GPU-Based Rendering for Ray Casting of Multiple Geometric Data

Sergei Vyatkin[1], Oleksandr Romaniuk[2], Oleksandr Dudnyk[2]

1. Synthesizing Visualization Systems Laboratory, Institute of Automation and Elecrtrometry SB RAS, RUSSIAN FEDERATION, Novosibirsk, 1 Akademika Koptyuga avenue, email: sivser@mail.ru

2. Department of Software Engineering, Vinnytsia National Technical University, UKRAINE, Vinnytsia, 95 Khmelnytske shose str., email: rom8591@gmail.com, dudnyk@vntu.edu.ua

Abstract: **We present a new GPU-based rendering method for ray casting of multiple geometric data. Our approach supports a polygonal data to calculate scattered light. Terrain is represented for the base of scalar perturbation functions. The geometric model is based on non-polygonal representation. We present a new representation scheme for freeform surfaces it is possible to combine basic surface and perturbation functions. To recognize the place it is often sufficient to fill areas with the generalized texture patterns, such as "themes". Themes designed beforehand and consumption of memory for storing them is not huge.**

Keywords: **freeform surfaces, recursive multilevel ray casting, shape texture, thematic texture, scattered light, volume-oriented visualization.**

I. Introduction

Using traditional polygonal representation for the example complex surface give rise to a range of problems such as visible surface determination, depth complexity handling, controlling levels of details, clipping polygons by viewing frustum, geometry transformations of large number of polygons [1, 2].

A method for constructing a triangle mesh whose vertices coincide with the zero-valued isosurface is the Marching Cubes algorithm [3]. Although it provides many greater capabilities, the use of voxel-based terrain in real-time virtual simulations also introduces several new difficulties. 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 [4]. Using a voxel-based model however, can achieve the same results at a much lower hardware requirement.

The proposed method includes the following main features: rendering second-order surfaces; rendering surfaces defined on a regular altitude grid (Shape texture); rendering freeform surfaces; scattered light visualization.

II. Volume-Oriented Rendering

Along with possibility to rasterizing 2D space, the main feature of the proposed method is rasterizing made by 3D space quadtree subdivision of pyramids of different levels, which constitute the whole pyramid of vision. Then a pyramid of the lowest level is binary-tree subdivided into voxels of the lowest level - Recursive Multilevel Ray Casting (RMLRC). In the latter case, extent space regions (in depth) can masked out. Depth subdivision of space performed on the logarithmic basis. The technique of RMLRC allows determining an intersection of a ray (pyramid) of any level with a surface effectively. It is also suitable for fast culling of a spatial region outside an object. The core of this approach is effective search of volume elements (hereinafter voxels), involved in current frame generation, fused with direct projection [5]. If geometry transformation is described by matrix (C) then new calculated matrix of quotients (Q)'=(C)T*Q*(C) does in the coordinate system P the same as the matrix (Q) in the coordinate system P. The matrix (C) of projecting transformation calculated once for particular frustum. So the use of projecting transformation generalizes the discussed algorithm for pyramidal volumes (frustums) and allows synthesize images with perspective. In our approach a virtual environment can be described using polygonal models, surfaces of second order, three-dimensional scalar functions, defined on discrete grid h=h(u,v), free-form surfaces, which are represented by composition of base surfaces and shape-driving functions. The proposed RMLRC algorithm has several advantages in processing mentioned surface models as compared with known algorithms. Therefore, for example, RMLRC algorithm applied for second order surfaces simplifies calculation of Phuong shading, since at the last level of subdivision derivative plane coefficients for the surface obtained. Further, the possibility of second order and free form surfaces composition allows producing objects that are more complex. Photorealistic visualization of complex surfaces, for instance, specified by a three-dimensional function, defined on discrete grid h=h(u,v) can be done without intermediate polygonal approximation. Such surfaces represented by differential height map, i. e. the algebraic carrier surface is given and in each grid node, only deviation from this surface needed. This representation facilitates calculation of contiguous levels of details as well as makes easy quality filtering. Geometry transformations apply only to the carrier surface and the height map is kind of surface

texture. Freeform surface representation differs from known representations, such as for example, Constructive Shell Representation [6], since the former is free of problems arising when complex surfaces are approximated with large number of Bezier patches, B-splines etc. (problems of rendering patches boundaries, problems of warped halfspaces). The proposed approach allows specifying surface representation as composition of base surfaces and shape-driving functions. Small number of such functions is enough to describe surfaces of any form including non-convex, with holes etc. (Fig. 1). Discussed surface representations and the approach to their processing facilitate description of such phenomena as waves, dynamic surface warping, morphing, deformations and animation of wide range of surfaces.

Fig. 1. Freeform surface (F-117) over terrain. Height map resolution 512x512. Screen resolution 1920 x 1080

III. Freeform Surfaces

Traditionally, the parametric form represents each patch in a freeform surface as a mapping from 2D parameter space to 3D space. Although parametric patches are powerful for constructing freeform surfaces, processing these patches poses fundamental problems, only two - constructive solid geometry (CSG) and boundary representation (Brep) - commonly represent solids exactly, that is, without approximation. In particular, "separation" and other such problems associated with curved halfspaces are hard to solve in parametric patches. We present a new representation scheme for freeform surfaces - it is possible to combine basic surface F(x,y,z) and shape-driving function R(x,y,z) or perturbation function, where shape-driving function represents smooth deformation of basic surface [7]. The shape-driving function is composed of several second-order functions using logic intersection and union operations, it is recursively evaluated during subdivision itself, and therefore, computation of resulting R (x, y, z) is minimal. At the same time, it is moderate in computations, so it can implemented in hardware to carry on realistic object outlook.

Because of application of the 3D space, subdivision algorithm the possibility appears to render surfaces, which usually consist of huge amount of patches such as freeform surfaces.

IV. Shape Texture

Non-regular terrain algorithms are more complex but have the potential to reduce the number of polygons that the system must process [1]. However, for many terrains with limited elevation gradient, the average expected reduction in the image generator load is small for irregular grid as compared to regular grid. Each irregular grid node consumes much more computational time and memory than that of regular. Systems, which employ irregular grids, are very limited in the number of LODs that can handled. In addition, it is very difficult to solve problems concerned with terrain deformation when using irregular grid (explosions, pits in the ground).

Volume oriented rendering and uniformity of object processing result in an efficient hidden surface removal and detection of spatial collisions. Chosen representation of terrain data is based on regular multi-level elevation map complemented with levels of detail [7]. This approach has several advantages, such as rapid generation and modification, efficient data storing and retrieving, over polygonal models.

V. Thematic Texture

Systematically analyzing geographer's work, Barr [8] has described it as operating with the "geographic matrix". For clearness, it can thought as three-dimensional, i.e. two dimensions correspond to the geo-coordinating system, and the third one corresponds to a description list. Each element represents a "geographic fact" (Fig. 2). For a large database, it is resource-intensive to obtain and process geospecific photo-texture for the entire gaming area. It assumed that most of the benefits of geospecific photo-texture could attained by combining a large amount of generic photo-texture with a small amount of geospecific photo-texture.

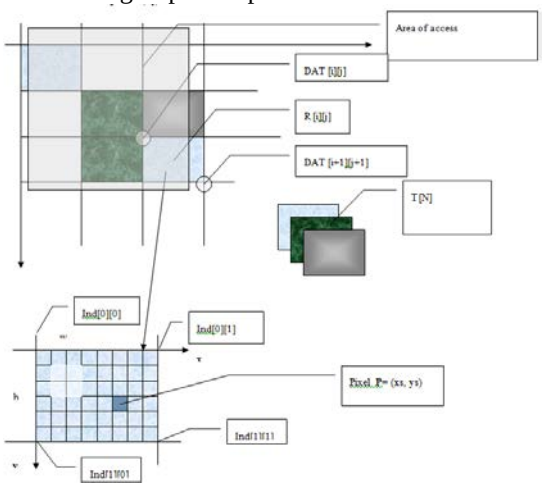

Fig. 2. Thematic texture

Simulation of big areas of terrain in the training system image generators requires the large capacity of memory for the photographic data storage. The usual approach to reduce this information is to compress the data with the help of the methods such as JPEG, etc. In the approach of the texture composition it supposed, that the observer (pilot) often is not interested in the exact, photographic information about the areas, covered by, for example, forest, water, mountains, etc. To recognize the place it is often sufficient to fill these areas with the generalized texture patterns (we will call them "themes" afterwards). Themes designed beforehand and consumption of memory for storing them is not huge. Area of the terrain photo-texture with the same theme approximated by the polygons, which are bounded by curves. Each of the

polygons has the pointer to the texture theme and to the procedure of generating the boundaries between the areas with the different themes. The procedures of generating the boundaries do not reproduce true boundaries exactly, but have the goal to keep their features. There are two main problems in the technique of the texture composition. The first is the generation of the boundaries between the areas with the different themes. To resolve this problem, the algorithms, which allows generating the different kinds of boundaries, designed. Among the possible kinds of the boundaries the following can be selected: fractal boundaries, following natural boundaries in the maps of the texture themes (for example, following the streets in the texture map of the town), procedural blend zones (beaches, etc.). The second problem of the technique of the texture composition is the filling the areas by the homogeneous texture pattern. Filling the big area of the terrain by the small texture pattern leads to the undesirable periodic, unnatural picture. The introduction of the different kinds of non-regularity, which keeps the features of the texture pattern, allows obtaining more natural picture.

The first stage is the usual rendering of polygons, describing the texture, and filling the service buffer by the pointers to the texture themes and kinds of boundaries. The second stage is access to the buffer through the perturbation map, to obtain the color of each texel. Perturbation map calculated beforehand. Note that approach is adapted for the generation of the fractal boundaries, but it allows to use the procedural blend zones also, and to generate boundaries, which coincide with the natural boundaries in the maps. To generate fractal boundaries the following actions must be done (some are simplified): access to the perturbation map by address, defined by the U and V coordinates of the given texel, to obtain the additions to the values of U and V. Obtaining the new values of U and V by the addition with the values from the perturbation map. Access to the buffer of the source map by the address, defined by new values of U and V, to obtain the index of the texture theme; access to memory of the texture themes, to obtain the color of the texel. Boundaries between regions, obtained by the method described, looks fractal due to the fractal character of the perturbation map used. Turbulent noise used to fill this map. As it is well known, the higher spatial frequencies of the turbulent noise have the smaller amplitude (and on the contrary). All of the boundaries, obtained by this method, are similar, but different themes can have different boundaries, with different widths (amplitudes) and "crooknesses" (frequencies). Therefore, the sequence of actions, described above, must be little more complex: access to the source map to obtain the characteristics of boundary; scaling of U and V coordinates to obtain required frequency of the noise, then access to the noise map (perturbation map). Multiplication of the values, obtained from the noise map, by the constant value to provide required amplitude of the noise, then modification of U and V; access to the source map to obtain the index of the texture map; access to memory of the texture themes to obtain the color of the texel.

Obviously, blending can become correct only if textural areas have coinciding borders. To satisfy a condition an additional database is introduced which describes faces of blending areas. These faces do not differ from others and their special role is to connect textural areas with different borders. For this purpose, such face owns border description of one area, and a texture of the other area. While the database created, these faces constructed by expanding polygons, i.e. by moving their edges. By the way, procedurally generated areas can obtained exactly the same way: if a certain polygon with any textural theme has a prescribed blended border, the procedure described above executed and a newly created polygon assigned descriptions of the area created.

The boundaries coinciding with natural borders within a textural picture are considered. Such borders can formed by processing the image from the original map buffer after all textural polygons have been rasterized. Beside difficulties with forming the borders between areas of different textures, there is a problem concerning inner filling of these areas by a homogeneous picture. Therefore, in the proposed approach the textural coordinates disturbed by a noise-map to provide singularities on the picture. It is possible to reuse noise values previously obtained while determining the textural theme index. Let us estimate memory and time consumption within the proposed approach. Note, that original map resolution can be less than final texture map resolution. Maximal spatial frequency of the final image determined by a spatial frequency of textural themes and a noise-map; therefore, they can be scaled, so that one texel of original map is transformed into an area filled by a certain picture with a fractal border. Thus, minimal size of original map determined by a required precision of border presentation. The image kept in an original map consists of contiguous areas each filled by a certain textural theme index, that is why these data can be safely compressed using, for instance, Color Cell Compression method (this will ensure up to six times compression). Besides, this task can accomplished during textural polygon rasterizing with no additional time consumption. As noise-map, contents do not explicitly emerge in the final image, periodically wrapped map can used for this purpose and unwanted regularity will not noticed. For instance, fractal border has a repeating structure only if it originally presented as a straight line parallel to one of textural coordinate axes. The task of rasterizing textured polygons and filling original map is not extremely difficult and can be handled by any sufficiently powerful universal processor, moreover the time is limited only by a needed uploading due to the camera’s motion. To determine the color of a texel several steps are completed: fetch operation from original map, from noise-map, then, modification of U and V coordinates, fetch operation from original map and from textural theme memory. As one may see, these operations are quite simple and, therefore, a conventional textural fetch operation can substituted. This implementation has several significant advantages, as follows: overall memory requirements are less then those for a global texture; texture animation features become available by simple means, as fractal border can made animated, if a shift is applied to a noise-map at each frame. In addition, noise magnitude can change resulting in an interesting effect. In analogous way, texture inside the distinct regions can animated. Besides, this approach would be naturally used also for all other feature

textures (fetch operation from original texture memory should be excluded). All this could be useful while rendering, for instance, water surface, flames, snow, etc.

It should be also mentioned that implementation described above requires original map, noise-map and textural theme (as MIP map) storing, Generation of levels of detail for textural themes and noise-maps is of no difficulty, however levels of detail for original map is not clear, because these data are indices, which cannot be blended like colors. A straightforward solution of this problem is to eliminate details that are smaller than corresponding texel size, i.e. only wholly covered texels handled during rasterizing original textural polygons.

VI. Scattered Light

The vertex shaders compute the light reaching the eye from a source or a reflective object and fog component. The inputs to the vertex shader are vertex position, transformation matrices, sunlight intensity, the sun direction, the various extinction and scattering coefficients. In this implementation using calculate them per-vertex in a vertex shader. If the Sun is not too low in the sky, this approximation gives good results at a low cost. For these assumptions, the illumination of the ground plane and light reaching the eye from it may easily derived (Fig. 3). To find the color perceived along a line of sight, we must consider both the light reflected by objects along the line of sight, attenuated by the inverting fog, and the light scattered towards the eye by fog along the line of sight.

Fig. 3. Terrain, F117 and scattered light. Height map resolution 512x512. Screen resolution 1920 x 1080

VII. Performance

The visualization time is reduced by using the computational resources of a graphics processing unit with compute unified device architecture (CUDA). The result of running the programs on different processing units is the same even if they may have a different number of streaming multiprocessors. A large portion of the cube will be computed in parallel. Among the functions of the graphics processing unit was to calculate the coordinates of points of the surfaces, normals, and illumination. Geometric transformations were performed by the central processing unit (CPU). Rendering results using CPU (i7-2700K) and GPU (GTX 550Ti, GTX 750 Ti, GTX 950) are shown in Table 1.

TABLE 1. Performance on CPU and GPUs

Resolution	i7-2700K	GTX 550Ti	GTX 750 Ti	GTX 950
256x256	802,65 ms	67,03 ms	31,07 ms	26, 55 ms
512x512	850,81 ms	71,05 ms	32,93 ms	28, 01 ms
1024x1024	856,52 ms	71,53 ms	33,15 ms	30, 22 ms

The surface obtained will be smooth (Fig. 1 and Fig. 3), and a small number of perturbation functions (16) will be necessary to create complex surface forms. The figure shows a result of modeling a scene object by means of free forms, whose description required 2Kbyte information, which is 500 times less than the polygonal description that would take 1Mbyte information.

VIII. Conclusion

The main advantages include ease of calculation of points on the surface with quick search and rejection of the regions not occupied by the scene objects. A factor of 100 or more decrease in the number of surfaces for describing curved objects. Operations of the Geometry Processor (CPU) becomes significantly easier and data stream from the Geometry Processor (CPU) to the Renderer (GPU) reduced. Paralleling of the system becomes easier because the total system performance determined by the performance of the Renderer (GPU) which operations can be easily distributed between different screen regions. All problems connected to terrain generation solved. The opportunity to describe objects with grid values, i.e., with shape texture map; it becomes possible to morph objects; it is achieved by plain interpolation or animation of shape texture. It can used to render clouds, explosions, waves on water, etc.

References

[1] R. Pajarola, E. Gobbetti. "Survey on semi-regular multiresolution models for interactive terrain rendering". *The visual computer. International Journal of Computer Graphics*, Vol. 23, P. 583–605, 2007.

[2] 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" *In Proceedings of Siggraph*, 1996, P. 109–118.

[3] W. E. Lorense, and H. E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm". *Computer Graphics, Proceedings of SIGGRAPH 87*, P. 163–169.

[4] R. Shu, C. Zhou, and M. S. Kankanhalli. "Adaptive marching cubes". *The Visual Computer*, Vol. 11, P. 202.

[5] Vyatkin S. I., Romanyuk A. N., and Savitska L. A. "Multi-level ray casting of function-based surfaces" *Journal of Physics: Conference Series*, 803, №. 1.

[6] P. Menon, and T. J. Watson "Constructive Shell Representation Freeform Surfaces". *IEEE Computer Graphics* 0373-17-16, 1994.

[7] S. I. Vyatkin. "Complex Surface Modeling using Perturbation Functions". *Optoelectronics, Instrumentation and Data Processing*. 43 (3), P. 226–231 (2007).

[8] R. Barr "Automated cartography and geographical information: Part 2", *Advances in Computer Graphics*, Volume 11, Springer, page 29, 1986.