An Adaptive Spatial Depth Filter for 3D Rendering IP

Chang-Hyo Yu and Lee-Sup Kim

Abstract－In this paper, we present a new method for early depth test for a 3D rendering engine. We add a filter stage to the rasterizer in the 3D rendering engine, in an attempt to identify and avoid the occluded pixels. This filtering block determines if a pixel is hidden by a certain plane. If a pixel is hidden by the plane, it can be removed. The simulation results show that the filter reduces the number of pixels to the next stage up to 71.7%. As a result, 67% of memory bandwidth is saved with simple extra hardware.

Index Terms－graphics, 3D rendering, depth test, Z buffer, early z test

I. INTRODUCTION

3D graphics rendering process needs lots of memory accesses which are Texture Mapping, Stencil Test, Depth Test and Color Blending. Because of the large memory accesses, the performance of the 3D rendering engine is highly depended on the bandwidth of memory system.

Among the operations of the 3D rendering engine, the texture mapping is the most memory intensive operation. In conventional high-performance rendering processors, texture mapping is performed before the *z*-test [4], [6], [7]. This architecture properly supports the semantics of the standard APIs such as OpenGL [9], but the major disadvantage is unnecessary texture mapping for invisible pixels; which causes a waste of the memory bandwidth. In order to remove such unnecessary operations, texture mapping should be performed after the *z*-test. However, a wider fragment queue for the pipeline execution is required. Moreover, the wide separation between reading and writing a *z*-value makes maintaining the frame memory consistency more difficult because more than one fragment may have the same pixel address.

Hyper-Z technology [3], ATI's architecture for their commercial product, keeps a reduced resolution of the *z*-buffer on the hierarchical *z*-buffer [8] and removes the fragments with *z*-test failures as early as possible; they are removed from the pipeline before texture mapping. After this step, texture mapping and the final *z*-test with the full screen frame memory are performed. With this scheme, a considerable amount (60–70% on the average) of fragments with *z*-test failures are detected and then discarded from the pipeline. However, the hierarchical *z*-buffer requires a very large data structure because it is constructed with the *z*-pyramid [8]. Maintaining the hierarchical *z*-buffer for every frame memory update may also bring an excessive computational burden.

Mid-texturing [5] used two-stage z-test operation. They used the 1st z-test before the texturing and 2nd z-test after the texturing. Their two-stage z-test shares the z-buffer in the frame buffer. Unlike the Hyper-Z [3], it needs not an auxiliary z-buffer for the 1st z-test but still needs the overhead for the memory bandwidth to read a z-value for the 1st z-test.

In this paper, we present a simple solution to perform the depth test partially before the texturing like Hyper-Z but we need only 1 bit or 2 bits for a fragment. And the data is managed by independently to the z-buffer of the frame buffer. Due to these features, the overhead from the early depth test (like the 1st z-test [5]) is very smaller than previous methods.

We previously presented the basic Depth Filter algorithm [1] conceptually. In this paper, we complete our algorithm such that it is more reliable by using a newly added algorithm for adaptive updating while

Manuscript received November 5, 2003; revised November 26, 2003.
KAIST
373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea+82-42-869-4442
Email: {mosmov,lskim}@ mvlsi.kaist.ac.kr

skipping unnecessary depth-read operations.

The remainder of the paper is arranged as follows. Section II shows the advanced algorithm of the basic depth filter. Section III shows the adaptation method and section IV shows the simulation environment and results. Section V discusses the results while the last section concludes this paper.

II. Depth Filter

We use a hierarchical concept wherein a depth filter performs an early-depth-test in the rasterizer. This newly added block uses only 1 bit or 2 bits, compared to 24 bits or 32 bits precision in the ATI's Hyper-Z [3] method. By this simple test, we reject a large portion of pixels.

Fig.1 shows the block diagram of the modified rendering engine. Since our previous basic depth filter [1] has a fixed filter position, it could not be used in actual situation. However, an adaptation block is newly added to the previous depth filter. It makes a signal to the depth filter so that the depth filter updates its filter to the optimal position.

In case of the 3-plane system of the basic depth filter, we already know that the depth filter has better performance due to a fine resolution of filtering, but if we use the third plane of 3-plane system as a special purpose mask plane, the depth filter has good result even in a less depth complexity cases. Fig. 2 shows the modified 3-plane system, called 2-plane Skipping Depth Buffer Reading (SDBR) system.

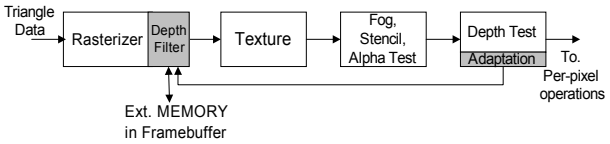

Fig. 1. A block diagram of the proposed architecture of 3D rendering Engine.

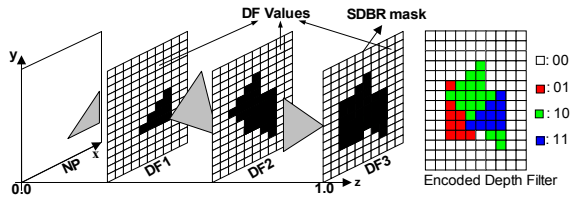

Fig. 2. A conceptual picture of 3-plane system or 2-plane SDBR system.

A conventional z-test needs a z value of a depth buffer. When a pixel is rendered to a certain coordinate for the first time, it is not necessary to read the z value from the depth buffer because the value of the depth buffer would be 1.0 (default setting). To remove (or skip) this needless read operation, the depth filter provides pertinent information. In Fig. 2, if we move DF3 to the Far plane, DF3 is used for storing whether the pixel has been rendered before. In other words, this plane holds the information whether the depth in this coordinate is valid or not. If the value of this plane is "1", the current coordinate had been updated before. Therefore its depth information is valid. When the value of this plane is "0", we only write a depth value to the depth buffer in the frame memory without reading the current depth value. The depth filter sends fragments to a pixel processor with this information, usually a 1 bit signal. The depth test block can save memory bandwidth wasted on reading useless depth.

III. Adaptive Updating Of The Filter Position

The depth filter uses the mask plane in the z-coordinate to remove the pixels. This plane should be located at an optimal position so as to yield a maximum rejected pixels. Therefore the position of the depth filter is one of the most important aspects in our system.

A. Characteristics of pixels

To find an optimum position of the filter, we focus on the pixel characteristics. A 3D application's object data is distributed by a user at any range of z-coordinates which are from the near-plane to the far-plane [0.0, 1.0]. However, there is no general closed form for the distribution of pixels. Since the pixels have only non-deterministic distributions in the general 3D application's object data, we use simple closed form of distribution function only to know and approximate the characteristics of pixel distribution. The simple models of distribution functions are a uniform distribution function, a triangle-shape function and a more complex polynomial function. These three types of pixel distributions are shown in Fig. 3.

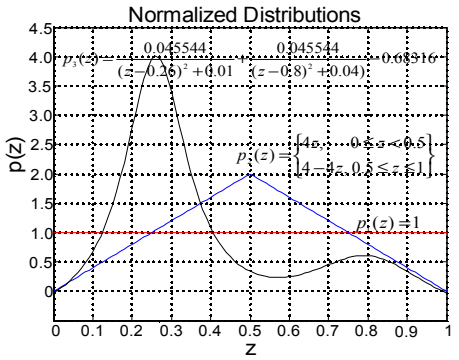

Fig. 3. Normalized distributions of the three types simplified models.

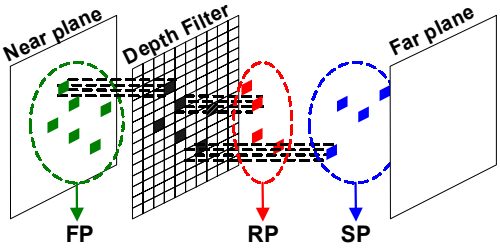

Fig. 4. FP and SP are passed to the next pipeline and RP is removed.

Let t be an arbitrary position of z-coordinates,

$$FP(t) = \int_0^t p(z)dz \quad BP(t) = \int_t^1 p(z)dz \qquad (1)$$

where $p(z)$ is the normalized density function of the pixel data. From these equations, $FP(t)$(Pixels in Front of t) and $BP(t)$ (Pixels Behind of t) always have a monotonic function of t because there is no negative value of $p(z)$. If the depth filter is on t, pixels can be divided into three parts as shown in Fig. 4.

The pixels, farther than t ($BP(t)$), are divided into two classes by the depth filter. The first is the sum of survived pixels, SP. These pixels have been tested by the depth filter but could not be rejected. The second is the sum of rejected pixels, RP, by the depth filter.

B. An optimal position of depth filter

To find a maximum RP, we use another convenient form of RP. RP can be written as

$$RP = BP - SP \quad = (TotalPixels - FP) - SP \qquad (2)$$

$$= TotalPixels - (FP + SP)$$

where $FP + BP = TotalPixels$, $SP = BP - RP$

By equation (2), the maximum value of RP can be obtained by the minimum value of (FP + SP). We find a minimum value of (FP + SP) instead of finding an optimal position of the depth filter. NRP is defined as (FP + SP). NRP is the negation of RP, which has a minimum value when RP has a maximum value. This is the first property to find an optimal position of the depth filter. And the following two properties are derived from the natural characteristics of the depth filter.

• Property 1: The depth filter has a maximum rejection ratio when NRP has a minimum.

• Property 2: The depth filter has a larger probability to reject pixels as the position of the depth filter moves to 1.0 in the z-coordinate because there are more pixels that can mask the depth filter.

• Property 3: The depth filter has a larger probability to reject pixels as the position of the depth filter moves to 0.0 in the z-coordinate because there are more pixels that can be rejected by the depth filter.

The $RP(t)$ can be approximated simply to $\alpha 1 \times t$ by property 2 and $\alpha 2 \times (1-t)$ by property 3. $\alpha 1$ and $\alpha 2$ are constants and t and $1-t$ are proportional factors from these properties.

We then rewrite RP(t)

$$(\alpha 1 \times (t)) \times (\alpha 2 \times (1-t)) = \beta t(1-t), \quad \alpha = \beta t(1-t) \quad (3)$$

where α is the proportion of $BP(t)$. α has a value from 0 to 1. If $RP(t)$ is 0, α is 0 and if $RP(t)$ is the same as $BP(t)$, α is 1. We assumed α to be 0.75 at the maximum (the simulation result shows about 72% ($RP(t)/(FP(t)+BP(t)) = 0.72$) of total pixels are rejected, then the proportion of α ($RP(t)/BP(t)$) is larger than 0.72, in test model (a)). Therefore the range of α is $0 \le \alpha \le 0.75$. Then we set the constant, β, to be 3 because the value of $t(1-t)$ is 0.25 at the maximum. Since the pixels of $FP(t)$ are not changed by the depth filter, $RP(t)$ only has a proportion of $BP(t)$. The function of $SP(t)$ is also monotonic because $SP(t)$ is the value of $BP(t) - RP(t)$ and none of functions have negative values.

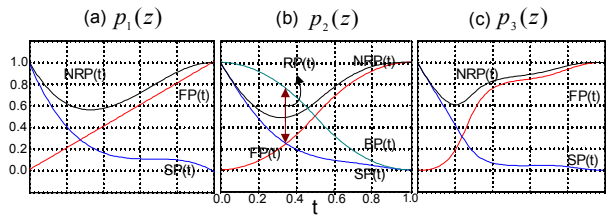

Fig. 5. *FP(t)* and *SP(t)* with their sum, *NRP(t)*. *RP(t)* has a maximum when *FP(t)* = *SP(t)*.

The curve of *SP(t)* has the same value as *BP(t)* at both sides and is lowered by *RP(t)*. In other words, the curve of *SP(t)* is a steeper version of the *BP(t)* curve. We rewrite the equations,

$$\begin{aligned} RP(t) &= BP(t)\times 3t(1-t) \\ SP(t) &= BP(t) - RP(t) = BP(t)\times\left(1-3t(1-t)\right) \\ NRP(t) &= FP(t) + SP(t) = FP(t) + BP(t)\times\left(1-3t(1-t)\right) \end{aligned} \quad (4)$$

The approximated optimal position, determined graphically, is obtained when *FP(t)* = *SP(t)*. Since *SP(t)* is also a monotone decreasing function, this approximation is acceptable. Simulated *NRP(t)*s of the models are shown in Fig.5. Our simplified models are well conducted to find an approximated optimum position. There can be an error at specific cases, for example, all objects are rendered in back-to-front order. However, the general 3D models have similar characteristics as shown here.

From this algorithm, we place an adaptation block into a conventional depth test block to adjust the filter position. The adaptation block makes a signal, which tracks the filter position to the cross point of FP and SP, to the depth filter at every frame.

The hardware overhead of our adaptation algorithm is very small; which is composed of two counters (for FP and SP) and leading-one detectors (for finding deference between the FP and SP) and some other operators. Therefore it is not burden to implement into a conventional depth test block.

IV. Simulation And Results

A. Simulation environment

For simulation, we used Graphics Architecture Testing Environment (GATE) [2] based on Microsoft visual C++. GATE models overall graphics hardware architecture through a modular approach, supports OpenGL, and offers easy modification and rapid testing of architecture. It also gathers computational statistics. The performance and other statistical data can be obtained from GATE.

We added the depth filter block into the rasterizer of the GATE and the adaptation block into the conventional depth test block. A single bit of the SDBR signal is fed to the conventional pipelines between the depth filter and the adaptation block.

B. Results

The test models are shown in Fig. 6. Model (a) is composed of 100 hexagonal columns generated randomly. Model (b) and (c) are general 3D objects. Model (d) is an image by texture mapping for testing the 2-plane SDBR system. It has 1.0 of depth complexity and is used for the worst case of the depth filter. All tests are conducted on a 512 x 512 screen and Back face culling is disabled. We use a replicated basic object so as to have high depth complexity.

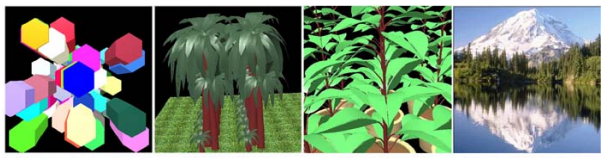

Fig. 6. Test models with different depth complexity. ( ) represents a model's depth complexity.

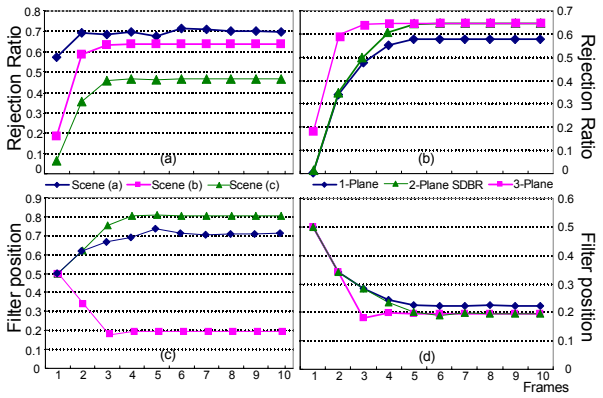

Fig. 7. Rejection ratios of the models and each system. Graph (a) shows the results of the three kinds of test models in the 3-plane system. Graph (b) shows the result of the three types of systems for test model (b). Graph (c) is the corresponding filter position of graph (a) and Graph (d) is that of graph (b)

Table 1. RR (Rejection ratio), ERR (Effective rejection ratio) of the test models.

RR (ERR)	1-Plane	2-Plane SDBR	3-Plane
Model (a)	63 (79.6)	70.9 (81.9)	71.7 (88.9)
Model (b)	57.9 (79.6)	63.2 (87.0)	63.6 (87.4)
Model (c)	30.6 (64.3)	35.3 (74.2)	40.4 (77.2)

Fig.7 shows the rejection ratio of the depth filter, (*RP(t)/Total Pixels*). All the systems and test models have large amounts of rejected pixels and all converge to the approximated maximum rejection in three frames. The 3-plane system tracks faster than the others and has the best rejection ratio. Although the 2-plane SDBR system is slower than the 3-plane system, its optimal rejection ratio is similar to the 3-plane system. The converged rejection ratios of the test models are shown in table 1.

V. DISCUSSIONS

We define a required memory bandwidth to compare our systems with a conventional system. The conventional method is not one of the other types of hierarchical methods such as the Hyper-Z [3], etc. Rather it is the conventional Z buffer method. We cannot compare with commercial methods because they are not in public.

Required Memory Bandwidth =
(FR × RR × 0 B/pixel) + (FR × SR × 16 B/pixel)
+FR × (1 – RR – SR) × 20 B/pixel + Depth Filter Overhead

Depth Filter Overhead =
(Read, Write) × FR × MR × Transferring Data Size

FR: fill rate, RR: rejection ratio, SR: SDBR rate and MR: miss rate

We assumed the entire test models need the memory operations which are Read-Z, Write-Z, Read-Color, Write-Color and Texture-Read. We assume that the all operations need 4 bytes. This calculation is based on the specifications of current conventional hardware and considered without any other overheads. For example, the overhead of the depth filter in case of the 1-plane system is to be 2 × 2.6G × 0.057 × 64 × 1 bit = 2.371GB/sec in the test model (b). We used the specifications of Radeon9700pro [4], which has a 2.6gigapixels/sec fill rate.

As shown in Fig. 8, the 2-Plane SDBR system has the best performance, not only for the complex model (model (a)) but also for the less complex model (model (d), a texture). In Fig. 8 (d), since test model (d) has 1.0 of depth complexity, there is no rejected pixel. Therefore the required memory bandwidth of the 1-Plane and the 3-Plane system is larger than that of conventional but even though the overhead of the depth filter, the 2-Plane SDBR system is always better than the conventional depth test. The effect obtained by skipping an unnecessary read operation is more efficient than the effect obtained by adding one plane from a 2-plane to a 3-plane system.

Hyper-Z [3] shows an effective rejection ratio of 44 ~ 93% in the test bench but they need large internal memory and synchronization of the internal memory and the depth buffer of the frame buffer. Our algorithm shows an effective rejection ratio of 64~88% with only small internal memory (or register) by the depth filter using the adaptive method.

VI. CONCLUSIONS

As more complex models become commonplace in 3D computer graphics, it becomes increasingly important to remove unnecessary operations that waste memory bandwidth. In this paper, we present an algorithm that saves memory bandwidth in the rendering engine of 3D graphics hardware.

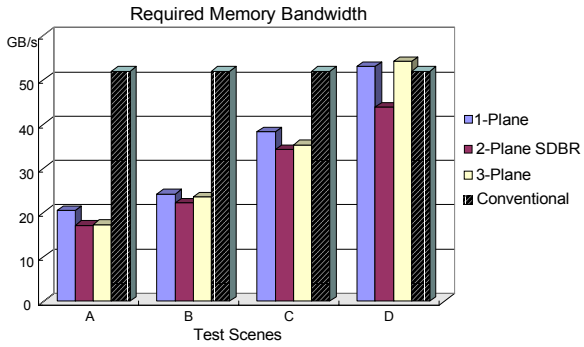

Fig. 8. Comparisons of the required memory bandwidth of all our systems and a conventional system.

Depth Filter is a filtering method that removes invisible pixels ahead of per-pixel pipelines. As a result, it avoids unnecessary per-pixel operations by adding extra hardware into a rasterizer placed in front of the per-pixel pipeline and a simple adaptive block into the conventional depth test block. Up to 71.7% of total pixels were removed by this method for the complex test models as well as non-complex test models. The maximum bandwidth gain obtained by the depth filter was 3.03. A novel adaptation algorithm makes the depth filter possible to use in actual situation; therefore it can be a useful method to alleviate memory bottlenecks in 3D rendering engines.

ACKNOWLEDGMENT

This research was sponsored by SAMSUNG electronics and KOSEF through the MICROS at KAIST, Korea.

REFERENCES

[1] C.H. Yu and L.S. Kim, "A hierarchical depth buffer for minimizing memory bandwidth in 3d rendering engine: depth filter", IEEE *Proceedings of the 2003 International Symposium on Circuits and Systems*, Vol. 2, pp 724–727, May, 2003.

[2] Inho. Lee, et al, "A hardware-like highlevel language based environment for 3d graphics architecture exploration", IEEE Proceedings of the 2003 International Symposium on Circuits and Systems, Vol. 2, pp 512–515, May, 2003.

[3] S. Morein, "ATI Radeon Hyper-Z technology", *In Hot 3D Proceedings, Graphics Hardware Workshop 2000*, Interlaken, Switzerland, August, 2000.

[4] ATI Corporation, Radeon9700pro, 2002, http://www.ati.com /products/pc/radeon9700pro/index.html and http://www.ati.com /developer/techpapers.html.

[5] W.C. Park, et al, "A mid-texturing pixel rasterization pipeline architecture for 3D rendering processors", IEEE *Proceedings of International Conference on Application-Specific Systems, Architectures and Processors*, pp 173 - 182, July, 2002.

[6] J. McCormack, R. McNamara, C. Gianos, L. Seiler, N. Jouppi, K. Correll, "Neon: a single-chip 3D workstation graphics accelerator", *Graphics Hardware Workshop 1998*, pp 123-132, August, 1998.

[7] L. Garber, "The wild world of 3D graphics chips," *IEEE Computer*, vol. 33, no. 9, pp 12–16, Sept, 2000.

[8] N. Greene, M. Kass and G. Miller, "Hierarchical z-buffer visibility,"ACM *Proceedings of SIGGRAPH"*, pp 231–238, Aug, 1993.

[9] R. Kempf and C. Frazier, *OpenGL reference manual*, Addison Wesley, 1996.

Chang-Hyo Yu received the B.S. and M.S. degrees in Electrical Engineering and Computer Science from KAIST, Korea, in 2001 and 2003 respectively, and now he is a Ph.D. candidate in Electrical Engineering and Computer Science from KAIST. His research interests include 3D graphics hardware design and multimedia programmable processor design.

Lee-Sup Kim received the B.S. degree in Electronics Engineering from Seoul National University, Korea, in 1982 and the M.S. and Ph.D. degrees in Electrical Engineering from Stanford University, Stanford, CA, in 1986 and 1990, respectively. He was a post-doctoral fellow at the Toshiba Corporation, Kawasaki, Japan, during 1990-1993, where he was involved in the design of the high performance DSP and single chip MPEG2 decoder. Since March 1993, he has been at KAIST. In November 2002, he became a Full Professor. During the year of 1998, he was on the sabbatical leave with Chromatic Research and SandCraft Inc. in Silicon Valley. His research interests are 3D graphics hardware design, LCD display controller design, Multimedia Programmable Processor design, and High speed and Low power digital IC design.