Sergey I. VYATKIN [1], Alexander N. ROMANYUK[2], Sergii V.PAVLOV[2], Maryna V. MOSKOVKO[2], Nursanat ASKAROVA[3], Azhar SAGYMBEKOVA[3], Waldemar WÓJCIK[4], Andrzej KOTYRA[4]

Institute of Automation and Electrometry, Siberian Branch of the Russian Academy of Sciences (1), Vinnytsia National Technical University (2), Kazakh National Research Technical University after K.I. Satpaeva (3), Lublin University of Technology (4)

doi:10.15199/48.2017.05.16

Fast ray casting of function-based surfaces

***Abstract**. This paper deals with the fast ray casting of high-quality images, a method of defining free forms without approximating them with polygons or patches, issues of using perturbation functions for animation of the surfaces of 3D objects. A method for visualizing functionally defined objects adapted for graphics processing units (GPU) is proposed.*

***Streszczenie.** W artykule zaprezentowano metodę rzutowania promieni wykorzystującą tzw. funkcję perturbacji zamiast aproksymacji za pomocą wieloboków w odniesieniu do obrazów wysokiej rozdzielczości w celu animacji powierzchni obiektów trójwymiarowych. Ponadto, zaproponowana została metoda wizualizacji obiektów z wykorzystaniem procesorów graficznych (GPU). (**Szybkie rzutowanie powierzchni opisanych za pomocą funkcji**).*

Keywords: geometric objects, perturbation functions, fast ray casting, graphics processing units.
Słowa kluczowe: obiekty geometryczne, funkcje perturbacji, szybkie śledzenie promieni, procesory graficzne

Introduction

In the generation of 3D scenes, a polygonal definition of object models is most frequently used which has a number of limitations. Wireframe models of 3D objects are approximate. Increasing the realism of graphic scenes involves increasing the level of detail for correct approximation of the surfaces of objects of the real world, with the rates of increase in the geometric complexity of 3D images exceeding the growth rate of the performance of graphics tools. Achieving photorealism usually requires more than one million polygons in the scene and a trend to further increase in the level of detail is observed. The amounts of data for visualization of three-dimensional objects with complex surface are close to voxel models. Polygonal models are in principle unsuitable for obtaining many visual effects necessary for a realistic displaying of scenes. Skeletal animation cannot provide high-quality animation of flexible materials or 3D metamorphosis or morphing of objects.

These drawbacks can be eliminated by means of analytical definition of objects and their rasterization using ray tracing algorithms. The functional representation describes most accurately the object geometry and has the smallest size of the required data. Procedures of functional representation demonstrate compact and flexible representation of surfaces and objects that are the results of logical operations on volumes. Its disadvantage is complicated geometrical processing and visualization in real-time.

The paper describes free forms based on the perturbation functions. It is shown that an adequate surface smoothness and a compact object description can be achieved using a limited number of base and perturbation functions. The aim of this work is to develop a method for visualizing functionally defined objects based on perturbation functions using graphics processing units.

The previous works

One of the main disadvantages of the known visualization methods is complexity of the calculation of points on the surface. Thus, the ray marching method does not guarantee detecting the surface, and, in addition, it is comparatively slow [1, 2, 3]. The method of determining the intersection of a ray with an implicitly defined surface is too complex to calculate the L- and G-parameters [4].

In the tracing method, finding the maximum radius when no point of the volume lies within the sphere is a nontrivial task [5]. Ray tracing with analysis of the interval for complex functions requires individual calculations for each ray and each interval along this ray [6]. In fast tracing, search for the rays intersecting the surface requires a lot of calculations and is not efficient enough as the clustering procedures of this method do not solve this problem completely [7]. A ray tracing method for imaging surfaces defined by algebraic polynomials of high degree is described in [8]. However, it is not easy to model real objects using polynomials. Nor is the accuracy of approximation of the initial function with a Bezier curve is guaranteed. Another disadvantage of this method is that transformation of objects to another coordinate system is a complex task. Therefore, the creation of dynamic scenes is problematic.

There is another technique for the visualization of analytically defined objects using GPU [9], which is based on conventional single-step tracing of rays. The highly parallel structure of a GPU makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel GPUs are done in parallel [10]. A distinctive feature of this method is that the step size is not constant but is chosen in each iteration where the radius of the sphere centred at the current point on the ray is determined. A disadvantage of the method is that finding a suitable radius is a difficult task. For static scenes, the authors of the algorithm preprocessed data. Therefore, as in the previous method, visualization of objects that change their shape and position in time requires a significant computational cost.

3D Models

Functionally defined objects are constructed from second-order surfaces with analytical perturbation functions, which ensure reaching a high coefficient of geometric compression of highly realistic three-dimensional objects. It is proposed to describe geometric objects (free forms) by defining the function of deviation (of the second order) from the basic surface of the second order (quadrics) [11].

The function is defined by a second-order algebraic inequality with three unknowns x, y, z in the form $F(\mathbf{X}) \geq 0$. The surfaces are considered as closed subsets of the Euclidean space E^3, which are defined by the describing function $F(\mathbf{X}) \geq 0$, where F is a continuous real function and $\mathbf{X} = (x, y, z)$ is a point defined by coordinate variables in the space E^3. The expression $F(\mathbf{X}) > 0$ defines the points inside the surface, $F(\mathbf{X}) = 0$ defines the points on the boundary, and $F(\mathbf{X}) < 0$ defines the points located outside and not belonging to the surface.

Free forms are constructed on the basis of quadrics. A free form is a composition of the basic quadrics and perturbations:

(1) $$F'(x,y,z)=F(x,y,z)+\sum_{i=1}^{N} f_i R_i(x,y,z)$$

where f_i is the form-factor; the perturbation function $R(x, y, z)$ is found as follows

(2) $$R_i(x,y,z)=\begin{cases} Q_i^3(x,y,z), if\ Q_i(x,y,z)\geq 0 \\ 0, if\ Q_i(x,y,z)<0 \end{cases},$$

where $Q(x, y, z)$ is the perturbing quadric.

In order to make the surface smooth, the degree should be higher than two (2). This condition ensures the continuity of the function and its derivative. The obtained surfaces are smooth, and creation of complex surface forms requires few perturbation functions.

The proposed method of describing objects of 3D scenes by basic surfaces and functions has a compact presentation, which allows one to reduce the volume of data transfer by a factor of 10 to 1000, depending on particular three-dimensional scenes and models. Objects with flat faces can be also defined fairly easily, e.g., a cube can be defined by three quadrics. Moreover, in solving the describing function in the form of the inequality $F(\mathbf{X}) \geq 0$, it is possible to visualize not only the surface, but also the internal structure of the object.

Two major types of elements of the set of geometric objects are simple geometric objects and complex geometric objects. A complex geometric object is a result of operations on simple geometric objects [9]. Let the objects G_1 and G_2 be defined as $f_1(\mathbf{X}) \geq 0$ and $f_2(\mathbf{X}) \geq 0$. The binary operation of the objects G_1 and G_2 means operation $G_3=\Phi_i(G_1, G_2)$ with the definition

(3) $$f_3=\psi(f_1(x,y,z), f_2(x,y,z))\geq 0,$$

where ψ is the continuous real function of two variables.

Fast ray casting

In the algorithms of rendering by recursive subdivision of an arbitrary-dimension space, the most important question is about intersection of an object defined as $F(\mathbf{X}) \geq 0$, with a division cell (a square, a bar neighborhood $B(P, \delta)=\{X_i|X_i-P|\leq\delta\}$).

It is clear that the exact solution of this problem (the system of inequalities) is possible only for rather simple functions $f(x, y, z)$. Even for a function expandable into polynomials in $B(P, \delta)$ to the *n*-th degree the exact solution is unsuitable because it will involve roots to the *n*-th degree.

However, the exact solution for our application. It is necessary is not needed and sufficient to answer the question whether there are contour points in the vicinity because the complete overlapping or the absence of object intersection by a cell is solved by simple inequalities. It is noteworthy that the approximate solution should take into account the case of potential intersection and be asymptotically accurate.

Thus, we will give another formulation of the problem as a definition of the set of points belonging to $B(0, \delta)$ and at the same time satisfying the equation $f(x, y, z) = 0$.

The set of points $B(P, \delta) = \{X{:}\mathrm{dist}(X, P) \leq \delta\}$ is called a δ-neighborhood of the point P in the metric space. Further considerations are concerned with the case of 3-D space because this is precisely the case of our interest, and the Manhattan metric $\mathrm{dist}(X, Y) = \max\{|X|, |Y|\}$, although these results are valid also for an arbitrary dimension and metric.

Let $f(x,y,z)$ be an analytical function in a three-dimensional space. Then in $B((0,0,0), \delta)$, it can be expanded into the Taylor series. Discarding terms with degree higher than some d, yields:

(4) $$f(x,y,z)=\sum_{0\leq i;i+j+k\leq d} f_{ijk}x^i y^j z^k .$$

Write expression (4) in the form:

(5) $$f(x,y,z)=\sum_{h=0}^{d} f^h(x,y,z)$$

where,

(6) $$f^h(x,y,z)=\sum_{h=i+j+k} f_{ijk}x^i y^j z^k$$

Here we assume that the following condition is fulfilled:

(7) $$|f(0,0,0)|>0,$$

because in the other side the contour involves the origin of coordinates, i.e., the trivial case.

Let us consider the inequality of triangle:

(8) $$|f(x,y,z)|\geq |f^0|-\sum_{h=0}^{d}|f^h(x,y,z)|$$

and the function: $f^i(x, y, z)$

(9)

$$|f(x,y,z)|\leq \sum_{i+j+k=h}|f_{ijk}x^i y^j z^k|\leq$$

$$\leq\left\{\sum_{i+j+k}|f_{ijk}|\right\}\left\{\max|x^i y^j z|\right\}\leq F_h\varepsilon^h$$

where:

(10) $$\varepsilon=dist((x,y,z);(0,0,0))=max\{|x|,|y|,|z|\} .$$

Thus, writing the test in the practically suitable form we have the following inequality (if it is true, then $f(x, y, z)$ has zeros in $B((0,0,0), \delta)$):

(11) $$F(\delta)=F_0-\sum_{h=1}^{d}F_h\delta^h\geq 0 \Leftrightarrow$$

$$|f_0|-\sum_{h=1}^{d}(|f_{0,0,h}|+|f_{0,1,h-1}|+\ldots+|f_{h,0,0}|)\delta^h\geq 0$$

Since the vicinity is usually a unit vicinity, $B((0,0,0), \delta=1)$, the formula is reduced to comparison of the modulus of the free term of the equation of the figure (a line or a surface) with the sum of modules of the rest of coefficients.

The main point at this stage is the efficient finding of the first intersection of the ray with the surface. This task is similar to visualization in volumetric tomography, where the density function is defined in the form of discrete data. In our case, we use an analytically defined density function, which allows a more efficient search for points on the surface. It is proposed to calculate the intersection of rays with the surfaces of three-dimensional objects using a method which does not have the above disadvantages characteristic of the previously discussed well-known methods of analytical definition of surfaces. For ease of understanding, we assume that the scene is in a unit three-dimensional cube. Perspective is not analyzed due to the fact that it reduces to transformation to another coordinate system. Therefore, we omit the initial transformations and pay more attention to the main part of the method. We assume that the observer looks along the Z axis (Fig. 1).

It is necessary to get the projection of the scene on the plane XY. The projection must represent a finite set of values. Rays pass through the plane of the cube XY, and each of them corresponds to a pixel on the image. The rays are limited by the front and rear faces of the cube. In the search for the points of intersection of the ray and the surface, each ray is divided along the Z axis to form a set of voxels. Thus, we obtain a density function along the ray, which depends on one variable. The task is to find the first point at which the function vanishes. Having determined this point for each ray, we can calculate the coordinate Z. Next, a normal is defined at each pixel. In the presence of all the coordinates and normal at each pixel, the local illumination model is used. The result is an image of a smooth object neglecting illumination.

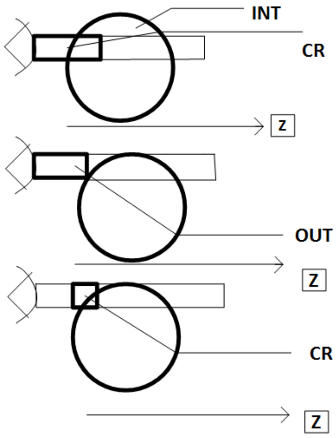

Fig.1. Ray casting

Implementation and performance

The visualization time is reduced by using the computational resources of a graphics processing unit with compute unified device architecture applied (CUDA) (NVIDIA). The CUDA system is a parallel programming model that allows implementing programs in C on a standard graphics processing unit. 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 number of computer processors allow parallel check of the intersection of several rays with the object simultaneously. Most of the graphics processing units that support CUDA have not less than 128 scalar cores. Therefore, a fairly large portion of the cube will be computed in parallel. The architecture of graphics processing units is based on many streaming multiprocessors with shared memory access for reading and writing. Each of the streaming processors contains eight scalar cores and a set of on-chip memory of four types. The number of registers may be 8192 or 16384, depending on the computational capabilities of the processing unit. The shared memory is 16 KB for each multiprocessor.

The constant memory cache (8 KB for each multiprocessor) and the texture memory cache (6 to 8 KB for each multiprocessor) were used only for reading. In the implementation of the proposed method, the effect of the speed of processors with memory on the performance was taken into account. The registers and shared memory were used to the maximum extent. In all other cases, the total memory of the graphic processing unit was used.

Among the functions of the graphics processing unit was to calculate the coordinates of points of the surfaces, normal, and illumination. Geometric transformations were performed by the central processing unit (CPU). The DirectX application-programming interface was used for visualization. Testing was performed on Intel Core2 CPU E8400 3.0 GHz, GPU and 9800 GT 470 GTX processors. Figures 2, 3 and 4 show the comparative results of testing of the dependence of the per-frame computation time on the number of defined perturbation functions for a particular test.

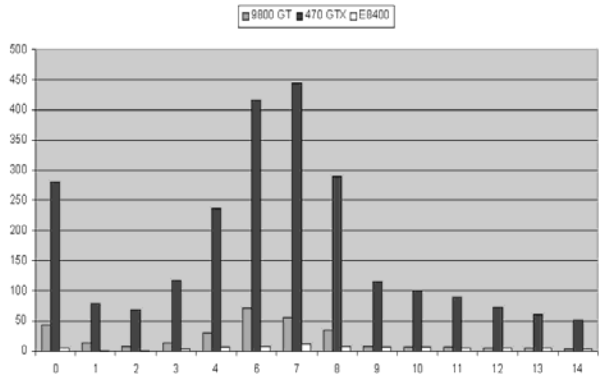

Fig.2. Timing diagram: the number of frames per second for different tests

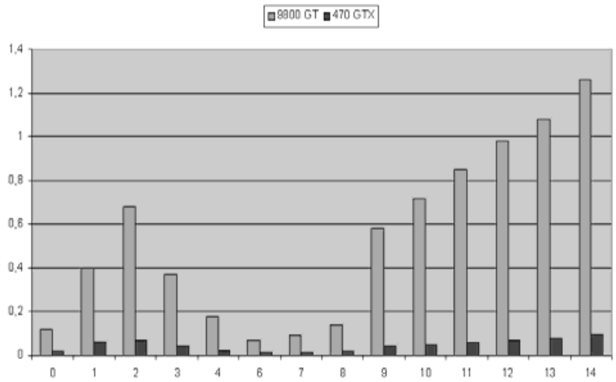

Fig.3. Timing diagram: the average time frame for different tests

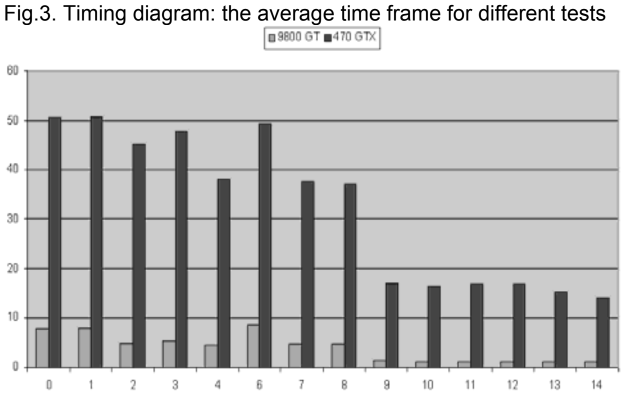

Fig.4. Timing diagram: the acceleration relative to the E8400 for different tests

Conlusions

The method of defining three-dimensional objects and the visualization method proposed in this paper have advantages over the existing approaches. 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; ease of animation and surface deformation.

Functional definition of objects is especially important in a number of computer graphics problems: in modeling soft or organic objects, 3D morphing, detection of collision of objects and constructive block geometry. The areas of application of functionally defined objects are molecular biology, interactive graphic visualization systems, CAD systems, 3D simulation systems, 3D web visualization, prototyping system, etc.

***Authors**: prof. Sergey I. Vyatkin, Institute of Automation and Electrometry, Siberian Branch of the Russian Academy of Sciences, Academician Koptug. ave. 1, 630090, Novosibirsk, Russia, E-mail: sergej.vyatkin.60@mail.ru; prof. Alexander N, Romanyuk, E-mail: romanyuk@vntu.edu.ua, prof. Sergii V.Pavlov, E-mail: psv@vntu.edu.ua, Ph.D. Maryna V. Moskovko, Vinnytsia National Technical University; Khmelnytske Shose 95, 21021 Vinnytsia, Ukraine; M.Sc. Nursanat Askarova, M.Sc. Azhar Sagymbekova,Kazakh National Research Technical University after K.I. Satpayeva, 22 Satbaev Street, 050013, Almaty City, Kazakhstan; prof. dr hab. inż. Waldemar Wójcik, E-mail: waldemar.wojcik@pollub.pl, dr hab. inż. Andrzej Kotyra, E-mail: a.kotyra@pollub.pl, Lublin University of Technology, Institute of Electronics and Information Technology, Nadbystrzycka 38a, 20-618 Lublin, Poland.*

REFERENCES

[1] Tuy H., and Tuy L., Direct 2-D Display of 3-D Objects, *IEEE Computer Graphics and Applications* 4(10) (1984), 29-33
[2] Perlin K., and Hoffert E. M., Hypertexture, *Computer Graphics*, 23 (Is. 3) (1989), 253-262
[3] Nowosielski L., Wnuk M., Siłaczuk M., The ray-tracing method for electromagnetic wave propagation modelling *Przegląd Elektrotechniczny*, R.90(7) (2014), 218-221
[3] Karla D., and Barr A. H., Guaranteed Ray Intersections with Implicit Surfaces, *Computer Graphics*, 23(3) (1989), 297-306.
[4] Hart J. C., Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces, *The Visual Computer*, 12(10) (1994), 527-545
[5] Mitchel D., Robust Ray Intersection with Interval Arithmetic, *Proc. on Graphics Interface*, Canadian Information Processing Society, Toronto (1990) 68-74
[6] Sherstuyk A., Fast Ray Tracing of Implicit Surfaces, *Computer Graphics Forum*, 18 (2) (1999), 139-147
[7] Reimers M., and Seland J., Ray Casting Algebraic Surfaces using the Frustum Form, *Computer Graphics Forum*, 27(2) (2008), 361-370
[8] Liktor G., Ray Tracing Implicit Surfaces on the GPU, *Comput. Graphics and Geometry* 10(3), 2008, pp. 36-53.
[9] Vyatkin S. I., Complex Surface Modeling Using Perturbation Functions, *Optoelectronics, Instrumentation and Data Processing*, 43(3), (2007), 226–231
[10] Sawicki D., Using the GPU to determining the area the flame in the vision diagnostic system, *IAPGOŚ*, 1 (2015), 80-85
[11] Romanyuk S.O, Pavlov S.V, Melnyk O.V, New method to control color intensity for antialiasing, *Proceedings on Control and Communications (SIBCON), Omsk,* (2015), *1-4*