Scientific Visualization, 2023, volume 15, number 1, pages 60 - 70, DOI: 10.26583/sv.15.1.05

Visualization of Geometric Models of Faceted Solids in Point Calculus

E.V. Konopatskiy[1,A], K.V. Ryabinin[2,B], A.A. Bezditnyi[3,C]

[A] Nizhny Novgorod State University of Architecture and Civil Engineering
[B] Perm State University
[C] Plekhanov Russian University of Economics Sevastopol branch

[1] ORCID: 0000-0003-4798-7458, e.v.konopatskiy@mail.ru
[2] ORCID: 0000-0002-8353-7641, kostya.ryabinin@gmail.com
[3] ORCID: 0000-0003-0528-9731, bezdytniy@gmail.com

Abstract

The paper considers the case of faceted solids and discusses visualisation of geometric solids in the form of a three-parameter set of points which belongs to a three-dimensional space. To visualize geometric solids, taking advantage of the modern GPU hardware acceleration, the Ray marching method is used. The implementation considers the definition of a signed distance function, which is reduced to the task of determining the set of intersection points of the projection rays with the rendered geometric solid. After that, for each pixel of the screen, its color is determined in accordance with whether the ray passes through the geometric solid or not. The analytical description of geometric solids and the solution of their intersection problem with projecting rays are solved within the framework of the point calculus mathematical apparatus. As a result, it was concluded that the proposed approach justifies itself, providing high rendering performance and the complete absence of visual artifacts when rendering faceted solids.

Keywords: solid-state modeling, geometric solid, point calculus, visualization, Ray marching, distance sign function, three-parameter set of points.

1. Introduction

The paper [1] formulated the problem of visualization of solid models presented as a three-parameter set of points in three-dimensional space. The work introduces a new concept of defining geometric solids in the form of a three-parameter set of points in three-dimensional space [2, 3]. The analysis of existing approaches to the visualization of a three-parameter set of points, which are described by a parametric equations system, carried out in [1], showed the absence of software solutions which an implement the visualization of three-dimensional solids obtained in accordance with the new concept. Given the analytical nature of the solid modeling engine to provide faster rendering on existing GPUs, it was decided to use the Ray marching method, which is used to render scenes in real time. The Ray marching method [4-6], by analogy with the Ray tracing method [7-10], has found wide application in solving a variety of scientific visualization problems in domestic and foreign practice. The variety of tasks, in turn, gave rise to many options for real-time visual representation of scenes. But in most cases, the task is reduced to determining the function of intersection of the rendered object with the ray and determining the signed distance function [11-13]. Based on the fact that geometric solids in the form of a three-parameter set of points in three-dimensional space are parametrized in point calculus [14-15], to determine the distance function, it is necessary to solve the problem of intersection of geometric solids with projecting rays in point calculus.

2. Determining the Intersection Point of a Ray with a Geometric Solid

In point calculus, the projecting ray is conveniently defined as a straight-line segment. Then the problem is reduced to determining the point of intersection of a straight-line segment with a geometric solid. A line is uniquely defined by two points. Then the beam of projecting rays can be specified by fixing one point in space, and the second one can be represented as the current point of the picture plane.

The point equation of a straight-line segment in point calculus has the following form:

$$N = P\overline{t} + Qt,$$

where P and Q are the points through which the line passes, which are determined by their coordinates;

t is the parameter;

$\overline{t} = 1 - t$ is the parameter's addition to 1.

Such a parametrization of the line ensures its passage through the point P at $t = 0$ and through the point Q at $t = 1$. If parameter $t \in (-\infty; +\infty)$, then we get an infinite line passing through the points P and Q. And for $t \in [0;1]$ we get the segment PQ. This feature can be used to determine the points of intersection of a ray with a geometric solid, if you place one point in front of the geometric solid, and the second behind it. Then the intersection points of the ray with the geometric solid will fall within the range of the parameter values $t \in [0;1]$. To determine a beam of rays, it is necessary to place a fixed point on one side of the geometric solid, and the current one, belonging to the plane, on the other side. Such a plane can be defined using any three points that do not lie on the same straight line. For example, forming a beam of rays passing through a point P (fig. 1), plane $A_1A_2A_3$ can be set by relations on the sides of the triangular simplex:

$$Q = A_1(1 - \lambda - \mu) + A_2\lambda + A_3\mu,$$

where λ and μ are the current parameters that determine the position of the point Q in plane $A_1A_2A_3$. Within the interior of a parallelogram formed on the basis of a $A_1A_2A_3$ triangle, these parameters vary from 0 to 1, but in total they can belong to the interval $(-\infty; +\infty)$.

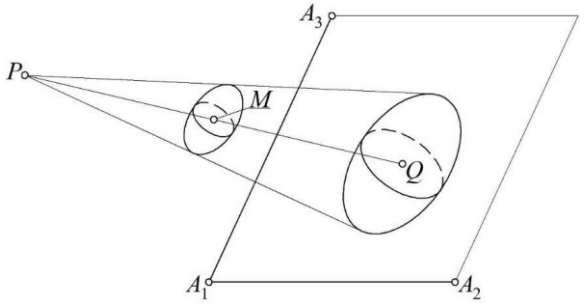

Fig. 1. Geometric scheme for determining a beam of rays in three-dimensional space

In Fig. 1, the current M point shows some geometric solid. In general, the point equation of a three-dimensional solid is determined by the following point equation:

$$M = Ap(u,v,w) + Bq(u,v,w) + Cr(u,v,w) + Ds(u,v,w),$$

where A, B, C and D are the points of a three-dimensional simplex (any four points that do not lie in the same plane);

$p(u,v,w)$, $q(u,v,w)$, $r(u,v,w)$ and $s(u,v,w)$ any continuous and differentiable functions of parameters u, v, w.

The condition for the current M point to belong to the space of a three-dimensional $ABCD$ simplex is:

$$p(u,v,w)+q(u,v,w)+r(u,v,w)+s(u,v,w)=1.$$

Based on this, any of the four free functions of u, v, w parameters can be excluded.

At the intersection of a straight-line segment and a geometric solid, for all common points, the point equation will be valid:

$$P\bar{t}+Qt=Ap(u,v,w)+Bq(u,v,w)+Cr(u,v,w)+Ds(u,v,w).$$

To determine the u, v, w and t parameters it is necessary to perform a coordinate-by-coordinate calculation [14]. For three-dimensional space we can formulate the following equation system:

$$\begin{cases} x_P\bar{t}+x_Qt=x_Ap(u,v,w)+x_Bq(u,v,w)+x_Cr(u,v,w)+x_Ds(u,v,w) \\ y_P\bar{t}+y_Qt=y_Ap(u,v,w)+y_Bq(u,v,w)+y_Cr(u,v,w)+y_Ds(u,v,w). \\ z_P\bar{t}+z_Qt=z_Ap(u,v,w)+z_Bq(u,v,w)+z_Cr(u,v,w)+z_Ds(u,v,w) \end{cases}$$

Thus, we obtain a system of three equations with four unknown variables. To solve it, we can consider that the edge values of the u, v, w parameters of a geometric solid determine the shell of its surface, the points of intersection that we need to find. Since a geometric solid can occupy completely different positions in space relative to the ray, it is impossible to determine in advance which two of the surfaces of the shell of the geometric solid will intersect the ray. Therefore, it is necessary to solve six systems – special cases, which are a consequence of the use of edge parameter values. In most cases, the parameters of point equations change from 0 to 1. Then it is necessary to fix in turn the values of the parameters ($u=0$, $u=1$, $v=0$, $v=1$, $w=0$, $w=1$) and obtain particular solutions to the system of parametric equations. From all the solutions, it is necessary to choose only those, which provide the obtained parameter values in the range from 0 to 1. There will be only two such solutions if the ray intersects the geometric solid, and one – if it touches the solid.

3. Visualization of Solid Models on the Example of Faceted Solids

Consider an example of determining the points of intersection of a segment with a solid of a tetrahedron (Fig. 2).

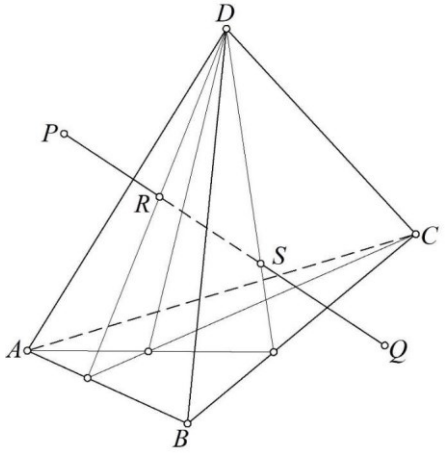

Fig. 2. Geometric scheme of the PQ segment intersection with the solid of the tetrahedron

In [1, 2], the point equation of the solid of a tetrahedron is given, which is determined by the points of the three-dimensional $ABCD$ simplex and three linear parameters u, v, w:

$$M=Auv\bar{w}+B\bar{v}\bar{w}+C\bar{u}v\bar{w}+Dw,$$

where $\bar{u}=1-u$, $\bar{v}=1-v$ и $\bar{w}=1-w$.

Based on the method described above, we get:

$$P\bar{t} + Qt = Auv\bar{w} + B\bar{v}w + C\bar{u}v\bar{w} + Dw$$

$$\Downarrow$$

$$\begin{cases} x_P\bar{t} + x_Qt = x_Auv\bar{w} + x_B\bar{v}w + x_C\bar{u}v\bar{w} + x_Dw \\ y_P\bar{t} + y_Qt = y_Auv\bar{w} + y_B\bar{v}w + y_C\bar{u}v\bar{w} + y_Dw \\ z_P\bar{t} + z_Qt = z_Auv\bar{w} + z_B\bar{v}w + z_C\bar{u}v\bar{w} + z_Dw \end{cases}$$

$$\Downarrow$$

$$\begin{cases} x_Auv\bar{w} + x_B\bar{v}w + x_C\bar{u}v\bar{w} + x_Dw - x_P = \left(x_Q - x_P\right)t \\ y_Auv\bar{w} + y_B\bar{v}w + y_C\bar{u}v\bar{w} + y_Dw - y_P = \left(y_Q - y_P\right)t \\ z_Auv\bar{w} + z_B\bar{v}w + z_C\bar{u}v\bar{w} + z_Dw - z_P = \left(z_Q - z_P\right)t \end{cases}.$$

Next, it is necessary to fix in turn the edge values of the current parameters of the tetrahedron solid ($u=0$, $u=1$, $v=0$, $v=1$, $w=0$, $w=1$) and solve the resulting systems of equations.

To conduct computational experiments, we accept the following test coordinates of the initial points:

$$A(0;0;0),\ B(3;5;0),\ C(8;3;0),\ D(4;2;7),\ P(7;5;4),\ Q(0;1;2).$$

Using the coordinates of the initial points, we obtain 6 systems of parametric equations. Systems for $v=0$ and $w=1$ have no solution. At $v=1$ and $w=0$ – the resulting solutions go beyond the range of parameter values $[0;1]$. At $u=0$ parameter values satisfy the required conditions $t=\frac{9}{22}$, $v=\frac{1}{4}$, $w=\frac{5}{11}$, and at $u=1$ – $t=\frac{12}{19}$, $v=\frac{4}{9}$, $w=\frac{52}{133}$. Thus, it turns out that on the interval of parameter values $t\in\left[\frac{9}{22};\frac{12}{19}\right]$ part of the segment PQ is inside the tetrahedron, and on the intervals $t\in\left(-\infty;\frac{9}{22}\right)$ and $t\in\left(\frac{12}{19};+\infty\right)$ – outside the tetrahedron $ABCD$.

The proposed approach can be used to visualize not only facets, but also other geometric solids. At the same time, in relation to faceted solids, another approach can be implemented based on the intersection of the projecting beam with the plane.

In the general case, a PQ segment (Fig. 2) can have two intersection points with a $ABCD$ pyramid, one intersection point (when the segment intersects the edges and vertices of the pyramid) and no intersection points at all.

As a special case, we define the S point of intersection of the PQ segment with the BCD face. To do this, we set the point equation of a straight line using the parameter λ: $S=P\bar{\lambda}+Q\lambda$, where $\bar{\lambda}=1-\lambda$. P and Q points are defined in the $ABCD$ simplex using the corresponding parameters:

$$P = Ap_P + Bq_P + Cr_P + Ds_P \Rightarrow \begin{cases} x_P = x_Ap_P + x_Bq_P + x_Cr_P + x_Ds_P \\ y_P = y_Ap_P + y_Bq_P + y_Cr_P + y_Ds_P \\ z_P = z_Ap_P + z_Bq_P + z_Cr_P + z_Ds_P \end{cases},$$

$$Q = Ap_Q + Bq_Q + Cr_Q + Ds_Q \Rightarrow \begin{cases} x_Q = x_Ap_Q + x_Bq_Q + x_Cr_Q + x_Ds_Q \\ y_Q = y_Ap_Q + y_Bq_Q + y_Cr_Q + y_Ds_Q \\ z_Q = z_Ap_Q + z_Bq_Q + z_Cr_Q + z_Ds_Q \end{cases},$$

where $p_P = \frac{V_{PBCD}}{V_{ABCD}}$, $q_P = \frac{V_{APCD}}{V_{ABCD}}$, $r_P = \frac{V_{ABPD}}{V_{ABCD}}$, $s_P = \frac{V_{ABCP}}{V_{ABCD}} = 1 - p_P - q_P - r_P$,

$p_Q = \frac{V_{QBCD}}{V_{ABCD}}$, $q_Q = \frac{V_{AQCD}}{V_{ABCD}}$, $r_Q = \frac{V_{ABQD}}{V_{ABCD}}$, $s_Q = \frac{V_{ABCQ}}{V_{ABCD}} = 1 - p_Q - q_Q - r_Q$.

The volumes V_i of the corresponding tetrahedra can be easily determined in terms of the coordinates of the vertices. In this case, multiplication by 1/6 can be abolished, because it will still shrink. As an example, we give a determinant for calculating the volume of a $ABCD$ tetrahedron. All other volumes of oriented tetrahedra in a $ABCD$ simplex can be found similarly.

$$V_{ABCD} = \begin{vmatrix} x_A - x_D & y_A - y_D & z_A - z_D \\ x_B - x_D & y_B - y_D & z_B - z_D \\ x_C - x_D & y_C - y_D & z_C - z_D \end{vmatrix}.$$

Next, we determine the value of the λ parameter, at which the volume of the $SBCD$ tetrahedron is equal to zero. In accordance with the point calculus $S-$ theorem [14], we obtain:

$$\begin{vmatrix} (p_Q - p_P)\lambda + p_P & (q_Q - q_P)\lambda + q_P & (r_Q - r_P)\lambda + r_P & (s_Q - s_P)\lambda + s_P \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} = 0.$$

$$\Downarrow$$

$$\lambda = \frac{p_P}{p_P - p_Q}; \quad \bar{\lambda} = \frac{p_Q}{p_Q - p_P}.$$

$$S = P\frac{p_Q}{p_Q - p_P} + Q\frac{p_P}{p_P - p_Q} \Rightarrow \begin{cases} x_S = x_P \dfrac{p_Q}{p_Q - p_P} + x_Q \dfrac{p_P}{p_P - p_Q} \\ y_S = y_P \dfrac{p_Q}{p_Q - p_P} + y_Q \dfrac{p_P}{p_P - p_Q} \\ z_S = z_P \dfrac{p_Q}{p_Q - p_P} + z_Q \dfrac{p_P}{p_P - p_Q} \end{cases}.$$

The resulting S point equation can also be determined in the original $ABCD$ simplex by replacing the P and Q points with the corresponding equations, but this is not necessary to visualize the tetrahedron.

In the same way, we define the point R:

$$R = P\frac{r_Q}{r_Q - r_P} + Q\frac{r_P}{r_P - r_Q} \Rightarrow \begin{cases} x_R = x_P \dfrac{r_Q}{r_Q - r_P} + x_Q \dfrac{r_P}{r_P - r_Q} \\ y_R = y_P \dfrac{r_Q}{r_Q - r_P} + y_Q \dfrac{r_P}{r_P - r_Q} \\ z_R = z_P \dfrac{r_Q}{r_Q - r_P} + z_Q \dfrac{r_P}{r_P - r_Q} \end{cases}.$$

As a result, we obtain the following conditions:

If $\lambda \in \left[\frac{r_P}{r_P - r_Q}; \frac{p_P}{p_P - p_Q} \right]$, the ray segment lies inside the solid of the tetrahedron.

If $\lambda \in \left(-\infty; \frac{r_P}{r_P - r_Q}\right) \cap \left(\frac{p_P}{p_P - p_Q}; +\infty\right)$, the ray segment does not belong to the solid of the tetrahedron.

Based on the relative position of the projection center and the picture plane, as shown in Fig. 1, $\lambda \in [0;1]$. At the same time, special cases are possible when the projecting ray belongs to one of the faces of the tetrahedron, coincides with its edge, or passes through the vertex. This does not affect the computational algorithm described above.

4. Software Implementation and Results of Computational Experiments

To conduct computational experiments on the visualization of faceted solids, a test program was written in the GLSL language, containing 167 lines of code (the source code is available under the MIT license on GitHub: https://github.com/icosaeder/tpps-raycast). Visualization is done in the Web browser using a free resource ShaderToy (https://www.shadertoy.com) that provides a convenient toolkit for launching programs intended for execution on a graphics card.

The aim of a test program was to check the ability to efficiently render faceted solids defined in point calculus. The block schema of the developed numerical algorithm is presented in Fig. 3.

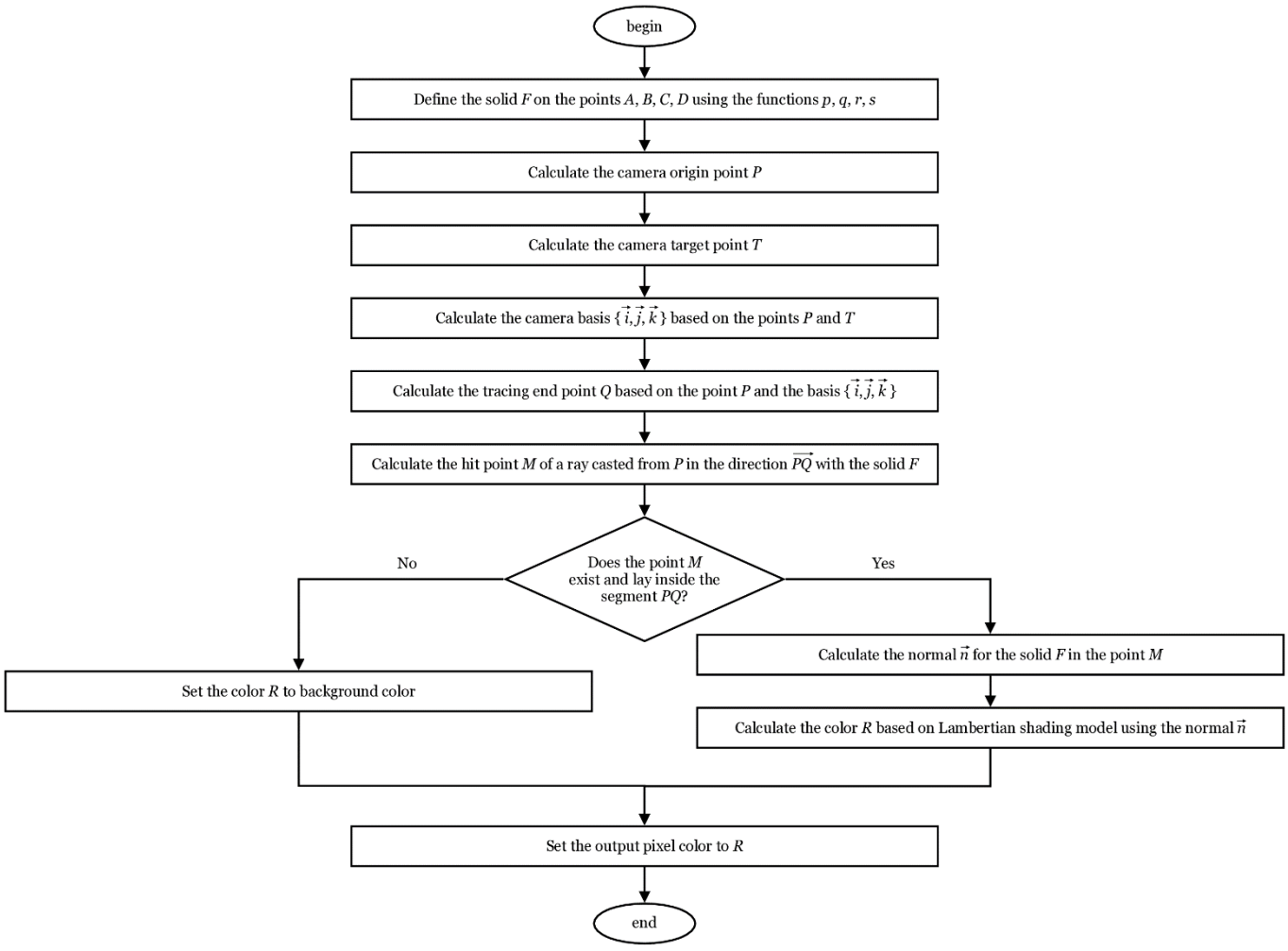

Fig. 3. Block schema of the proposed rendering algorithm

The input for this algorithm is a definition of a faceted solid F in point calculus using the points A, B, C, D, and the functions p, q, r, s as described in Section 2. The output is the color

of the individual pixel of an image. When executed for each pixel of an image, this algorithm renders a given faceted solid F from a point of view determined by points P, T, and Q.

Although the proposed algorithm is written in a sequential manner, it assumes an implicit parallel execution on a graphics card within a SIMD (Single Instruction Multiple Data) paradigm. The parallelism is achieved by executing this algorithm concurrently for a subset of pixels. The number of pixels in a subset is determined dynamically by the graphics card based on its hardware capabilities.

The software implementation is based on the numerical solution of systems of equations using the iterative Newton method [16]. At each i-th iteration, a new approximation of the solution $\sigma^{(i)} = \{x^{(i)}, y^{(i)}, z^{(i)}\}$ of the F system is calculated by the formula:

$$\sigma^{(i)} = \sigma^{(i-1)} - J\left(\sigma^{(i-1)}\right)^{-1} F\left(\sigma^{(i-1)}\right),$$

where J is the Jacobian of the F system, which is calculated numerically by the formula:

$$J(\sigma) = \begin{pmatrix} \left(\frac{\partial F}{\partial x}(\sigma)\right)_x & \left(\frac{\partial F}{\partial y}(\sigma)\right)_x & \left(\frac{\partial F}{\partial z}(\sigma)\right)_x \\ \left(\frac{\partial F}{\partial x}(\sigma)\right)_y & \left(\frac{\partial F}{\partial y}(\sigma)\right)_y & \left(\frac{\partial F}{\partial z}(\sigma)\right)_y \\ \left(\frac{\partial F}{\partial x}(\sigma)\right)_z & \left(\frac{\partial F}{\partial y}(\sigma)\right)_z & \left(\frac{\partial F}{\partial z}(\sigma)\right)_z \end{pmatrix}.$$

The parameters u, v, w alternately act as the unknown variables x and y, and the parameter t always acts as the unknown variable z. Substitutions are carried out according to the following scheme for all 6 systems described in Section 2:

1. $u = 0, x = v, y = w, z = t.$
2. $u = 1, x = v, y = w, z = t.$
3. $v = 0, x = u, y = w, z = t.$
4. $v = 1, x = u, y = w, z = t.$
5. $w = 0, x = u, y = v, z = t.$
6. $w = 1, x = u, y = v, z = t.$

Partial derivatives in the composition of the Jacobian are calculated in accordance with the difference scheme, for example:

$$\frac{\partial F}{\partial x}(\sigma) = \frac{F(\{x + \Delta, y, z\}) - F(\{x - \Delta, y, z\})}{2\Delta}.$$

It was experimentally found that an acceptable accuracy of the solution is achieved when $\Delta = 10^{-3}$.

The first approximation $\sigma^{(0)}$ is taken in our case identically equal to zero. However, for some F, such a $\sigma^{(0)}$ may lead to a loss of solution, since the Jacobian at the first iteration may turn out to be degenerate, which will make it impossible to calculate $\sigma^{(1)}$ due to the lack of an inverse matrix for the degenerate Jacobian. To address a problem, the code implements a check for Jacobian degeneracy at the first iteration, and if it is degenerate, the identity matrix is substituted for it, which corrects the first approximation.

Newton's method has two stopping conditions. The first is the achievement of the specified accuracy of the solution, which is determined by the formula:

$$\sqrt{(x^{(i)} - x^{(i-1)})^2 + (y^{(i)} - y^{(i-1)})^2 + (z^{(i)} - z^{(i-1)})^2} < \Delta.$$

The second stopping condition consists in exceeding the threshold number of steps, which indicates that the system does not have a solution. In our case, a threshold value of 100 steps was experimentally selected. This constant is due to the fact that, according to our observations, for faceted solids, the corresponding systems are solved on average in 7–10 steps (if they have a solution). Thus, the threshold of 100 steps with a margin provides reliable protection against loss of the solution, and, at the same time, does not greatly overload the GPU with unnecessary iterations.

In order for the user to have the opportunity to view the rendered solid from all sides, a model of an interactive orbital camera is implemented. The camera position expressed by P point (which corresponds to the beginning of the segment mentioned in Section 2) is calculated by the following formula:

$$P = \{\sin\theta \cdot \sin\varphi\,, \cos\varphi\,, \cos\theta \cdot \sin\varphi\} \cdot r,$$

where θ, φ are the polar angles calculated based on the coordinates of the mouse cursor in the visualization area, r is the radius of the camera's orbit, in our case calculated as twice the maximum value of the coordinates of the A, B, C, D, points, which define a three-dimensional simplex for constructing a rendered solid.

The center of the scene (the point the camera "looks" at) is selected T point – the center of mass of the solid. For example, for a tetrahedron:

$$T = \frac{A+B+C+D}{4} \Rightarrow \begin{cases} x_T = \dfrac{x_A + x_B + x_C + x_D}{4} \\ y_T = \dfrac{y_A + y_B + y_C + y_D}{4} \\ z_T = \dfrac{z_A + z_B + z_C + z_D}{4} \end{cases}.$$

The orts of the camera coordinate system are defined as follows:

$$\vec{k} = \frac{T - P}{|T - P|},$$

$$\vec{\imath} = \frac{\{0,1,0\} \times \vec{k}}{|\{0,1,0\} \times \vec{k}|},$$

$$\vec{\jmath} = \frac{\vec{\imath} \times \vec{k}}{|\vec{\imath} \times \vec{k}|}.$$

The Q point corresponding to the end of the segment mentioned in Section 2 is determined by the formula:

$$Q = P + \frac{\vec{\imath} \cdot p_x + \vec{\jmath} \cdot p_y + \vec{k}}{|\vec{\imath} \cdot p_x + \vec{\jmath} \cdot p_y + \vec{k}|} d_{max},$$

where p_x, p_y are the normalized coordinates of the screen pixel through which the tracing is performed (and whose color you want to determine), d_{max} is the maximum tracing distance (in our case, the constant 100 is taken).

To increase the visual quality of the displayed faceted solid and ensure the perception of its three-dimensionality by the user, a Lambertian shading model has been implemented. In this case, the $\vec{n}$ normal is calculated as the gradient of the distance function using the difference scheme:

$$\vec{d} = \{\delta(P + \vec{\imath} \cdot \Delta, Q) - \delta(P, Q), \delta(P + \vec{\jmath} \cdot \Delta, Q) - \delta(P, Q), \delta(P + \vec{k} \cdot \Delta, Q) - \delta(P, Q)\},$$

$$\vec{n} = \frac{\vec{d}}{|\vec{d}|},$$

where δ is the function for determining the distance to a solid on a segment PQ.

All the calculations described above are performed for each screen pixel in parallel leveraging the SIMD paradigm. The results of visualization of faceted solids are shown in Fig. 4 and 5.

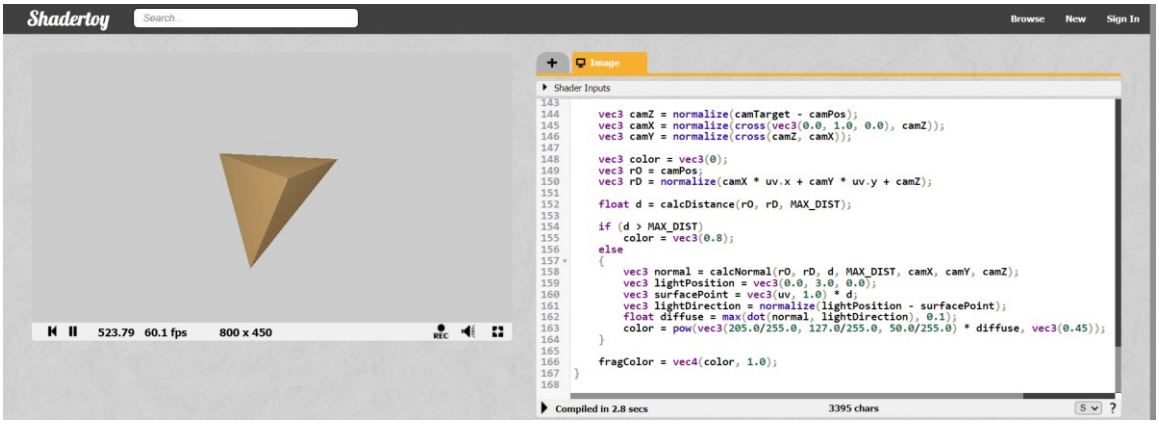

Fig. 4. Visualization of the tetrahedron solid in the service environment ShaderToy

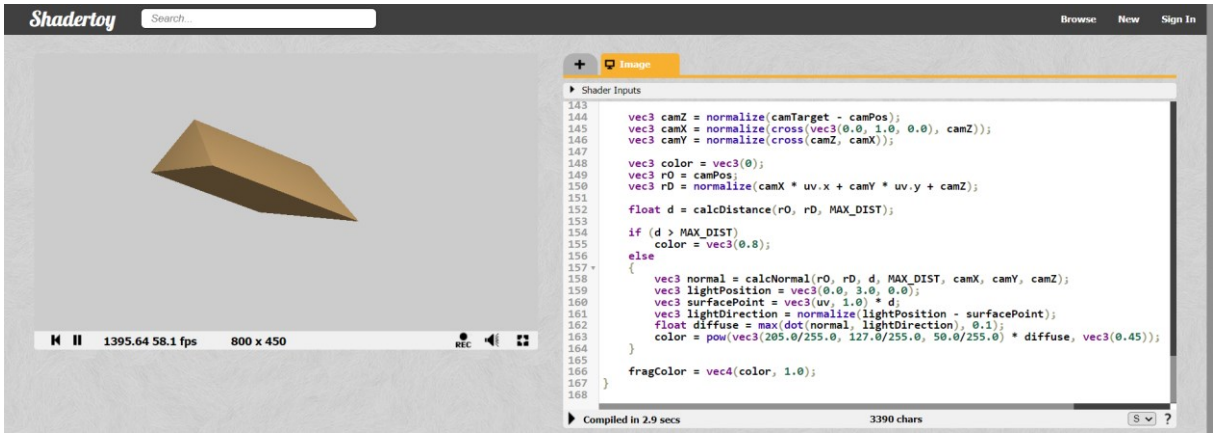

Fig. 5. Visualization of the prism solid in the service environment ShaderToy

5. Conclusion

As can be seen from the results of computational experiments (Fig. 4–5), the proposed approach to the visualization of solid geometric models justifies itself to a sufficient extent. Rendering performance delivers an image refresh rate of around 60 frames per second, matching the refresh rate of today's general-purpose monitors. At the same time, high image fidelity is maintained without any visible artifacts. However, when visualizing curvilinear solids, the appearance of artifacts is still possible. This is due to the fact that Newton's method cannot provide sufficient accuracy for the numerical solution of the system of equations in that case. Proceeding from this, the prospect of further research is the improvement of the method for determining the signed distance function for the visualization of curvilinear geometric solids with the subsequent integration of the developed software solutions into the scientific visualization system SciVi [17-19].

The proposed approach to visualization of solid geometric models is primarily oriented to application in computer-aided design, solid-state and information (BIM – Building Information Modeling) modeling systems. However, given the wide application of three-dimensional models in various sectors of the economy, after improvement, the proposed approach can find application not only in mechanical engineering, construction, and architecture, but also in science, medicine, design, education, as well as in advertising, film, and video game industry. It seems promising to use the proposed approach to the definition of solid models in point calculus to develop a new technology for generating full-fledged volumetric images in three-dimensional space by analogy with holographic ones.

References

1. Konopatskiy E.V., Bezditnyi A.A. The Problem of Visualizing Solid Models as a Three-Parameter Point Set. Scientific Visualization, 2022. Vol. 14. No. 2. pp. 49-61. DOI: 10.26583/sv.14.2.05.

2. Konopatskiy E.V., Bezditnyi A.A., Lagunova M.V., Naidysh A.V. Principles of solid modelling in point calculus. IoP conference series: Journal of Physics: Conf. Series 1901 (2021) 012063. DOI: 10.1088/1742-6596/1901/1/012063.

3. Konopatskiy E.V., Bezditnyi A.A. Solid modeling of geometric objects in point calculus. Proceedings of the 31st International Conference on Computer Graphics and Vision (GraphiCon 2021) Nizhny Novgorod, Russia, September 27-30, 2021. Vol. 3027. pp. 666-672. DOI: 10.20948/graphicon-2021-3027-666-672.

4. Kettunen M., D'Eon E., Pantaleoni J., Novák J. An unbiased ray-marching transmittance estimator. ACM Transactions on Graphics. 2021. 40(4). DOI: 10.1145/3450626.3459937.

5. Coulon R., Matsumoto E. A., Segerman H., Trettel S. J. Ray-marching thurston geometries. Experimental Mathematics. 2022. DOI: 10.1080/10586458.2022.2030262.

6. Xu T., Zeng W., Wu E. Viewport-resolution independent anti-aliased ray marching on interior faces in cube-map space. Paper presented at the Proceedings - SIGGRAPH Asia 2021 Technical Communications. SA 2021. DOI: 10.1145/3478512.3488598.

7. Debelov V.A., Kozlov D.V. Visualization of Light Polarization to Debug Ray Tracing Algorithms. Scientific Visualization. 2013. Vol. 5. No 4. pp. 71-87.

8. Zhdanov D.D., Ershov S.V., Deryabin N.B. Object-oriented model of photorealistic visualization of complex scenes. Scientific Visualization. 2013. Vol. 5. No. 4. pp. 88-117.

9. Sokolov V.G., Voloboy A.G., Potemin I.S., Galaktionov V.A. Overview of BSDF Reconstruction Methods for Rough Surfaces. Scientific Visualization. 2022. Vol. 14. No 3. pp. 132-151. DOI: 10.26583/sv.14.3.10.

10. Wald I., Zellmann S., Usher W., Morrical N., Lang U., Pascucci V. Ray tracing structured AMR data using exabricks. IEEE Transactions on Visualization and Computer Graphics. 2021. 27(2). pp. 625-634. DOI: 10.1109/TVCG.2020.3030470.

11. Zhang K., Luan F., Wang Q., Bala K., Snavely N. PhySG: Inverse rendering with spherical gaussians for physics-based material editing and relighting. Paper presented at the Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2021. 5449-5458. DOI: 10.1109/CVPR46437.2021.00541.

12. Shen T., Gao J., Yin K., Liu M., Fidler S. Deep marching tetrahedra: A hybrid representation for high-resolution 3D shape synthesis. Paper presented at the Advances in Neural Information Processing Systems. 2021. No. 8. pp. 6087-6101.

13. Soukov S.A. Combined signed distance calculation algorithm for numerical simulation of physical processes and visualization of solid solids movement. Scientific Visualization. 2021. 12(5). DOI: 10.26583/SV.12.5.08.

14. Konopatskiy E.V., Bezditnyi A.A. Point tools of geometric modeling, invariant relating to parallel projection. Geometry & Graphics. 2021. Vol. 9. No. 4. pp. 11-21. DOI: 10.12737/2308-4898-2022-9-4-11-21.

15. Konopatskiy E.V., Bezditnyi A.A., Kokareva Ya.A., Kucherenko V.V. Features visualization of geometric objects in the BN-calculus. Scientific Visualization. 2020. Vol. 12. No. 2. pp. 98-109. DOI: 10.26583/sv.12.2.08.

16. Remani C. Numerical Methods for Solving Systems of Nonlinear Equations. Lakehead University. 2013. 38 p.

17. Ryabinin K.V., Baranov D.A., Belousov K.I. Integration of Semograph information system and SCIVI visualizer for solving the tasks of lingual content expert analysis. Scientific Visualization. 2017. Vol. 9. No. 4. pp. 67-77. DOI: 10.26583/sv.9.4.07.

18. Ryabinin K.V., Kolesnik M.A., Akhtamzyan A.I., Sudarikova E.V. Cyber-Physical Museum Exhibits Based on Additive Technologies, Tangible Interfaces and Scientific Visualization. Scientific Visualization. 2019. Vol. 11. No 4. pp. 27-42. DOI: 10.26583/sv.11.4.03.

19. Ryabinin K.V., Belousov K.I. Visual Analytics of Gaze Tracks in Virtual Reality Environment. Scientific Visualization. 2021. Vol. 13. No 2. pp. 50-66. DOI: 10.26583/sv.13.2.04.