and shading of voxel-based terrain is more difficult than it is for height-based terrain. In the cases that triangle meshes are generated for multiple resolutions, arises the cracking problem. A method for patching cracks on the boundary plane between cells triangulated at different voxel resolutions was described in [23].

In paper [24] was proposed architecture for real-time visualization of non-polygonal terrain.

This paper describes a hybrid method to define and visualize terrain. To speed up the rendering graphics accelerators are used.

Terrain based on scalar perturbation functions [25]. Chosen representation of terrain data is based on regular multi-level elevation map complemented with levels of detail. This approach has several advantages (rapid generation and modification, efficient data storing and retrieving) over polygonized terrain models.

2. Non-polygonal terrain representation

Open simply connected set of points on the plane will be called the flat area [25]. Let *D* be a flat area, and $\overline{\overline{\mathrm{D}}}$ its closure. Then enter in the plane coordinate system (u, v). Where x, y, z are the Cartesian coordinates of the points in the space E^3. Three functions on the set $\overline{D}$ there:

$$x = \varphi(u,v),\ y = \psi(u,v),\ z = \chi(u,v), \tag{3}$$

Functions (3) have the following properties. Since (u_1, v_1) and (u_2, v_2) are different points of the set $\overline{\overline{\mathrm{D}}}$. $\mathrm{M}_1(\mathrm{x}_1, y_1, z_1)$ and $\mathrm{M}_2(\mathrm{x}_2, y_2, z_2)$ are the points of the space E^3. Coordinates are calculated by the formula (3):

$$\begin{aligned} &x1 = \varphi(u1,v1),\ \ y1 = \psi(u1,v1),\ \ z1 = \chi(u1,v1), \\ &x2 = \varphi(u2,v2), y2 = \psi(u2,v2),\ \ z2 = \chi(u2,v2), \end{aligned} \tag{4}$$

The set of points *M* (*x*,*y*,*z*) is called a simple surface. Then built a complex surface F, which is the graph of a function defined in 3-dimensional space. The value of $h(G(d_F))$ characterizes the deviation of the point d_F, on the surface *F* from the point *dp*

$$\vec{\mathrm{v}} = (\vec{\mathrm{d}}_{\mathrm{F}} \text{—} \vec{\mathrm{d}}_{\mathrm{P}}) \tag{5}$$

Complex surface area can be defined as the set of points in $\Re^3$, defined by the vector equation

$$\vec{\mathrm{F}} = G(\vec{\mathrm{v}}) + \vec{\mathrm{n}} \cdot h(G(\vec{\mathrm{v}})); \forall\, \vec{\mathrm{v}} \in \Re^3, \tag{6}$$

where $\vec{n}$ is the normal to the base plane.

This paper considers representation of terrain based on the base planes. In this case, the transformation *G* is a parallel projection directed oppositely to the normal vector of the base plane. We will use the notion of the terrain *F* as a combination of the base planes and the perturbation domain; it may have a rectangular contour or be defined by vector equation (6).

3. Rendering method

Terrain is defined by the base plane and scalar perturbation functions [26]. As a basic surface used a plane, and then the direction of the carrier plane normal must match the longitudinal direction of the parallelepiped. It's a region of perturbation function definition.

Multilevel height map is calculated. The initial height map the level n if the array dimension is 2^n x 2^n. Data for the level n-1 (2^{n-1} x 2^{n-1}) are obtained by choosing a maximum from four adjacent values of the level n. The zero level consists of one value.

Let's find coordinates of univariate bar - voxel V_0, which will be assigned pair vectors

$$P_0=(x0,y0,z0) \text{ and } P_1=(x1,y1,z1),\ V_0=\{P_0,P_1\}. \tag{7}$$

Further, coordinates of voxel V_0 by means of transformations G are converted in coordinate system height map:

$$\{(x0,y0,z0),\ (x1,y1,z1)\} \Rightarrow \{(u0,v0,h0),\ (u1,v1,h1)\} \tag{8}$$

Using the transformation matrix T in the height map coordinate system, which being multiplied to the matrix of geometric transformation M and gives a resulting matrix of transformation G. G=T*M;

$$T = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,0\,1\,0 \\ 0\,1\,0\,0 \\ 0\,0\,0\,1 \end{bmatrix} \tag{9}$$

Then, voxel transformed coordinates *(u, v, h, a)* in coordinate system of height map are calculated from *(x,y,z)* voxel coordinates in model space by multiplying a vector of point in model space to matrix G.

$$G\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ h \\ a \end{bmatrix} \tag{10}$$

Then voxel subdivision on Z coordinate is used

$$P_{n\,i} = P_{n\,i-1}\,,\ P_{f\,i} = (P_{n\,i-1} + P_{f\,i-1})/2\ .V_i = \{P_{n\,i},\ P_{f\,i}\}, \tag{11}$$

where V_i is a voxel of i –level of recursion, $P_{n\,i}\ P_{f\,i}$ is the coordinates of near and far-away voxel of i-level subdivision.

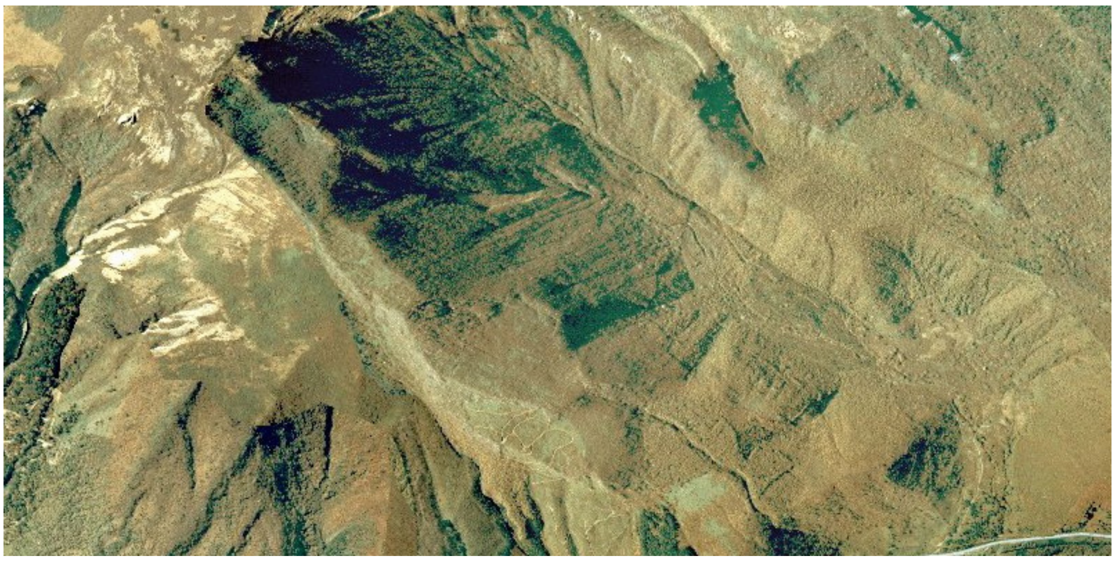
Fig-1. Hybrid-based Terrain

At each step of the partition voxel level of detail is calculated

$$\frac{1}{2^{level\ l}} < L_p < \frac{1}{2^{level+1}} \tag{12}$$

For this calculated size of rectangle being voxel projection on the multilevel height map.

Figures 1 shows the result of hybrid-based terrain modeling (height map resolution – 1024x1024). Hybrid method is a combination of height-based method and voxel-based approach.

1. Implementation and performance

Two applications which visualize the terrain based on scalar perturbation functions have been realized. The first uses only CPU for calculations. The second uses GPU for calculation of depth, normal and illumination, and CPU for geometric transformations. For image display both versions used DirectX. Testing of productivity of the offered variants of realization has been made. Compute Unified Device Architecture (CUDA) from NVIDIA was used. CUDA is a model of parallel programming. Together with a set of software, she allows to realize programs in language C for execution on a graphics accelerator. Testing was performed on the processor Intel Core i7-2700K, GTX 550Ti and GTX 750 Ti. A textured DEM of terrain (50km x 80 km) is rendered on a 1920 x 1080 view port using this method. The spatial resolution of the DEM and the texture is 1 m and 10 cm, respectively. Rendering results are shown in Table 1.

Table 1.

Height map resolution	i7-2700K	GTX 550Ti	GTX 750 Ti
256x256	802,65 ms	67,03 ms	31,07 ms
512x512	850,81 ms	71,05 ms	32,93 ms
1024x1024	856,52 ms	71,53 ms	33,15 ms

The main feature of terrain visualization in this method lies in the fact that the rendering is weakly dependent on the resolution elevation map. (See Table 1).

4. CONCLUSION

During the recursive voxel subdivision on each level, the centers of the voxels onto basic plane are projected. The current interval projection is calculated, this governs the level of detail. A cruder approximation of the initial function is chosen for a larger interval. If a more accurate representation is required, then we perform bilinear or bicubic interpolation of values of heights from the last level of detail. The computed coordinates, as well as in the case of ordinary RGB texture map, will define address.

The altitude corresponding to this address and a level of details is calculated, and use it to modify coefficients of the plane or quadric equation. As a result will be obtained a smooth surface of arbitrary shape modulated with the values from the altitude map. But the problems solved by this algorithm require much more complicated methods within the traditional approach. Indeed, the common way to present terrain with polygons requires an abundance of polygons. Besides, the number of additional problems arises such as high depth complexity, hidden polygons removal, priorities, switching between levels of detail, clipping polygons by the pyramid of vision, etc. Such problems do not appear in the proposed method.

Rendering method, described above, uses a graphics accelerator for most of the calculations. We can use parallel calculations in GPU to accelerate rendering. We successfully integrated proposed visualization method into the standard rendering pipeline. Verify the performance for the different scenes. For considered tests the application with GPU average ten times faster, than the version using only CPU.

References

[1] R. Ferguson, R. Economy, W. Kelly, and P. Ramos. “Continuous Terrain Level of Detail for Visual Simulation”, *Proceedings of Image V Conference*, June, 1990. pp. 144-151.

[2] L.L. Scarlatos “A Refined Triangulation Hierarchy for Multiple Levels of Terrain Detail”, *Proceedings of Image V Conference,* June, 1990. pp. 115-122.

[3] L.L. Scarlatos. Adaptive Terrain Models for Real-Time Simulation, Proceedings of the Digital Electronic Terrain Board Symposium, Wichita, KS, October 1989, pp. 219-229.

[4] L.L. Scarlatos (1989). A Compact Terrain Model Based On Critical Topographic Features, Proceedings of Auto Carto 9, Baltimore, MD, April 1989, pp. 146-155.

[5] L. Scarlatos and T. Pavlidis. Adaptive Hierarchical Triangulation, Proceedings of Auto-Carto 10, Baltimore, MD, March 1991, pp. 234-246.

[6] L. Scarlatos and T. Pavlidis. Hierarchical Triangulation Using Terrain Features, Proceedings of Visualization '90, San Francisco, CA, October 1990, pp. 168-175.

[7] L.L. Scarlatos. An Automated Critical Line Detector for Digital Elevation Matrices, Technical Papers of 1990 ACSM-ASPRS Annual Convention, Volume 2, Denver, CO, March 1990, pp. 43-52.

[8] L. Scarlatos and T. Pavlidis. Adaptive Hierarchical Triangulation, Proceedings of Auto-Carto 10, Baltimore, MD, March 1991, pp. 234-246.

[9] L. Scarlatos and T. Pavlidis. Hierarchical Triangulation Using Cartographic Coherence, CVGIP: Graphical Models and Image Processing, 54 (2), March 1992, pp. 147-161.

[10]L.L. Scarlatos and T. Pavlidis. "Real Time Manipulation of 3D Terrain Models", 1993 ACSM/ASPRS Annual Convention & Exposition Technical Papers, Volume 3, New Orleans, LA, February 1993, pp. 331-339.

[11]L.L. Scarlatos and T. Pavlidis. Techniques for Merging Raster and Vector Features with 3D Terrain Models in Real Time, 1993ACSM/ASPRS Annual Convention & Exposition Technical Papers, Volume 1, New Orleans, LA, February 1993, pp. 372-381.

[12]R. Pajarola, and E. Gobbetti. Survey on Semi-Regular Multiresolution Models for Interactive Terrain Rendering. *The Visual Computer: International Journal of Computer Graphics,* Volume 23, Number 8, 2007, pp. 583–605.

[13]P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, F. Nick, and G. A. Turner. Real-Time, Continuous Level of Detail Rendering of Height Fields. *Proceedings of SIGGRAPH 1996*, pp. 109–118.

[14]M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Alrich, and M. B. Mineev-Weinstein. ROAMing Terrain: Real-time Optimally Adapting Meshes. *Proceedings of the 8th Conference on Visualization '97*, pp. 81–88.

[15]S. Rottger, W. Heidrich, P. Slusallek, and H-P. Seidel. Real-Time Generation of Continuous Levels of Detail for Height Fields. *Proceedings of WSCG '98*, pp. 315–322.

[16]H. Hoppe. Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering. *Proceedings of the Conference on Visualization '98*, pp. 35–42.

[17] D. Cohen and A. Shaked "Photo-realistic imaging of digital terrain". *Proceedings of Eurographics'93*, Barselona-Spain, 6-10 September 1993, Volume 12 Number 3, pp. 363-373.

[18]D. W. Paglieroni and S. M. Petersen "Parametric heights field ray-tracing". *Proceedings of Graphics Interface'92,* 1992, pp. 192-200.

[19]G. Vezina and P. K. Robertson "Terrain perspectives on a massively parallel SIMD Computer". *Proceedings of CG International'91*, Springer-Verlag, 1991, pp. 163-188.

[20]P. Pitot, Y. Duthen, and R. Caubet "A parallel architecture for ray-casting". *Computer Graphics'89*, 1989, pp. 463-472.

[21]G. Agranov and C. Gotsman "Algorithms for Rendering Realistic Terrain Image Sequences and Their Parallel Implementation". *Proceedings of Graphicon'95*, 1995, pp. 153-161.

[22]W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm". Computer Graphics, *Proceedings of SIGGRAPH 87*, Volume 21, Issue 4, pp. 163–169.

[23]R.Shu, C. Zhou, and M. S. Kankanhalli. Adaptive Marching Cubes. *The Visual Computer*, Volume 11, pp. 202–217.

[24]S.I. Vyatkin, B.S. Dolgovesov, A.V. Yesin et al. Parallel architecture and algorithms for real-time synthesis of high-quality images using voxel-based surfaces // GraphiCon 2000 Proceedings, 2000. pp. 117-123.

[25]S.I. Vyatkin, Complex Surface Modeling Using Perturbation Functions, *Optoelectronics, Instrumentation and Data Processing*, Volume 43, Number 3, 2007, pp. 226-231.

[26]Vyatkin Sergei I., Dolgovesov Boris S. Voxel-Based Terrain Generation Using Scalar Perturbation Functions // Proc. of 13-th International Conference on Computer Graphics GraphiCon '2003, (Moscow, September 5-10, 2003). P. 165-168.