# RESEARCH ON GRAPHICS ALGORITHM AND REAL-TIME DRAWING TECHNOLOGY IN 3D DATA FIELD

*Yili Tan[1], Yourong Wang[2]*
*[1]Department of Statistics, College of Science, North China University of Science and technology, Tangshan, China,*
*[2]Department of Basic, Tangshan College, Tangshan, 063000, China*
*E-mail: tangyili_tyl@163.com*

**Abstract:** With the continuous development of graphic image processing technology, three-dimensional data is being widely used in various fields. However, difficulty still exists in graphic and image processing in three-dimensional data field. This paper applied ray projection algorithm and interpolation algorithm for graphic processing and combined acceleration algorithm with texture mapping technology to draw real-time images. By drawing a semi-sphere with the computer, the effect of the algorithms was detected. Meanwhile, in order to reduce the influence of image noise in the three-dimensional data field, three-dimensional polishing algorithm was applied for optimization. It was found that the drawn images were clear and the drawing speed was fast, suggesting that the algorithms were helpful in drawing 3D graphics and worth being promoted for wide application.

**Keywords**: 3D data, Graphics Algorithm, Real-Time Rendering, Drawing Technology, Noise Processing

## 1. Introduction

With the continuous development of the Internet and multimedia technology, the computer's configuration memory and disk are constantly expanding, leading to the realization of drawing of images and graphics through the computer [1], which improves the utilization of data in the three-dimensional data field and makes people play a dominant role in the process by realizing the interaction between people, data and images [2].

Though scientific calculation, the data are displayed in visual forms through graphics algorithm and rendering technique. Therefore, the method is well applied in various fields [3]. However, real time drawing of images in the three-dimensional data field remains difficult as it requires the overcoming of severe calculation challenges.

Facing complex and large-scale scenes, the current graphics hardware and software technology can not generate film-level three-dimensional motion image sequence instantly [4-6].Tang [7] constructed a three-dimensional real-time rendering algorithm test platform, eliminated the differences between various hardware with render pipelines and realized the conversion from the three-dimensional model to two-dimensional high-quality images with good display effects.

Macdougall et al. [8] applied the new atomic signature detection software to 3D visualization technology and found that tiled array of three-dimensional display independent drive combined with GPU could display high-definition, sync and function-related pictures, with intuitive visual effects. GPU is a microprocessor to calculate images on computers or mobile devices, which is capable of transferring and driving information displayed by the computer system, delivering the scanned signals to the displayer, controlling the displayer and connecting the displayer with to the PC motherboard [9].

This paper studied the algorithms and real-time rendering techniques of graphics in order to provide some technical support for the drawing of 3D images. It was found that ray projection algorithm, interpolation algorithm, acceleration algorithm and texture mapping algorithm had a positive effect on the graphic rendering in 3D data field.

## 2. Graphics Algorithm

### 2.1. Ray projection algorithm

Ray projection algorithm displays the user information on the screen through parallel projection or perspective projection.

The ray projection algorithm in the three-dimensional data field is shown in Figure 1.

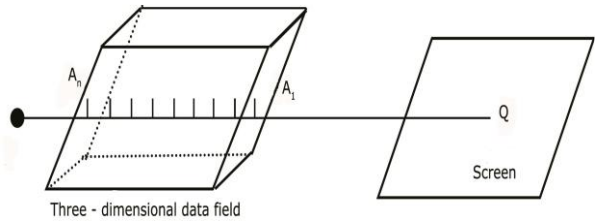


*Figure 1: The ray projection algorithm in the three-dimensional data field*

The three-dimensional spatial discrete data are preprocessed and the data of the three-dimensional data field is resampled. As shown in Figure 1, a ray is emitted from the pixel point Q on the screen to the three-dimensional data field along the observation direction, which enables Q to enter the data field. $A_1$ refers to the starting point entering the data field, $A_n$ refers to the leaving point. The distance between the two points is equally sampled. Set the sampling point to be $A_i$ ($1 \le i \le n$), the coordinate to be (x, y, z) and the spacing to be $\Delta s$, then the spacing $\Delta s'$ between pixel points on screen 1 is calculated. If $\Delta s'$ is greater than the data sampling spacing $\Delta s''$, then $\Delta s = \Delta s'$; otherwise, $\Delta s = \Delta s''$. Since the value of the sampling points does not include the starting and ending point, the remaining seven vertices are solved by triple linear interpolation. As for the image synthesis and color rendering, the first step is to find the corresponding color for the data value of the sapling point. Then, the color values corresponding to the sampling points are output and the output values correspond to the rendering color of the pixel point Q.

The specific formula is as follows:

Make

$$A_{(i)} = A_{(1)} + \overrightarrow{Dir} * i \tag{1}$$

In the equation, $\overrightarrow{Dir}$ refers to the ray direction and $i$ refers to the sampling sequence number.

## 2.2. Interpolation algorithm

In the trilinear interpolation, the sampling point may also be a data point that does not exist. Therefore, the value of $A_i$ is calculated through the surrounding eight vertices using trilinear interpolation.

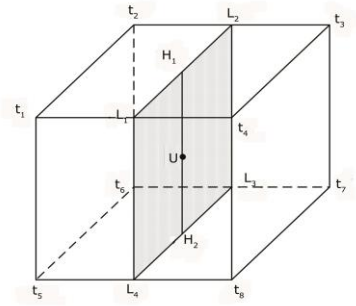


*Figure 2: Three-dimensional interpolation sampling points*

As shown in Figure 2, $t_1$ to $t_8$ are gridded 3D data points, expressed with $w_i = f(A_i), i = 1,2,\ldots,8$. U refers to the resampling point and the value of the interpolated point can be solved by the above eight points. The three-dimensional interpolation obtains the approximation of the required points through 3 rounds and 7 times of one-dimensional interpolation based on the data of the eight points. Set $t_5$ as the origin of the axis and the side length of the figure is 1. Four times of linear interpolation is performed on the y direction based on line segments of ($T_5$, $t_8$), ($t_7$, $t_6$), ($t_1$, $t_4$), ($t_3$, $t_2$).

$$\begin{cases} l_1 = yw_4 + (1-y)w_1 \\ l_2 = yw_3 + (1-y)w_2 \\ l_3 = yw_7 + (1-y)w_6 \\ l_4 = yw_8 + (1-y)w_5 \end{cases} \tag{2}$$

Whereby the vertex function values of $L_1$ to $L_4$ of the shaded portion can be obtained, $l_i = f(L_i), i = 1,2,3,4$. Subsequently, linear interpolation is performed twice on ($L_1$, $L_2$) and ($L_4$, $L_3$) in the z-direction.
Thus, the $H_1$ and $H_2$ function values are:

$$\begin{cases} h_1 = (l_3 - l_4)z + l_4 \\ h_2 = (l_2 - l_1) + l_2 \\ h_1 = (w_7 - w_8)yz + (w_6 - w_1)(1-y)z + w_8 y + (1-y)w_5 \\ h_2 = (w_4 - w_1)yz + (w_2 - w_1)(1-y)z + w_4 + (1-y)w_1 \end{cases} \tag{3}$$

Through the last linear interpolation in the x direction, the following can be obtained:

$$F(U) = (h_1 - h_2)x + h_2 \tag{4}$$

Then, the value of the $A_i$ sampling point is:

$$F(U) = (1-x)(1-y)(1-z)w_5 + (1-x)y(1-z)w_8 + (1-x)yzw_7 + (1-x)(1-y)zw_6 + x(1-y)(1-z)w_1 + xy(1-z)w_4 + xyzw_3 + x(1-y)zw_2 \quad (5)$$

# 3. Implementation of Drawing Technology

## 3.1. Acceleration algorithm

The acceleration algorithm proposed in this paper is to remove the rendering elements that cannot be recognized by the naked eye from the scene through back removal, visual field removal and occlusion removal. When the direction of the normal vector of the drawing element deviates from the viewing direction, back removal is applied. When the drawing element is outside the visual range of the viewing body, visual field removal is selected. When the drawing element is blocked by other opaque drawing elements, occlusion removal is considered. At the same time, the acceleration algorithm can handle the details of the hierarchy [10]. In this paper, the IBR algorithm is used to improve the drawing speed, including elve drawing, bulletin boards and substitute technologies. Besides, the algorithm also works for the representation of clouds and flames.

## 3.2. Texture mapping

As the light projection algorithm can not reflect the texture details on the surface of the object, this paper adds the texture mapping algorithm. There are color, geometry and process texture based on the different forms of material texture. Color texture includes pattern and text. Geometric texture is in a surface micro-geometrical shape.

Process texture is mainly in water, smoke and other irregular shapes and mostly dynamic natural scene. Texture mapping is the process of mapping the texture pixels in the texture space to pixels in the screen space [11]. The common two-dimensional texture can display the texture in a three-dimensional scene to form an image. The function is expressed as:

$$p(m,n) = \begin{cases} 0, |m\times 8| \times |n\times 8| \text{ o}ddnumber \\ 1, |m\times 8| \times |n\times 8| \text{ even number} \end{cases} \quad (6)$$

In the equation, x and y are the values within the unit square area, the value of m and n is less than or equal to 0 and greater than or equal to 1 respectively. In order to map a two-dimensional image into a three-dimensional scene, we need to establish the correspondence between the three-dimensional spatial coordinates (x, y, z) and the texture space coordinates (m, n), which means to perform parameter setting on the object surface and calculate the parameters afterwards to obtain the texture value at (m, n) point. A frequently adopted mapping method is spherical mapping. The spherical parameter equation is:

$$\begin{cases} x = \cos(2\pi m)\cos(2\pi n) \\ y = \sin(2\pi m)\cos(2\pi n) \\ z = \sin(2\pi n) \end{cases} \quad (7)$$

Then, the counter seeking parameters of a random point $(x, y, z)$ on the sphere is obtained:

$$(m,n) = \begin{cases} (0,0), (x,y) = (0,0) \\ (\frac{1-\sqrt{1-(x^2+y^2)}}{x^2+y^2}x, \frac{1-\sqrt{1-(x^2+y^2)}}{x^2+y^2}y), (x,y) \neq (0,0) \end{cases} \quad (8)$$

# 4. Experimental Testing

Applying the above methods, spherical real-time rendering is carried out in the three-dimensional data field, combined with vertex changes and chip coloring.

The light chip coloring algorithm is as follows:

```
void Light Shader (float position :TCOORD1,
float normal :TCOORD2,
out float color :COLOR,
uniform float global Ambient,
uniform float light Color,
uniform float light Position,
uniform float shininess,
uniform float eye Position,
uniform float sa,
uniform float sc,
uniform float sd,
uniform float se) {
 float f Position = position.xyz;
float f Normal = normalize(normal);
float f Emissive = sa;
 float f Ambient = sc * global Ambient;
 float f Light = normalize(light Position – f Position);
 float f Diffuse Light = max(dot (f Normal, f Light), 0);
 float f Diffuse = sd * light Color * f Diffuse Light;
 float f Vertical = normalize(eye Position– f Position);
float f Horizontal = normalize(f Light + f Vertical);
float f Specular Light = pow (max(dot(f Normal, f Horizontal), 0), shininess);
 if (f Diffuse Light <= 0) {
 f Specular Light = 0;
 }
 float f Specular = se * light Color * f Specular Light;
```

```
color.xyz = f Emissive + f Ambient + f Diffuse + f Specular;
color.w = 1;
}
```

The algorithm for calculating the spotlight lighting effect is as follows:

```
float dual Cone Spot Light (float f Position, Light light) {
float f Vertical = normalize(f Position–light position);
float cos Outer Cone = light cos Outer Cone;
float cos Inner Cone = light cos Inner Cone;
float cos Direction = dot(f Vertical, light direction);
return Smoot Step (cos Outer Cone, cos Inner Cone, cos Direction);
}
void spot Lighting with Attenuation(Light light, float f Position, float f Normal,
float eye Position, float shininess,
out float diffuse Result, out float specular Result) {
// Compute attenuation
float atten Factor = attenuation(f Position, light);
// Compute spotlight effect
float spot Light Effect = dual Cone Spot Light (f Position, light);
// Compute the diffuse lighting
float f Light = normalize(light position – f Position);
float diffuse Light = max(dot(f Normal, f Light), 0);
diffuse Result = atten Factor * spot Light Effect * light color * diffuse Light;
// Compute the specular lighting
float f Vertical = normalize(eye Position – f Position);
float f Horizontal = normalize(f Light +f Vertical);
float f Specular Light = pow (max(dot (N, f Horizontal), 0), shininess);
if (diffuse Light <= 0) {
f Specular Light = 0;
}
Specular Result = atten Factor * spot Light Effect * light
}
```

Then, grinding algorithm is applied to perform noise processing on the drawn sphere. The specific algorithm is as below:

The biorthogonal wavelet multivariate of the grid C is represented by $\{C_0, C_1, ....C_n\}$, $C_0$ refers to the grid basis, $R_\mu$ refers to the detail of the $\mu$ layer of grid,

$$C_\mu = C_{\mu-1} \oplus R_\mu, \mu = 1,2,...,n$$

The details after the $\vartheta$ layer are added to the grid in sequence and grinded with the grinding algorithm.

# 5. Detection Results

## 5.1. Drawing of the spherical body

The initial sphere body is drawn with the computer, as shown in Figure 3.

After the establishment of the three - dimensional coordinate system, the data in front of the 3D surface is displayed correctly while that in the back end is blocked.

At the same time, the data on the left and right sides are also blocked in varying degrees.

Combined with the above algorithms, grinding and noise cancelling are performed, the result of which is shown in Figure 4.

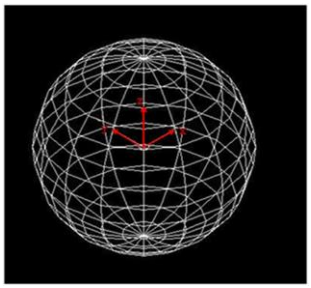

*Figure 3: Initial sphere body image*

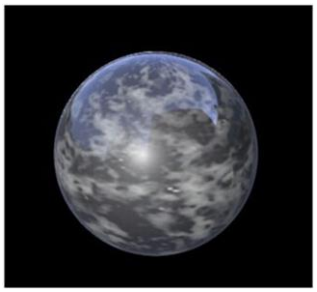

*Figure 4: Final sphere body rendering image*

## 5.2. Coloring algorithm and spotlight lighting effect detection

Petal shape is added to the spherical surface, with coloring and the spotlight, as shown in Figure 5.

Under the influence of the spotlight, the color of the upper part of the surface is bright while the lower part is in single green.

The back of the spherical body is still blocked, while the front is displayed in the screen with good effects.

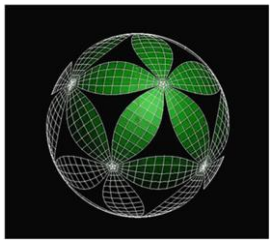

*Figure 5: The colored spherical body with spotlight lighting effect*

## 6. Conclusion

In general, the structure of three-dimensional data can be divided into wireframe modeling, geometric voxel modeling and surface modeling [12]. The techniques that influence the drawing of graphics are styling techniques, graphics transformation algorithms and display technology, with many more algorithms involved [13]. In this paper, ray projection algorithm, interpolation algorithm, acceleration algorithm and texture mapping algorithm are applied. As a kind of direct body rendering algorithm, the ray projection algorithm is also the most classic image rendering algorithm, which is combined with the interpolation algorithm in this study [14]. The interpolation algorithm can calculate the known discrete data to encrypt the data that is not known around and interpolate images with low resolution to ones with high resolution [15]. Accelerated algorithm is an important algorithm for implementing rendering techniques, which can improve the realistic effect of 3D graphics by removing unnecessary algorithms, with high potential for improvement [16]. The texture mapping technique makes the display of the details of the three-dimensional image clearer [17]. In this paper, we used the above algorithms to study the graphic drawing in 3D data field and detected the results with computer, which found that the real-time rendering three-dimensional effect was better with the noises processed, suggesting that the combination of the algorithm and real-time rendering technology could further improve the drawing of graphics in 3D data field and therefore deserved to be developed.

## Acknowledgments

This work is supported by Natural Science Foundation of Hebei Province (NO. A2015209040).
Conflicts of Interest: The authors declare no conflict of interest.

## References

[1] Prisacariu, V A, Kähler O, Murray D W, et al D 2015 Real-time 3D tracking and reconstruction on mobile phones. IEEE Transactions on Visualization & Computer Graphics, 21(5): 557.
[2] Bakker, P M, Gerritsen, S, Cao, Q, et al 2015 3D RTM-based wave-path tomography: theory and applications to synthetic and field data. In: 2015 SEG Annual Meeting, New Orleans, Louisiana on 18-23 October, 2015, pp. 5259-5264.
[3] Oh, J W, Alkhalifah, T 2016 3D elastic-orthorhombic anisotropic full-waveform inversion: Application to field OBC data. In: 2016 SEG International Exposition and Annual Meeting, Dallas, Texas on 16-21 October, 2016, pp. 1206-1210.
[4] Scherzer D, Yang L, Mattausch O, et al 2011 A Survey on temporal coherence methods in real-time rendering. Eurographics State of the Art Reports, Llandudno, UK, pp. 2378–2408.
[5] Tesnim, A, Jacques, C, Joseph, S 2013 Rigorous implementation of real-time systems u2013 from theory to application. Mathematical Structures in Computer Science, 23(4): 882-914. Thanou, D, Chou, P, Frossard, P 2015 Graph-based compression of dynamic 3D point cloud sequences. IEEE Transactions on Image Processing, 25(4): 1765-1778.
[6] Thanou D, Chou P, Frossard P. Graph-based compression of dynamic 3D point cloud sequences[J]. IEEE Transactions on Image Processing, 2015, 25(4):1765-1778.
[7] Tang, K. 2013 Research and development of test platform for 3D real-time rendering algorithm. Computer Knowledge & Technology, 3: 144-147.
[8] Macdougall, P J, Henze, C E, Volkov, A 2016 Volume-rendering on a 3D hyperwall: A molecular visualization platform for research, education and outreach. Journal of Molecular Graphics & Modelling, 70: 1-6.
[9] Mittal, S, Vetter, J S 2015 A Survey of CPU-GPU Heterogeneous Computing Techniques. ACM Computing Surveys, 47(4): 1-35.
[10] Radwan, I, Dhall, A, Goecke, R 2013 Monocular image 3D human pose estimation under self-occlusion. In: Proceedings of the 2013 IEEE International Conference on Computer Vision, Sydney, Australia on 1-8 December 2013, pp. 1888-1895.
[11] Pascali, M A, Petronio, C 2012 Branched covers of the sphere and the prime-degree conjecture. Annali Di Matematica Pura Ed Applicata, 191(3): 563-594.
[12] Dai, H, Liu, Z, Hu, J 2010 Multi-scale 3D geological digital representation and modeling method based on octree algorithm. In: 6th International Conference on IEEE, Chengdu, China on 23-25 Sept. 2010, pp. 1-3.
[13] Zhang B L, Yang J C, Lee H R. Image Space Coordinates Extraction Algorithm Based on the Perspective Projection Transformation Matrix. Applied Mechanics & Materials, 2014, 644-650:4368-4371.
[14] Liu L P, Sun Y X, Guan T J, et al 2014 Improved rapid interpolation ray casting algorithm. Advanced Materials Research, 846-847: 1247-1251.
[15] Yaroslavsky, L P 2015 Signal sinc-interpolation: A fast computer algorithm. Bioimaging, 4(4): 225-231. Zhang, B L, Yang, J C, Lee, H R 2014 Image Space Coordinates Extraction Algorithm Based on the Perspective Projection Transformation Matrix. Applied Mechanics & Materials, 644-650: 4368-4371.
[16] Garcia-Molla V M, Liberos A, Vidal A, et al 2014 Adaptive step ODE algorithms for the 3D simulation of electric heart activity with graphics processing units. Computers in Biology & Medicine, 44(C): 15.
[17] Yan Y, Chen X, Gao F, et al. 2015 A roof-contour guided multi-side interpolation method for building texture-mapping using remote sensing resource. In: 2015 IEEE Geoscience and Remote Sensing Symposium, Milan, Italy on 26-31 July 2015, pp. 2998-3001.