Sergei I. Vyatkin,
Alexander N. Romanyuk,
Sergei A. Romanyuk,
Peter Velichko

FUNCTION-BASED TECHNOLOGY OF VISUALIZATION

Abstract

For visualization of high-realistic expressing is realized transition from the scan of two-dimensional space to three-dimensional.

Task four types of the free forms of objects alongside with polygonal, implicit perturbation functions, scalar perturbation functions and 3D-data of volume (voxel's array) is considered. Rasterization of enumerating primitives without partitioning them on polygons is proposed.

1. Introduction

Real-time computer graphics oriented to 3-D scene visualization has attained appreciable success nowadays. Though a sufficiently high realism of real-time scene imaging has been attained, some problems are still present (e.g., imaging of large terrain regions), where it is necessary to store and visualize scenes containing a greater number of polygons than it is implemented in the present-day systems. For instance, the problem of imaging of mountains requires for initial description hundreds of thousands of polygons. On the other hand, exact modeling of shapes of the car, airplane, submarine frames requires thousands of spline-surfaces (curvilinear areas defined by polynomial functions) whereas the case of definition by polygons will require tens and sometimes hundreds of thousands of polygons. Moreover, polygonal 3-D graphics with scanning of polygons in the image plane is not three-dimensional in the full sense of the word. Information presented to the user in such an approach is incomplete. The main point is the absence of information on object depth. This case implies not the absence of the Z-coordinate of the surface point but the absence of information about the beam passing through the object.

We present results of some investigations concerned with modeling of a system in which it is proposed, to use along with the polygonal representation, object representation by free forms in the form of real and scalar functions and volume spaces of voxel arrays. The possibility of freeform volume visualization is investigated. A recursive algorithm of rendering with object space division with regard to perspective is proposed for visualization.

2. Representation of object free forms.

The characteristic feature of the proposed freeform representation is:

Firstly, the fact that the main primitives are presented by second-order surfaces, i.e., quadrics [1]. A primitive-quadric is the basis for constructing the rest of objects. The quadric is defined by a real continuous descriptive function of three variables (x1, x2, x3) En in the form $F(X) \geq 0$. Quadrics are considered as closed subsets of the Euclidean space En defined by the descriptive function $F(X) \geq 0$ where F is the continuous real function; X= (x1,x2,x3) - is the point at En specified by the coordinate variables. Here $F(X) > 0$ specifies points inside the quadric, $F(X) = 0$ – the points at the boundary, and $F(X) < 0$ - the points lying outside and not belonging to the quadric.

Using the quadrics, the first class of free forms is constructed with the use of real perturbation functions. This type of free forms is suitable in generation of artificial (man-made) objects.

Secondly, the second class of free forms is proposed with the use of scalar perturbation functions with respect to the basic plane or the quadric, for example, to generate a terrain or sculpture models.

2.1 Perturbation functions in the implicit form.

It is proposed to describe complex geometric objects by defining the function of deviation (of the second order) from the basic quadric. Free forms are constructed on the basis of quadrics. The freeform is a composition of the basic quadric and the perturbation perturbation function. In other words, the composition of the basic quadric and the deviation function is a new perturbation function, i.e., a derivative for another basic quadric. The surface

obtained will be smooth, and a small number of perturbation functions 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 4Kbyte information, which is 500 times less than the polygonal description that would take 2Mbyte information. Thus, the problem of object construction reduces to the problem of quadric surface deformation in a desired manner rather than to approximation by primitives (polygons or patches represented by B-spline surfaces). In addition, while solving the descriptive function in the form of inequality $F(X) \geq 0$, we can visualize not only the surface but also the internal structure of the object

2.2 Perturbation functions in the scalar form.

It is proposed to describe complex geometric objects by defining (in the scalar form) the second-order function of deviation from the basic surface or (in the simplest form) from the basic plane. A terrain is a particular case of such objects; it is defined by means of the basic plane and the perturbation function defined in an infinitely long parallelepiped. Values of the perturbation function are specified at the parallelepiped cross-section by a 2-D height map. As a basic surface we may use a plane, and then the direction of the carrier plane normal must match the longitudinal direction of the parallelepiped - the region of perturbation function definition.

The dream of every simulator designer is to render a terrain as easily as a texture. The algorithm considered does this. Mountain landscape generated for altitude/height map 200x200). A terrain model is coded as differential height map, i.e. the carrier surface is defined by algebraic means and only deviation from this basic surface is stored in the each node. Such modeling method simplifies creation of smooth detail levels and shading. The data of height grid is not subject to geometry transformations as the triangle vertices do. The geometry transformations are only required for the carrier surface. During the recursive voxel subdivision on each level we project the centers of the voxels onto some plane. The computed coordinates, as well as in the case of ordinary

RGB texture map, will define address in the so-called “altitude map” or “shape texture” [1]. We calculate the altitude corresponding to this address and a level of details, and use it to modify coefficients of the plane or quadric equation. As a result we will obtain 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. The Geometry Processor works with the single plane. The corresponding traversing of the tree and the set of masks provide the right priority order. The backside of a terrain is rejected automatically. The clipping terrain by the pyramid of vision becomes unnecessary since sampling of just required altitudes from the altitude map is provided automatically by the rendering algorithm. To switch between levels of detail the same procedure is used as for the ordinary texture.

3. Rendering technique.

We used the multilevel ray casting algorithm [2-8], which performs efficient search for volume elements - voxels which participate in image generation. At the first step of recursion, the initial viewing pyramid is divided into four smaller subpyramids in the screen plane. At the stage of division of space along the quaternary tree, 2-times compression and transfer by ±1 along two coordinates. If the intersection is determined, then the subpyramid is subject to the next recursion level. Subpyramids that do not intersect with the object are not subject to further immersion to recursion, this corresponds to elimination from consideration of square screen spaces which the subpyramid (and, consequently, the object surface) is not mapped to. The viewing pyramid is subdivided until reaching the maximal set level of recursion. The technique has an advantage that it allows discard of large parts of empty space at an early

stage. While searching for voxels containing the imaging object surface areas, the pyramidal space is traversed along the quaternary tree whose leaves are roots of binary trees. The multilevel ray casting technique allows us to determine effectively and quickly belonging of rays of different levels (pyramids) to surfaces, and discard space regions outside the objects.

While visualizing the surfaces a test is verified for belonging of only intersected voxels (unit volume elements), external and internal voxels are discarded. To improve imaging realism and extend the class of objects imaged (translucent structures with internal density distribution, 3-D textures), it is necessary to image the internal translucent object structure. For this purpose, not only voxels lying on the surface but also voxels inside the object must participate in imaging. Therefore, while dividing the volume the internal object parts are not discarded, for them algorithm recursion is performed further. Scanning of the scene along the Z-coordinate, which corresponds to scanning of the volume through the depth, is not interrupted upon meeting the surface but is continued until the volume is scanned completely or a certain value of transparency higher than a threshold value is stored. To reduce the computation time the algorithm is adapted to quick passage of homogeneous spaces of objects, for which it is unnecessary to scan the volume completely reaching the last recursion level, it is necessary to “skip” empty or homogeneous spaces along the Z-coordinate and immediately calculate the color and the total transparency. Since ray passage through empty space makes no contribution to the final image, then the skip of the empty space is able to make the processing substantially fast and does not affect the image quality.

3.1 Paged Texture Memory

Many of today’s PC architectures employ the “universal memory” architecture. Main memory is used for program execution, database storage, and texture storage. While this approach minimizes cost, it is difficult to guarantee a sustained fill rate because the available memory bandwidth is constantly being spread over these various non-uniform processes. To complicate matters, even

the largest system memories are too small to accommodate modern terrain databases that are completely covered with geospecific phototexture. An end-to-end system solution is necessary to handle the texture requirements of the simulation environment. To compensate for the varying demands on system memory, a texture memory cache is usually located on the rendering engine. But caches that are sufficiently large enough to smooth out the variations use precious gates in the core of the rendering chipset. Frequently used textures such as those used for special-effect animations and common database features may be locked into part of this memory. The remaining space can be used to hold the terrain texture in the vicinity of the eyepoint. Real-time software pages terrain texture into this space from secondary storage (usually hard disks) based on the eyepoint's geographical location within the database. Obviously, the various system busses and secondary storage access times must be sized to handle the expected paging rates. Texture compression helps relieve the bandwidth demands and, depending on where the texture is decompressed, may permit the use of a smaller texture memory.

4. Conclusion

Our investigation in the volume-oriented visualization technology have made it possible to reveal some advantages in both the scene representation technique and the rendering algorithm oriented to real-time implementation. The change over from rasterization in the image plane "back-end" or the image-space end of the graphics pipeline) to volume rendering, ("fronted" or the object-space end of the graphics pipeline) in combination with the proposed object definition techniques, though increases the amount of real-time computation, as a whole, nevertheless it results in some merits improving the scene imaging realism.

The main merits of our approach are the following:

—reduced number of surfaces for describing curvilinear objects (representation of objects by freeform surfaces reduces 100 times and more the database description compared with their representation by polygons)

—reduction of the load on the geometry processor and decrease of data flow from it to the video processor

—sufficiently simpler construction of terrain because the preliminary surface triangulation and the viewing pyramid clipping are unnecessary (to change the level of detail we use a mechanism similar to the usual texture sampling [8])

—the computation time in terrain generation is practically independent of the height map resolution and depends only on the screen resolution

—the possibility to process voxel arrays bounded by freeform surfaces

—simple animation and morphing of scenes.

A statistics will bring for the example for voxel-based terrain. Where is not required triangulation of terrain, in the given technology main, on than it is based, is a hierarchical mechanism by control levels of detail. This practically reminds MipMapping usual texture of color, as well as in the event of the texture of color a time of calculations does not depend on permits of height map. If conduct a get fat analogy with the texture, all its positive characteristics are inherited given technology, to which possible refer including and simplicity an animation such surface without computing expenses on geometric transformations of tops of triangles, if such surface was tessellated.

Possible also metamorphosis of non-homeomorphic objects that without breakups impossible and following splicing in the event of polygonal surface. Another feature of the proposed method is the possibility to define dynamic objects like waves or surface movements by defining time dependent function, which alters scalar field values. As a result, one can visualize propagating waves, surface deformations (explosions, waterspouts of navy battle, sinking of a submarine in a wavy sea with semi-transparent water, cloud shape deformations, etc.). Objects can move and penetrate within each other, can change shape and size.

All these merits of our approach form the ground for creating a new class of computer visualization systems for various applications. The proposed

algorithm of rastering along with the possibility to visualize arbitrary surfaces of free forms and inhomogeneous volume spaces offers a wide scope of application.

References

1. S.I. Vyatkin, B.S. Dolgovesov, A.V. Yesin et al. "Geometric Modeling and Visualizatiom of Free Forms" // Autometry - 1999.-N6.

2. S.I. Vyatkin, B.S. Dolgovesov, A.V. Yesin et al. Voxel Volumes volume-oriented visualization system, International Conference on Shape Modeling and Applications (March 1-4, 1999, Aizu-Wakamatsu, Japan) IEEE Computer Society, Los Alamitos, California, 1999, pp. 234-241.

3. S.I. Vyatkin, B.S. Dolgovesov, S.E. Chizhik, “Synthesis of virtual environment with object-space recursive subdivision” - Graphicon’98 Proceedings, S. Klimenko et al. (Eds). 1998.

4. S.I. Vyatkin, B.S. Dolgovesov, A.V. Yesin et al. "The Real-Time System of Volume Visualization VOXEL-VOLUMES", 4 International Conference - Recognition of Objects and Analysis of Images, Novosibirsk-1998.

5. S.I. Vyatkin, B.S. Dolgovesov, A.V. Yesin et al. “The Visualization of Non-Homogeneous Structures in Real-Time with Multi-Level Recursive Ray Casting”, 4 International Conference - Recognition of Objects and Analysis of Images, Novosibirsk-1998.

6. S.I. Vyatkin, B.S. Dolgovesov, A.V. Yesin et al. “The Visualization of Free Forms and Voxel Volumes in Real Time” //- Graphicon’99 Proceedings, S. Klimenko et al. (Eds). 1999.

7. S.I. Vyatkin, B.S. Dolgovesov, A.V. Yesin et al. "Algorithm of Visualization of Free Form Surfaces" // Software and Systems - Moscow, 1999.

8. S.I. Vyatkin, B.S. Dolgovesov, A.V. Yesin et al. Geometric Modeling and Visualization of Functionally Defined Objects // Optoelectronics, Instrumentation and Data Processing. – 2000 -N6.