A New Culling Scheme for Low Power 3D Graphic Processors

Chanho Lee[1] and Kyeongeun Choi[2]
[1] School of Electronic Engineering, Soongsil University, Seoul, Korea
511 Sangdo-dong, Dongjak-Gu, Seoul, Korea
E-mail: chlee@ssu.ac.kr
[2] Samsung Electronics, Suwon, Korea

Abstract: Recently, portable devices employ applications using 3D graphics such as 3D games and 3D navigations. The portable devices require small area and low power consumption. We propose an efficient culling scheme for low power 3D graphics processors. The proposed culling scheme consists of the selection and back-face culling in the geometry engine and the elimination of pixels outside in the rasterizer engine. The new scheme reduced both the hardware complexity and the number of operation cycles of culling operations. We design a 3D graphic pipeline using Verilog-HDL according to the proposed scheme, and verify it on an FPGA prototyping board. The latency of the proposed architecture is reduced by 15 cycles and the gate count of the synthesized result is reduced by 8%.

1. Introduction

3D graphic processing represents the generation of 2D images from 3D objects, virtual camera, light sources, lighting models, and textures, and is also called rendering [1]. The 3D graphic processing had required high performance workstation for 3D graphic accelerators with 3D graphic software. Nowadays, personal computers have 3D graphic accelerators, and can show realtime 3D graphic processing and animation [2]. Recent development of 3D graphics hardware for mobile environment enables the production of digital contents based on 3D graphics which is more attractive than those based on 2D graphics. More mobile devices such as cellular phones, PMP, and PDA start to employ 3D graphic contents [3]. 3D graphic processing requires much more arithmetic operations than 2D graphics due to the complex processing algorithm.

The conventional 3D graphic processors for workstations or personal computers work with high performance CPUs and practically unlimited power is supplied. However, mobile devices have limited power supplied by batteries which requires low energy operation and the mobile microprocessors may not share enough computing power with the 3D graphic processors. The 3D graphic processing for mobile environment cannot employ the same algorithm as that in the desktop environment [1]. An efficient architecture can reduce the load of an embedded processor and energy consumption.

The conventional clipping algorithm in the geometry engine generates a new vertex using vertices inside and outside the clipping window when a triangle or a line crosses the boundary, and requires a lot of arithmetic calculation which results in many operation cycles or arithmetic units.

In this paper, we propose a new culling scheme which substitutes the conventional clipping and back-face culling methods. The proposed culling scheme is performed by a cull-and-sort unit and scan conversion engine in the rasterizer as shown Figure 1. The cull-and-sort unit detects triangles which are completely outside the clipping window and removes them. If part of a triangle is inside the clipping window, it is removed in the edge walk and the span processing of the scan conversion engine. Triangles are set up by the cull-and-sort unit of the geometry engine instead of the scan conversion stage, and back-face culling is achieved simultaneously. The culling in the scan conversion unit requires only several comparators, and the increase in area is minimal. We design a 3D graphic pipeline according to the proposed scheme, and verify the operation.

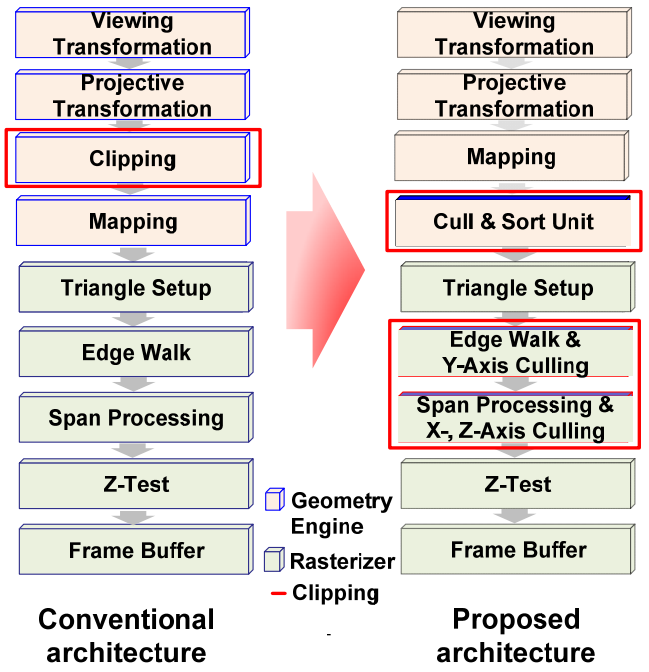

Figure 1. Proposed culling scheme for fixed 3D graphic pipelines

2. Proposed Algorithm

2.1 Conventional Clipping algorithm

The rasterization requires a lot of arithmetic units and operation cycles. A viewer or camera can capture a limited range of volume due to the limited view angle. Therefore, the rasterization of the 3D objects outside the clipping window is useless, and the objects or polygons may be eliminated before delivered to the rasterization engine. It is the goal of clipping operation. The position of a polygon is investigated using the coordinates of vertices if it is kept, removed, or modified by generating new vertex (or vertices) in the conventional clipping algorithm. The

polygons totally outside the boundary are eliminated. Cohen-Sutherland algorithm [4] and Liang-Barsky algorithm [5] for 2D clipping operation was combined and extended to 3D clipping algorithm [2]. It requires complex operation for reconstruction of triangles in the clipping window if a triangle intersects a clipping boundary, and often causes the generation of additional triangles as shown in Figure 2. The additional triangles cause pipeline stall in geometry engine and the increase of the number of vertices to be delivered to the scan conversion unit [7].

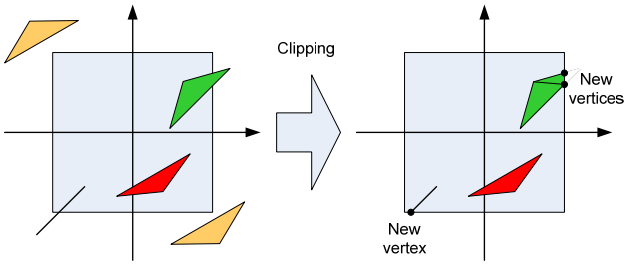

Figure 2. Conventional clipping algorithm which generates new vertices.

2.2 Cull-and-Sort unit

The most complex calculation in the conventional clipping algorithm is the generation of new vertices for the triangles which crosses a boundary of the clipping window. The proposed algorithm removed the operation, and introduces a cull-and-sort unit which calculates flag information according to Cohen-Sutherland algorithm [4], and the information is used to determine the location of vertices. The cull-and-sort unit checks if a triangle is completely outside the clipping window (or view frustum) or at least one vertex is inside. The triangles completely outside the window are eliminated and are not transferred to the scan conversion unit as shown in Figure 3. The box in the Figure 3 is a view frustum and five triangles which are located outside the box are eliminated, and one triangle which is inside the box is delivered to the scan conversion unit. Two triangles are intersects at least one boundary of the box. The conventional algorithm generates new vertices on the boundary surfaces and reconstructs new triangles, which results in four triangles. The proposed algorithm delivers the two triangles without any modification. The operation is quite simple and does not increase the number of vertices and triangles. The back-face culling can also be completed at the same time.

Since a triangle needs to be set up for the clipping operation, the cull-and-sort unit does the job so as to reduce the burden of the triangle setup unit of scan conversion engine. The mapping transformation is merged with the projective transformation so as for the triangles to be transferred from the cull-and-sort unit to the scan conversion engine directly. As a result, the complexity of the geometry engine is lowered. The architectural enhancement enables the geometry engine to perform culling action in a cycle and to avoid the pipeline stall to the end of geometry calculation.

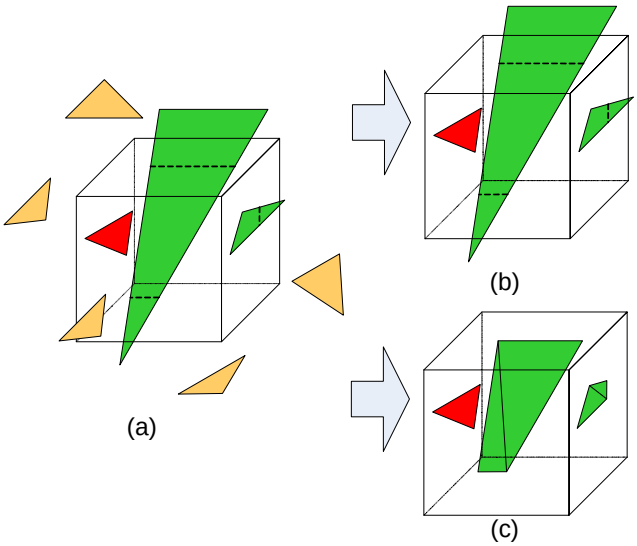

Figure 3. Comparison of cull and sort unit with the conventional clipping method. (a) Before culling. (b) Proposed. (c) Conventional.

2.3 Edge walk with Y-axis culling

Edge walk unit determines the colors and the coordinates of edges which are start and end points of spans while increasing Y-axis values by one. At least one vertex is inside the clipping window since triangles outside the clipping window are already eliminated. Y-axis culling is obtained by adding a comparator unit in the proposed scheme. The normal edge walk process is performed for each pixel in the edge, and then the y-axis value of the pixel is compared with that of the boundary. If the position is outside the boundary, the edge walk process is stopped as shown in Figure 4.

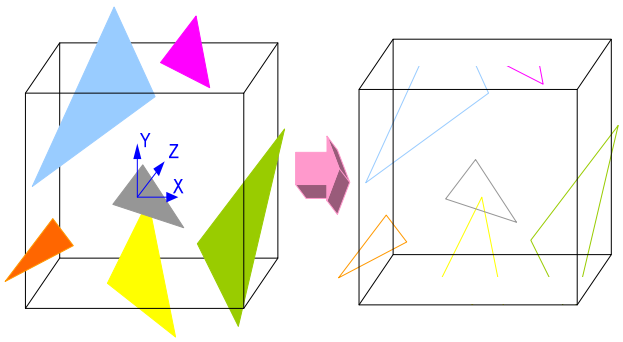

Figure 4. Result of Y-axis culling using the edge walk unit.

2.4 Span processing with X- and Z-axis culling

Span processing unit determines the colors and the coordinates of spans while increasing X-axis value by one, which fills colored pixels inside a triangle. X-axis and Z-axis culling are obtained by adding two comparators in the proposed scheme. A span may include pixels outside the clipping window since the x-axis value of the pixels in the edge is not investigated. The x-axis values and z-axis values are compared with those of the boundary during the

normal span processing, and the processing is stopped when a pixel is determined to be outside the boundary as shown in Figure 5. The culling process in the scan conversion unit loads little burden with three comparators and the corresponding propagation which may be negligible.

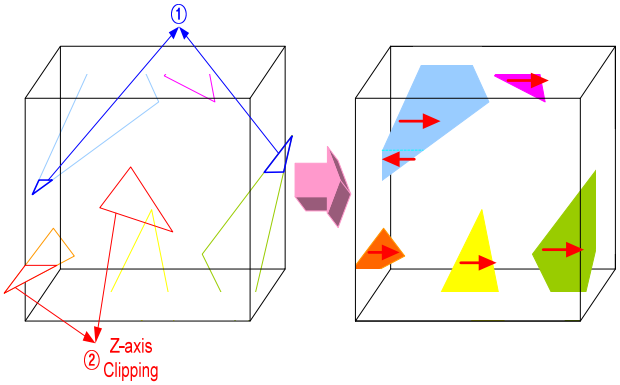

Figure 5. Result of span processing including X- and Z-axis clipping

3. Design and Verification

We design a fixed 3D graphic pipeline accelerator based on the proposed culling scheme using Verilog-HDL, and implement it on an FPGA. Table 1 shows the comparison results of synthesized area and pipeline latency of a conventional 3D graphic engine and the proposed 3D graphic engine. The engines are synthesized using Synopsys Design Compiler and a 0.25um CMOS standard cell library at the target operating frequency of 100MHz. The latency is decreased from 47 cycles of the conventional scheme to 31 cycles of the proposed scheme, which is the improvement of 34%. The synthesized area of the conventional geometry engine is 89,100 gate counts while that of the proposed engine is 63,300 gate counts, which is improvement of 29%. The synthesized area of the scan conversion unit is slightly increased from 172,300 gate counts to 177,880 gate counts, which is the increment of 3%, due to the addition of culling function. The overall area is decreased from 261,300 gate counts to 241,000 gate counts, which is the improvement of 8%.

The proposed 3D graphic engine is verified in a verification system using a Xilinx Virtex5 FPGA as shown in Figure 6. Test vector are composed of four models of Pawn, Sphere, Cone, and Torus with the number of vertices of 1,518, 504, 504, and 1,020, respectively. An ARM processor provides the 3D model data the transformation engine with the cull-and-sort unit, the calculated vertex data are returned to the processor. The processor supplies the vertex data and color information to the scan conversion unit with the culling function. The final pixel data are transferred to a TFTLCD controller for display. The results show that the proposed scheme works fine as shown Figure 7.

Table 1. Comparison results of conventional 3D graphics engine and proposed 3D graphics engine

	Conventional Architecture[7]		Proposed Architecture	
	Area [gates]	Latency [cycle]	Area [gates]	Latency [cycle]
Transformation	56,900	18	56,900	18
Clipping	32,100	16~49	6,300	1
Triangle setup	90,800	10	90,700	9
Edge walk	69,500	2	71,500	2
Span processing	12,000	1	15,600	1
Total	261,300	47	241,000	31

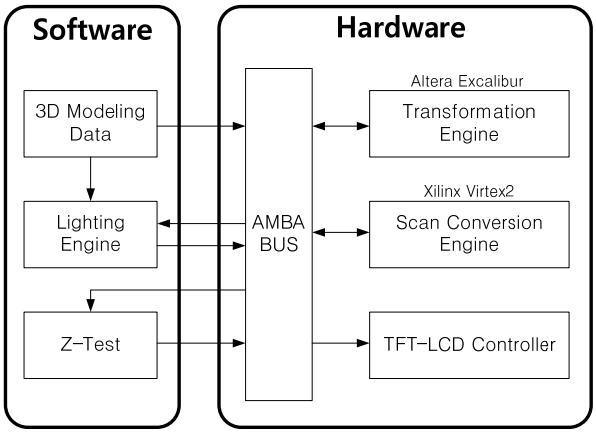

Figure 6. Block diagram of verification system

Figure 7. The image of the verification system

4. Conclusion

We propose a new efficient culling algorithm for low power 3D graphic pipelines, and verify it by designing and implementing a system based on the proposed algorithm. The complex clipping operation which requires many operation cycles and arithmetic units is divided into two stages. The first stage in the geometry engine eliminates triangles completely outside the clipping window, and the operation is quite simple. The second stage in the scan

conversion unit eliminates the pixels outside the clipping window with only three additional comparators and without any extra operation cycles to the normal scan conversion operation. The proposed algorithm can reduce the area and latency of 3D graphic pipelines. A 3D graphic pipeline is designed using Verilog-HDL and implemented on an FPGA. The implementation results show that the proposed algorithm reduces the area and the latency by 8% and 34%, respectively, compared with the conventional algorithm.

Acknowedgmentt

This work was supported by Seoul R&BD Program and EDA tools were supported by IDEC.

References

[1] Gopi K. Kolli, "3D Graphics Optimizations for ARM Architecture", Game Developers Conference, San Jose, CA, March 2004.

[2] Tomas A. Moller and Eric Haines, "Real Time Rendering", A K Peters, p.30, p.95-96, 2002.

[3] Eungon Lee, “Effects of 3D graphics on the experience of customers in on-line shopping mall with virtual reality,” Master's thesis, Yonsei University, Febrary, 2003.

[4] F. Devai, “An analysis technique and an algorithm for line clipping", IEEE Conference on Information Visualization, pp.29-31, July 1998.

[5] Y-D. Liang and B. Barsky, “A New Concept and Method for Line Clipping", ACM Transactions on Graphics, vol.11, pp.276-290, January 1984.

[6] Jeemyeong Lee, “Design of geometry engine for mobile 2D/3D graphic pipelines,” Master's thesis, Soongsil University, June, 2007.