Perspective-Correct Computation Pixels Color for Systems of Three-Dimensional Rendering

Oleksandr Romaniuk[1], Oleksandr Dudnyk[1], Serhii Pavlov[2], Olena Tsikhanovska[3]

1. Department of Software Engineering, Vinnytsia National Technical University, UKRAINE, Vinnytsia, 95 Khmelnytske shose str., email: rom8591@gmail.com, dudnyk@vntu.edu.ua

2. Department of Biomedical Engineering, Vinnytsia National Technical University, UKRAINE, Vinnytsia, 95 Khmelnytske shose str., email: psv@vntu.edu.ua

3. Department of Economics of Enterprises and Corporations, The Vinnytsyia Educational and Research Institute of Economics Ternopil National Economic University, UKRAINE, Vinnytsia, 37 Gonta str., email: vnnie.epik@gmail.com

Abstract: **In computer graphics, the projections of three-dimensional images are considered on a two-dimensional picture plane. Flat geometric projections are divided into two main classes: central and parallel. The difference between them is determined by the relationship between the center of the projection and the projection plane. If the distance between them is finite, then the projection will be central, and if it is infinite, then the projection will be parallel. In real space reflection of rays from objects is perceived at the location of the observer, that is, on the principle of central projection. Correct reproduction of colors takes place provided that the components of the color intensities of the corresponding surface points in the global (object) and screen coordinate systems coincide.**

The authors propose methods to improve the performance of texture mapping and realism of shading, in particular, the methods perspective-correct texturing and Phong shading.

Keywords: **texturing, texture mapping, rendering, computer graphic, shading, Phong shading.**

I. Introduction

When forming graphical images is solved by a twofold task – improve performance and enhance realism. Today the performance of graphical tools sufficient for the formation of images according to their visual properties similar to the photos, you have achieved photographic quality.

To ensure high realism is important to consider the perspective transformation of polygons in the determination of pixel colors: in the texture mapping and shading.

The perspective-correct formation of colors is used both for shading and texture mapping.

When coloring the surfaces of three-dimensional objects, the methods of Gourund and Phong are most often used. The question of the prospective correct reproduction of colors according to these methods is considered, respectively, in [1, 2, 3].

In the tasks of texturing, you need to find the relationship between the screen coordinates and texture coordinates. In order to ensure high productivity, the linear and quadratic functions are often used in perspective-correct texture mapping [4, 5]. In such approaches, a molded image may have artifacts and does not always faithfully reproduce the perspective of an object. In order to increase the realism of perspective-correct texturing [4, 5, 6], use of nonlinear functions, the calculation of which involves the implementation of labor-intensive operations.

II. Shading with the Perspective

Ignoring the depth of the object in the calculation of the vectors leads to error computing its orthogonal components, which can be calculate by the formula

$$\Delta I = I_A + (I_B - I_A)\cdot\frac{u\cdot z_1}{z_2 - u\cdot(z_2 - z_1)} - I_A -$$
$$-(I_B - I_A)\cdot u = (I_B - I_A)\cdot\left(u - \frac{u\cdot z_1}{z_2 - u(z_2 - z_1)}\right) = \tag{1}$$
$$= (I_B - I_A)\cdot u\cdot\left(1 - \frac{1}{\frac{z_2}{z_1} + u\left(1 - \frac{z_2}{z_1}\right)}\right),$$

replacing the intensity value of the color value of the orthogonal component.

For perspective-correct reproduction of colors using Phong shading it is necessary to use non-linear interpolation of normal vectors using a variable t_w. Unfortunately, the calculation t_w according to the formula [1]

$$t_w = \frac{Z_{\grave{A}w}\cdot t_v}{Z_{Bw} - t_v(\cdot Z_{Bw} - Z_{Aw})} \tag{2}$$

provides for the execution of a division operation for each current value of t_v. Consider the approximation of t_v to simplify the hardware implementation. Since the dependence is nonlinear, using linear interpolation on the whole interval variable is excluded. Approximation t_w second degree polynomial $a\cdot t_v^2 + b\cdot t_v + c$. Find unknown a, b, c. To do this we set up a system of equations using three points $t_v = 0,\ t_v = 1,\ t_v = 1/2.$

$$\begin{cases} c = 0, \\ a + b + c = 1, \\ \frac{1}{4}\cdot a + \frac{1}{2}\cdot b + c = \frac{Z_{\hat{A}w}}{Z_{Bw} + Z_{Aw}}. \end{cases}$$

The system has such a solution

$$a = \frac{2\cdot\left(Z_{Bw} - Z_{Aw}\right)}{\left(Z_{Bw} + Z_{Aw}\right)},\ b = \frac{\left(3\cdot Z_{Aw} - Z_{Bw}\right)}{\left(Z_{Bw} + Z_{Aw}\right)},$$

$$c = 0.$$

If $\hbar = \frac{Z_{Bw}}{Z_{Aw}}$, then $a = \frac{2\cdot\left(\hbar - 1\right)}{\left(\hbar + 1\right)},\ b = \frac{\left(3 - \hbar\right)}{\left(\hbar + 1\right)}.$

The quadratic approximation gives satisfactory results only for $\hbar \le 3$. In figure 1 shows a graph of change of the absolute error of the approximation from t_v , $\hbar$.

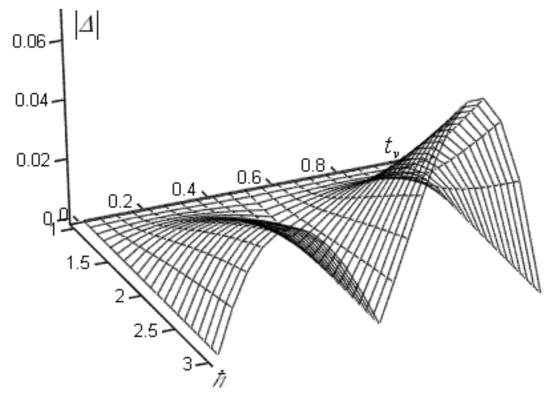

Fig. 1. The dependence of the modulus of the absolute error of the approximation from t_v, $\hbar$

Higher accuracy of approximation can be achieved if the use of piecewise quadratic interpolation on two periods of change t_v. For $0 \le t_v \le 0{,}5$

$$a = \frac{8\cdot Z_{Aw}\cdot\left(Z_{Bw} - Z_{Aw}\right)}{\left(Z_{Bw} + Z_{Aw}\right)\left(3\cdot Z_{\hat{A}w} + Z_{Aw}\right)},$$

$$b = \frac{\left(3\cdot Z_{Aw} + Z_{Bw}\right)}{\left(Z_{Bw} + Z_{Aw}\right)\left(3\cdot Z_{\hat{A}w} + Z_{Aw}\right)},\quad c = 0.$$

For $0{,}5 < t_v \le 1$

$$a = \frac{-8\cdot Z_{Bw}\cdot\left(Z_{Aw} - Z_{Bw}\right)}{\left(Z_{Bw} + Z_{Aw}\right)\left(3\cdot Z_{\hat{A}w} + Z_{Aw}\right)},$$

$$b = \frac{2\cdot\left(9\cdot Z_{Aw} - 5\cdot Z_{Bw}\right)\cdot Z_{Bw}}{\left(Z_{Bw} + Z_{Aw}\right)\left(3\cdot Z_{\hat{A}w} + Z_{Aw}\right)},$$

$$c = \frac{3\cdot\left(Z_{Aw} - Z_{Bw}\right)^2}{\left(Z_{Bw} + Z_{Aw}\right)\left(3\cdot Z_{\hat{A}w} + Z_{Aw}\right)}.$$

The analysis showed that in this case $\hbar = 2, 3, 4, 5$ the maximum modulus of the relative error does not exceed, 1% , 4%, 8%, 13%. With regard to three-dimensional objects, $\hbar$, as a rule, does not exceed 3.

Consider using approximation by third-order polynomial of the form $a\cdot t_v^3 + b\cdot t_v^2 + ct + d$. For finding the unknown we set up a system of four equations. To do this, make the value of the polynomial equal t_w (see formula 2) in points $t_v = 0,\ 1/3,\ 2/3,\ 1$. Find:

$$a = \frac{9\cdot\left(Z_{Bw} - Z_{Aw}\right)^2}{\left(2\cdot Z_{Bw} + Z_{Aw}\right)\cdot\left(Z_{Bw} + 2\cdot Z_{Aw}\right)},$$

$$b = \frac{-9\cdot\left(Z_{Bw} - Z_{Aw}\right)\left(Z_{Bw} - 2\cdot Z_{Aw}\right)}{\left(2\cdot Z_{Bw} + Z_{Aw}\right)\cdot\left(Z_{Bw} + 2\cdot Z_{Aw}\right)},$$

$$c = \frac{\left(2\cdot Z^2_{Bw} - 4\cdot Z_{Aw}\cdot Z_{Bw} + 11\cdot Z_{Aw}\right)}{\left(2\cdot Z_{Bw} + Z_{Aw}\right)\cdot\left(Z_{Bw} + 2\cdot Z_{Aw}\right)}.$$

The analysis showed that when using the cubic interpolation achieves better accuracy compared to piecewise-quadratic interpolation. For example, when $\hbar = 2, 3, 4, 5$ the maximum modulus of the relative error does not exceed, 0,64 %, 2,9 %, 6,3 %, 10,6 %.

III. Improving the Performance of Texturing with Perspective

Finding texture coordinates is a time consuming procedure because it requires the execution of complex operations for each pixel according to the formula [4, 5].

$$u = \frac{Ax + By + C}{Gx + Hy + I},\quad v = \frac{Dx + Ey + F}{Gx + Hy + I}.$$

If $x_{i+1} = x_i + 1$, then

$$u_{i+1} = \frac{A_1\cdot(x_i + 1) + B_1\cdot y_i + C_1}{D\cdot(x_i + 1) + E\cdot y_i + F} = \frac{(A_1\cdot x_i + B_1\cdot y_i + C_1) + A_1}{(D\cdot x_i + E\cdot y_i + F) + D}.$$

For u_0 the formula has the form:

$$u_0 = \frac{A_1\cdot(x_0 + 0) + B_1\cdot y_i + C_1}{D\cdot(x_0 + 0) + E\cdot y_i + F} = \frac{A_1\cdot x_0 + B_1\cdot y_i + C_1}{D\cdot x_0 + E\cdot y_i + F}.$$

For u_1:

$$u_1 = \frac{A_1 \cdot (x_0 + 1) + B_1 \cdot y_i + C_1}{D \cdot (x_0 + 1) + E \cdot y_i + F} = \frac{A_1 \cdot x_0 + A_1 + B_1 \cdot y_i + C_1}{D \cdot x_0 + D + E \cdot y_i + F}.$$

For u_2:

$$u_2 = \frac{A_1 \cdot (x_0 + 2) + B_1 \cdot y_i + C_1}{D \cdot (x_0 + 2) + E \cdot y_i + F} = \\ = \frac{A_1 \cdot x_0 + A_1 \cdot 2 + B_1 \cdot y_i + C_1}{D \cdot x_0 + D \cdot 2 + E \cdot y_i + F}.$$

Thus, for u_n the formula has the form:

$$u_n = \frac{A_1 \cdot (x_0 + n) + B_1 \cdot y_i + C_1}{D \cdot (x_0 + n) + E \cdot y_i + F} = \\ = \frac{A_1 \cdot x_0 + A_1 \cdot n + B_1 \cdot y_i + C_1}{D \cdot x_0 + D \cdot n + E \cdot y_i + F}. \quad (2)$$

A similar formula can be written for v_n.

$$v_n = \frac{A_2 \cdot x_i + B_2 \cdot y_o + B_2 \cdot n + C_2}{D \cdot x_i + E \cdot y_0 + E \cdot n + F}.$$

From the given formulas it is visible that for calculation of each texel computer needs to perform 2 operations of division, 8 operations of addition and 8 multiplications.

As can be seen from formulas (2), the value of the expressions $A_1 \cdot x_0 + B_1 \cdot y_i + C_1$ and $D \cdot x_0 + E \cdot y_i + F$ for each n remain unchanged, and therefore can be calculated once for each rasterization line using formula:

$$u_t = A_1 \cdot x_0 + A_1 \cdot n + B_1 \cdot y_i + C_1, \;\; u_b = D \cdot x_0 + E \cdot y_i + F.$$

where u_t and u_b constant part of the numerator and denominator of formula (1) in accordance.

Therefore, u_n can be calculated according using the formula

$$u_n = \frac{u_t + A_1 \cdot n}{u_b + D \cdot n}. \quad (3)$$

Thus, for calculating coordinates of all texels, variables, u_{1t} and u_{1b} are constant, and their value is sufficient to calculate once. Similarly you can calculate v_n.

This simplification reduces the number of operations of addition and multiplication to 4 for each.

Based on these mathematical relationships it is possible to offer algorithm of parallel calculation of texels coordinates: for each rasterization line at first calculates the parameters u_t and u_b, then calculate values of the numerator and denominator in parallel by the formula:

$$w_n = u_t + A_1 \cdot n, \;\; v_n = u_b + D \cdot n,$$

then the division operation to calculate the coordinates

$$u_n = \frac{w_n}{v_n}.$$

The sequence of operations of the algorithm shown in figure 2.

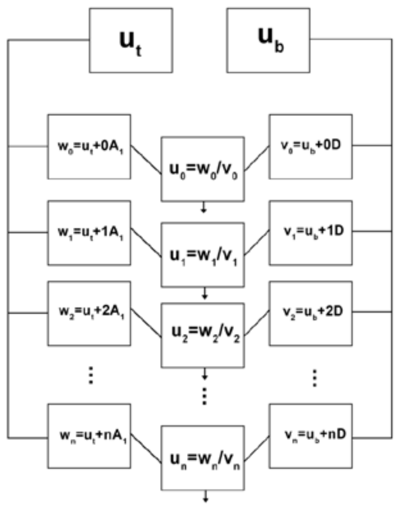

Fig. 2. Parallel calculating texel coordinates

Since for each x, n is increased by 1, there is such a formula:

$$w_n = w_{n-1} + A_1, \;\; v_n = v_{n-1} + D. \quad (4)$$

In accordance with the formulas (4), we can offer a consistent algorithm for calculating texture coordinates: w_n and v_n for each x calculated by adding to w_{n-1} and v_{n-1}, that was calculated for x-1, A_1 and D. The sequence of operations of the algorithm shown in figure 3.

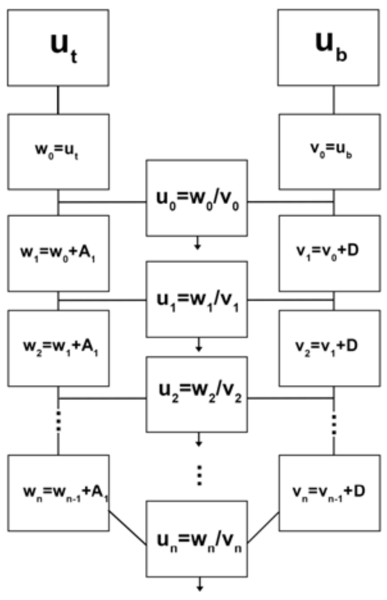

Fig. 3. Sequential calculating texel coordianates

The computing of texture coordinates by this algorithm can also be accelerated using parallel computing. One possible approach to parallelization is the parallel rasterization of several lines simultaneously. However, this approach will not be productive enough in cases when the number of pixels per row is much higher than the number of rows. Therefore, it is advisable to use means of parallel calculations within a single line.

Let's consider parallel computing of cordiant texels for pixels located at even and odd positions in the rasterization line (fig. 4).

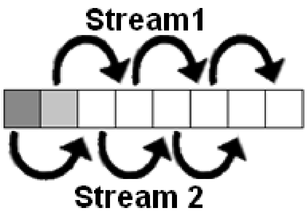

Fig. 4. Parallel computing texels coordinates for two threads on the same rasterization line

If $w_n = w_{n-1} + A_1$, a $v_n = v_{n-1} + D$ then:

$$w_{n+1} = (w_{n-1} + A_1) + A_1,\ \ v_{n+1} = (v_{n-1} + D) + D.$$

Thus, parallel computation of the texture coordinates of pixels on odd and even positions is possible according to the formulas:

$$w_n = w_{n-2} + 2A_1,\ v_n = v_{n-2} + 2D. \tag{5}$$

In this case, the coordinates texels for the first two points are defined by the formula (3).

Based on the formulas (5) and (5) to establish the relationship which allows parallel rasterization the line in an random number of threads using formulas:

$$w_n = w_{n-k} + kA_1,\ \ v_n = v_{n-k} + kD,$$

whre k – the number of concurrent threads.

It is also possible a parallel calculation of the coordinates in two streams by simultaneous rasterization of line in two directions (fig. 5). From right to left according to the formula (4), and from left to right according to the formula:

$$w_n = w_{n+1} - A_1,\ \ v_n = v_{n+1} - D.$$

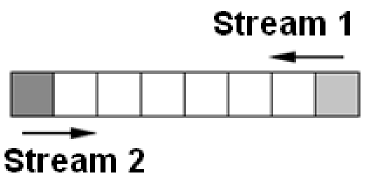

Fig. 5. Parallel computation of the coordinates in the two streams with a counter direction of rasterization

The figure 6 show that the proposed method makes it possible to increase productivity perspective-correct texturing 26%. Testing was conducted on the Intel i7 2600K CPU and GPU AMD RX460.

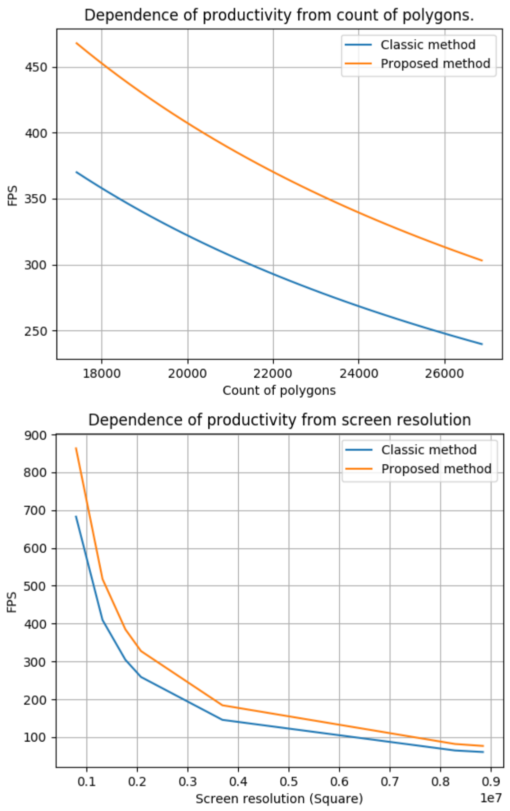

Fig. 6. Comparison of the performance of the proposed method with the classical

III. Conclusion

Proposed methods to improve the performance of texture mapping and realism of shading, in particular, the methods perspective-correct texturing and Phong shading.

References

[1] G. F. Ahmed, R. Barskar, J. Bharti, and N. S. Rajput, “Content Base Image Retrieval Using Fast Phong Shading,” 2010 International Conference on *Computational Intelligence and Communication Networks*, 2010.

[2] R. F. Lyon, “Phong Shading Reformulation for Hardware Renderer Simplification”, *Apple Technical Report #43*, August 2, 1993.

[3] D. A. Kulagin, "Models of shading. Flat model. Shading on Guro and Fong ", *Computer graphics. Theory, algorithms, examples on C++ and OpenGL*. [Online]. Available http://compgraphics.info/3D/lighting/shading_model.php

[4] P. S. Heckbert, “Survey of Texture Mapping,” *IEEE Computer Graphics and Applications*, vol. 6, no. 11, pp. 56–67, 1986.

[5] F. Tsai and H. C. Lin, “Polygon-based texture mapping for cyber city 3D building models,” *International Journal of Geographical Information Science*, vol. 21, no. 9, pp. 965–981, 2007.

[6] X. U. Ying, “An Improved Texture Rendering Technique Based on MipMap Algorithm”, *Journal of Mianyang Normal University*, 2013, 5: 017.