DOI: 10.15276/hait.03.2020.4
UDC 004.925.8

Interactive shape modeling using functionally defined objects

Olexandr N. Romanyuk
Vinnytsyia National Technical University, Vinnitsa, Ukraine
ORCID: 0000-0002-2245-3364
Sergey I. Vyatkin
Institute of Automation and Electrometry, SB RAS, Novosibirsk, Russia
ORCID: 0000-0002-1591-3588
Pavlo I. Mykhaylov
CEO 3D GNERATION GmbH, Dortmund, Germany
ORCID: 0000-0001-5861-5970
Roman Y. Chekhmestruk
3D GENERATION UA, Vinnitsa, Ukraine
ORCID: 0000-0002-5362-8796

ABSTRACT

Creating digital models is a complex task in computer graphics. Animation developers usually use two methods. The models are either sculpted from a traditional material such as clay or plasticine, and then the model must be digitized. Models can also be created using one of several commercial (or custom) modeling systems, such as MAYA or SoftImage. Since clay can be molded to create smooth surfaces and precise details, most designers very often use this method. It would be useful to give users the same opportunity as modeling from clay or plasticine, but in virtual space. So that the designer can deform the work piece, add details, and remove unnecessary parts. In addition, virtual shopping malls, virtual worlds, scientific visualization, design, construction, and so on, require huge costs to transmit three-dimensional geometric data over the network. This requires a compact description of three-dimensional objects. Considering these requirements, methods were developed with the following features. Innovations in the interactive modeling interface that take advantage of functional model assignment. This is the orientation and positioning of the sculpting tool relative to the surface. The paper describes the interactive modeling of deformation forms of models based on perturbation functions. Such objects are characterized by a high degree of smoothness and are described by a small number of functions. They are easy to deform and create shapes similar to modeling from plasticine. The proposed method of deformation of the functionally based models with fast visualization allows to provide the interactivity and a realistic appearance of the resulting shapes. An interactive modeling of deformations is presented. The process of interactive modeling of geometric shapes defined by perturbation functions is described. A method for interactive modeling of functionally defined objects without preliminary triangulation is proposed. This allows more accurate definition of 3D shapes and simplifies the modeling system. The algorithm for finding the minimum common parent for the objects, the algorithm for adding an object (perturbation) to the scene, and the algorithm for selecting the objects in the scene were developed for this purpose. A method for visual representation of free forms and analytical perturbations for interactive modeling is developed. An interactive scene editor has been created with the ability to save the result both as a scene file and as a bitmap image. The set of primitives for constructing scenes has also been expanded, and the properties of new primitives have been investigated. When creating the editor, work was done to optimize the rasterization algorithm. A method adapted for graphic processing units is used for rapid rendering of 3D models. The considered scientific problem can be used to facilitate the modeling of 3-dimensional surfaces with different types of deformations, which can be relevant for solving applied problems.

Keywords: interactive modeling; functionally defined objects; deformation; perturbation functions

For citation: Romanyuk O. N. Vyatkin S. I. Mykhaylov P. I., Chekhmestruk R. Y. Interactive shape modeling using functionally defined objects. *Herald of Advanced Information Technology* . 2020;Vol.3, No.3: 149–162. DOI: 10.15276/hait.03.2020.4

1. INTRODUCTION

In most modeling applications, such as engineering, medical, and others, there is a great need for approaches that allow modeling and visualization of the material properties of an object and its behavior under the action of forces applied to it. Medical application requires the modeling of soft body tissues [1]. In the work [1], a finite element method was used to model elastic deformations.

Technical applications are needed mainly to model elastic deformations of objects whose components consist of isotropic materials under the conditions of applicability of Hooke's law (elasticity theory). Review work [2] most of the previous work on modeling deformable objects. There are works with the boundary element method for modeling linear elastic homogeneous objects [3], as well as using of iterative dynamic schemes for modeling nonlinear elasticity [4–5]. For nonlinear elastic objects of high detail, it is still not possible to simulate deformations in real time. For a simplified linear formulation of the elasticity theory, it is

possible to solve large-scale systems of equations in real time synchronously with graphical processing, or, to speed up the method, part of the calculations can be performed in advance. In addition, depending on the internal structure of the object, you can use the finite element method or the boundary element method.

This paper describes the various approaches and methods and tools used based on the application of linear elasticity theory, finite element and boundary element methods with the pre-computation of the complete basis of solutions (Green's functions). The simulation results visualization is implemented as an application and is used in real-time interactive, user-defined elastic deformations of functionally specified objects.

Creating digital models is a complex task in computer graphics. Animator developers usually use two methods. The models are sculpted from traditional material such as clay or plasticine, and then the model must be digitized. Models can also be created using one of several commercial (or custom) modeling systems, such as MAYA or SoftImage. Since clay can be molded to create smooth surfaces and precise details, most designers use this method very often.

It would be useful to give users the same opportunity as modeling from clay or plasticine, but in virtual space. So that the designer can deform the workpiece, add details, and remove unnecessary parts. Such manipulations are possible using of functionally defined shapes.

There are known methods based on implicit surfaces [6], triangle meshes [7], images [8], volumes [9] and function-based models [10]. The possibility of interactive rendering of shapes of 3D objects was considered [11–13]. Examples of explicit and implicit functions are soft objects [14], droplet models [15], and perturbation functions [16]. Known sculpting interfaces, skeletally driven modeling, Morse theory, and quad-meshing. Sculpting interfaces allow to construct and manipulate 3D shapes using the methods of sculpting. This methods build up shape detail by smoothly adding or removing material [17], by cloning existing 3D geometry [18] and preserving surface features [19]. Skeletally driven modeling focuses on the creation of objects where the surface shape is defined by a skeletal structure. In this method attempts to create shape as a combination of a number of skeletal implicit surface primitives [20–21]. Morse theory is a tool for analyzing the topology of surfaces through the critical points of a function defined on the surface. Given a Morse function, the Reeb graph is a graph where each connected component of each level set of maps to a single point on the graph [22]. Quad-meshing methods attempt to define the flow-lines of shape [23].

A deformation modeling method based on perturbation function [16; 24] is proposed. This paper extends the work [24]. In this work, we consider an interactive modeling approach, which is a tool for creating 3D objects without preliminary triangulation. For rapid rendering, a method involving the use of graphics processors is used. A source environment is developed, which allows interactive generation or editing of 3D objects using perturbation functions. The proposed software tools significantly simplify generation of functionally defined models.

2. PROBLEM STATEMENT

It is often rather difficult to calculate the defining function at a given point in functional definitions of surfaces. For this reason, preliminary surface triangulation is often performed for interactive modeling [25–26]. The available function-based object definition methods are limited to a rather narrow group of modeled surfaces and slow visualization. One of the main disadvantages of available visualization methods is the difficulty in calculating surface points. Some methods do not guarantee finding surface points. Methods based on a sculpting objects proposed model construction using implicit functions [20]. Such surface is tesselated into a mesh that can be further manipulated. Adding complex shape details to this mesh will finally destroy any constructive connection between the original skeleton and the final 3D shape. For polygonal models, it is difficult to ensure transformations of geometric objects using operations on functions that are necessary to perform unary and binary geometric operations and to detect collisions between the objects. In the case of polygonal object definition, it is rather difficult to compute surface deformation because each apex of the triangular grid has to be subjected to geometric calculations. Today’s realistic graphics requires a compact scene definition with an accurate description of 3D objects, which ensures effective implementation of various geometric operations with object models in simulations of the behavior of interacting objects. Thus, the low level of realism of polygonal models and limited functional capabilities in terms of formation of graphical scenes give rise to an academic and simultaneously applied problem. Solving this problem requires the development of theoretical grounds for modeling and visualization of three-dimensional objects of a virtual medium,

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

which would eliminate the drawbacks typical for the polygonal definition of object models.

The following tasks are set.

A. Develop a way to visually represent free forms and analytical perturbations for interactive scene creation and editing. Create a pre-modeling tool (which consists of specifying their type, position, size, etc.) and then transmitting data for direct image generation. Implement the ability to save the result, both as an image and as a scene file-for further improvement.

B. Extend the set of primitives for building scenes. The task is to implement an interactive user interface for creating and editing functionally defined surfaces using perturbation functions.

It is often rather difficultto calculate the defining function at a given point in functional definitions of surfaces. For this reason, preliminary triangulation of the surface is often performed for interactive modeling [25]. Available methods of object definition on the basis of functions are limited to a rather narrow group of modeled surfaces and slow visualization. One of the main drawbacks of available visualization methods is the difficulty in calculating surface points. Some methods do not guarantee finding surface points.

In this paper, we consider an interactive modeling approach, which is a tool for creating three-dimensional objects without preliminary triangulation. A method that involves using graphics processing units is used for rapid visualization. A source environment is developed, which allows interactive generation or editing of three-dimensional objects with the use of perturbation functions. The proposed software tools significantly simplify generation of functionally defined models.

3. BLENDING MATHEMATICAL BASIS OF THE METHOD

Discrete methods such as the finite element method are commonly used to represent deformable objects in mathematical modeling. In this case, the object is divided into a finite set of elements, and the conditions of physical equilibrium for each of the elements are determined. These conditions are recorded as a very large system of equations with tens of thousands of unknowns. Such a system is usually degenerate, but after imposing the appropriate boundary conditions, it becomes well defined and has a unique solution. Finding a solution requires extremely intensive computation, requiring a large amount of RAM and execution time.

It is also desirable to optimize the solution algorithm, especially in relation to its execution speed.

Elasticity theory characterizes the deformation of elastic objects using displacements $\vec{u}(\vec{x})$ that transform the points of the object as follows: $\vec{x} \to \vec{x} + \vec{u}(\vec{x})$.

The strain tensor is determined by partial derivatives:

$$\phi_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} + \frac{\partial u_k}{\partial x_i}\frac{\partial u_k}{\partial x_j} \right), \quad (1)$$

where it is assumed the summation over repeated indices. The strain tensor characterizes changes in distances in a deformed object so that the length of a small element $d\vec{x}$ is transformed as follows

$$\left|d\vec{x}\right|^2 \to \left|d\vec{x}\right|^2 + 2\phi_{ij}\,dx_i\,dx_j \quad . \quad (2)$$

The distribution of forces in an elastic body is characterized as follows. Consider a slice of a body and a small element $d\vec{S}$.

This vector is directed along the normal to the slice, the modulus $\left|d\vec{S}\right|$ is equal to the area of the element. The force $d\vec{f}$ acting through this element between the parts of the object separated by the slice is defined as $d\vec{f}_i = \sigma_{ij}\,dS_j$. The coefficient σ_{ij} is called the tension tensor.

It is symmetric $\sigma_{ij} = \sigma_{ji}$, also known as the tension tensor, and can be written similarly:

$$\sigma = \left(\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz}\right)^T . \quad (3)$$

Hooke's law relates the tension and deformation tensors: $\sigma = D\phi$, where the matrix D is independent of deformations and characterizes material properties. This linear relation holds for small deformations under the additional assumption that the undeformed object has a zero tension tensor. In the case of small displacements, only linear terms can be stored in the deformation tensor by writing $\phi_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$, the resulting approximation is called small deformation theory.

For a homogeneous isotropic material, the matrix D is represented as a linear combination $D = \lambda D_1 + \mu D_2$, where $D_{1,2}$ are the 6x6 matrix constants.

The energy of a deformed object is given by an integral $E = \int dV\left(\frac{1}{2}\phi^T + U\right)$, where the first term represents pure elastic energy, and the second is associated with a potential field of external forces $\vec{f} = -\nabla U$, for example, the field of gravity. The equilibrium conditions corresponding to the

minimum energy have the form $f_i = \partial\sigma_{ij} / \partial x_j$. In terms of unknown displacements, this yields a system of second-order linear partial differential equations.

When substituting in the right-hand part of this system forces of the form $f_i(\vec{x}) = \delta_{ik}\delta(\vec{x} - \vec{x}_0)$, where δ_{ik} is the Kronecker symbol and $\delta(\vec{x} - \vec{x}_0)$, is the Dirac function. The solution of the system for the data $u_i(\vec{x}, \vec{x}_0, k) = G_{ik}(\vec{x}, \vec{x}_0)$ is called the Green's function of the system. This function describes how the effect of a force applied locally at a point $\vec{x}_0$ propagates to a point $\vec{x}$. If this function is known, the solution of the system for arbitrary right-hand parts $\vec{f}\ (\vec{x})$ can be found as

$$u_i(\vec{x}) = \int d^3 x_0 G_{ik}(\vec{x}, \vec{x}_0) f_k(\vec{x}_0). \qquad (4)$$

4. INTERACTIVE MODELING METHOD

4.1. Geometric Objects and Operations

Objects are constructed using basic quadrics, perturbations, and set-theoretic operations.

The basic equation of the quadric is

$$F(x, y, z) = A_{11}x^2 + A_{22}y^2 + A_{33}z^2 + A_{12}xy + \\ + A_{13}xz + + A_{23}yz + A_{14}x + A_{24}y + A_{34}z + A_{44} \geq 0. \qquad (5)$$

Equation of the basic quadric with perturbations is

$$F'(x, y, z) = F(x, y, z) + \sum_{i=1}^{N} f_i R_i(x, y, z), \qquad (6)$$

where the perturbation function R(x, y, z) is

$$R_i(x, y, z) = \begin{cases} Q_i^3(x, y, z), \text{ if } Q_i(x, y, z) \geq 0 \\ 0, \text{ if } Q_i(x, y, z) < 0 \end{cases}, \qquad (7)$$

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

In order to create a complex scene, it is necessary to describe in it a certain number of primitives necessary for a particular task. The displayed object with which the rasterization algorithm interacts through queries represents the entire three-dimensional scene. Therefore, the geometric model should allow you to design objects and their compositions of unlimited complexity. This is achieved primarily by using Boolean Union and intersection operations. The whole scene is a tree view, each node of which is a constructor object that performs logical operations on its descendants, and the vertices of the tree are primitives used by the system. Now when the rasterizing algorithm makes a request to the constructor object, this object refers to its descendants, converts the result, and gives the appropriate response to the request. In this case, the descendants can be either a primitive or another constructor object. When applying geometric operations, rotations, displacements, scaling, to the constructor object, it performs all these operations with its descendants.

Thus, a scene is a binary tree whose root is an object that represents the entire three-dimensional scene and has the property to respond to a request for an intersection with a volume; the nodes contain operators, joins, or intersections-the operator is also an object, and the leaves are objects represented by quadrics.

On the basis of quadrics, you can build more complex smooth surfaces. This is achieved by deforming the base quadric. *Figure 1* shows two variants of deformation of one ellipsoid by another with variation of one parameter. In General, with such a perturbation, the volume of the quadric can be distorted both in the direction of decreasing and increasing its value. As a result, the perturbation quadric is a second-order surface with higher-order local values (mostly fourth-order).

Fig. 1. **Free form objects**

The class library of functionally defined objects has a static map for the names of objects registered in the library and for creating functions. The names of the registered objects are automatically defined as tokens during parsing of the scene. The task of the preparatory phase is to build a geometric model. At the input of the program, there are several text files, each of which describes a logically complete element of the scene and all its necessary parameters. With the help of the preprocessor, these files are translated into a single file containing the full description of the scene. The resulting file incoming to the input parser, that one, which parses the construction of the text and carries out the construction of the model. Upon successful completion of the construction, the model undergoes a projective transformation, after which the preprocessing phase is completed.

The proposed method of describing objects of three-dimensional scenes by base surfaces and perturbation functions has a compact description, which allows reducing from 10 to 1000 times the

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

amount of transmitted data depending on specific three-dimensional scenes and models. Objects with flat faces are also easily defined, for example, a cube can be defined by three quadrics. In addition, when solving the describing function in the form of inequality $F(X) \geq 0$, then it becomes possible to display the internal part of the object.

4.2. Interactive Deformation

Primitives reduce the problem of designing an object to the problem of deformation of the base surface in the desired way, and not to approximation, this process resembles modeling a model from plasticine using geometric operations presented in [16]. To calculate the coordinates of the points of the deformable surface, the method of determining collisions of functionally specified objects is used [24]. This collision detection method is based on the intersection relation of objects and recursive subdivision of object space to find the first point of contact of objects.

Deformation consists in the possibility of adding to any point on the surface perturbation with parameters specified by the tool. The tool sets the scope and type of perturbation. The user, as a simple example, selects the tool; it can be a thin cylinder. To do this, a library of objects (tools) was created. That is, you can add a perturbation with predefined parameters to any point on the surface using the selected tool. There is also a function of selecting an object in the scene to perform operations with it: rotation, compression stretching, movement, etc. Thus, the perturbation is added to the surface of the object in the place indicated with the mouse. As mentioned, the data in memory is stored as a tree, in which each scene object is represented as a function description with its properties and a subtree. The tree is displayed on the screen, and you can copy, cut, and paste individual branches or the entire tree. It is also possible to call the properties window for each element of the tree and in this window to change the properties of the element. Actions in the main window of the program are performed on the selected element in the tree. The copy function of the geometry of the scene is needed to get a list of objects. Because when you change scenes, you should be able to restore it to its initial state. Using the operation of crossing objects of the scene with the tool placed in the x, y, z coordinates, you can consider objects that can be acted on by the tool.

For the function of determining whether a point belongs to an object, adding depth levels to the object, and pointers to the ancestor, a flag is applied that will signal whether the subtree participated in the rendering. Before returning from a function, the pointer to an object is remembered if it has no descendant. Thus, the roots of the tree are found. A field is created in Vxobject that will show the depth of the tree. The depth of the subtree is calculated just before finding the youngest ancestor for all objects in the list. Since, if the depth is calculated during the construction of the scene, it may be necessary to recalculate the depths of all nodes after the merge or intersection operations, and this is a long time. The depth of only the roots is calculated, and then-the desired depth of the node.

4.3. Interface Concept

Task 1. You must be able to add a perturbation with predefined parameters to any point on the surface (tool). This perturbation sets the scope, and it is on this scope that we must perturb our object.

Task 2. You need to be able to select an object on the stage and be able to do some operations with it: rotation, compression, stretching, and movement. You can select objects using the task 1 tools that define the scope. For example, a thin cylinder.

Task 3. Write to a file. We must be able to save a scene to a file.

Task 4. Interface. Program response to mouse events (modification of MRS vectors (move, rotate, scale)). Selecting the MRS action. List of tools. Low-resolution rendering during modification.

The function of copying the scene geometry is necessary to obtain the list of objects, because there should be a possibility of recovering the scene to the initial state after it is changed. The algorithm of finding the minimum common parent for the objects is shown in *Fig. 2*, while *Fig. 3* illustrates the algorithm of adding an object (perturbation) to the scene. By using the operation of intersection of the objects of the scene with the tool placed in the x, y, z coordinates, it is possible to consider the objects subjected to the action of the tool. For the function of determining whether the point belongs to the object and for adding the depth levels to the object and references to the parent, the code has a flag, which signalizes subtree participation in rendering. After that, the reference to the object is stored if this object has no descendants. Thus, the tree roots are found. A field in the class VxObject is created, which indicates the tree depth (*Fig. 3*). The subtree depth is calculated directly before finding the minimum common parent for all objects in the list. If the depth is calculated during scene construction, it may become necessary to recalculate the depths of all nodes after the unit or intersection operations, which requires a lot of time; for this reason, only the depths of the roots are calculated.

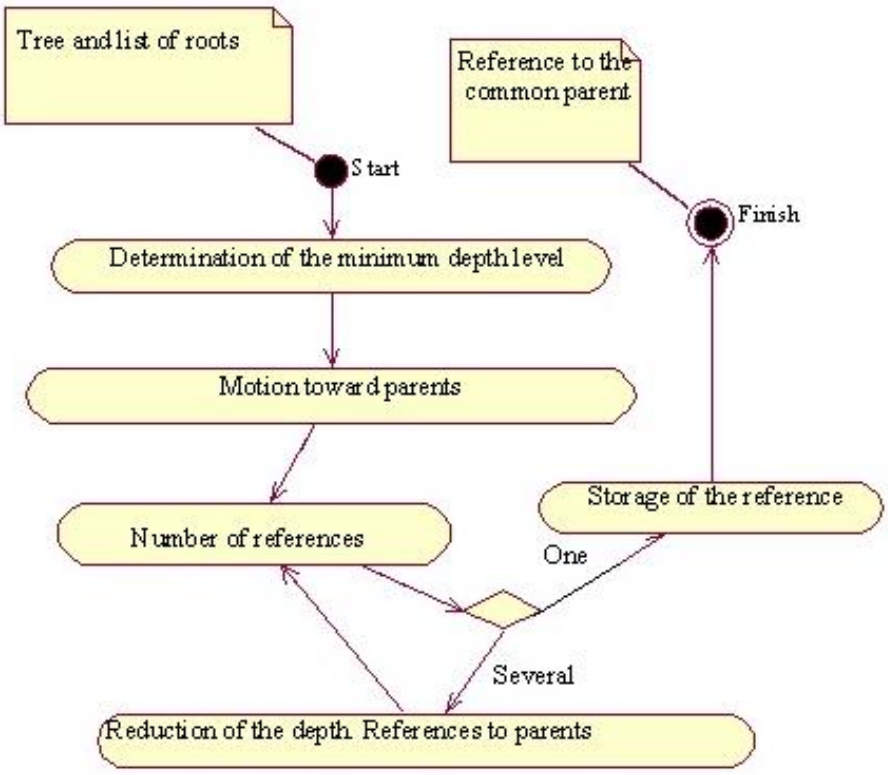


Fig. 2. **The algorithm of finding the minimum common parent for the objects**

Figure 4 illustrates the algorithm of choosing the objects in the scene. An important issue is the possibility of deformation of the initial shape, not only of adding new perturbations.

Algorithm for adding an object to the scene, in the place specified by the mouse:

Find X, Y converted to the coordinates of the scene coordinates of the bmp, which displays the image. Multiply the entire object matrix by the camera matrix.

Run the scene construction algorithm and at the last level of binary division, see if the X, Y coordinates match the coordinates of the point that is currently being calculated. If they match, take the current Z coordinate. We build images for speed with a small level of recursion and without displaying them on the screen. We move the object we want to add to the received coordinates. Thus, we add an object to the surface of another object in the place that we specify with the mouse.

Algorithm for finding objects in the scene, in the place selected with the mouse:

Find the X, Y, Z coordinates of the scene where the mouse hit. Run the scene construction algorithm and at the last level of binary division, see if the X, Y coordinates match the coordinates of the point that is currently being calculated. If they match, we start adding all the objects in the scene that participated in the calculation of this point to the list. We build images for speed with a small level of recursion and without displaying them on the screen.

You can use the interface to interactively add, delete, copy, move objects in space, and add or remove properties. Objects can be moved either by dragging the mouse on the screen, or by dragging sliders, as well as manually substituting coordinate values.

An important point is the ability to isolate and work only with perturbations. So that existing perturbations change during deformation, and not only new ones are added. Part of the interactive system is the program VxDemo, designed to visualize functionally defined objects. Input data for VxDemo is a text file (with the extension SCN) created in any text editor or interactively. The file contains a scene description. The text of a scene file is a set of tokens that can be divided into the following categories: graphic primitives, primitive properties, operators, texture, lighting parameters. The output is the resulting images of scenes in graphical format. For changing the parameters of the scene display use the Options window, which is called through the View menu item of the same name. The tool sets the scope and type of perturbation. To do this, some information is graphically provided to the user (including the

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

selection of an object, for example, by color, or by bounding box, drawing axes, etc.). All rendering of "support" work with operations is done using OpenGL – for speed and functionality. Thus, the main features of the rendering class include allocation of rendering to a separate thread, support for OpenGL rendering (compatibility of buffers, depth test, etc.). Formation of bounding box for the objects. A separate thread is also needed so that, whenever the user changes the scene, the rendering can be quickly interrupted and restarted.

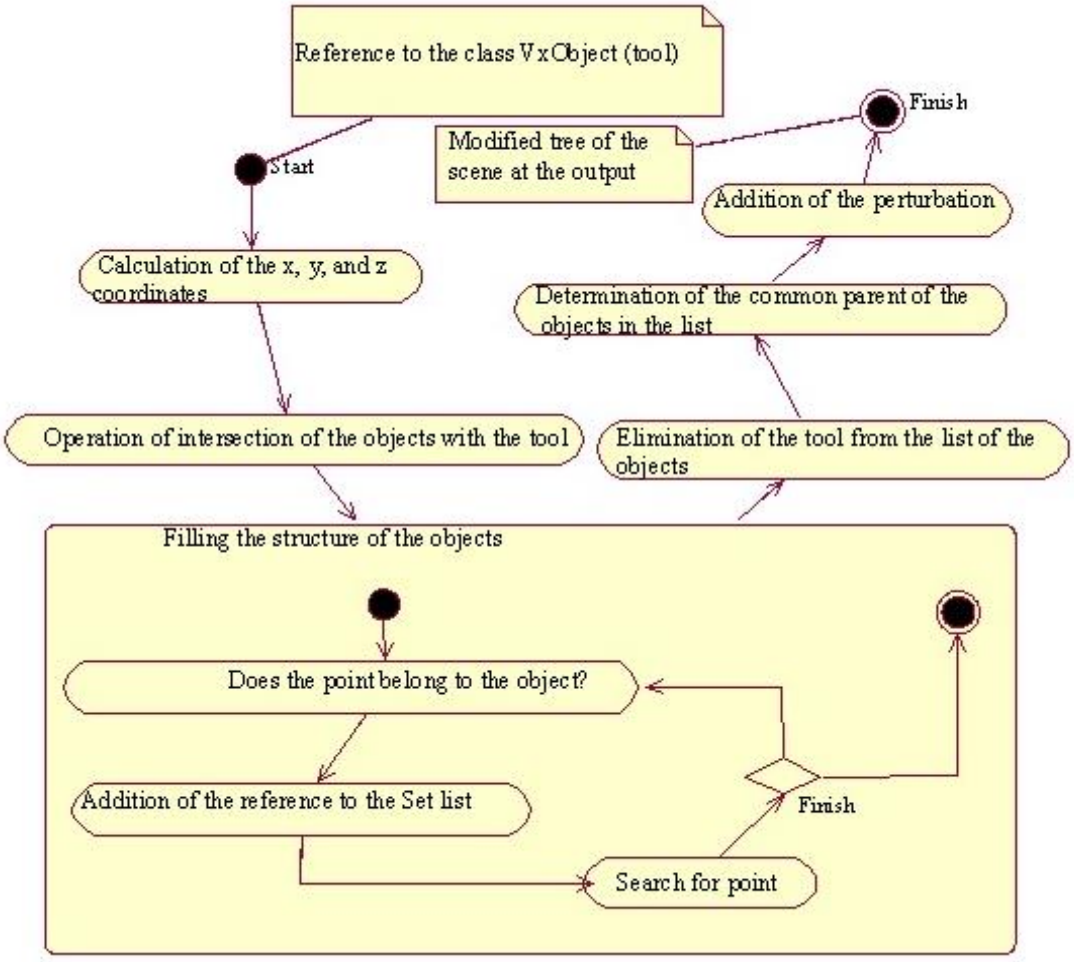


Fig. 3. **The algorithm of adding an object (perturbation) to the scene**

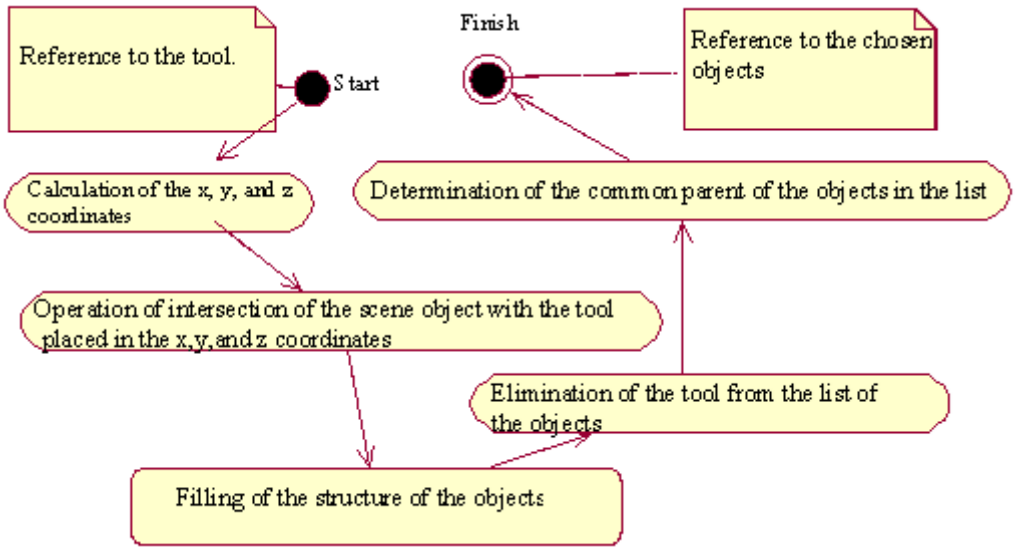


Fig. 4. **The algorithm of choosing the objects in the scene**

The necessary algorithms and a set of C++ classes for the modeling system have been developed: Recursive multi-level ray casting for scenes containing functionally defined objects [24]; conversion from polygonal data format and volume data to a functional description of the objects, manipulation and triangulation of functionally defined objects, developed using these classes. Thus, the system is designed as a set of classes (Framework) to facilitate application development.

All manipulations are object-oriented: first, the object is selected, and then the command is selected. Select an object or objects directly by left clicking in the rendering window (you can see a list of object names at the top of this window) or using the scene tree control (*Fig. 1*) from the context menu.

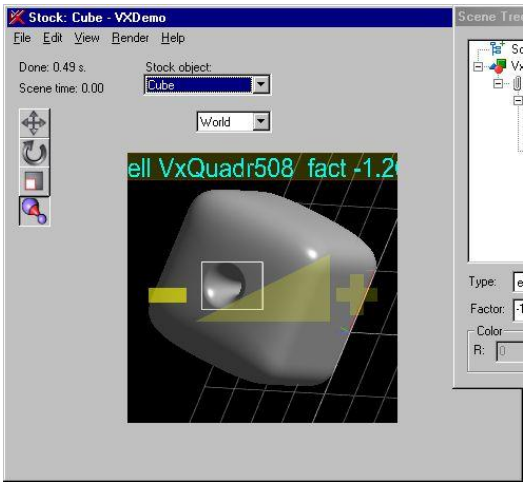


Fig. 5. **Main application window. The perturbation object is selected for deformation**

During the deformation of the objects, you must perform the following steps

1. To add a new perturbation to an arbitrary location of an object without using a tree control, first set the mode for manipulating the deformation (by clicking the button see, *Fig. 1*). You just need to left-click and move the mouse cursor to the point of the object where you want to add the disturbance. You need to hold the left button down and move the button to the left to add a disturbance with a negative factor, or to the right to add a disturbance with a positive factor. If you move the mouse and do not release the left button, the greater the coefficient (in absolute value) of the new disturbance.

2. If you select objects in the scene with a mouse click and there are perturbations among the selected objects, their factors change without adding new perturbations. This can be useful if you need to change an existing disturbance – just select it while holding down the “Ctrl” key.

Interactive modeling stuff is grouped in VxManipulator class.

Proposed interface for design system provides four basic manipulations: Move, Rotate, Scale and Deform (*Fig. 5*).

5. VISUALIZATION

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 slow [27–28]. The method of determining the intersection of a ray with an implicitly defined surface is too complex to calculate the L- and G-parameters [29]. In the tracing method, finding the maximum radius when no point of the volume lies within the sphere is a nontrivial task [30]. Ray tracing with analysis of the interval for complex functions requires individual calculations for each ray and each interval along this ray [31]. 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 [32].

A ray tracing method for imaging surfaces defined by algebraic polynomials of high degree is described in [33]. 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.

There is another technique for the visualization of analytically defined objects using graphics processing units (GPU) [34] which is based on conventional single-step tracing of rays. 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 centered 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.

The aim of this work as well is to develop a method for visualizing functionally defined objects based on perturbation functions using graphics processing units.

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

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

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. 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. Further, a normal is defined at each pixel. In the presence of all the coordinates and normals at each pixel, the local illumination model is used. The result is an image of a smooth object neglecting illumination.

The visualization time is reduced by using the computational resources of a graphics processing unit with compute unified device architecture (CUDA) (NVIDIA, (USA). 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.

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). 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. *Figure 6*, *Fig. 7* and *Fig. 8* show the comparative test results of the dependence of the per-frame computation time on the number of defined perturbation functions for a particular test.

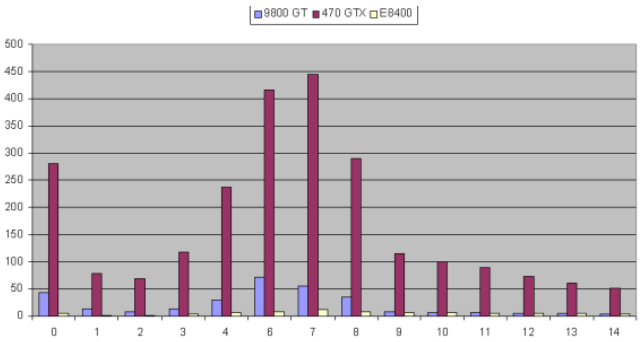


Fig. 6. **Timing diagram: the number of frames per second for different tests**

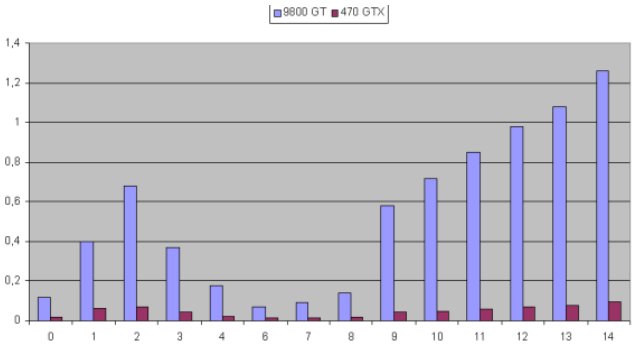


Fig. 7. **Timing diagram: the average time frame for different tests**

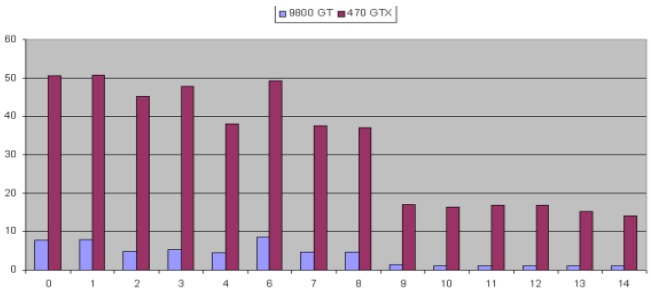


Fig. 8. **Timing diagram: the acceleration relative to the E8400 for different tests**

Table 1 and *Table 2* shows the computation time, the number of operations per second and acceleration. The *Tables* shows data from 16 tests: the rendering time and the number of frames per second.

Table 3 shows the characteristics of a specific test. The number of perturbation functions is shown for each test.

6. CONCLUSIONS

Many methods for interactive modeling and editing of functionally defined models involve intermediate triangulation of the edited surface for the purpose of decreasing the computation time,

Table 1. **Time computation, the number of operations per second and acceleration, tests 1-8**

Testing values	Types of CPU	0	1	2	3	4	6	7	8
Time	9800 GT	0,1164	0,39797	0,68046	0,3689	0,17703	0,07047	0,09109	0,13953
	470 GTX	0,01782	0,06266	0,07235	0,04251	0,02109	0,01203	0,01125	0,0172
	E8400	0,90156	3,17578	3,27562	2,0336	0,80265	0,59422	0,42344	0,63796
FPS	9800 GT	42,95533	12,56376	7,34797	13,55381	28,2438	70,95218	54,89077	35,83459
	470 GTX	280,5836	79,79572	69,1085	117,6194	237,0792	415,6276	444,4444	290,6977
	E8400	5,545943	1,574416	1,526429	2,458694	6,229365	8,414392	11,80805	7,837482
Accelera-tion	9800 GT	7,745361	7,979948	4,813832	5,512605	4,533977	8,432241	4,648589	4,572207
	470 GTX	50,59259	50,68273	45,27464	47,83816	38,05832	49,39485	37,63911	37,0907

Table 2. **Time computation, the number of operations per second and acceleration, tests 9-16**

Testing values	Types of CPU	9	10	11	12	13	14	15	16
Time	9800 GT	0,57969	0,71531	0,85265	0,97703	1,0797	1,25798	0,04205	0,10891
	470 GTX	0,04345	0,05016	0,05688	0,06829	0,08079	0,09734	0,0061	0,01798
	E8400	0,74001	0,82063	0,95875	1,1536	1,23343	1,36859	2,97159	21,98563
FPS	9800 GT	8,6253	6,989976	5,864071	5,11755	4,630916	3,974626	118,9061	45,90947
	470 GTX	115,0748	99,68102	87,90436	73,21716	61,88885	51,36634	819,6721	278,0868
	E8400	6,756665	6,09288	5,215124	4,334258	4,053736	3,653395	1,682601	0,227421
Accelera-tion	9800 GT	1,276562	1,147237	1,124436	1,180721	1,142382	1,087927	70,66801	201,8697
	470 GTX	17,0313	16,36025	16,85566	16,89266	15,26711	14,05989	487,1459	1222,783

which complicates the modeling method itself and the entire system as a whole. Moreover, the computation accuracy is deteriorated by these intermediate manipulations. It should be also noted that successful triangulation is not possible for all geometric shapes; hence, there are constraints on the complexity of modeled shapes. In the proposed approach, we managed to avoid additional triangulation operation, while the interactive regime is retained. This is ensured by using appropriate methods of definition of 3D objects, visualization and detection of object collisions, geometric operations, and possibility of local and global deformation.

As a result, a source environment that allows interactive formation and/or editing of functionally defined objects and a software system that significantly simplifies construction of functionally defined objects with the use of perturbation functions were developed.

We have proposed a method modeling of deformations as well. The following tasks were solved: program response to mouse events (and, accordingly, selection and modification of objects).

Table 3. **The main characteristics of the specific test**

	Number of objects	Number of perturbation functions									
		0	1	2	3	4	5	6	7	8	9
0	1	4									
1	22	4	4	4	1	1	4	1	1	3	1
2	22	4	4	4	1	1	6	1	1	3	1
3	3	1	1	3							
4	3	1	1	3							
5	1	0									
6	10	0	0	0	0	0	0	0	0	0	0
7	1	2									
8	1	4									
9	1	5									
10	1	6									
11	1	7									
12	1	8									
13	1	9									
14	1	10									
15	1	1									
16	10	1	1	1	1	1	1	1	1	1	1

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

Selecting an object (or several objects) in the scene and giving the user the opportunity to do some operations with it (them). Selecting an operation – affine MRS transformations (move, rotate, scale), geometric operations and deformation; the ability to work with a list of tools; writing to a file and loading the scene tree; the ability to create new tags and recognize them.

The software model of the interactive modeling method of functionally specified objects is implemented in the high-level language C++. This language is well suited for the implementation of the object model of the described system, as it is object-oriented.

Developed and implemented the necessary methods and C++ classes: the method of binary search of image elements; C++ classes are functionally defined objects; C++ classes to render functionally defined objects; C++ classes interface system interactive volume-based geometric modeling with a simple extension mechanism for new algorithms and features.

These hierarchical classes form the core of a package that can be extended to add functionality or change characteristics.

Testing was performed on graphics processors of the company N-video.

REFERENCES

1. Cotin, S., Delingette, H. & Ayache, N. “A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation”. *Visual Comp* 16. 2000. p. 437–452. DOI: 10.1007/PL00007215.

2. Sarah, F, Gibson, F. & Mirtich, B. “A survey of deformable modeling in computer graphics. Cambridge”. *MA: Mitsubishi Electric Research Laboratories*. Technical Report TR97-19. 1997.

3. James, D. L. & Pai, D. K. “Artdefo: accurate real time deformable objects”. *In SIGGRAPH ’99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: *ACM Press/AddisonWesley Publishing Co*. 1999. p. 65–72. DOI: 10.1145/311535.311542.

4. Picinbono, G., Delingette, H. & Ayache N. “Real-Time Large Displacement Elasticity for Surgery Simulation: Non-linear Tensor-Mass Model”. In: Delp S.L., DiGoia A.M., Jaramaz B. (eds) “Medical Image Computing and Computer-Assisted Intervention – MICCAI 2000”. Lecture Notes in Computer Science. *Springer*. Berlin, Heidelberg: 2000; Vol.1935. DOI: 10.1007/978-3-540-40899-4_66.

5. Zhuang, Y. & Canny, J. “Real-time and physically realistic simulation of global deformation”. *In: SIGGRAPH 1999. ACM Press.* New York. 1999. 270 p. DOI: 10.1201/9781439864135-16.

6. Perry, R. N. & S. F. Frisken Kizami. “A System for Sculpting Digital Characters”. *In SIGGRAPH’01*. 2001. p. 47–56.

7. Agrawala, M., Beers, A. C. & Levoy, M. “3D painting on scanned surfaces”. *In: SI3D ’95: Proceedings of the 1995 Symposium on Interactive 3D Graphics*. ACM. New York. 1995. p. 145–150. DOI: http://doi.acm.org/10.1145/199404.199429.

8. Oh, B. M., Chen, M., Dorsey, J. & Durand, F. “Image-based modeling and photo editing”. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques – SIGGRAPH ’01*. 2001. DOI:10.1145/383259.383310.

9. Huff, R., da Rosa, R. S., Nedel, L. & Freitas, C. M. D. S. “Volume sculpting based on geometric tools. Journal of the Brazilian Computer Society”. 2009; 15(2): 3–18. DOI:10.1007/bf03194498.

10. Levinski, K. & Sourin, A. (n.d.). “Interactive function-based artistic shape modeling”. *First International Symposium on Cyber Worlds.* Proceedings. 2002. DOI:10.1109/cw.2002.1180921.

11. Szeliski, R. & Tonnesen, D. “Surface modeling with oriented particle systems”. *Computer Graphics (SIGGRAPH’92)*. 1922; 6(2): 185–194. DOI: 10.1145/142920.134037.

12. Welch, W. & Witkin, A. “Free-Form Shape Design Using Triangulated Surfaces”. *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques – SIGGRAPH’94. ACM Publ.* New York, USA: 247–256.

13. Curless, B. & Levoy, M. A volumetric method for building complex models from range images. Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques – SIGGRAPH ’96. 1996. DOI: 10.1145/237170.237269.

14. Wyvill, G., McPheeters, C. & Wyvill, B. “Data structure for soft objects”. *The Visual Computer*. 1986; 2(4): 227–234. DOI: 10.1007/bf01900346.

15. Wyvill, B. Guy, A. & Galin, E. “Extending the CSG Tree. Warping, Blending, and Boolean Operations in an Implicit Surface Modeling System”. Computer Graphics Forum. 1999; 18 (2): 8–24. DOI: 10.1111/1467-8659.00365.

16. Vyatkin, S.I. “Complex Surface Modeling Using Perturbation Functions”. *Optoelectronics, Instrumentation and Data Processing*. 2007; Vol. 43 No. 3: 226–231. DOI: 10.3103/S875669900703003X.

17. Galyean, T. A. & Hughes, J. F. “Sculpting: An Interactive Volumetric Modelling Technique”. *Proceedings SIGGRAPH ′91, Computer Graphics*. 1991; 25 (4): 267–74.

18. Takayama, K., Schmidt, R., Singh, K., Igarashi, T., Boubekeur, T. & Sorkine, O. “GeoBrush: Interactive Mesh Geometry Cloning”. *Computer Graphics Forum*. 2011; 30(2): 613–622. DOI: 10.1111/j.1467-8659.2011.01883.x

19. Stãnculescu, L., Chaine, R., Cani, M.-P. & Singh, K. “Sculpting multi-dimensional nested structures”. *Computers & Graphics*. 2013; 37(6): 753–763. DOI:10.1016/j.cag.2013.05.010.

20Bloomenthal, J. “An introduction to implicit surfaces”. Morgan-Kauffman, San Francisco: 1997.

21. Zanni, C., Bernhardt, A., Quiblier, M. & Cani, M.-P. “SCALe-invariant Integral Surfaces”. *Computer Graphics Forum*. 2013; 32(8): 219–232. DOI:10.1111/cgf.12199.

22. Biasotti, S., Giorgi, D., Spagnuolo, M. & Falcidieno, B. “Reeb graphs for shape analysis and applications”. *Theoretical Computer Science*. 2008; 392(1-3): 5–22. DOI:10.1016/j.tcs.2007.10.018.

23. Bessmeltsev, M., Wang, C., Sheffer, A. & Singh, K. “Design-driven quadrangulation of closed 3D curves. ACM Transactions on Graphics. 2012; 31(6): DOI: 10.1145/2366145.2366197.

24. Vyatkin, S. I., Romanyuk, A. N., Savytska, L. A., Troianovska, T. I. & Dobrovolska, N. V. “Real-Time Deformations of Function-Based Surfaces using Perturbation Functions”. *Journal of Physics: Conference Series.* 2018. 1015. 032115. DOI: 10.1088/1742-6596/1015/3/032115.

25. Levinski, K. & Sourin, A. (n.d.). “Interactive Function-Based Shape Modeling for Cyberworlds”. *International Conference on Cyberworlds*. 2004. DOI: 10.1109/cw.2004.41.

26. Levinski, K. & Sourin, A. “Interactive polygonisation for function-based shape modelling”. *Eurographics 2002.* 2002. p. 71–79.

27. Tuy, H. & Tuy, L. “Direct 2-D Display of 3-D Objects”. *IEEE Comput. Graph.* 1984; Appl. 4 (10): 29–33. DOI: 10.1145/74333.74359.

28. Perlin, K. & Hoffert, E. M. “Hypertexture”. *Comput. Graph*. 1989; 23 Is.3: 253–262. DOI.org/10.1145/74333.74359.

29. Karla, D. & Barr, A. H. “Guaranteed Ray Intersections with Implicit Surfaces”. *Comput. Graph*. 1989; 23 (3): 297–306. DOI: 10.1145/74333.74364.

30. Hart, J. C. “Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces”. *The Visual Comput*. 1994; 12(10): 527–545. DOI: 10.1007/s003710050084.

31. Mitchell, D. P. “Robust Ray Intersection with Interval Arithmetic”. In: *Proceedings of Graphics Interface '90*. 1990. p. 68–74.

32. Sherstyuk, A. “Fast Ray Tracing of Implicit Surfaces”. *Computer Graphics Forum*. 1999; 18(2): 139–147. DOI: 10.1111/1467-8659.00364.

33. Reimers, M. & Seland J. “Ray Casting Algebraic Surfaces using the Frustum Form”. *Computer Graphics Forum.* 2008; 27(2): 361–370. DOI: 10.1111/j.1467-8659.2008.01133.x.

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

DOI: 10.15276/hait.03.2020.4
УДК 004.925.8

Інтерактивне моделювання форм з використанням функціонально заданих об'єктів

Олександр Н. Романюк
Вінницький національний технічний університет, Вінниця, Україна
ORCID: 0000-0002-2245-3364
Сергій І. Вяткін
Інститут автоматики та електрометрії, СВ РАН, Новосибірськ, Росія
ORCID: 0000-0002-1591-3588
Павло І. Михайлов
CEO 3D GENERATION GmbH, Дортмунд, Німеччина
ORCID: 0000-0001-5861-5970

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

Роман Ю. Чехместрук
3D GENERATION UA, Вінниця, Україна
ORCID: 0000-0002-5362-8796

АНОТАЦІЯ

Створення цифрових моделей - складне завдання комп'ютерної графіки. Розробники анімацій зазвичай використовують два методи. Моделі виліплюються з традиційного матеріалу, такого як глина або пластилін, а потім моделі оцифровують. Моделі також можна створювати з використанням однієї з декількох комерційних (або призначених для користувача) систем моделювання, таких як MAYA або SoftImage. Оскільки з глини можна виліпити гладкі поверхні та точні деталі, більшість дизайнерів дуже часто використовують цей метод. Було б корисно дати користувачам такі ж можливості, як ліплення з пластиліну або глини, але у віртуальному просторі. Щоб дизайнер міг деформувати заготовку, додати деталі та видалити непотрібні частини. Крім того, віртуальні торгові центри, віртуальні світи, наукова візуалізація, проектування, будівництво і т. д. вимагають величезних витрат на передачу тривимірних геометричних даних по мережі. Для цього потрібно компактний опис тривимірних об'єктів. З урахуванням цих вимог були розроблені методи з такими особливостями. Інноваційний інтерфейс інтерактивного моделювання, що використовує призначення функціональних моделей. Це орієнтація та розташування інструменту для ліплення щодо поверхні. У статті описується інтерактивне моделювання форм деформацій моделей на основі функцій збурення. Такі об'єкти характеризуються високим ступенем гладкості та описуються невеликою кількістю функцій. Їх легко деформувати і створювати форми, схожі на ліплення з пластиліну. Запропонований метод деформації функціонально заданих моделей з швидкою візуалізацією дозволяє забезпечити інтерактивність і реалістичність одержуваних форм. Представлено інтерактивне моделювання деформацій. Описано інтерактивне моделювання геометричних форм, заданих функціями збурення. Запропоновано метод інтерактивного моделювання функціонально заданих об'єктів без попередньої тріангуляції. Це дозволяє точніше створювати тривимірні форми та спрощує систему моделювання.
Для цього були розроблені алгоритм знаходження мінімального загального предка для об'єктів, алгоритм додавання об'єкта (збурення) в сцену та алгоритм вибору об'єктів у сцені. Розроблено метод візуального поданння вільних форм і аналітичних збурень для інтерактивного моделювання. Був створений інтерактивний редактор сцен з можливістю збереження результату як у вигляді файлу сцени, так і у вигляді растрового зображення. Було також розширено набір примітивів для побудови сцен і досліджені властивості нових примітивів. При створенні редактора була прорoблена робота по оптимізації алгоритму растеризації. Для швидкого рендеринга 3D-моделей використовується метод, адаптований для графічних процесорів. Розглянута наукова задача може використовуватися для полегшення моделювання тривимірних поверхонь з різними типами деформацій, що може бути актуально для вирішення прикладних завдань

Ключові слова: інтерактивне моделювання; функціонально задані об'єкти; деформація; функції збурення

DOI: 10.15276/hait.03.2020.4
УДК 004.925.8

Интерактивное моделирование форм с использованием функционально заданных объектов

Александр Н. Романюк
Винницкий национальный технический университет, Винница, Украина
ORCID: 0000-0002-2245-3364

Сергей И. Вяткин
Институт автоматики и электрометрии, СО РАН, Новосибирск, Россия
ORCID: 0000-0002-1591-3588

Павел И. Михайлов
CEO 3D GNERATION GmbH, Дортмунд, Германия
ORCID: 0000-0001-5861-5970

Роман Ю. Чехместрук
3D GENERATION UA, Винница, Украина
ORCID: 0000-0002-5362-8796

АННОТАЦИЯ

Создание цифровых моделей – сложная задача компьютерной графики. Разработчики анимаций обычно используют два метода. Модели вылепливаются из традиционного материала, такого как глина или пластилин, а затем модели оцифровывают. Модели также можно создавать с использованием одной из нескольких коммерческих (или пользовательских) систем моделирования, таких как MAYA или SoftImage. Поскольку из глины можно вылепить гладкие поверхности и точные детали, большинство дизайнеров очень часто используют этот метод. Было бы полезно дать пользователям такие же возможности, как лепка из пластилина или глины, но в виртуальном пространстве. Чтобы дизайнер мог деформировать заготовку, добавить детали и удалить ненужные части. Кроме того, виртуальные торговые центры, виртуальные миры, научная визуализация, проектирование, строительство и т. д. требуют огромных затрат на передачу трехмерных геометрических данных по сети. Для этого требуется компактное описание трехмерных объектов.

С учетом этих требований были разработаны методы со следующими особенностями. Инновационный интерфейс интерактивного моделирования, использующий назначение функциональных моделей. Это ориентация и расположение инструмента для лепки относительно поверхности.
В статье описывается интерактивное моделирование форм деформаций моделей на основе функций возмущения. Такие объекты характеризуются высокой степенью гладкости и описываются небольшим количеством функций. Их легко деформировать и создавать формы, похожие на лепку из пластилина. Предлагаемый метод деформации функционально заданных моделей с быстрой визуализацией позволяет обеспечить интерактивность и реалистичность получаемых форм. Представлено интерактивное моделирование деформаций. Описано интерактивное моделирование геометрических форм, заданных функциями возмущения. Предложен метод интерактивного моделирования функционально заданных объектов без предварительной триангуляции. Это позволяет точнее создавать трехмерные формы и упрощает систему моделирования. Для этого были разработаны алгоритм нахождения минимального общего предка для объектов, алгоритм добавления объекта (возмущения) в сцену и алгоритм выбора объектов в сцене. Разработан метод визуального представления свободных форм и аналитических возмущений для интерактивного моделирования. Был создан интерактивный редактор сцен с возможностью сохранения результата как в виде файла сцены, так и в виде растрового изображения. Был также расширен набор примитивов для построения сцен и исследованы свойства новых примитивов. При создании редактора была проделана работа по оптимизации алгоритма растеризации. Для быстрого рендеринга 3D-моделей используется метод, адаптированный для графических процессоров. Рассмотренная научная задача может использоваться для облегчения моделирования трехмерных поверхностей с различными типами деформаций, что может быть актуально для решения прикладных задач.

Ключевые слова: интерактивное моделирование; функционально заданные объекты; деформация; функции возмущения.

ABOUT THE AUTHORS

Olexandr N. Romanyuk – Doctor of Technical Sciences, Professor, Department of Software, Vinnytsyia National Technical University, Vinnitsa, Ukraine
rom8591@gmail.com

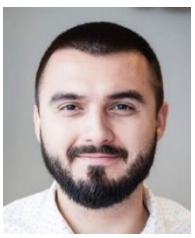
Олександр Н. Романюк – д-р техніч. наук, проф. каф. програмного забезпечення, Вінницький Національний технічний університет, Вінниця, Україна

Александр Н. Романюк – д-р технич. наук, проф. каф. программного обеспечения, Винницкий национальный технический университет, Винница, Украина

Sergey I. Vyatkin – Candidate of Technical Sciences, senior scientific researcher of Synthesizing Visualization Systems Laboratory at Institute of Automation and Electrometry, SB RAS, Novosibirsk, Russia
sivser@mail.ru

Сергій І. Вяткін – кандидат техніч. наук, ст.науч.сотр. лабораторії синтезуючих систем візуалізації, Інститут автоматики та електрометрії, СВ РАН, Новосибірськ, Росія

Сергей И. Вяткин – кандидат технич. наук, ст.науч.сотр. лаборатории синтези-рующих систем визуализации, Институт автоматики и электрометрии, СО РАН, Новосибирск, Россия



Pavlo I. Mykhaylov – CEO 3D GENERATION GmbH 90-92 (Germany), Dortmund, Germany
pm@3dgeneration.com

Павло І. Михайлов – генеральний директор 3D GNERATION GmbH (Німеччина), CEO 3D GNERATION GmbH, Дортмунд, Німеччина

Павел И. Михайлов – генеральный директор 3D GNERATION GmbH (Германия), CEO 3D GNERATION GmbH, Дортмунд, Германия

Roman Y. Chekhmestruk - Candidate of Technical Sciences, Technical Director 3D GENERATION UA, 3D GENERATION UA, Vinnitsa, Ukraine
Rc.ua@3dgeneration.com

Роман Ю. Чехместрук – кандидат техніч. наук, технічний директор 3D GENERATION UA, 3D GENERATION UA, Вінниця, Україна

Роман Ю. Чехместрук – кандидат технич. наук, технический директор 3D GENERATION UA, 3D GENERATION UA, Винница, Украина

Received 07.08.2020
Received after revision 12.09.2020
Accepted 21.09.2020

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)