DOI:10.15276/aait.04.2019.2
UDC 004.925.8

BLENDING FUNCTIONALLY DEFINED SURFACES

Olexandr N. Romanyuk[1)]
ORCID: 0000-0002-2245-3364, rom8591@gmail.com
Sergey I. Vyatkin[2)]
ORCID: 00000002-1591-3588, sivser@mail.ru
Svitlana G. Antoshchuk[3)]
ORCID: 0000-0002-9346-145X, asgonpu@gmail.com
Pavlo I. Mykhaylov[4)]
ORCID: 0000-0001-5861-5970, pm@3dgeneration.com
Roman Y. Chekhmestruk[5)]
ORCID: 0000-0002-5362-8796, Rc.ua@3dgeneration.com
[1)] Vinnytsyia National Technical University, Khmelnytsky Hwy, 95, Vinnitsa, 21021, Ukraine
[2)] Institute of Automation and Electrometric, SB RAS, Ak. Koptyuga, 1, Novosibirsk, 630090, Russia
[3)] Odessa National Polytechnic University, Shevchenko Avenue, 1, Odessa, 65044, Ukraine
[4)] CEO 3D GNERATION GmbH, Viktoriastraße, 15, Dortmund, 44137, Germany
[5)] 3D GENERATION UA, Pirogova Str., 37, Vinnitsa, 21021,Ukraine

ANNOTATION

Smooth surfaces with perturbation functions for the creation of complex shapes are considered. The method for describing objects in three-dimensional scenes with a base surface and perturbation functions that have a compact description is proposed. One of the positive properties of functionally defined objects in comparison with other methods of specifying models is the simplicity and efficiency of their geometric transformations, in particular, three-dimensional morphing and collision detection of objects. The most common model for visualizing three – dimensional images is the polygonal approximation. Along with many advantages, this model has its drawbacks. By modeling real objects, an approximate polygonal model is constructed. To increase the image quality, it is often necessary to increase the number of polygons. An increase in the number of polygons results in an increase in rendering time and memory usage. Changing the scale of an object introduces additional problems because you cannot change quickly and efficiently the number of polygons for the object model. You can get rid of such shortcomings by applying analytical volume assignment and rasterization using ray-tracing algorithms. Analytical volume assignment does not require a large amount of memory. The problem of synthesis of realistic images is relevant for various simulators, virtual studios and three-dimensional games. Now, there are already works on visualization of functionally defined surfaces, but their application is limited to a rather narrow class of surfaces and slow visualization. The algorithms used are difficult to optimize, which also imposes restrictions on practical application. The paper proposes to use a special class of volumes, which are called "free forms". Each free form represents a base surface and a perturbation on that surface. The base surface and perturbation are given by polynomials of the second degree-quadrics. To achieve smoothness, the perturbation function is raised to the third degree. The aim of the work is to create an application that, according to a given analytical task, calculates the frame depth and surface normal in each pixel with the help of perturbation quadrics. This application should use the computing resources of the graphics processing units as much as possible. There have been attempts to create algorithms to visualize volumes given analytically, but most of them used only the CPU for calculations, and the processing time was too long for practical application. Moreover, these algorithms were not designed for parallel processing. In contrast, the proposed algorithm uses a graphics-processing unit for most of the calculations. In this case, the calculations on the graphics accelerator occur in parallel, and the method effectively uses this feature. Due to parallel processing and the absence of the need to transfer a large amount of data from the shared memory to the memory of the graphics accelerator, the speed of visualization increases compared to the option that uses only the CPU. The clock speed of processors in graphics accelerators is less than the CPU frequency. However, for a certain class of tasks performance using graphics accelerators will be better, due to the large number of processors.

Keywords: Functionally Defined Surfaces; Perturbation Functions; Blending Operation

***For citation*: Olexandr N. Romanyuk, Sergey I. Vyatkin, Svitlana G. Antoshchuk, Pavlo I. Mykhaylov, Roman Y. Chekhmestruk Blending Functionally Defined Surfaces. *Applied Aspects of Information Technology.* 2019; Vol. 2 No.4: 271–282. DOI:10.15276/aait.04.2019.2**

1. INTRODUCTION

When scanning two-dimensional space it is impossible to obtain a full three-dimensional image. The information that is provided to the user in such technology is incomplete. The main thing is the lack of information about the depth of the object, meaning not the lack of Z – coordinates of the surface point, and the lack of information about the beam passing through the object. In addition, the visualization of scientific data is very relevant geometric transformations – unary, binary and more complex geometric operations, which are most optimally solved in the framework of solid modeling and structural geometry for functional models. For example, set-theoretic operations that can be used to construct objects and their compositions of unlimited complexity. This is achieved primarily by the use of Boolean operations of union and intersection (disjunction and conjunction). The process of modeling molecular structures becomes much more efficient when using functionally defined primitives and volume-oriented visualization. An alternative visualization technology based on the ray-casting method and the functional definition of primitives is proposed, in contrast to the traditional three-dimensional scene display technology based on the method of rendering triangles using the z-buffer algorithm. The fundamental difference of the proposed method from the others is non-polygonal representation of the free form surfaces of the scene using the algorithm of ray rasterization stage. The rendering mechanism differs from those implemented in modern computer graphics accelerators. Objects are represented functionally, which ensures the compactness of the database. The use of the ray-casting algorithm solves the problems typical for systems with polygon-based rasterization and "Big Data problems". To get a full three-dimensional scene of molecular structures, it is necessary to have a huge number of triangles (hundreds of millions). To describe the structures it is necessary to have a large number of triangles (Fig. 1). Since there are too many triangles in the scene, if you draw the whole scene, the accelerator will not display it in real time. Two-dimensional images are good for showing atom-to-atom bonds, but this is not enough when it is necessary to describe molecular structures. For example, molecular designers need to know how one functional group enters another structure, i.e. a three-dimensional picture is needed, with the ability to rotate the scene, move, etc. In addition, three-dimensional models are needed to show inter-molecular interactions in the simulation of some complex molecular movements. Hypothetically, there are 10^{180} molecular structures (the whole universe), only 10^{6} are known, and only 10^{3} are commercially used. It is also necessary to store a huge amount of information about molecular structures, it is necessary to compress. When calculating the forecast of new structures it is necessary to solve differential equations. The problems of molecular mechanics lack the computing power of the most modern supercomputers.

The most common model for visualization of three – dimensional images is a polygonal approximation. Along with many advantages, this model has its drawbacks. When modeling real objects, an approximate polygonal model is constructed. Increasing the number of polygons leads to an increase in visualization time and memory usage. In addition, changing the scale of an object causes additional problems because you cannot quickly and efficiently change the number of polygons for an object model. Such drawbacks can be eliminated by applying an analytical setting of volumes and rasterizing them using ray tracing algorithms.

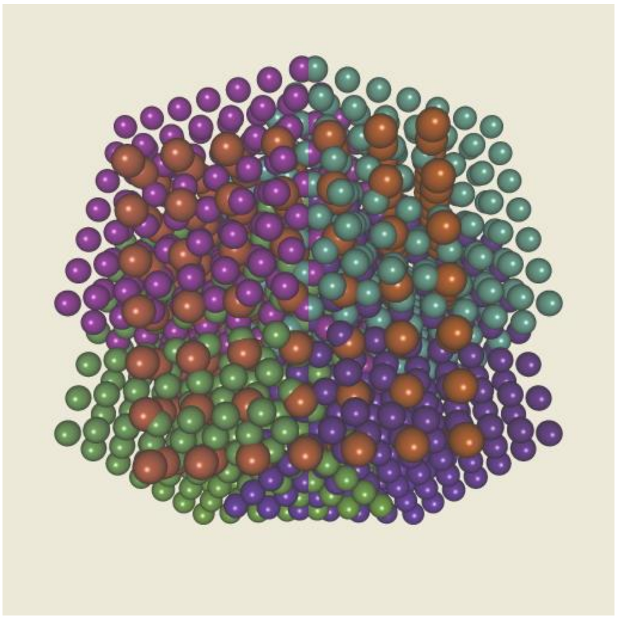


Fig. 1. **Molecular structures**
Source: **compiled by the authors**

The analytical description does not require a large amount of memory. The problem of synthesis of realistic images is relevant for various simulators, virtual studios and three-dimensional games. Now, there are already works on visualization of functionally defined surfaces, but their use is limited to a rather narrow class of surfaces and slow visualization. The algorithms used are difficult to optimize, which also imposes restrictions on practical application. An important application task can be the application of the proposed functional assignment of objects for web visualization, for example, a functionally based extension. Functionally defined objects can be used in conjunction with standard X3D and VRML models.

It is proposed to use a special class of volumes, which are called "free forms". Each free form represents a base surface and a perturbation on that surface. The base surface and the perturbation are given by polynomials of the second degree – quadrics. To achieve smoothness, the perturbation function is raised to the third degree.

A number of works [1-6] show functional ways of describing the visualized objects, allowing to significantly reducing the amount of databases for a certain class of objects in comparison with a polygon task. However, real - time display of objects specified in this way is associated with a significant increase in the required calculations due to the high order of describing functions.

Note the following basic functional ways of specifying primitives. Convolution surfaces [2], [3], [4] are an integral representation of implicitly defined surfaces, known in computer graphics as blobby models [5; 6], metaspheres [7], and soft objects [8]. These surfaces combine the flexibility of blobby models and the compactness of skeletal models [9]. At the same time, they have their drawbacks, which include the complexity of calculating surface points.

2. PROBLEM STATEMENT

When forming three-dimensional scenes, the most commonly used polygonal assignment of object models, which at the current level of computer graphics has a number of limitations.

Frame models of three-dimensional objects are approximate. Increasing the realism of the reproduction of graphic scenes provides for an increase in the level of detail for the correct approximation of the surfaces of real-world objects, and the growth rate of geometric complexity of three-dimensional images exceeds the growth rate of graphics performance. To achieve photorealism, it is necessary to use more than 1,000,000 polygons in the scene, and there is a tendency to further increase the metallization. Already in many applications, the number of scene triangles is comparable to or greater than the number of pixels occupied on the screen, which negates the advantages of the polygonal approach. If the objects are so detailed that the projection size of the graphic primitives is smaller than the pixel size of the screen, you can use points in complex primitives without losing the quality of the visualization. In this case, there is no need for a complex data structure, including polygonal grids, textures and texture coordinates. There is a question of expediency of visualization by polygons of models with a high degree of detail.

As the complexity of the scene increases, the efficiency of the polygon method decreases exponentially. Reducing the size of triangles leads to an increase in the amount of memory used and the excess of preparatory operations over pixel, which affects the time of formation of graphic scenes

Polygon models allow you to visualize only the outer layer of an object, while many applications require an internal structure.

The structure of polygon meshes is linear and they do not provide support for multiscale, so working with large meshes is difficult and requires computationally complex simplification methods. Dynamic control of detail consumes considerable computational resources, requires continuous recalculation not only of the coordinates of the vertices of triangles, but also of the illumination parameters. With frequent switching between levels, there is an effect of “undulation” of the surface, which is not peculiar to real objects.

When forming the contours of objects, which in General are curvilinear, the number of triangles in the approximating polygon must be comparable to the number of pixels forming the trajectory.

It is difficult to introduce attributive information into polygonal models, and the algorithms for visualization of topological operations are quite time-consuming. When approximating triangles, there are problems of high depth complexity, discarding invisible faces, determining and changing levels of detail, clipping triangles visibility pyramid, etc. the Amount of data in the visualization of three-dimensional objects with a complex surface, close to voxel models.

Additional problems are caused by changing the scale of the object, because it offers the possibility of changing the number of polygons for the object model and recalculation of image points. When an observer approaches an object, when changing the level of detail, new faces must be added to make the silhouette of the object look realistic.

When visualizing translucent structures with an internal density distribution, there is a problem of displaying inhomogeneous structures bounded by complex surfaces in the case of specifying them by polygons.

The polygonal model essentially does not allow receiving many visual effects necessary for realistic selection of a scene.

With skeletal animation, you cannot make high-quality animation of flexible materials, as well as perform complex morphing of the geometry of objects.

For polygon models, it is difficult to implement new effects on geometric objects, due to the introduction of operations on functions, which is necessary when modeling complex movements of bodies. It is difficult to implement both unary and binary geometric operations, as well as to determine the collision of objects.

The deformation of surfaces is also very expensive in the case of a polygon assignment, since each vertex of a triangular grid must be subjected to geometric calculations.

Today, realistic graphics require a compact description of the scene with an accurate description of the geometry of three-dimensional objects, providing an effective implementation of various geometric operations with models of objects to simulate the behavior of interacting bodies.

The free-form representation created by mean of the analytical perturbation functions have the following advantages: fewer surfaces for mapping curvilinear objects, short database description, fewer operation for geometric transformations and data transfer, simple animation and deformation of objects and surfaces.

Thus, the low realism of polygonal models and limited functionality for the formation of graphic scenes cause a scientific and applied problem, to solve which it is necessary to develop theoretical foundations for modeling and visualization of three-dimensional objects of the virtual environment, allowing eliminating the shortcomings characteristic of the polygonal task of models.

The relevance of the research topic is determined by the fact that this area is currently in a state of transition to qualitatively new results.

In connection with intensive development of methods and means of computing technologies, the solution of these questions is constantly new and actual task.

3. BLENDING

The operation joining several surfaces in a complex object with a smooth surface is called blending. The main difficulties and requirements to blending are: tangency of a blend surface with the base surfaces; easy intuitive control of the blending surface shape; necessity to perform for blended objects all the computations possible for unblended objects including set-theoretic operations; blend interference or ability to blend on blends and as the particular case complex vertices or corners blending; at least C^1 continuous blending function in the entire domain of definition; blending definition of basic set-theoretic operations, such as intersection, union and subtraction; single edge blending or localizing the blend to a region about intersection curve of two faces; added and subtracted materials blends; the ability to produce constant-radius blending; no restriction of circular cross sections or the requirement of variable-radius blends; exact representation for blends instead of many approximation; automatic clipping of unwanted parts of the blending surface; blending of two non-intersecting surfaces; functional constraints; aesthetic blends constrained by appearance.

An example of 2D method is

Let A=0 represents a line of the form:

$$x_A + y_A + c_A = 0,$$

starting from four line equations *A*=0, *B*=0, *P*=0, *Q*=0, we can combine them into a curve as follows:

$$(1-u)AB + uPQ = 0,$$

if P=0 and Q=0 come together, the equation is

$$(1-u)AB + uP^2 = 0,$$

$$u = \frac{P}{P+Q}.$$

Examples of 3D methods are bellow.

A solid is defined as $f(P) \le 1$.

Intersection is

$$I(f_1, f_2 ..., f_n) = (f_1^p + f_2^p + ... + f_n^p)^{1/p},$$

Union is

$$U(f_1, f_2 ..., f_n) = (f_1^{-p} + f_2^{-p} + ... + f_n^{-p})^{-1/p}$$

p is a positive real number.

$$\lim_{p\to\infty} I(f_1, f_2 ..., f_n) = \min(f_1, f_2, ... f_n)$$

$$\lim_{p\to\infty} U(f_1, f_2 ..., f_n) = \max(f_1, f_2, ... f_n)$$

The blend surface is defined as [10]:

$$1 - \left(1 - \frac{A}{r_A}\right)^w - \left(1 - \frac{B}{r_B}\right)^w = 0 \qquad (1)$$

where: r_A and r_B determine the range of the blend: when $r_A = A$, the blend surface becomes B=0.

The exponent w also known “thumb weight” and controls the nearness of the blend to the surfaces. Larger values of w give a super elliptic cross-section, which approximates sharp intersection more and more.

An example is

Substitute (A+M-r) for P in the Liming formula:

$$(1-u)A^{э}B^{э} + u(A^{'} + B^{'} - r)^2 = 0 \qquad (2)$$

where: $A^{ý} = c_A A, \quad B^{ý} = c_B B$.

Constant-radius blends can be implemented with the composition of constant-radius offsets.

This article presents the results of some studies on modeling and visualization of three-dimensional scenes. It is proposed to use the task of objects in free forms in the form of real functions.

4. FUNCTIONALLY DEFINED OBJECTS BASED ON PERTURBATION FUNCTIONS

Functionally defined surfaces are constructed from second-order surfaces with analytical perturbation functions, thereby achieving a high coefficient of geometric compression of highly realistic three-dimensional objects. It is proposed to describe geometric objects (free forms), setting the function of deviation (second order) from the base surface of the second order (quadrics). The function is given by an algebraic inequality of the second degree with three unknowns x, y, z as F(x, y, z) ≥ 0. We consider surfaces as closed subsets of the Euclidean space defined by the describing function F(x, y, z) ≥ 0. Where F is an analytical function, (x, y, z) is the point specified by the coordinate variables in Euclidean space. Here F(x, y, z) > 0 specifies the points inside the surface, F(x, y, z) = 0 are the points on the boundary, and F(x, y, z) < 0 are the points outside and not belonging to the surface.

Function of the perturbation in implicit form. It is proposed to describe complex geometric objects, setting the function of deviation (second order) from the base quadric as:

$$Q(P[x,y,z]) = q_{xx}x^2 + q_{yy}y^2 + q_{zz}z^2 + q_{xy}xy + \\ + q_{xz}xz + q_{yz}yz + q_x x + q_y y + q_z z + q \quad . \tag{3}$$

Free forms are built based on quadrics. The free form is obtained by deformation of the base quadric by means of perturbation functions [11-14]

$$F'(x,y,z) = F(x,y,z) + \sum_{1}^{N} R_i(x,y,z), \tag{4}$$

where: R (x, y, z) is the perturbation, if is the form factor

$$R_i(x,y,z) = \begin{cases} Q_i^3(x,y,z), if\ Q_i(x,y,z) \geq 0 \\ 0, if\ Q_i(x,y,z) < 0 \end{cases} \tag{5}$$

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

The resulting surface will be smooth, and a small number of perturbation functions are required to create complex surface shapes. The degree of perturbation can be chosen arbitrarily. This will determine the smoothness of the transitions from the base surface to the perturbation region, and, accordingly, the size of the perturbation itself. In order for the surface to be smooth, the degree must be greater than two must. This condition guarantees continuity of the function and its derivative. Thus, the problem of object construction is reduced to the problem of deformation of the base surface in the desired way, and not to the approximation of its primitives.

Consider a simple object with one base quadric (ellipsoid) and two perturbation functions (positive and negative form factors) (Fig. 2).

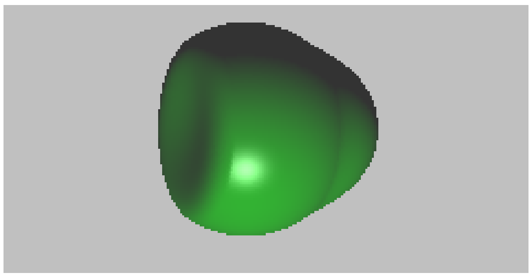


Fig. 2. **Perturbed quadric**
Source: **compiled by the authors**

As already mentioned above, in order for the surface to be smooth enough that the degree of perturbation was more than two. This ensures that the function itself and its derivative are continuous. However, the second derivative will be represented as a function with discontinuities. In addition, these gaps will be on the border of perturbations. It should be clarified how will be affected by these breaks over the image. In the case where the degree is less than three, you can see (Fig. 3; Fig. 4 and Fig. 5) that the normal function is continuous along each of the coordinates by virtue of the continuity of the derivatives. However, unfortunately, the derivative of this function has discontinuities, since it appears as the second derivatives of the density function. Given that a local lighting model that depends on the normal is used, the perturbation boundaries will be highly noticeable. Despite the fact that the surface itself has a smooth appearance, visually it will be perceived as a union or intersection of surfaces. Such surfaces are not of great interest, because visually close to this result could be obtained by set-theoretic operations on the base volumes in the form of quadrics. A visualization of the volumes specified by this method is less time-consuming task. Moreover, most importantly, such volumes are not completely smooth. Therefore, the third degree of perturbation functions is used, which ensures the smoothness not only of the surface itself, but also of the normal (Fig. 6 and Fig. 7). In this case, transitions to the perturbation region are not visually distinguished. Such objects look more realistic.

This effect is especially noticeable when the glare hits the perturbation boundary. In objects with the second degree of perturbation, the glare is broken (Fig. 3; Fig. 4 and Fig. 5). Highlights without breaks are shown in Figures 6 and 7.

Objects and operations can be of the following types: Functionally Based Primitive Object, Functionally Based Complex Objects, and Functionally Based Objects Operation.

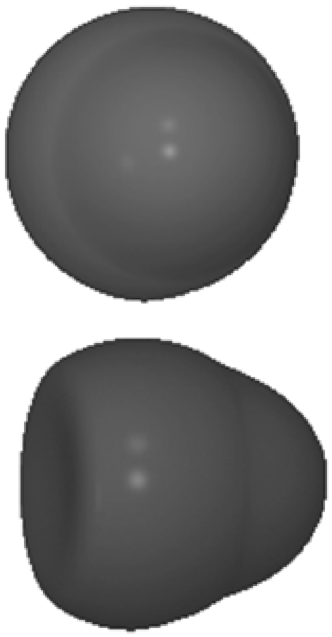


Fig. 3. **The degree of perturbation is two**
***Source:* compiled by the authors**

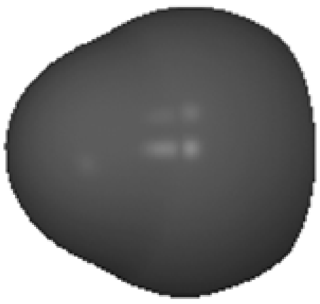


Fig. 4. **The degree of perturbation is two**
***Source:* compiled by the authors**

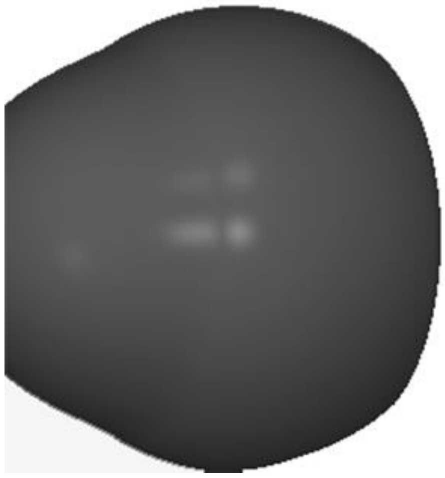


Fig. 5. **The degree of perturbation is two**
***Source:* compiled by the authors**

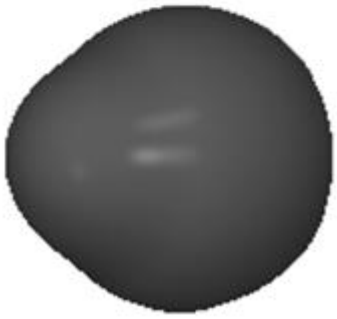


Fig. 6. **The degree of perturbation is three**
***Source:* compiled by the authors**

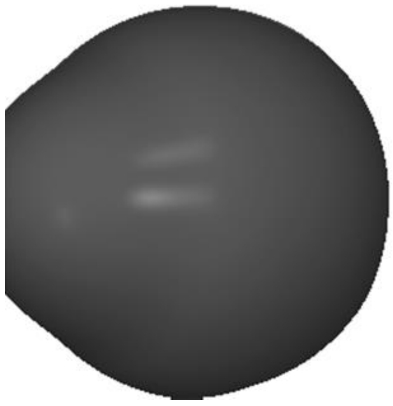


Fig. 7. **The degree of perturbation is three**
***Source:* compiled by the authors**

In equation (4), 1 and N are the lower and upper limits of summation. The lower limit is set equal to unity (one perturbation function), and the upper limit corresponds to several perturbation functions (the value depends on the form complexity).

The resultant surface is smooth and a few perturbation functions are needed for generating complex surface forms if the volume of data decreases by more than a factor of 100 and more as compared to the polygonal description.

Because functional objects have a more compact description compared to the traditional description of scenes in the form of triangles, each displayed object can store its properties without significantly increasing the cost of storage in the system memory. Therefore, with a small amount of memory, you can add to objects almost any number of parameters for the light model that can significantly increase the quality and complexity of the displayed scene. To be able to superimpose texture on the surface and volume of the displayed object, the array objects were supplemented with the property of converting coordinates from model space M to texture space T.

When the array object is asked for the value of a property, it converts the coordinates of the current voxel from M to T: (X, Y, Z) $\rightarrow$ (U, V, W) and uses these coordinates as an address in the array. If, for example, the array is two-dimensional, then one coordinate is simply not used. For complex functional surfaces (with different kinds of concavities and convexities), the two-dimensional texture mapping method described above may not give positive results: there will be unacceptable distortions. Modeling and animating textured objects is not straightforward using implicit surfaces. You can define a three-dimensional texture in a global image frame. However, the surface will simply move into texture space when the object deforms, resulting in strange visual effects. In the soft geometry design approach, different 3D textures can be attached to each implicit primitive. In addition, mixed at surface points where multiple primitives have a non-zero contribution. However, animating the surface will result in visual artifacts, such as time-varying interferences between texture patterns. In any case, being able to display and coherently animate a 2D texture on an implicit surface that deforms over time would be much more likely than 3D textures to simulate a kind of shell covering an animated object. This is one of the most difficult problems in modeling implicit surfaces, because implicit surfaces do not provide a parameterization on which to attach texture coordinate. However, determining the initial texture mapping is not a big problem, if you calculate sample points, you can attach the initial mapping to these points. This can be done using any standard approach that works regardless of the topological surface type. For example, to define the initial texture map, we use a mesh of triangular patches. These selection points of the object repeat all transformations of this object, they are used only for texturing, and they are not displayed. A much more complex problem in computer animation is the way the texture is animated when the surface of an object deforms over time (these deformations can even include splitting and merging!). Quadrics with perturbations can crumble into fragments and reassemble during animation, as it happens, for example, with mercury droplets. The above approaches no longer work for complex surface animations either. If you use local primitive-based polygonization, where the texture coordinates can be directly related to the sample points defined in each local primitive frame. However, this approach, which works well enough for uniform color, will give pretty poor results when using arbitrary texture patterns: parts of the texture will experience stiff movement during smooth deformation of the implicit surface, while other parts corresponding to the blending areas will simply appear or disappear during movement. Therefore, for such tasks we use the following approach: An approach based on a three-dimensional vector field is used to animate 2D textures on an implicit surface that deforms over time [15]. Sample points of fixed texture coordinates are animated using different vector fields. Vector fields are provided that implement various behaviors such as “sticking“, “twisting“, “elasticity”, “jitter” of textures, etc. Most of these fields are related to the individual movement of the underlying primitives, since the implementation was performed in a constructive implicit surface representation.

5. RENDERING

The algorithm for visualization of objects given by quadrics and its implementation are considered in the work [16]. For ease of understanding, we assume that the scene is in a single three-dimensional cube. The perspective will not be considered, because it is reduced to the transition to another coordinate system. Since the visibility pyramid will be a cube in the new coordinate system. Therefore, let's omit the initial transformations and pay more attention to the main part of the algorithm. We assume that the observer is looking along the z-axis. It is necessary to obtain a projection of the scene on the XY plane. The projection must be a finite set of values. Therefore, the entire cube will be divided into "bars" so that each bar corresponds to a pixel in the image. Each of the bars will divide along the Z-axis, forming a set of vowels. Since the dimensions of the bar in the XY plane are much smaller than in the direction of the z-axis, the bar can be considered as a beam. Thus, we obtain a density function along the beam, which depends on one variable. The task will be to find the first point at which the function turns to zero. Finding such a point for each beam, the depth of the frame will be known. These approximations reduce the problem to the problem of ray tracking. Next, you can calculate the normal in each pixel. In addition, with depth and normal data in each pixel, you can use a local lighting model. The result is an image of a smooth object, taking into account the lighting. The main part of the work is to effectively find the first intersection of the beam with the surface.

This task resembles the tasks of visualization of volume data, which is often used, for example, in tomography. In such problems, the density function is given. The main difference is that in such problems we are dealing with discretized data. In addition, in our case there is an analytically given density function. This allows you to search more efficiently. Another significant difference is that in the case of analytically given density functions, it is possible to morph surfaces.

The main task is to adapt and implement the visualization algorithm so that it effectively uses the computational resources of graphics accelerators. First, consider the basic idea of the initial algorithm. During the search, the cube is divided into smaller parts, for which the intersection test with the volume is checked. The process of division can be divided into two parts. First, the cube is divided into four parts in the XY plane. Further, each part is considered separately. If there is no intersection with the specified volume, then this part is excluded from further consideration, and the color of all pixels corresponding to this part of the cube takes the background value. In addition, with those parts with which there can be an intersection, a similar division procedure is carried out. In General, this process ends when only one pixel corresponds to the part in question. Now every part with which there can be an intersection can be considered as a ray directed along the z-axis. The second step considers the rays corresponding to the obtained parts, and a binary search is performed to find the nearest intersection point with the volume. The advantage of this approach is that it is possible to discard at an early stage large parts of the cube that do not have a given volume.

The main difference of the adapted algorithm is that the first part is omitted. Therefore, the cube is divided into parts in the XY plane according to the pixels in the image. In the second part, the algorithm remains the same. In this case, the reduction of time for visualization is achieved due to the effective use of computing resources of the graphics accelerator. In order to justify the effectiveness of this approach, it is necessary to consider the programming model that was used. NVIDIA's Compute Unified Device Architecture (CUDA) was used for implementation. Testing was carried out on a computer with an Intel Core 2 CPU E8400 3.0 GHz, and a GeForce 8800 GTX graphics accelerator. Performance was tested on fourteen different scenes from free forms given by quadrics.

Five scenes contained more than one free form. A depth buffer is used to render such scenes. In addition, for each object, a full pass through the frame is made. This approach allows the second version to effectively use multiple graphics accelerators at the same time. Using the depth buffer, you can combine free forms not only with each other, but also with the scene specified by the polygons.

The rendering time depends on the complexity of the object. More perturbations are required to specify objects that are more complex. As mentioned, speed is highly dependent on memory speed. Therefore, the best performance can be achieved by using only registers.

6. TEST RESULT

Reducing the time for visualization is achieved through the efficient use of computing resources graphics accelerator architecture CUDA (Compute Unified Device Architecture) from NVIDIA. The implementation took into account the impact of memory speed. Registers and shared memory are maximized. In all other cases, the shared memory of the graphics accelerator is used.

The functions of the graphics accelerator included the calculation of coordinates of points of surfaces, normals and lighting. Geometric transformations were performed by the Central processor (CPU or CPU), as well as rasterization of geometric primitives in the tile grid and the formation of a list of fragments with the calculation of all necessary parameters. The DirectX application-programming interface was used for visualization. Testing was performed on Intel Core2 CPU E8400 3.0 GHz and GPU 470 GTX. Figure 8 shows the results of testing the dependence of the frame calculation time on the number of specified perturbation functions for a particular test of the two methods. On the abscissa axis-test numbers, on the ordinate axis-average time per frame (in seconds). The diagram shows that on average, the calculation time decreased by an order of magnitude. At the same time, the transport delay doubled. Note that performance has increased not only for medium and small objects, but also for large objects with a large number of disturbances. As the object is split into tiles and the number of perturbations of the fragments decreases.

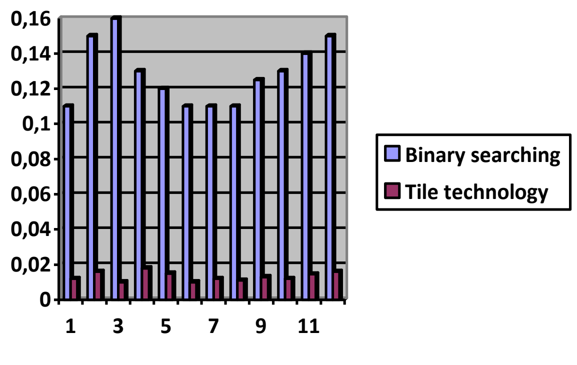


Fig. 8. Test chart: average time per frame for different tests

Source: **compiled by the authors**

One of the main drawbacks of known visualization methods is the complexity of calculating surface points. Therefore, the method of marching on the beam does not guarantee the detection of the surface, in addition, it is slow [17, 18]. In [19] we describe a method for calculating the intersection of a beam with a surface given in an implicit form, but the calculations of - and - parameters are very complex. In the sphere-tracing method, finding the greatest radius so that no point of the volume lies inside the sphere is a nontrivial task [20]. In ray tracing with interval analysis, many calculations are required for complex functions because they are needed individually for each ray and for each interval along that ray [21]. In fast ray tracing, the search for rays crossing surfaces is complex and not efficient enough, since the clustering methods of this method do not solve this problem completely [22].

In [23] a ray tracking method is described for visualization of surfaces defined algebraically by polynomials of high degree. However, it is difficult to model real objects using polynomials. It is also not guaranteed how closely the initial function will approximate the Bezier curve. Another disadvantage of this method is that translating an object to another coordinate system is not an easy task. Therefore, creating dynamic scenes is problematic.

There is another method of visualization using GPU analytically specified objects [24], based on the usual step-by-step tracking of rays. The difference is that the step size is not constant, but is selected at each step. At each step, there is a ball centered at the current point on the beam. The disadvantage is that finding a suitable radius is a non-trivial task. For static scenes, the authors of the algorithm used data preprocessing. Therefore, as in the previous method, the visualization of objects that change their shape and position over time requires significant computational costs.

7. CONCLUSIONS

The proposed method of describing objects of three-dimensional scenes with basic surfaces and perturbation functions has a compact description, which allows reducing from 10 to 1000 times the amount of data transmitted depending on specific three-dimensional scenes and models. Objects with planar faces are also easy to define, for example, a cube can be defined with three quadrics. In addition, when solving the describing function as an inequality, you can display the inside of an object.
By changing the perturbation functions, you can change the shape of the object. Interpolating the coefficients of the quadratic equation from the initial state to the selected final state, we obtain a smooth flow of the object from one state to another. The same manipulations can be done with the basic quadric. It is also possible to interpolate the perturbation factor. Interesting effects can be achieved by changing the position and scale of the perturbation quadric relative to the base. It is worth considering in more detail the case when it is necessary to move smoothly from one object to another. This effect can be achieved by constructing a new density function from the functions of these objects. This is essentially just an interpolation of these functions. Note that these objects can be different. That is to have different base quadrics, different sets of perturbation functions. For many objects, this effect is difficult to implement when using a polygon job model. However, many effects associated with the transformation of coordinates, which are implemented using the polygon model, can be similarly implemented on analytically specified objects.

REFERENCES

1. Arvid Perego. "Introduction to Algebraic Surfaces". 2009p. http://www-math.sp2mi.univ-poitiers.fr/~sarti/corso_Perego.pdf.

2. Pizaine Guillaume, et al. "Vessel Geometry Modeling and Segmentation using Convolution Surfaces and an Implicit Medial axis". *In: Biomedical Imaging: From Nano to Macro, IEEE International Symposium on. IEEE.* 2011. p. 1421–1424. ISSN 1945-7928. DOI: 10.1109/ISBI.2011.5872666.

3. Alexe, A., Barthe, L., Cani, M. P. & Gaildrat, V. "Shape Modeling by Sketching using Convolution Surfaces". *In ACM SIGGRAPH 2007courses* (39 p.). ACM. ISBN 978-1-4503-1823-5. (2007, August). DOI:10.1145/1281500.1281550.

4. Abbena, E., Salamon, S. & Gray, A. "Modern Differential Geometry of Curves and Surfaces with Mathematica". *CRC Press.* New York. 2016. 1016 p.

5. Pharr, M., Jakob, W. & Humphreys, G. "Physically Based Rendering: from Theory to Implementation". *Morgan Kaufmann Publishers.* San Francisco: 2004. 1168 p.

6. Rocchini, C., Cignoni, P., Ganovelli, F., Montani, C., Pingi, P. & Scopigno, R. "The Marching Intersections Algorithm for Merging Range Images", *Vis. Comput.* 2004; 20(2-3): 149-164. DOI: 10.1007/s00371-003-0237-8.

7. Gomes, A., Voiculescu, I., Jorge, J., Wyvill, B. & Galbraith, C. "Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms". *Springer*-Verlag. London: 2009. 351 p.

8. Ian Stephenson. "Production Rendering: Design and Implementation". *Springer* Science & Business Media. (1 December 2004). 302 p.

9. Raposo, A. & Gomes, A. (201923 WSCG 2019 27). *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision. Sun.* 16.

10. Jun 2019. Greg Turk & Brien, J. "Introduction to Implicit Surfaces". *ACM Transactions on Graphics*. October 2002; Vol 21 No 4: p. 2–12. DOI: 10.1145/571647.571650.

11. Bellet, T., Arnould A. & Le Gall, P. "Rule-based Transformations for Geometric Modeling". *Research Notes* 2011-1. XLIM-SIC. UMR CNRS 6172. University of Poitiers.

12. Akenine-Möller, T., Haines, E. & Hoffman, N. "Real-Time Rendering", *Peters/CRC Press.* 2018. 1045 p.

13. Vyatkin, S. I., Romanyuk, A. N. & Voit, B. L. "Perturbation Functions and Operations in Geometric Modeling". *Measuring and Computing Devices in Technological Processes*. 2017; No.3 (59): 117–120.

14. Cagniart, C., Boyer, E. & Ilic, S. "Probabilistic Deformable Surface Tracking from Multiple Videos". In *European Conference on Computer Vision.* 2010. DOI: 10.1007/978-3-642-15561-1_24.

15. Romanyuk, O. N, Vyatkin, S. I. & Antoshchuk, S. G. "3D Vector Fields Visualization Using Graphics Processing Units". Research and Modeling of Information Processes and Technologies. *Herald of Advanced Information Technology.* 2019; Vol. 2 No.3: 173–182. DOI: 10.15276/hait.03.2019.1.

16. Vyatkin, S., Romaniuk, O. & Dudnyk, O. "GPU-Based Rendering for Ray Casting of Multiple Geometric Data". *ACIT – Advanced Computer Information Technologies*. ACIT 2018. *International Conference, Ceske Budejovice, CZECH REPUBLIC*, 1-3 June 2018. http://acit.tneu.edu.ua.

17. Geng, J. "Three-dimensional Display Technologies". *Advances in Optics and Photonics.* 2013; 5(4): 456–535. DOI: 10.1364/AOP.5.000456.

18. Satherley, R. & Jones, M. "Hypertexturing Complex Volume Objects", *The Visual Compute*r. 2002; 18(4): 226–235. DOI: 10.1007/s003710100143.

19. Kutulakos, K. N. & Seitz, S. M. "A theory of Shape by Space Carving". *Int. J. Comput. Vis.* 2000; 38 (3): 199–218. DOI: 10.1023/A:1008191222954.

20. Adamson, A. & Alexa, M. "Ray Tracing Point set Surfaces". *In Proceedings of Shape Modeling International 2003.* May 2003. DOI: 10.1109/SMI.2003.1199627.

21. Balsys, R. & Suffern, K. "Visualisation of Implicit Surfaces". *Computer & Graphics*. 2001; 25: 89–107. DOI: 10.1016/S0097-8493(00)00110-2.

22. Singh, J. & Narayanan, P. "Real-Time Ray Tracing of Implicit Surfaces on the GPU". *IEEE Trans Vis Comput Graph.* 2010; 16(2): 281–272. DOI: 10.1109/TVCG.2009.41.

23. Reimers, M. & Seland, J. "Ray Casting Algebraic Surfaces using the Frustum Form". *Eurographics.* 2008; Vol. 27 No. 2: 361–370. DOI: 10.1111/j.1467-8659.2008.01133.x.

24. Liktor, G. "Ray Tracing Implicit Surfaces on the GPU", *Computer Graphics and Geometry*. 2008; Vol.10, No.3: 36–53.

Conflicts of Interest: the authors declare no conflict of interest

Received 12.09.2019
Received after revision 07.11.2019
Accepted 02.12.2019

DOI:10.15276/aait.04.2019.2
УДК 004.925.8

БЛЕНДІНГ ФУНКЦІОНАЛЬНО ЗАДАНИХ ПОВЕРХОНЬ

Олександр Никифорович Романюк[1)]
ORCID: 0000-0002-2245-3364 rom8591@gmail.com
Сергій Іванович Вяткін[2)]
ORCID: 0000-0002-1591-3588, sivser@mail.ru
Світлана Григорівна Антощук[3)]
ORCID: 0000-0002-9346-145X, asgonpu@gmail.com
Павло Ігорович Михайлов[4)]
ORCID: 0000-0001-5861-5970 pm@3dgeneration.com
Роман Юрійович Чехместрук[5)]
ORCID: 0000-0002-5362-8796, Rc.ua@3dgeneration.com
[1)] Вінницький національний технічний університет, Хмельницьке шосе. 95. Вінниця, 21021,Україна
[2)] Інститут автоматики та електрометрії. пр-т Коптюга 1. Новосибірск, 630090, Росія
[3)] Одеський національний політехнічний університет. пр. Шевченка, 1. Одеса, 65044,Україна
[4)] CEO 3D GNERATION GmbH, 15, Viktoriastraße. Дортмунд, 44137, Німеччина
[5)] 3D GENERATION UA. вул. Пирогова 37. Вінниця, 21021,Україна

АНОТАЦІЯ

Найбільш поширена модель для візуалізації тривимірних зображень - полігональне наближення. Поряд з рядом переваг, така модель має і свої недоліки. Моделюючи реальні об'єкти, будується наближена полігональна модель. Для п підвищення якості зображення найчастіше необхідно збільшувати кількість полігонів. Збільшення кількості полігонів передбачає збільшення часу візуалізації і обсягу використовуваної пам'яті. Додаткові проблеми вносить зміна масштабу об'єкта, тому що не можна швидко і ефективно змінити кількість полігонів для моделі об'єкта. Від таких недоліків можна позбутися, застосовуючи аналітичне п одання об'ємів і растеризація їх за допомогою алгоритмів трасування променів. Аналітичне завдання об'ємів не вимагає великого обсягу пам'яті. Проблема синтезу реалістичних зображень актуальна для: різних тренажерів, віртуальних студій і тривимірних ігор. На даний момент вже існують роботи по візуалізації функціонально заданих поверхонь, але їх застосування обмежене досить вузьким класом поверхонь і повільною візуалізацією. Використовувані алгоритми складно оптимізувати, що також накладає обмеження на практичне застосування. У роботі пропонується використовувати особливий клас об'ємів, які називаються «вільні форми». Кожна вільна форма є базова поверхня та збурення на цій поверхні. Базова поверхня і збурення задаються поліномами другого ступеня-квадрікою. Щоб досягти гладкості, функція збурення зводиться в третю ступінь. Метою роботи є розробка програми, яка за заданим аналітичним завданням з використанням квадріків зі збуреннями обчислює глибину кадру і нормалі до поверхні в кожному пікселі. Цей додаток має максимально можливо використовувати обчислювальні ресурси графічного акселератора. Вже були спроби створення алгоритмів візуалізації об'ємів, заданих аналітично, але більшість з них використовували тільки ЦПУ для обчислень, і час обробки було занадто великим для практичного застосування. Ці алгоритми не були призначені для паралельної обробки. На відміну від них, запропонований в роботі алгоритм використовує графічний акселератор для більшої частини обчислень. При цьому обчислення на графічному акселераторі відбуваються паралельно, і метод ефективно використовує цю особливість. За рахунок паралельної обробки і відсутності необхідності пересилання великої кількості даних із загальної пам'яті в пам'ять графічного акселератора, збільшується швидкість візуалізації порівняно з варіантом, що використовують тільки ЦПУ. Тактова частота процесорів в графічних акселераторах менше, ніж частота ЦПУ. Але для певного класу задач продуктивність з використанням графічних акселераторів буде вищою за рахунок великої кількості процесорів.

Ключові слова: функціонально задані поверхні; функції збурення; операція сгладження

DOI:10.15276/aait.04.2019.2
УДК 004.925.8

БЛЕНДИНГ ФУНКЦИОНАЛЬНО ЗАДАННЫХ ПОВЕРХНОСТЕЙ

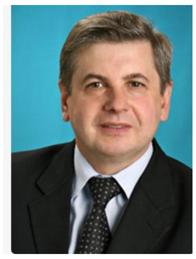
Александр Никифорович Романюк[1]
ORCID: 0000-0002-2245-3364, rom8591@gmail.com
Сергей Иванович Вяткин[2]
ORCID: 0000-0002-1591-3588, sivser@mail.ru
Светлана Григорьевна Антощук[3]
ORCID: 0000-0002-9346-145X, asgonpu@gmail.com
Павел Игоревич Михайлов[4]
ORCID: 0000-0001-5861-5970, pm@3dgeneration.com
Роман Юрьевич Чехместрук[5]
ORCID: 0000-0002-5362-8796, Rc.ua@3dgeneration.com
[1] Винницький национальный технический университет, Хмельницкое шоссе, 95. Винница, Украина
[2] Институт автоматики и электрометрии, пр-т Коптюг, 1. Новосибирск, 630090,Россия
[3] Одеський национальный политехнический университет, пр. Шевченко, 1. Одесса, 65044, Украина
[4] CEO 3D GNERATION GmbH, 15, Viktoriastraße, Дортмунд, 44137, Германия
[5] 3D GENERATION UA, ул. Пирогова, 37. Винница, 21021, Украина

АННОТАЦИЯ

Наиболее распространенная модель для визуализации трехмерных изображений–полигональное приближение. Наряду с множеством преимуществ, такая модель имеет и свои недостатки. Моделируя реальные объекты, строится приближенная полигональная модель. Для увеличения качества изображения чаще всего необходимо увеличивать количество полигонов. Увеличение количества полигонов, влечет за собой увеличение времени визуализации и объёма используемой памяти. Дополнительные проблемы вносит изменение масштаба объекта, потому что нельзя быстро и эффективно изменить количество полигонов для модели объекта. От таких недостатков можно избавиться, применяя аналитическое задание объёмов и растеризации их при помощи алгоритмов трассировки лучей. Аналитическое задание объёмов не требует большого объёма памяти. Проблема синтеза реалистичных изображений актуальна для: различных тренажеров, виртуальных студий и трехмерных игр. На данный момент уже существуют работы по визуализации функционально заданных поверхностей, но их применение ограничено довольно узким классом поверхностей и медленной визуализацией. Используемые алгоритмы сложно оптимизировать, что также накладывает ограничения на практическое применение. В работе предлагается использовать особый класс объёмов, которые называются «свободные формы». Каждая свободная форма представляют собой базовую поверхность и возмущение на этой поверхности. Базовая поверхность и возмущение задаются полиномами второй степени–квадриками. Чтобы добиться гладкости, функция возмущения возводится в третью степень. Целью работы является создание приложения, которое по заданному аналитическому заданию с помощью квадрик с возмущениями вычисляет глубину кадра и нормали к поверхности в каждом пикселе. Данное приложение должно максимально возможно использовать вычислительные ресурсы графического акселератора. Уже были попытки создания алгоритмов визуализации объёмов, заданных аналитически, но большинство из них использовали только ЦПУ для вычислений, и время обработки было слишком большим для практического применения. И эти алгоритмы не были предназначены для параллельной обработки. В отличие от них, предложенный в работе алгоритм использует графический акселератор для большей части вычислений. При этом вычисления на графическом акселераторе происходят параллельно, и метод эффективно использует эту особенность. За счёт параллельной обработки и отсутствия необходимости пересылки большого количества данных из общей памяти в память графического акселератора, увеличивается скорость визуализации по сравнению с вариантом, использующим только ЦПУ. Тактовая частота процессоров в графических акселераторах меньше, чем частота ЦПУ. Но для определенного класса задач производительность с использованием графических акселераторов будет лучше, за счёт большого числа процессоров.

Ключевые слова: функционально заданные поверхности; функции возмущения; операция сглаживания

ABOUT THE AUTHORS



Olexandr Nikiforovich Romanyuk, Dr. Sci. (Eng), Professor of Department of Software, Vinnytsyia National Technical University, Khmelnytsky Hwy, 95, Vinnitsa, 21021, Ukraine
ORCID: 0000-0002-2245-3364, rom8591@gmail.com,
Research field: formation and processing of graphic images

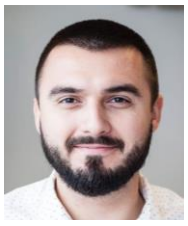
Романюк, Олександр Никифорович, доктор технічних наук, професор кафедри програмного забезпечення, Вінницький національний технічний університет, Хмельницьке шосе. 95. Вінниця, 21021,Україна

Sergey Ivanovich Vyatkin, PhD (Eng), senior scientific researcher of Synthesizing, Visualization Systems Laboratory at Institute of Automation and Electrometric, SB RAS, Ak. Koptyuga, 1, Novosibirsk, 630090, Russian Federation
ORCID: 0000-0002-1591-3588, sivser@mail.ru
Research field: Formation and Processing of Graphic Images

Вяткін, Сергій Іванович, кандидат технічних наук, с.н.с. лабораторії синтезуючих систем візуалізації, Інститут автоматики та електрометрії, пр-т Коптюга, 1, Новосибірск, 630090, Російська Федерація

Svitlana Grigoryevna Antoshchuk, Dr. Sci. (Eng), Professor, Institute of Computer System, Odessa National Polytechnic University, Shevchenko Avenue, 1, Odesa, 65044, Ukraine
ORCID: 0000-0002-9346-145X, asgonpu@gmail.com
Research field: Formation and Processing of Graphic Images

Світлана Григорівна Антощук, доктор технічних наук, директор інституту комп'ютерних систем, Одеський національний політехнічний університет, пр. Шевченка, 1, Одеса, 65044, Україна



Pavlo Igorovich Mykhaylov, general director CEO 3D GNERATION GmbH (Germany), Viktoriastraße, 15, Dortmund, 44137, Germany
ORCID: 0000-0001-5861-5970, pm@3dgeneration.com
Research field: Formation and Processing of Graphic Images

Павло Ігорович Михайлов , генеральний директор 3D GNERATION GmbH (Німеччина), Viktoriastraße 15, Дортмунд, 44137, Німеччина

Roman Yurievich Chekhmestruk, PhD (Eng), Technical Director 3D GENERATION UA, Pirogova Str., 37, Vinnitsa, 21021, Ukraine
ORCID: 0000-0002-5362-8796, Rc.ua@3dgeneration.com
Research field: Formation and Processing of Graphic Images

Роман Юрійович Чехместрук, кандидат технічних наук, технічний директор 3D GENERATION UA, вул. Пирогова 37, Вінниця, 21021, Україна