**UDC 004.925.8**

**Olexandr N. Romanyuk**[1], Doctor of Technical Sciences, Professor, Department of Software, E-mail: rom8591@gmail.com, ORCID: 0000-0002-2245-3364
**Sergey I. Vyatkin**[2], Candidate of Technical Sciences, senior scientific researcher of Synthesizing Visualization Systems Laboratory, E-mail: sivser@mail.ru, ORCID: 0000-0002-1591-3588
**Svitlana G. Antoshchuk**[3], Doctor of Technical Sciences, Professor, Director of Institute of Computer System, E-mail: asgonpu@gmail.com, ORCID: 0000-0002-9346-145X
[1]Vinnytsia National Technical University, Khmelnytsky Hway, 95, Vinnytsia, Ukraine, 21021
[2]Institute of Automation and Electrometry, Siberian Branch of the Russian Academy of Sciences, Ac. Koptyuga Avenue, 1, Novosibirsk, Russia, 630090
[3]Odessa National Polytechnic University, Shevchenko Avenue, 1, Odessa, Ukraine, 65044

# 3D VECTOR FIELDS VISUALIZATION USING GRAPHICS PROCESSING UNITS

***Abstract.*** *The paper describes a method of visualization of three-dimensional vector fields adapted for GPUs. The aim of this work is to develop and implement a method of visualization of three-dimensional vector fields, effectively using GPUs. The software for visualization of three-dimensional vector field based on algorithms developed by the authors is created. This application provides visualization of three-dimensional vector fields through an interactively controlled animation sequence. The main criteria for evaluating the performance of visualization algorithms are the ease of interpretation and performance. The paper deals with the problems of adaptation of the computational model of vector field visualization algorithms to the implementation based on the GPU. Effective data representation for methods implemented based on vertex and pixel shaders of graphic processors is developed. The generalized model of calculations based on the graphic processor is offered. The program for interactive visualization of sections of a three-dimensional field of speeds by means of animation is created. A method of decomposition of a three-dimensional texture cube to represent a three-dimensional vector field is developed. All proposed algorithms are implemented in the form of software modules that can be used to build a visualization system. This paper describes a method of ray casting for visualizing three-dimensional vector fields. The distinguishing features of this method are the separation of the screen into cells (spans) and the pipelining of calculations using an intermediate description of the frame in the form of a list of primitives. Splitting calculations into two phases using an intermediate frame description allows achieving maximum performance at the stage of pixel calculations that require the most resources and determine the performance of the system as a whole. The advantages of such an approach over the frame-buffer visualization method are shown. The use of modern graphics equipment allows achieving the best results in terms of performance. Three-dimensional vector fields are used in scientific visualization, image processing and for special effects.*

***Keywords:*** *3D vector fields;, ray casting; scalar fields; rendering; visualization*

## Introduction

Progress in the field of numerical mathematical physics in recent years, due to the rapid growth of computer power, made it possible to carry out complex computational experiments of mathematical modeling of natural objects and phenomena. Very common in environmental studies and problems of hydrogasdynamics, a special case of such modeling is the calculation of the three-dimensional velocity field of the medium in a limited volume on a regular or irregular grid.

The results of numerical simulation are large arrays of vector and scalar data. The successful outcome of the experiment largely depends on how the data obtained will be presented to the researcher, and what conclusions he will and what conclusions he will make based on the resulting picture [1-5]. In this regard, the visualization of the velocity vector field becomes an integral part of many problems of numerical simulation. The stage of visualization of the results obtained during the computational experiment should provide the researcher with comprehensive information about the structure, flow rate and other characteristics.

The importance of the stage of visual interpretation of the numerical approximation of three-dimensional vector fields is often underestimated and is left to the standard mathematical packages that implement outdated visualization techniques, such as arrow diagrams or color-coding. Flow velocity vector with this method is displayed as an arrow, the direction and magnitude of which correspond to the value of the field at the point. The disadvantage of arrow diagrams is the low resolution when displaying a vector field. It is easy to see that in the visualization of vortex structures there is a mutual intersection of vector symbols, which leads to noise of the image and the impossibility of its interpretation.

Method of color-coding vector field puts in correspondence to each value of the velocity field, the value of the color space HSL (hue-saturation-brightness). The direction of the field determines the component of the color hue according to the color wheel model. Brightness and saturation are selected according to the value of the vector value at that point. The color coding method allows for real-time visualization, but the interpretation of the visualization results requires some mental effort. This circumstance makes the images obtained by this algorithm not available for understanding without additional mental effort.

At the same time, very little attention is paid to the methods of visualization of three-dimensional fields, while it is this family of algorithms that is the only approach that allows the user to transmit information about the vector field as a whole, i.e. almost at every point of the considered area. The images obtained by this family of methods continuously cover the entire visualized area of the vector field, as opposed to the arrow diagrams. Visualization does not require additional interpretation of the results and gives the researcher a clear and intuitive picture of the vector field, containing information about the structure and speed in different areas.

One of the features of this family of methods is the ability to implement graphics hardware. The hardware architecture of modern graphics processors has a complex structure that provides great opportunities for controlling the visualization of a three-dimensional scene. The presence of a graphics processor (GPU –Graphics Processing Unit) and fast SDRAM memory on the graphics card itself allows you to consider a personal computer as a dual-processor machine with shared memory. Such graphics systems allow flexible control of the pipeline rasterization of three-dimensional objects. This gives you the opportunity to completely redefine the operations and implement the methods of the visualization entirely on the GPU.

Implementation of vector field's visualization methods using GPU resources allows achieving high performance of the visualization system by transferring part or all stages of the algorithm to the GPU. The use of this approach makes it possible to visualize a three-dimensional unsteady vector field in an interactive mode.

**Three-dimensional vector fields**

Field theory is a branch of mathematics, but the concept of field underlies many concepts of modern physics. In general, it is said that in space a field of some value u is given, if at each point of space, the value of this value is determined. For example, in the study of the gas flow it is necessary to investigate several fields: the temperature field (at each point, the temperature has a certain value), the pressure field, the velocity field and other fields. The field of magnitude u is called stationary (or steady), if u does not depend on the time t. Otherwise, the field is called non-stationary (or unsteady). Thus, the value *u* is a function of the point *M* and time *t.* There are two types of fields: scalar and vector.

Let *D* be some area on the plane or in space. If in the *D* region of each point *M(x,y,z)* of space or point *M(x,y)* of the plane at each time *t,* by a certain law, the value of the scalar quantity u is put in correspondence. Then the function *u(x,y,z,t)* in the case of space or u (x, y, t) in the case of the plane is called a scalar field. The concepts of the scalar field and the function defined in area D are the same. An example of a scalar field can be the field of air temperature in a room, if the temperature is considered as a function of the point. At points closer to the heat source, the temperature is higher than at points farther from the heat source. We can give examples such as the field of illumination, the field of mass density and the like. To get a more complete picture of the scalar field, its graphical image is used - the level surface in space and the level line on the plane. A level line is widely used in the preparation of topographic and weather maps. On topographic maps, the level line is the line at which points the same height above sea level is marked. On meteorological maps, two types of level lines are built – isotherms (lines of the same temperature) and isobars (lines of the same pressure).

A vector field is a map that maps each point in the space under consideration to a vector with a beginning at that point. For example, the wind velocity vector at a given time is different at different points and can be described by a vector field. Examples of a vector field are velocity and acceleration fields in a flowing liquid or gas, a gravitational force field, an electrostatic intensity field, and the like. In general, an example of a vector field can serve as a field of forces of any nature.

If in a certain area of space each point *M* is put in accordance with a certain law vector $\vec{v}$ , the vector function $\vec{v}(M)$ is called a vector field or vector field. Thus, a vector field is a vector function of a space point

$$\vec{v} = \vec{v}(M) = \vec{v}(x, y, z) .$$

Examples of a vector field are velocity and acceleration fields in a flowing liquid or gas, a


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

gravitational force field, an electrostatic intensity field, and the like. In general, an example of a vector field can serve as a field of forces of any nature. The projection of the vector $\vec{v}(M)$ corresponding to the point *M* on the coordinate axes will be denoted by *P = P(x,y,z), Q = Q(x,y,z), R = R(x,y,z).* Then we can define the vector field through the components:

$$\vec{v} = P\vec{i} + Q\vec{j} + R\vec{k}.$$

Thus, a vector field, you can define three scalar functions of *P, Q, R*. Let the functions and their partial derivatives in the variables *x, y, z* are continuous functions. That is, each point of space is mapped to a vector (the value of the vector field at a given point in space). This vector differs for different points in space, that is, the vector field takes different values at different points in space. At each point in space, the field vector has a certain value and certain, except when the field turns to zero, direction in this space.

Imaging 3D vector fields has applications primarily in scientific visualization [6-17]. Much of the shape is a function of directional information as well. Sculptors, computer graphics researchers have recognized the importance of direction in process of image creation and shape. Therefore, methods that can image directional information have wide application across both scientific and artistic areas.

At the present stage of development of computer graphics, graphics performance sufficient for most applications achieved by improving the components and manufacturing technology of video cards and by increasing the number of processor units and their clock frequency. However, the main approaches to the creation of photorealistic images remain unchanged and do not meet the requirements of many applications of three-dimensional graphics, and this contradiction cannot be resolved without developing a new theoretical basis.

These drawbacks eliminated by means of analytical definition of objects and their rasterization using ray-tracing algorithms. Analytical definition of geometric objects does not require large amounts of memory. There are works on visualization of functionally defined surfaces, such as convolution surfaces [18], implicitly defined surfaces, known in computer graphics as blobby models [19], metaspheres [20], soft objects [21], etc. However, their use is limited to a rather narrow class of modeled surfaces and slow visualization. The employed algorithms are difficult to optimize for real-time visualization, and this imposes restrictions on their practical application.

One of the main disadvantages of the known visualization methods is complexity of the calculation of points on the surface [22]. Thus, the ray marching method does not guarantee detecting the surface, and, in addition, it is slow [23]. The method of determining the intersection of a ray with an implicitly defined surface is too complex to calculate the *L*- and *G*-parameters [24]. In the tracing method, finding the maximum radius when no point of the volume lies within the sphere is a nontrivial task [25]. Ray tracing with analysis of the interval for complex functions requires individual calculations for each ray and each interval along this ray [26]. In fast tracing, search for the rays intersecting the surface requires many calculations and is not efficient enough as the clustering procedures of this method do not solve this problem completely.

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 guaranteed. Another disadvantage of this method is that transformation of objects to another coordinate system is a complex task. Therefore, the creation of dynamic scenes is problematic.

There is another technique for the visualization of analytically defined objects using graphics processing units. This technology based on conventional single-step tracing of rays. A distinctive feature of this method is that the step size is not constant but chosen in each iteration. 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** is to develop a method of visualization of three-dimensional vector fields, effectively using GPUs, and to develop software for visualization of three-dimensional vector fields through an interactively controlled animation sequence.

## Method of visualization of three-dimensional vector fields

We developed approach for visualizing 3D vector fields using graphics processing units. Visualization of three-dimensional vector fields is necessary when it is necessary to show the structure of the flow, to display three-dimensional datasets comparable to images obtained experimentally. Tools to solve such problems are indispensable in computational Aero - and hydrodynamics.

A vector field on Euclidean space is defined as a vector-function of the point of space that maps this space on itself. That is, each point of space is mapped to a vector, the value of the vector field at a given point in space. This vector differs for different points in space, that is, the vector field takes different values at different points in space. At each point in space, the field vector has a certain value and a certain (except when the field turns to zero) direction in this space.

The local behavior of the 3D vector field approximated by computing a local streamline that starts at the center of voxel *(x, y, z*) and moves out in the directions:

$$P_0 = (x + 0.5, y + 0.5, z + 0.5) \quad , \qquad (1)$$

$$P_i = P_{i-1} + V(\lfloor P_{i-1} \rfloor)\Delta s_{i-1} \Big/ \left\| V(\lfloor P_{i-1} \rfloor) \right\|, \quad (2)$$

where: $V(\lfloor P \rfloor)$ the vector from the input vector field at lattice voxel is $(\lfloor P_x \rfloor, \lfloor P_y \rfloor, \lfloor P_z \rfloor)$ , $\Delta s_i$ is the distance to the vector field from $P_i$ to the nearest cell face.

The local streamline adverted backwards by the negative of the vector field as well

$$P_0' = P_0,$$
$$P_i' = P'_{i-1} - V(\lfloor P'_{i-1} \rfloor)\Delta s'_{i-1} \Big/ \left\| V(\lfloor P'_{i-1} \rfloor) \right\|. \quad (3)$$

The entire integral convolution for output voxel $F'(x, y, z)$ is given

$$F'(x, y, z) = \frac{\sum_{i=0}^{l} F(\lfloor P_i \rfloor) h_i + \sum_{i=0}^{l'} F(\lfloor P'_i \rfloor) h'_i}{(\sum_{i=0}^{l} h_i + \sum_{i=0}^{l'} h')_i} \quad , \quad (4)$$

where: $F(\lfloor P \rfloor)$ is the input voxel corresponding to the vector at position $(\lfloor P_x \rfloor, \lfloor P_y \rfloor, \lfloor P_z \rfloor)$ , $l = i$ such that $s_i \le L < s_{i+1}$ .

The numerator represents the line integral of the filter kernel times the input voxel field, *F*.

**Ray casting**

Using equations (2), (3) and (4) can be calculated cell faces. Input vector field and input texture are 3D. The output of the 3D algorithm is a scalar field. This field is rendered using two ways.

The first method is as follows [27]. We construct a 3D surface *F*, which is a graph of the function determined in the 3D space *z = f (x, y).* Definition of a 3D surface based on a scalar field is a set of some base surface P located in the same coordinate system as F and related to the base surface P of the height map. The height map is a two-dimensional rectangle, which is called the domain of perturbation DP of the base surface P inside which the perturbation function h(u, v) is specified. In turn, the height map defines the perturbation proper. The domain of definition of the function *h(u, v)* is *Dh(u,v) = {U, V }*, where *U* and *V* are the rectangle sizes. The height map is related to the base surface in the following way: there exists a transformation $G(R3 \Rightarrow R2)$ from the coordinate system containing *F* and *P* to the coordinate system of the height map. Most often, this transformation is parallel projection.

The value of the function $h\left(G\left(dF\right)\right)$ on the surface *F* from the point *dP* , which is the projection of this point onto the surface *P*. In other words, the value of the function $h\left(G\left(dF\right)\right)$ is equal to the absolute value of the vector

$$v = \left(dF - dP\right). \qquad (5)$$

Therefore, the domain of the complex surface can be defined as a set of points in R3 determined by the vector equation

$$\mathrm{F} = G\left(v\right) + \mathrm{n}h\left(G\left(v\right)\right); \forall v \in R3 \, , \qquad (6)$$

where: n is the normal to the base surface.

If *v* is located outside the perturbation domain, then the vector $h\left(G\left(dF\right)\right)$ is equal to zero, and the vector *F* is a vector on the base surface.

Thus, it is possible to use the table of numbers for defining the form of the perturbing surface and the function of interpolation over the nodal values taken from the table as the function h. In this case, one can argue that a scalar field is defined in the perturbation domain *DP*.

Considering the structure of the geometric model, the 3D surface is formed by the base surface of the second order and the perturbation function, which is defined in an infinitely long parallelepiped. The values of the perturbation function are given in the cross section of the parallelepiped by a two-dimensional height table. As the base surface can be used plane, then the direction of the normal of the carrier plane must coincide with the longitudinal direction of the parallelepiped - the area of determination of the perturbation function.


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

To implement the perturbation function of the landscape type, it is necessary to create another object of the geometric model, the initial data for which is a sample of values located at the nodes of a rectangular grid, obtained based on measurements of the heights of the real area relative to sea level. Since during rasterization the perturbation must evaluate the maximum of its function on a three-dimensional or one-dimensional interval, the maps of the levels of detail are pre-compiled for the efficiency of calculations. The source data is level n if the mesh dimension is $2n \times 2n$. The data for level $n-1$ is obtained by selecting the maximum of four adjacent values of level $n$, the other three are not taken into account further, and i.e. we obtain the grid dimension $2n-1 \times 2n-1$. Level 0 consists of a single value-the maximum throughout the elevation map. When determining the maximum perturbation, the characteristic projection size of the current interval is calculated, based on which the level of detail is selected.

For a larger interval, a coarser approximation of the original function is chosen accordingly. If a more accurate representation is required than is available, bilinear or bucolic interpolation of the elevation values that make up the last level of detail is performed. This approach allows you to reduce the number of calculations by increasing the amount of memory to store additional data and, mainly, by dynamically adjusting the complexity of calculations by the criterion of the required accuracy of the result.

The second method is as follows. We convert the scalar field to a functionally defined analytical description [28].

The essence of the method implies the transformation of scalar field to a functional description based on perturbation functions. The transformation is ensured by means of rigorous mathematical calculations rather than by iterative or other approximate methods, which lead to partial loss of information. After conversion, visualize functionally specified description.

This task is similar to visualization in volumetric tomography, where the density function defined in the form of discrete data. For ease of understanding, we assume that the scene is in a unit three-dimensional cube (Fig. 1). Perspective not analyzed because 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 limited by the front and rear faces of the cube.

In the search for the points of intersection of the ray and the object, each ray subdivided along the Z-axis to form a set of voxels. Thus, we obtain a density function along the ray, which depends on one variable.

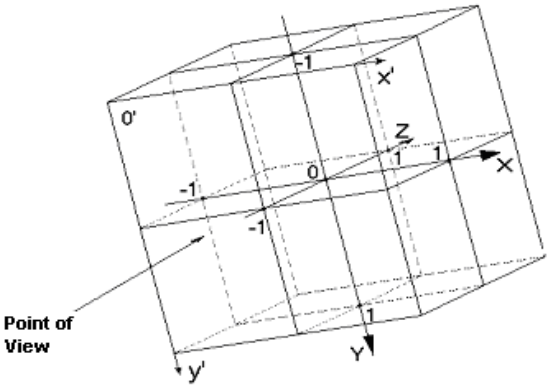


Fig 1. Unit cube

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 defined at each pixel. In the presence of all the coordinates and normals at each pixel, the local illumination model is used.

To increase performance, we offer virtual buffer imaging technology. Binary search for the image elements of functionally defined objects, related to the so-called frame buffer visualization method, was proposed in [22]. The frame buffer method presupposes the storage and modification of each image pixel in the frame memory and rasterizes the primitives in the random access memory. The proposed virtual visualization technology, i.e., the virtual buffer method is the rasterization of primitives into intermediate portions of the screen memory and the reuse of these portions for constructing a full frame.

The main drawback of this method [28] is that there are many superfluous calculations in the scene carried out for small and medium-sized objects. This is directly due to the architecture of graphics processing units as conditional transitions are very time-consuming operations, instead of which parallel data processing was used. In order for the program to work effectively, one should account for the peculiarities of hardware. Lately, an increase in the processing power occurs specifically because of parallel computations. We propose an increase in the visualization performance by increasing the transport delay, which is the main disadvantage of virtual visualization technology [22].

The rasterization process is divided into two stages and distributed between the central processor and the graphics processing unit. The central processor divides the object space according to a quaternary tree. In the search algorithm, the cube is divided into smaller parts, for which the object intersection test is performed. The division process takes two steps. The first step is when the cube is divided into four parts in the plane XY. Then each part is considered separately. If there is no intersection with a given object, then this part is neglected further on. If there is intersection with other parts, the same division procedure is repeated. In the general case, the process is finished when the part under consideration corresponds to a tile of a certain size. The advantage of this approach is that, at an early stage, one can neglect the larger parts of the cube that do not have a given object. The primitives of the intermediate description are the fragments where geometric objects intersect with spans.

The second step of computations includes processing the list of objects and determining the visibility and pixel color, both of which are carried out by the graphics processing unit. The fragments of the object are input to the graphics processing unit. Then the fragment is tested for intersection with a ray directed along the Z-axis, and the binary search for the nearest intersection point of the ray with the object is carried out [22]. The problem is to find the first point at which the function vanishes. Having determined such a point for each ray, one can determine the z coordinate. Then the normal is determined in each pixel.

Having all coordinates and normal's in each pixel, one can apply a local lighting model. The result is an image of a smooth object with account for lighting.

The visualization time is reduced by the effective use of the computational resources of the graphics processing unit with CUDA architecture. The method was implemented with account for the influence of the speed of operation with memory. Registers are used to a maximum degree, and memory is used jointly. In all other cases, the processors work with the general memory of the graphics processing unit. Fig. 2 shows the results of testing the dependence of the frame calculation time on the number of perturbation functions for a particular test of two methods are shown (Table 1). On the abscissa axis – test numbers, on the ordinate axis – average time per frame (in seconds). The chart shows that on average, the calculation time has decreased by an order of magnitude. At the same time, the transport delay doubled. Note that the performance has increased not only for medium-sized and small objects, but also for large objects with a large number of disturbances. As the object is divided into cells and the number of perturbations of the fragments decreases.

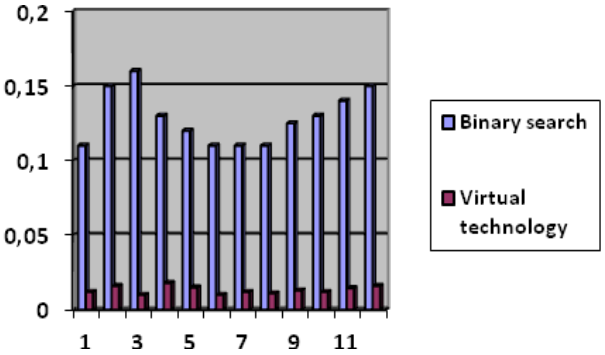


Fig 2. The test results of the two methods of visualization

Table 1. Distribution of the number of perturbations for the frame objects in different tests

| Test No. | Number of objects in the frame | Types of perturbations | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 1 | 4 | - | - | - | - | - | - |
| 2 | 22 | 4 | 4 | 4 | 1 | 1 | 4 | 3 |
| 3 | 22 | 4 | 4 | 4 | 3 | 3 | 1 | 3 |
| 4 | 3 | 1 | 1 | 3 | - | - | - | - |
| 5 | 3 | 1 | 1 | 4 | - | - | - | - |
| 6 | 10 | 2 | - | - | - | - | - | - |
| 7 | 1 | 4 | - | - | - | - | - | - |
| 8 | 1 | 5 | - | - | - | - | - | - |
| 9 | 1 | 6 | - | - | - | - | - | - |
| 10 | 1 | 7 | - | - | - | - | - | - |
| 11 | 1 | 8 | - | - | - | - | - | - |
| 12 | 1 | 9 | - | - | - | - | - | - |

The functions of the graphics processing unit included calculating the coordinates of the surface points, normals, and lighting. The central processor carried out the geometric transformations, rasterized primitives in the span grid, and formed a list of fragments with determination of all the necessary parameters. For the visualization, the DirectX applied programming interface was used. The visualization time 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


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

on different processing units is the same even if they may have a different number of streaming multiprocessors. A large number of computer processors allow parallel check of the intersection of several rays with the object simultaneously. Most of the graphics processing units that support CUDA have not less than 128 scalar cores. Therefore, a large portion of the cube will have computed in parallel. The architecture of graphics processing units based on many streaming multiprocessors with shared memory access for reading and writing. Each of the streaming processors contains eight scalar cores and a set of on-chip memory of four types. The number of registers may be 8192 or 16384, depending on the computational capabilities of the processing unit. The shared memory is 16 KB for each multiprocessor.

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

Among the functions of the graphics, processing unit was to calculate the coordinates of points of the surfaces, normal's, and illumination. Geometric transformations performed by the central processing unit (CPU). The DirectX application-programming interface used for visualization. Testing performed on GT 470 GTX processors. The tests showed the possibility of interactive visualization of three-dimensional vector fields. Figure 3 shows a three-dimensional vector field.

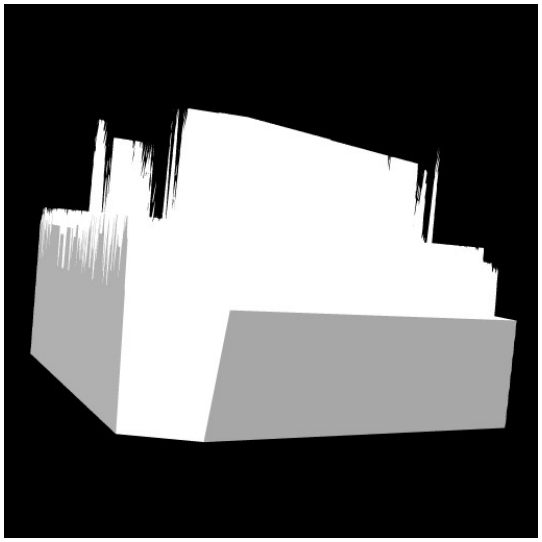

Fig 3. 3D vector field

**Conclusions**

This paper proposes method of visualization of three-dimensional vector fields with effectively using GPUs. We proposed an effective method for rasterization (search for the image elements of the surface), which comes down to dividing the screen into spans and pipelining the computations by means of intermediate description of the frame in the form of a list of primitives. The division of computations into two steps with the use of intermediate description of the frame makes it possible to reach maximum performance at the step of pixel computations that require the greatest resources and determine the system performance in general.

The method of defining 3D vector fields and the visualization method proposed in this paper have advantages over existing approaches. Both the input vector field and input texture are three-dimensional. The output of the rendering is a 3D image or scalar field.

The main advantages of the proposed method of specifying the field and methods of their visualization include: the simplicity of calculating the volume points with a quick search and rejection of areas not occupied by the volume of the scene.

3D vector field definition is especially important in a number of computer graphics problems, for example, in scientific visualization.

This paper is intended for researchers with interest in the possibility of making the flow or the process visible. In physics, typical examples are force fields, force field is the field of some force, depending on the position in the space of the body on which this force acts, or closely related to the force of the field strength. Other typical examples are velocity field, for example, liquid or gas flow velocity, displacement field, in a deformed elastic medium, current density vector, energy flow vector or flow density vector of some material particles, in diffusion, temperature gradient vector, concentration or pressure vector, etc.

**References**

1. Masson, M. E. J., & Loftus, G. R. (2003). "Using Confidence Intervals for Graphically Based Data Interpretation," *Canadian Experimental Psychology*, Vol. 57, No. 3, pp. 203-220.

2. Ware, C. (2012). Information Visualization: Perception for Design. 3rd Edition. N.-Y., *Morgan Kaufmann Publishers*, 537 p.

3. Kosara, R., Healey, C. G., Interrante, V., Laidlaw, D. H., & Ware, C. (2003). "Thoughts on User Studies: Why, How, and When", *Computer*

*Graphics and Applications*, Vol. 23, No. 4, July/Aug. 2003, pp. 20-25.

4. Kosara, R., Miksch, S., Hauser, H., Schrammel, J., Giller, V., & Tscheligi, M. (2002). "Useful Properties of Semantic Depth of Field for Better f+c Visualization", Proc. Joint Eurographics-IEEE TCVG *Symposium. IEEE Trans. Visualization and Computer Graphics*, Vol. 3, No. 2, Visualization 2002 (VisSym '02), pp. 205-210.

5. Swan, J. E., Gabbard, J. L., Hix, D., Schulman, R. S., Kim, K. P. (2003). A Comparative Study of User Performance in a Map-Based Virtual Environment, *IEEE Virtual Reality* 2003, Mar. 2003, pp. 259-266. **DOI:** 10.1109/VR.2003.1191149.

6. Cabral, B., & Leedom, L. C. (1993). "Imaging Vector Fields Using Line Integral Convolution," *Computer Graphics, SIGGRAPH '93 Proc.*, Vol. 27, Aug. 1993, pp. 263-272.

7. Cabral, B., & Leedom, L. C. (1995). "Highly parallel vector visualization using line integral convolution". *In Seventh SIAM Conference on Parallel Processing for Scientific Computing*. Feb. 1995. pp. 802-807.

8. Turk, G., & Banks, D. (1996). Image-Guided Streamline Placement. *Proc. SIGGRAPH'96*, pp. 453-460.

9. Globus, A., Levit, C., & Lasinski, T. A. (1991). "Tool for Visualizing the Topology of Three-Dimensional Vector Fields", *Proc. Visualization'91*, pp. 33-40.

10. Maxwell, S. E., & Delaney, H. D. (1990). "Designing Experiments and Analyzing Data: A Model Comparison Perspective", Belmont, California: Wadsworth.

11. Wainer, H., & Thissen, D. (1993). Graphical data analysis. In: G. Keren, & C. Lewis, (Eds.). "A handbook for data analysis in the behavioral sciences: Statistical issues". Hillsdale, NJ: Lawrence Erlbaum, pp. 391-457

12. Tufte, E. (1983). "The Visual Display of Quantitative Information". *Graphics Press*.

13. Cleveland, W. S. (1985). The Elements of Graphing Data. Wadsworth.

14. Loftus, G. R., & Masson, M. E. J. (1994). "Using Confidence Intervals in Within-Subject Designs". *Psychonomic Bulletin and Rev.*, Vol. 1, pp. 476-490.

15. Bancroft, G. V., Merritt, F. J., Plessel, T. C. Kelaita, P. G., McCabe, R. K., & Globus, A. (1990). "FAST: A multiprocessed environment for visualization of computational fluid dynamics". In Proc. of Visualization 90, pp. 14-27,

16. Bryson, S., & Levit,t C. (1991). "The virtual windtunnel: An environment for the exploration of three-dimensional unsteady flows". In Visualization'91, pp. 17-24.

17. Jobard, Bruno & Erlebacher, Gordon & Hussaini, Mohammed. (2002). "Lagrangian-eulerian advection of noise and dye textures for unsteady flow visualization", *IEEE Transactions on Visualization and Computer Graphics*, 8, 3, 2002, pp. 211-222. DOI: 10.1109/TVCG.2002.1021575.

18. (2011). Pizaine, G., et. al. "Vessel geometry modeling and segmentation are using convolution surfaces and an implicit medial axis". In: Biomedical Imaging: From Nano to Macro, *IEEE International Symposium on. IEEE*, 2011, pp. 1421-1424. **DOI:** 10.1109/ISBI.2011.5872666.

19. Muraki, S. (1991). "Volumetric Shape Description of Range Data using "Blobby Model"", *Computer Graphics* 25 (4), pp. 227-235.

20. Nishimura, H., Hirai, M., & Kawai, T. (1985). "Object Modeling by Distribution Function and a Method of Image Generation". *Trans. Institute of Electronics and Communication Engineers of Japan* J68-D (4), pp. 718-725.

21. Wyvill, G., McPheeters, C., & Wyvill, B. (1986). "Data Structure for Soft Objects". *The Visual Comput.* 2 (4), pp. 227-234.

22. Vyatkin, S., Romanyuk, A., & Savitska, L. (2017). "Multi-level ray casting of function-based surfaces", *Journal of Physics: Conference Series*, 803, No. 1. DOI:10.1088/1742-6596/803/1/012180.

23. Tuy, H., & Tuy, L. (1984). "Direct 2-D Display of 3-D Objects". *IEEE Computer Graphics and Application*. 4 (10), pp. 29-33.

24. Karla, D., & Barr, A. H. (1989). "Guaranteed Ray Intersections with Implicit Surfaces". *Computer Graphics* 23 (3), pp. 297-306.

25. Hart, J. C. (1994). "Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces", *The Visual Computer* 12 (10), pp. 527-545.

26. Mitchel, D. (1990). "Robust Ray Intersection with Interval Arithmetic". *In Proc. on Graphics Interface, Toronto:* Canadian Information Processing Society, pp. 68-74.

27. Vyatkin, S. I. (2007). "Complex Surface Modeling Using Perturbation Functions". [Optoelectronics, Instrumentation, and Data Processing. 43 (3), 226–231]. Article in Optoelectronics Instrumentation and Data Processing 43(3):226-231 · June, 2007. DOI: 10.3103/S875669900703003X

28. Vyatkin, S. I., & Dolgovesov, B. S. (2018). "Compression of Geometric Data with the Use of Perturbation Functions". *Optoelectronics, Instrumentation and Data Processing*, Vol. 54, No. 4, pp. 1-7. Vyatkin, S. I., & Dolgovesov, B. S. Optoelectron. Instrument. Proc. (2018) 54: 334. https://doi.org/10.3103/S8756699018040039.

Received 30.05.2019


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

**УДК 004.925.8**

[1]**Романюк, Олександр Никифорович,** доктор техніч. наук, професор, професор кафедри програмного забезпечення, E-mail: rom8591@gmail.com, ORCID: 0000-0002-2245-3364
[2]**Вяткін, Сергій Іванович,** кандидат техніч. наук, старший науковий співпрацівник лабораторії синтезуючих систем візуалізації, E-mail: sivser@mail.ru, ORCID: 0000-0002-1591-3588
[3]**Антощук, Світлана Григорівна,** доктор техніч. наук, професор, директор інституту комп'ютерних систем, E-mail: asgonpu@gmail.com, ORCID: 0000-0002-9346-145X
[1]Вінницький національний технічний університет, Хмельницьке шосе,95, м. Вінниця, 21021, Україна
[2]Інститут автоматики та електрометрії Сибірського відділення Російської академії наук, проспект Ак. Коптюга, 1, м. Новосибірськ, 630090, Російська Федерація
[3]Одеський національний політехнічний університет, проспект Шевченка, 1, м. Одеса, Україна, 65044

# ВІЗУАЛІЗАЦІЯ 3D-ВЕКТОРНИХ ПОЛІВ З ВИКОРИСТАННЯМ ГРАФІЧНИХ ПРОЦЕСОРІВ

***Анотація**. У статті описаний метод візуалізації тривимірних векторних полів, адаптованих для графічних процесорів. Метою даної роботи є розробка та реалізація методу візуалізації тривимірних векторних полів з ефективним використанням GPU. Створено програмне забезпечення для візуалізації тривимірного векторного поля на основі алгоритмів, розроблених авторами. Програма забезпечує візуалізацію тривимірних векторних полів через інтерактивно керовану послідовність анімації. Основними критеріями оцінки продуктивності алгоритмів візуалізації є простота інтерпретації та продуктивності. У статті розглянуто проблеми адаптації обчислювальної моделі алгоритмів візуалізації векторного поля до реалізації на основі GPU. Розроблено ефективне представлення даних для методів, що реалізуються на основі вершинного та піксельного шейдерів графічних процесорів. Запропоновано узагальнену модель обчислень на основі графічного процесора. Створена програма для інтерактивної візуалізації ділянок тривимірного поля швидкостей за допомогою анімації. Розроблено метод декомпозиції тривимірного текстурного куба для зображення тривимірного векторного поля. Всі запропоновані алгоритми реалізовані у вигляді програмних модулів, які можна використовувати для побудови системи візуалізації. У роботі описаний метод лиття рейкастингу для візуалізації тривимірних векторних полів. Відмінними особливостями цього методу є поділ екрана на комірки (проміжки) та конвеєрне обчислення за допомогою проміжного опису кадру у вигляді списку примітивів. Розбиття обчислень на дві фази з використанням проміжного опису кадру дозволяє досягти максимальної продуктивності на етапі піксельних обчислень, які потребують найбільше ресурсів, та визначити продуктивність системи в цілому. Показано переваги такого підходу над методом візуалізації кадру-буфера. Використання сучасної графічної техніки дозволяє досягти найкращих результатів у плані продуктивності. Тривимірні векторні поля використовуються для наукової візуалізації, обробки зображень та для спеціальних ефектів.*

***Ключові слова:** 3D векторні поля; рей кастинг; скалярні поля; рендерінг; візуалізація*

**УДК 004.925.8**

[1]**Романюк, Александр Никифорович**, доктор технич. наук, профессор кафедры программного обеспечения, E-mail: rom8591@gmail.com, ORCID: 0000-0002-2245-3364
[2]**Вяткин, Сергей Иванович**, кандидат технич. наук, старший научный сотрудник лаборатории синтезирующих систем визуализации, E-mail: sivser@mail.ru, ORCID: 0000-0002-1591-3588
[3]**Антощук, Светлана Григорьевна**, доктор технич. наук, директор института компьютерных систем, E-mail: asgonpu@gmail.com, ORCID: 0000-0002-9346-145X
[1]Винницкий национальный технический университет, Хмельницкое шоссе, 95, г. Винница, Украина, 21021
[2]Институт автоматики и электрометрии Сибирского отделения Российской академии наук, проспект Ак. Коптюга, 1, г. Новосибирск, Российская Федерация, 630090
[3]Одеський национальный политехнический университет, проспект Шевченко, 1, г. Одесса, Украина, 65044

# ВИЗУАЛИЗАЦИЯ 3D-ВЕКТОРНЫХ ПОЛЕЙ С ИСПОЛЬЗОВАНИЕМ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ

***Аннотация***. *В статье описан метод визуализации трехмерных векторных полей, адаптированных для графических процессоров. Целью данной работы является разработка и реализация метода визуализации трехмерных векторных полей с эффективным использованием GPU. Создано программное обеспечение для визуализации трехмерного векторного поля на основе алгоритмов, разработанных авторами. Программа обеспечивает визуализацию трехмерных векторных полей через интерактивно управляемую последовательность анимации. Основными критериями оценки производительности алгоритмов визуализации является простота интерпретации и производительности. В статье рассмотрены проблемы адаптации вычислительной модели алгоритмов визуализации векторного поля к реализации на основе GPU. Разработан эффективное представление данных для методов, реализуемых на основе вершинного и пиксельного шейдеров графических процессоров. Предложена обобщенная модель вычислений на основе графического процессора. Создана программа для интерактивной визуализации участков трехмерного поля скоростей с помощью анимации. Разработан метод декомпозиции трехмерного текстурной куба для изображения трехмерного векторного поля. Все предложенные алгоритмы реализованы в виде программных модулей, которые можно использовать для построения системы визуализации. В этой работе описан метод литья лучей для визуализации трехмерных векторных полей. Отличительными особенностями этого метода является разделение экрана на ячейки (промежутки) и конвейерное вычисления с помощью промежуточного описания кадра в виде списка примитивов. Разбивка вычислений на две фазы с использованием промежуточного описания кадра позволяет достичь максимальной производительности на этапе пиксельных вычислений, требующих больше ресурсов, и определить производительность системы в целом. Показаны преимущества такого подхода над методом визуализации кадра-буфера. Использование современной графической техники позволяет достичь лучших результатов в плане производительности. Трехмерные векторные поля используются для научной визуализации, обработки изображений и для специальных эффектов.*

***Ключевые слова***: *3D векторные поля; рей кастинг; скалярные поля; рендеринг; визуализация*

**Olexandr Nikiforovich Romanyuk**, Doctor of Technical Sciences, Professor,
*Research field*: formation and processing of graphic images

**Sergey Ivanovich Vyatkin**, Candidate of Technical Sciences, senior scientific researcher
*Research field:* information support of technological processes

**Svitlana Grigorivna Antoshchuk**, Doctor of Technical Sciences, Professor,
*Research field:* formation and processing of graphic images


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