THE DESIGN OF THE PERSPECTIVE TEXTURE MAPPING FOR 3D COMPUTER GRAPHICS IN RASTERIZER MERGED FRAME BUFFER TECHNOLOGY

Seung-Gi Lee*, Woo-Chan Park*, Won-Jong Lee*, Woo-Nam Jung* and Tack-Don Han*

*Dept. of Computer Science, Yonsei University, Seoul 120-749,Korea

***E-mail:** {sklee, chan, airtight, wnjung, hantack}@kurene.yonsei.ac.kr

Abstract

Recently, more advanced image quality and many techniques are needed in 3D computer graphics. Especially texture mapping support is the major issue in the design of high performance 3D graphics system. In this paper, a high-performance 3D graphics hardware is designed in the point of texture mapping. Proposed system is composed of a single-chip of rasterizer and frame buffer using processor-memory integration method, which is adapted for process of the high-performance 3D graphics. Also, texture mapping unit is designed using the processor-memory integration technology to provide sufficient bandwidth. Therefore, the tremendous memory bandwidth and the long latency required for texture mapping can be reduced by this approach.

I. INTRODUCTION

A remarkable advance in 3D computer graphics industry has been progressed for last ten years. High performance graphics accelerators make it possible to process real-time images of high quality not to be compared with last ones. Recently, Many researchers studied various methods for removing aliasing and mapping techniques, i.e., texture mapping, bump mapping, and environment mapping for realistic scene and this trend is accelerated in these days [1].

To support realistic 3D computer graphics image, the texture mapping is a major issue in the design of 3D graphics hardware. But, the tremendous memory access and the long latency are required to support a texture-mapped image. While, processor-memory integration technology was adapted in the high performance 3D graphics system [9], [14], [15]. This approach is the most promising candidate for high performance 3D graphics system. Then the main focus of these systems is to reduce memory bandwidth between the frame buffer and the rasterizer, but texture mapping is not considered.

In this paper, the structure of a rasterizer integrated with a frame buffer is designed using the processor-memory integration technology. And texture mapping unit is designed using the processor-memory integration technology to provide sufficient bandwidth, as similar to [10]. The proposed system presents the exact method for the perspective projection texture mapping. By using perspective projection method, the distortion invoked by linearly interpolating 2D texture data can be eliminated.

This paper is organized as follows. As a background, Section 2 describes the rasterization process, Section 3 describes the texture mapping overall, and Section 4 discusses the interpolation technique for perspective texture mapping and the problem invoked by transforming from 3D object space into 2D screen space and the rendering algorithm are analyzed. Section 5 provides proposed 3D rendering system and describes the differences with previous one. Section 6 summarizes the conclusions and outlines areas for future research.

II. RASTERIZING PROCESS

Rasterizing need quite processing. Because interpolation is needed to compute color, texture and transparency before final image dispatch to frame buffer. One triangle is defined three vertices (x, y, z) and color values of each vertex RGB α. Before scan conversion processing, the increase of triangle of edge walk and span interpolation must be computed.

In the way of calculating the rate of increase, triangular slope, parameters of gradient, coordinate division for perspective texture and etc. must be processed division against several parameters. In this stage, at least 5 number of lookup tables are needed for possible processing of prefect pipeline [12]. Using the reciprocal table with guard bit is practical way that is reduced latency and size of rasterizer [13]. And the rate of increase, that is the color, depth and texture coordinate value of triangle, is computed by the same reciprocal values.

$$\frac{\partial z}{\partial x} = \frac{(z_2 - z_1)\cdot(y_3 - y_1) - (z_3 - z_1)\cdot(y_2 - y_1)}{(x_2 - x_1)\cdot(y_3 - y_1) - (x_3 - x_1)\cdot(y_2 - y_1)}$$

$$\frac{\partial z}{\partial y} = \frac{(z_3 - z_1)\cdot(x_2 - x_1) - (z_2 - z_1)\cdot(x_3 - x_1)}{(x_2 - x_1)\cdot(y_3 - y_1) - (x_3 - x_1)\cdot(y_2 - y_1)}$$

So, reference of previously computed value is reduced overhead which can be occurred in calculating the rate of increase.

III. TEXTURE MAPPING

When image are mapped in object, colors of the each pixel of object are mapped by correspond color of image. In general, color, which is mapped by image, is needed to conceptually several stages [3]. Sequential images must be reconstructed first, because of image are stored in sampled array. Then next, displayed image must be twisted or bent for properly wrapped with some kind of projective distorted object. (Perhaps, this is occurred presentation of perspective distortion.) Distorted mage is removed high frequency component by filtering. In resampling stage, aliasing are occurred, because of high frequency component. Pixel what are processed texture, are obtained proper color by resampling stage.

Practically, above filtering stage are approximately performed by one of the several ways. Mip-mapping is the most popular method [6]. There are many normalized techniques of above basic texture mapping. It is no need to construct 2-dimensional map. Sampling and filtering can be applied with 1-dimensional or 3-dimensional image [8].

0-7803-6470-8/00/ $10.00 © 2000 IEEE

Texture image are applied to polygon that is assigned texture coordinate to polygon vertex. This texture coordinate index texture image, and are interpolated each pixel of polygon. In present many graphic system, texture mapping are supported by hardware. In the most case, the time required to make scene with texture mapping is less than without texture mapping. Texture mapping method, which are not confined to texture, can be used interesting application graphic drawing method, with air-brush, volume rendering, Phong shading and environment mapping [7].

IV. LINEAR INTERPOLATION FOR PERSPECTIVE PROJECTION MAPPING

1. Affine and Projective Mapping

Two kinds of mapping, what are affine and projective, are used in the 3D graphics. Normal feature of projective mapping from (u,v) to (x,y) of 2D is:

$$x=\frac{au+bv+c}{gu+hv+i},\ y=\frac{du+ev+f}{gu+hv+i}$$

These are more simply expressed by homogeneous matrix notation.

$$(xw \quad yw \quad w)=(uq \quad vq \quad q)\begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$$

There are scale, rotation, translation and shear in the affine mapping. A 2-D projective mapping is affine if $g=h=0$ and $i\neq 0$.

2. Polygon Rendering with Linear Interpolation

Linear interpolation algorithm is [4]:

(1) Associate a record containing the parameters of interest with each vertex of the polygon.

(2) For each vertex, transform object screen coordinate to homogeneous screen space $(xw;\ yw;\ zw;\ w)$ using 4×4 object to screen matrix.

(3) Clip the polygon against plane equation each of the six sides of the viewing frustum (view volume), linearly interpolating all the parameters when vertices are created.

(4) Perform a homogeneous division to compute $x=wx/w,\quad y=yw/w,\quad z=zw/w$.

(5) Scan convert in screen space, by linear interpolation of all parameters, using the parameter values at each pixel for shading.

The step (1) to (4) from this algorithm are processed in geometry computation stage, and the last step is processed in rasterization stage. By the way, for object space coordinate are normalized to screen coordinate system, in this algorithm, homogeneous coordinate are simply performed by divided by scale factor. These invoke the problem in the processing of screen coordinate like texture mapping.

3. Flaws of Interpolation

Linear interpolation is usually used Gouraud shading, Pong shading and texture mapping as well. It is wrong, however, to perform linear interpolation in screen space of parameter, which associated with object space. Form object space to screen space, are performed perspective projection mapping. So above linear interpolation algorithm are only applied mapping by parallel projection or viewing direction, and linear transformation, which screen coordinate space is mapping with. Demerit of linear interpolation is especially appeared with texture mapping. Texture mapping with linear interpolation can be presented perspective scene. Texture is also appeared discontinuity along the horizontal line across the vertex. And, the polygon with several plane are occurred rubber sheet effect and rotational variation. Linear interpolation is only correctly applied, when that is rotational invariation with affine mapping. To make 3D scene, basically use perspective projection method, so linear interpolation is not proper form screen space to object space. Triangulation method is often used, as solution of this problem. There is polygon subdivision, which is the method of linear interpolation in object space parameter of new vertices. A polygon is partitioned in several small polygon then approximation at polygon. This is reduced fast pixel processing rate by increased polygon.

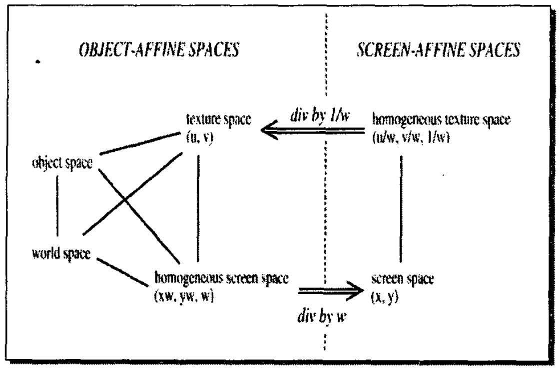

Fig. 1. Interrelationship of the coordinate system required for standard texture mapping.

4. Rational Linear Interpolation

Rational linear interpolation is proposed to solve above problem [5]. Fig. 1. presents interrelationship of the coordinate system required for standard texture mapping. Object-affine space is mean object space and possible space with affine mapping through translation. Screen-affine space is mean screen space and possible space with affine mapping through translation. Object space and screen space are translated with perspective projection mapping.

If parameters are affine with object space and $w=1$, homogeneous screen coordinate are computed by translate object space coordinate point, homogeneous parameter vector $(r_1,\ r_2,\ \cdots,\ r_n)$ is affine with screen space. Therefore this vector is interpolated by linear interpolation in screen space [Fig. 1].

In this point, proposed methods are added to interpolation parameter list with other parameter without geometry translation processing stage. For computing as many as n parameter, each vertex is divided by $n+1$ times and interpolated $n+1$ parameters. For computing each parameter, n homogeneous parameters is divided by interpolated $\frac{1}{w}$ per each pixel.

$$r_i(x,y) = \frac{r_i(x,y)/w(x,y)}{1/w(x,y)} = \frac{a_i x + b_i y + c_i}{Ax + By + C}$$

In most arithmetic processor, fast division method are multiply n times with a reciprocal of multiplier. If the value w is the same with every vertex, parameters are screen-affine against polygon. So, each pixel is avoided division processing.

5. Rendering Algorithm with Rational Linear Interpolation

The following algorithm is rational linear interpolation [12].

(1) Associate a record containing the n parameters of interest $(r_1, \ r_2, \ \cdots, \ r_n)$ with each vertex of the polygon.

(2) For each vertex, transform object space coordinates to homogeneous screen space using 4×4 object to screen matrix, yielding the values $(xw, \ \ yw, \ \ zw, \ \ w)$.

(3) Clip the polygon against plane equations for each of the six sides of the viewing frustum, linearly interpolating all the parameters when new vertices are created.

(4) At each vertex, divide the homogeneous screen coordinates, the parameters r_i, and the number $\frac{1}{w}$ to construct the variable list $(x, \ y, \ z, \ r_1/w, \ r_2/w, \ \cdots, \ r_n/w, \ 1/w)$.

(5) Scan convert in screen space by linear interpolation of all parameters, at each pixel computing $\frac{r_i/w}{1/w}$ for each of the n parameters; use these values for shading.

In these algorithm, step (1)~(3) are as same as linear interpolation algorithm. The other two stage however, compute perspective division for perform polygon to perspective projection.

V. RENDERING SYSTEM

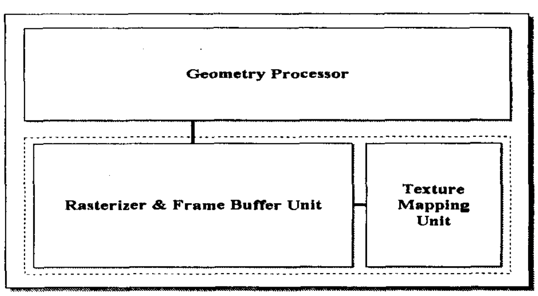

Fig. 2. System Model

Fig. 2 is the system model that proposed by [9], which is basically supposed to be system model in this paper. Contrary to conventional system, this model separates geometry processing unit from CPU completely and put it into separate graphic chip, which makes CPU be free from conventional geometry processing load. Besides, this model composed rasterizing unit and frame buffer unit as single chip form, and includes separate texture mapping unit for texture mapping that process abundant texture data.

There are several problems in system model that proposed by [9]. The amount of transmission data between rasterizer-framebuffer units is much larger than the one of system model that locates polygon processing unit before rasterizing unit, since polygon processing unit exists in geometry processing unit. And the other problem is that this model composes texture/bump mapping address generation logic per edge processor for reducing the hardware size, but it causes span process time delay about four times when texture mapping is running. Besides it didn't consider any distortion that is caused by perspective projection mapping.

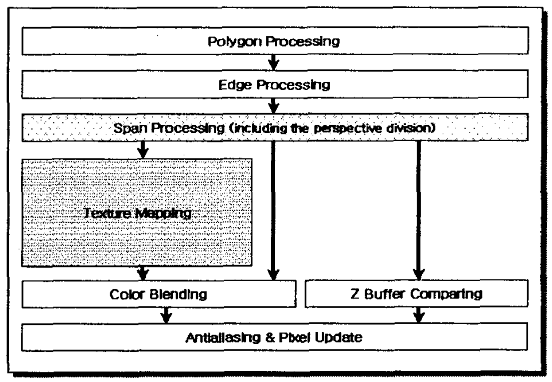

Fig. 3. Rasterizing Flow Chart

Setup stage for polygon processing is moved from geometry processing unit to the front part of rasterizing unit in the proposed model in this paper [12], which is able to solve the bandwidth problem between geometry-rasterizer unit. Besides, it executes perspective division in the span processing stage for resolving distortion problem. Perspective division unit is composed of reciprocal lookup table for obtaining value of $\frac{1}{w}$ and multiplier to multiply this value to each parameter. Differently from the system is proposed by [9], the datapath for texture data address generation and the datapath for scan conversion is not separated and overlapped for more efficient structure and process [Fig. 3].

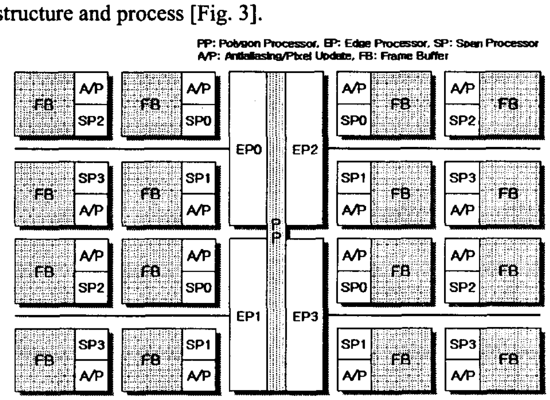

Fig. 4. Proposed Rasterizer-Frame Buffer Structure

Fig. 4 shows the rasterizer-frame buffer structure is proposed in this paper. Proposed structure is composed of one polygon processing unit and four edge processing units, and frame buffer memory is interleaved to the form of 16 memory banks, each bank has one span processing unit and antialiasing/pixel update unit. And perspective division unit in the span processing unit is able to improve image quality.

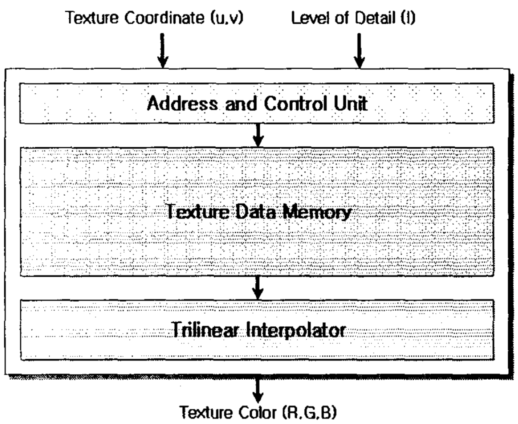

Fig. 5. Texture Mapping Unit

Abundant texture memory is required for efficient texture mapping. Off-chip memory with low bandwidth for texture memory causes access delay and becomes a large obstacle to 3D graphic processing. For resolving this problem, texture mapping logic and texture memory is composed in single chip by processor-memory integration technology [Fig. 5].

This unit is composed of address and control unit for decide each level of detail, 16 bank typed texture memory for trilinear mip-mapping, and trilinear interpolation unit for texture data interpolation [10]. Two bilinear interpolation unit and one linear interpolation unit is used for fast texture data computation in the trilinear interpolation [11].

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, the method of optimizing the rasterizing step by overlapping the interpolation step of screen coordinates, screen parameters, and texture coordinates, which is a common part of scan conversion and the texture mapping, is presented.

To solve the distortion problem invoked by handling perspective-projected polygon, perspective division is executed in span processing unit. Also, the bandwidth problem between rasterizer and frame buffer is solved by using processor-memory integration method, and the memory latency and bandwidth problem is solved by separating texture mapping unit, which requires a large amount of memory, from a rasterizer into a single chip.

Currently, the implementation of the simulation environment to verify the designed system is progressing. High-performance rendering system that supports the detail representation techniques of 3D objects, that is, the bump mapping, the environment mapping, and further Phong shading will be designed.

VII. ACKNOWLEDGMENTS

This work is supported by National Research Laboratory Projects from Ministry of Science & Technology of Republic of Korea.

REFERENCES

[1] Frederick M. Weinhaus and Venkat Devarajan, "Texture Mapping 3D Models of Real-World Scenes," *ACM Computing Surveys*, Vol. 29, No. 4, pp. 325-365, December 1997.

[2] Paul S. Heckbert, "Survey of Texture Mapping," *IEEE Computer Graphics and Applications*, Vol. 6, No. 11, pp. 56-67, November 1986.

[3] Paul S. Heckbert, "Fundamentals of texture mapping and image warping," M.sc.thesis, Dept. of EE and CS, University of California, Berkeley, June 1989.

[4] Paul S. Heckbert, "Generic Convex Polygon Scan Conversion and Clipping," Graphics Gems, Andrew Glassner, ed., Academic Press, Boston, 1990.

[5] P. S. Heckbert and H. P. Moreton, "Interpolation for Polygon Texture Mapping and Shading," *State of the Art in Computer Graphics: Visualization and Modeling*, Springer-Verlag, pp. 101-111, 1991.

[6] Lance Williams, "Pyramidal parametrics," *Computer Graphics (SIGGRAPH '83 Proceedings)*, Vol. 17, No. 3, pp. 1-11, July 1983.

[7] Paul Haeberli, Mark Segal, "Texture Mapping as a Fundamental Drawing Primitive," *Proceedings of the 4th Eurographics Workshop on Rendering*, pp. 259-266, June 1993, Paris, France.

[8] D. R. Peachey, "Solid texturing of complex surfaces," *Computer Graphics (SIGGRAPH '85 Proceedings)*, Vol. 19, No. 3, pp. 279-286, July 1985.

[9] Chun-Ja Choi, Woo-Chan Park, Tack-Don Han, "High Performance Rendering System using a Rasterizer Merged Frame Buffer," *Proceedings of The 26th KISS Fall Conference (III)*, Vol. 26, No. 2, pp. 9-11. October 1999.

[10] Andreas Schilling, Günter Knittel, and Wolfgang Strasser, "Texram: A Smart Memory for Texturing," *IEEE Computer Graphics and Applications*, Vol. 16, No. 3, pp. 32-41, May 1996.

[11] Tzi-cker Chiueh, "Heresy: A Virtual Image-Space 3D Rasterization Architecture," *1997 SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pp. 69-77, 1997.

[12] Anders Kugler, "The Setup for Triangle Rasterization," *11th Eurographics Workshop on Graphics Hardware*, pp. 49-58, August 1996, Poitiers, France.

[13] D. DasSarma, D. Matula, "Measuring the Accuracy of ROM Reciprocal Tables," *IEEE Transactions on Computers*, Vol. 43, No. 8, pp. 932-940, August 1994.

[14] http://www-ee.kaist.ac.kr/~ssl/research/ramp/ramp.html

[15] Steven Molnar, John Eyles and John Poulton, "PixelFlow: High-Speed Rendering Using Image Composition," *Computer Graphics (SIGGRAPH '92 Proceedings)*, Vol. 26, No. 2, pp. 231-240, July 1992.