УДК 004.92:512

K.S. KRASNIKOV
Dniprovsky state technical university, Kamianske

THE USE OF QUATERNIONS FOR THE PROJECTION OF 3D-OBJECTS ON A PLANE

The article describes a mathematical description of 3D points mapping to a plane for a graphical image of a 3D scene, which consists of 3D objects. Instead of a common approach in the form of transformation matrices, the article proposes vectors transformation using quaternions as a useful approach for the problem. The method was tested on examples and the results are presented in the figures. This approach can be used to visualize the results of mathematical model calculations for the metallurgical process on the screen.

Keywords: vector 3D-graphics, quaternion, perspective projection.

К.С. КРАСНІКОВ
Дніпровський державний технічний університет, м. Кам'янське

ВИКОРИСТАННЯ КВАТЕРНІОНІВ ДЛЯ ПРОЕКЦІЇ 3D-ОБ'ЄКТІВ НА ПЛОЩИНУ

У статті викладено математичний опис відображення 3D-точок на площину для графічного зображення 3D-сцени, яка складена з 3D-об'єктів. Замість розповсюдженого підходу у вигляді матриць перетворень стаття пропонує перетворення векторів з використанням кватерніонів як корисний для означеної задачі підхід. Метод був протестований на прикладах і результати представлено на рисунках. Даний підхід може бути використаний для візуалізації результатів обчислення математичної моделі металургійного процесу на екрані.

Ключові слова: векторна 3D-графіка, кватерніон, перспективна проекція.

К.С. КРАСНИКОВ
Днепровский государственный технический университет, г. Каменское

ИСПОЛЬЗОВАНИЕ КВАТЕРИНИОНОВ ДЛЯ ПРОЕКЦИИ 3D-ОБЪЕКТОВ НА ПЛОСКОСТЬ

В статье изложено математическое описание отображения 3D-точек на плоскость для графического изображения 3D-сцены, которая состоит с 3D-объектов. Вместо распространенного подхода в виде матриц превращений статья предлагает преобразование векторов с использованием кватернионов как полезное для указанной задачи подход. Метод был протестирован на примерах и результаты представлены на рисунках. Данный подход может быть использован для визуализации результатов вычислений математической модели металлургического процесса на экране.

Ключевые слова: векторная 3D-графика, кватернион, перспективная проекция.

Problem definition

During numerical research on a computer, the researcher receives a set of numbers, which are the results of a mathematical model of a real process, for example in metallurgy — the refining of a metal melt by cored wire in a ladle with inert gas blowing [1]. The size of the resulting set of numbers, especially in the three-dimensional formulation of a complex physical problem, may result in thousands of numbers in the end of the calculation, which necessitates the transformation of this set into more convenient forms for perception. Such representations can be integral indicators, in particular, the coefficient of variation, the minimum or the maximum concentration of some substance, etc.; graphic representation of the process state. And if obtaining the specific integral indicators is usually simple, then to obtain a graphic image, especially three-dimensional, you need to create a separate mathematical model.

In this way, it is often necessary to create and implement not only a mathematical model of the physical process on a computer, but also — a mathematical model of graphic representation of the resulting numerical results. An example is a racing computer game based on two models: automotive mechanics — «physics» of game; and a graphic image of three-dimensional car models and its environment on the screen — the game «graphics». Usually the most popular and a high rate games have high-quality three-dimensional «physics» and «graphics», because event realism is achieved in such games.

Related publications

One of the best ways to get realistic images of three-dimensional objects is the ray-tracing — a tracking of the light beam through the «screen point» and simulating the beam interaction with three-dimensional objects [2]. With all the advantages, this method takes a significant amount of processor time — it is necessary to calculate the intersection of the beam with the surfaces for each point of the screen, taking into account its reflection. An alternative modern method is the projection of triangles on a plane [3,4] (and their coloring). The speed of the alternative method is large enough to display three-dimensional objects on the screen of modern personal computers with a real-time user interaction. At the same time, the photographic quality of the image can be neglected.

Goal of investigation

The purposes of the work are obtaining formulas for 3D-point projection on a plane using quaternions and checking the result in a computer program.

Presentation of the research material

In the real process that is modeled, there are items that play certain roles. These objects are replaced by simple or complex geometric bodies — their three-dimensional models — for the purpose of graphic representation. For example, in the mentioned in the introduction metallurgical process, you can define the following objects: ladle and wire. Their surface can be displayed using such geometric bodies: a ladle — an open cylinder, a wire — a set of cylinders. Or, if a spherical granule is involved in the process, it can be displayed by a sphere.

For a sphere or cylinder image, first, we need to define the coordinates of the surface points P_s i P_c. It is convenient to get them using the corresponding coordinate system (Fig. 1-2):

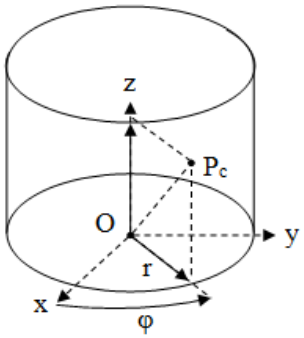

Fig. 1. Point on the cylinder surface

$$P_c = (r\cos\varphi,\quad r\sin\varphi,\quad z), \tag{1}$$

$$P_s = (r\cos\varphi\sin\theta,\quad r\sin\varphi\sin\theta,\quad r\cos\theta), \tag{2}$$

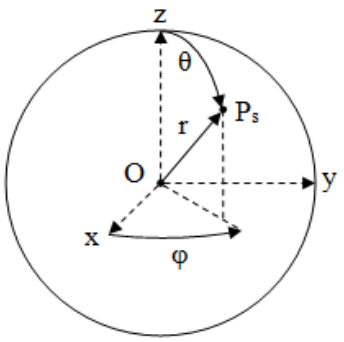

Fig. 2. Point on the sphere surface

Using dividing of the axes r, φ, θ, z by steps, you can get a set of quadrangles, which reproduce the corresponding surface with the chosen accuracy. Each quadrilateral ABCD consists of two triangles (ABC and ACD), which normal vectors matches each other (Fig. 3):

$$\bar{n} = \frac{\overline{BC} \times \overline{AB}}{\left\| \overline{BC} \times \overline{AB} \right\|}, \tag{3}$$

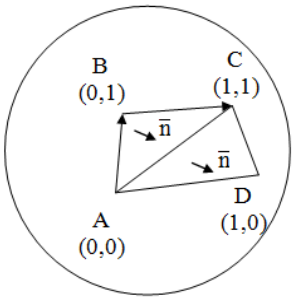

Fig. 3. Points of triangles. In brackets there are texture coordinates u & v for edge drawing

Each surface triangle corresponds to:

1) Three vertices (A, B, C) that determine the location of a triangle in space.

2) The normal vector $\bar{n}$ required to illuminate the triangle, depending on the location of the light source.

3) Two sides of the triangle and only one of them we will see in the final graphic representation of the triangle. They can be differentiated by the vertices' order in the image — A, B, C or C, B, A. This will be useful if we need to draw both external and internal sides of the surface (only one is showed by default).

Thus, the point coordinates on the geometric body surface are determined in relation to the reference point O without taking into account the position of the body itself with respect to O. For the purpose of model positioning and orienting three types of transformations are used: a parallel transfer, a rotation around the x, y, z axes and a scaling relative to O.

In order to get an image of a 3D object, such as a triangle, on a plane (C, $\vec{n}$) we define the coordinates of the triangle points $\vec{r}_{1..3}$ (Fig. 4), in the reference frame that corresponds to the spectator, Ot (Fig. 5).

The points $\vec{r}'_{1..3}$ (Fig. 5) are determined using quaternion q_m, taking own 3D-object rotation and isotropic scaling into account, and positioning — using radius vector $\vec{t}_m$.

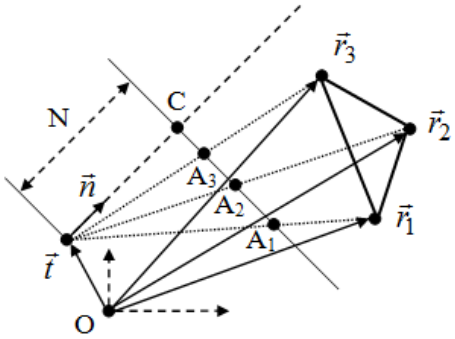

Fig. 4. Reference point O, triangle, point and direction of view and the plane for projection

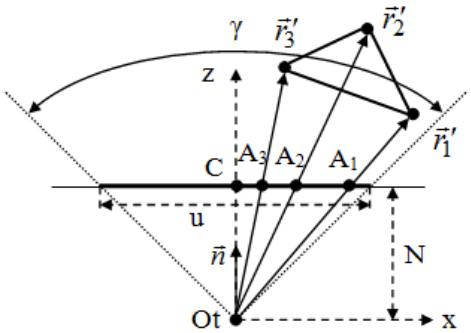

Fig. 5. Triangle and plane for projection in reference frame of spectator Ot

The direction of observation $\vec{n}$ is determined by azimuth φ and zenith θ angles:

$$\vec{r}_i' = q_v\left(q_m\vec{r}_i q_m^{-1} + \vec{t}_m - \vec{t}\right)q_v^{-1} = q_v q_m \vec{r}_i q_v^{-1} q_m^{-1} + q_v\left(\vec{t}_m - \vec{t}\right)q_v^{-1}$$

$$q_v q_m = \left(\tilde{q}_v\tilde{q}_m - \vec{q}_v\cdot\vec{q}_m; \tilde{q}_v\vec{q}_m + \tilde{q}_m\vec{q}_v + \vec{q}_v\times\vec{q}_m\right);$$

$$q_v = q_y q_z;\ q_y = \left(\cos\frac{\theta}{2}; \sin\frac{\theta}{2}\vec{e}_y\right);\ q_z = \left(\cos\frac{\varphi}{2}; \sin\frac{\varphi}{2}\vec{e}_z\right);$$

$$q_v = \begin{pmatrix} \cos\frac{\theta}{2}\cos\frac{\varphi}{2}; \cos\frac{\theta}{2}\sin\frac{\varphi}{2}\vec{e}_z + \\ + \cos\frac{\varphi}{2}\sin\frac{\theta}{2}\vec{e}_y + \sin\frac{\varphi}{2}\sin\frac{\theta}{2}\vec{e}_x \end{pmatrix},$$

where q_v — versor, which corresponds to the observation direction; $\tilde{q}$ & $\vec{q}$ — scalar and vector part of versor q; $\vec{e}_y$ & $\vec{e}_z$ — unit vectors of corresponding axes. Quaternion q_m can be determined using multiplication of scalar by versor:

$$q_m = \sqrt{s}\left(\tilde{q}_m; \vec{q}_m\right),$$

where s — coefficient of isotropic scaling.

Transformation of vector $\vec{r}$ using quaternion q is defined by well-known formulas:

$$\vec{r}^{\,*} = q\vec{r}q^{-1}$$

$$\vec{r}_x^{\,*} = \left(\tilde{q}^2 + \vec{q}_x^2 - \vec{q}_y^2 - \vec{q}_z^2\right)\vec{r}_x + \left(2\vec{q}_x\vec{q}_y - 2\tilde{q}\vec{q}_z\right)\vec{r}_y + \\ + \left(2\vec{q}_x\vec{q}_z + 2\tilde{q}\vec{q}_y\right)\vec{r}_z$$

$$\vec{r}_y^{\,*} = \left(2\vec{q}_x\vec{q}_y + 2\tilde{q}\vec{q}_z\right)\vec{r}_x + \left(\tilde{q}^2 - \vec{q}_x^2 + \vec{q}_y^2 - \vec{q}_z^2\right)\vec{r}_y + \\ + \left(2\vec{q}_y\vec{q}_z - 2\tilde{q}\vec{q}_x\right)\vec{r}_z$$

$$\vec{r}_z^{\,*} = \left(2\vec{q}_x\vec{q}_z - 2\tilde{q}\vec{q}_y\right)\vec{r}_x + \left(2\vec{q}_y\vec{q}_z + 2\tilde{q}\vec{q}_x\right)\vec{r}_y + \\ + \left(\tilde{q}^2 - \vec{q}_x^2 - \vec{q}_y^2 + \vec{q}_z^2\right)\vec{r}_z$$

The mentioned formulas should be simplified, if there is a possibility, for example, for the versor q we can use, well-known expression:

$$q\vec{r}q^{-1} = \vec{r} + 2\vec{q} \times \left(\vec{q} \times \vec{r} + \tilde{q}\vec{r}\right)$$

In the case of perspective projection point A on the screen is determined by dividing the vector $\vec{r}$ (or x and y components only — for saving z) by coordinate z:

$$\mathrm{A} = \frac{N}{\vec{r} \cdot \vec{e}_z}\vec{r}$$

It is convenient to normalize the coordinates x and y of point A to the interval (-0,5; 0,5) using the viewing angle γ:

$$\mathrm{A}' = \mathrm{A}u^{-1} \qquad u = 2N\tan\frac{\gamma}{2}$$

$$\mathrm{A}' = \frac{\vec{r}}{(\vec{r} \cdot \vec{e}_z)2\tan\frac{\gamma}{2}}$$

Due to the aspect ratio of the screens, which isn`t equal to one usually, it is necessary to scale, for example, the coordinate x:

$$\mathrm{A}''_{\mathrm{x}} = \mathrm{A}'_{\mathrm{x}}\, s \qquad s = \frac{w}{h},$$

where w and h — correspond to width and height of screen in pixels.

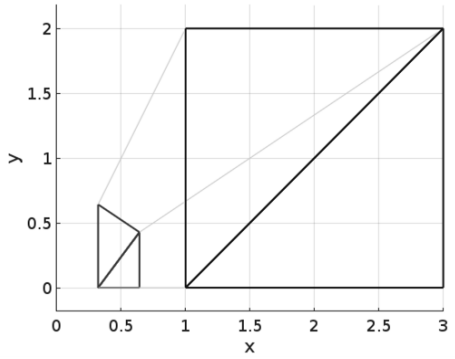

Fig. 6. Perspective square projection, top view

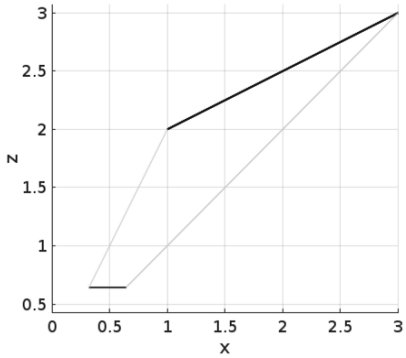

Fig. 7. Perspective projection, side view

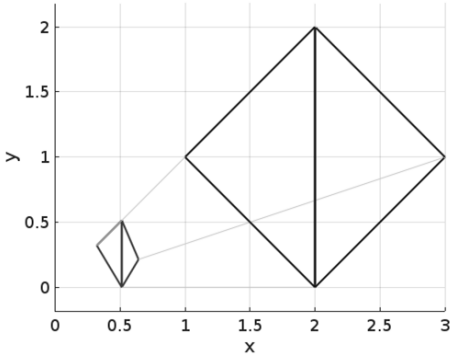

Fig. 8. Perspective rhombus projection, top view

If the orthogonal projection of a point is required, then the formula is simplified:

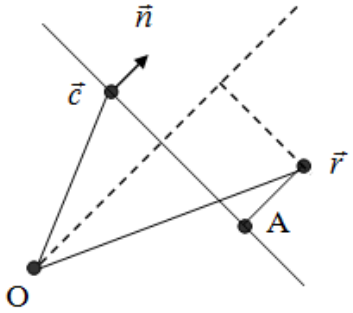

Fig. 9. Orthogonal projection of the point r on the plane

$$\mathrm{A} = (\vec{c} \cdot \vec{n})\vec{n} + [\vec{r} - (\vec{r} \cdot \vec{n})\vec{n}],$$

where c — point of view, r — point for projection, A — point on plane.

During the above operations, an important value is determined — the "depth" of the point, which will be useful during the visibility test — the image point (pixel) is ignored if its depth is greater than the depth of the drawn point. It saves the computational time (lighting for this point is not calculated) and allows you to draw geometric bodies in any order. A two-dimensional map of pixels on the screen (image) corresponds to a two-dimensional depth map with the same height and width. At the beginning of creating a new image, all values on the depth map are set to the maximum value of 1. The points on the other side of the triangle are also ignored — only one of the sides is colored.

We use the simplest model of lighting to determine the color of each pixel:

$$\bar{c}_f = \bar{c}(\bar{d} \cdot \bar{n}),$$

where $\bar{c}$ — vector (with components r — red, g — green, b — blue) color of the geometric body, $\bar{d}$ — direction of light beam, $\bar{n}$ — normal vector of triangle (known for each pixel).

In order to increase realism the simplest surface self-immersion can be taken into account. For its implementation we use a map of depth and a map of normals, which correspond to the image. The presence of surface curvature, that causes shading, is determined by the depth of the pixels using Jensen's inequality. The

intensity of shading is proportional to the angle between the normal of this pixel and the normal of its neighbors — the difference of the corresponding depths must be within the specified limits z_{max} (Fig. 10).

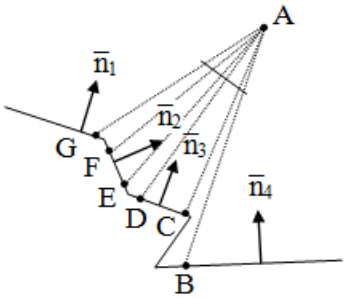

Fig. 10. Simulating the effect of self-immersion

Consider shading at points D and E:

$$\bar{c}_s = \bar{c}_f (1-s),$$

$$s = \begin{cases} \cos^{-1}\left(\bar{n}_2 \cdot \bar{n}_3\right) & \left(|z(D)-z(E)| < z_{max}\right) \wedge \\ & \wedge \left(z\left(\frac{D+E}{2}\right) > \frac{z(D)+z(E)}{2}\right), \\ 0 & \end{cases}$$

where z_{max} — maximum difference in depth of pixels for applying shading, $\bar{c}_f$ — pixel color (rgb). The obvious drawback of this approach is the lack of shading at point B, if z_{max} is not big enough. On the other hand if z_{max} is big it will lead to excessive shading of distant surfaces.

In order to get a black and white image with the edges of the quadrangles, we use the texture coordinates u and v (after linear interpolation between the vertices of the triangle for each pixel, Fig. 3):

$$\bar{c}' = \begin{cases} (0,0,0) & \left(u \le l_u\right) \vee \left(v \le l_v\right) \vee \\ & \vee \left(u > 1-l_u\right) \vee \left(v > 1-l_v\right), \\ (1,1,1) & \end{cases}$$

where l — the line's thickness, respectively, for u and v (taking into account the aspect ratio of the screen).

Results and conclusions

Unlike the transformation matrix, the use of quaternions has the advantage of a smaller number of data to store: 4 against 9 for the matrix. Also, in our case, the mapping of 3D objects using transformation matrices is redundant, because we use translation, rotation and isotropic scaling only. In addition, the number of arithmetic operations during quaternion multiplication is known to be less than in multiplication of transformation matrices. Thus, the use of quaternions for a given task is rational. The author is grateful to the head of applied mathematics department Samokhvalov Sergey Yevgenovich for an interesting idea of quaternion usage in 3D-scene projection on a plane.

Today, computers include graphic processors, which in further research can be programmed with using of quaternions to visualize thousands of triangles and comparing of matrix computing time.

References

1. Krasnikov K.S. The mathematical model of the cored wire injection in the molten steel during blowing on the ladle-furnace / S.E. Samokhvalov, V.P. Piptyuk // Regional interuniversity compendium of scientific works "System technologies". — № 5 (100). — Dnipro, 2015. – pp. 4-14.
2. Pharr M.; Humphreys G. Physically based rendering: from theory to implementation / Pharr M. – Amsterdam : Elsevier/Morgan Kaufmann. ISBN 0-12-553180-X. – 2004. – 1101 p.
3. Verhola A. P. Engineering graphics: drawings, computer graphics: tutorial / Ed. A. P. Verhola. — Kiev: Caravela, 2005. — 304 p. — ISBN 966-8019-35-0
4. John Vince. Mathematics for Computer Graphics. 3rd Edition. — Springer, 2010. — 293 p. — ISBN: 978-1-84996-022-9