Hit-Test Unit을 이용한 Ray Tracing의 구현

최규열 · 정덕진

Implementation of Ray Tracing using Hit-Test Unit

K. Y. Choi, D. J. Chung

인하대학교 전자재료공학과

Abstract

The synthesis of the 3D images is the most important part of the virtual reality. The ray tracing is the best method for reality in the 3D graphics. But the ray tracing requires long computation time for the synthesis of the 3D images. So, we implements the ray tracing with software and hardware. Specially we designs the hit-test unit with FPGA tool for the ray --tracing.

I. 서론

미래의 컴퓨터 산업은 가상현실(Virtual Reality)을 기반으로 하여 발전하게 된다. 현재의 2차원 평면 그래픽을 이용하여 생성된 영상으로는 가상현실을 실현하기에 역부족이다. 가상현실에서 가장 중요한 것은 인간으로 하여금 컴퓨터로 합성된 상황을 현실로 인지하도록 하는 것이기 때문에 사실감이 떨어지는 2차원 그래픽으로는 가상현실을 구현함에 있어 무리가 따르게 된다. 하지만, 3차원 그래픽을 이용할 경우에는 보다 현실감 있는 가상현실 시스템을 구축할 수 있게 된다.

3차원 그래픽을 구현하는 기법으로는 Ray Tracing, Z-buffering, Ray Casting, Radiosity 기법 등 여러 가지 기법이 사용된다. 그중 가장 사실감 있는 기법은 Ray Tracing이다. Ray Tracing은 빛을 방향을 추적하여 화면을 구성하는 기법이다. 사람의 눈으로 들어오는 빛의 세기와 색상을 계산하여 화면을 구성하기 때문에 물체의 밝기, 색상 등이 현실에 가깝게 합성된다는 이점을 가지고 있다. 또한 빛의 방향을 추적하는 기법을 사용하므로 자연스럽게 원근감이 표현된다. 즉, 인간이 시각을 통해서 보는 기법 그대로 화면을 구성하는 것이다. 따라서, 다른 기법에 비해서 가장 현실에 가까운 화상을 구현할 수 있게 된다. 즉, 최고 품질의 가상현실을 위해서는 Ray Tracing이 가장 적합하다.

하지만, Ray Tracer의 가장 큰 문제점은 영상의 각 픽셀에 해당하는 시선벡터마다 수많은 연산을 해야 하기 때문에, 다른 기법에 비해 화면을 생성시키는데 필요한 시간이 매우 길다는 것이다. 이 문제를 해결하기 위하여 본 연구에서는 소프트웨어와 고속 병렬 곱셈기, 덧셈기 등을 사용하여 고속처리가 가능한 Ray Tracer를 설계하였다. 연산시간을 크게 요구하지 않는 부분은 소프트웨어로 처리하고, 반복연산으로 인해 많은 연산을 요구하는 부분만을 하드웨어로 구현하였다. 설계된 Ray Tracer는 병렬 구조를 지원하기 때문에 필요할 경우 여러 개의 하드웨어를 중복 사용하여 병렬처리함으로서 빠른 속도로 화면을 생성시킬 수 있다.

2. Ray Tracing

2.1 Ray Tracing의 특징

3차원 컴퓨터 그래픽[1][2]의 최종 목적은 영상을 실제의 사진과 같이 사실감 있게 합성해내는 것이다.(Photo Realism) 하지만, 컴퓨터 그래픽을 이용하여 사진과 똑같은 이미지를 구현한다는 것은 거의 불가능하다고 할 수 있다. 때문에 3차원 컴퓨터 그래픽에서 추구하는 것은 실제 사진이 아니라, 실제 사진에 매우 가까운 영상을 얻는 것이라 할 수 있다. 이러한 조건에 가장 적합한 알고리즘이 바로 Ray Tracing이다. Ray Tracing[3]은 빛 하나 하나를 모두 추적하기 때문에 사실적인 영상의 합성이 가능하다. 하지만, 광선 하나 하나를 모두 추적해야 하기 때문에 많은 연산량을 필요로 하는 단점을 안게 된다.

2.2 Ray Tracing의 개요

Ray Tracing은 빛의 진행방향을 추적하여 영상을 합성하는 방법이다. 이 Ray Tracing에 접근하는 방법은 두 가지가 있다. 한 가지는 광원으로부터 Camera까지의 진행방향을 추적하는 방법이고, 또 다른 한 가지는 Camera로부터 광원까지의 진행방향을 추적하는 방법이다. 전자를 정방향 Ray Tracing이라고 하고, 후자를 역방향 Ray Tracing이라고 한다.

일반적으로 Ray Tracing이라 하는 것은 정방향 Ray Tracing이 아닌 역방향 Ray Tracing을 가리킨다. 역방향 Ray

Tracing은 영상 합성에 필요한 빛만을 추적하게되므로 정방향 Ray Tracing에 비해 비교적 적은 계산 과정으로 3차원 영상을 합성하게 된다. 하지만 이 방법 역시 Polygon과 시선 벡터와의 교점연산을 수행하기 위해 많은 연산시간을 요하게 된다. 정방향 Ray Tracing에 비해 연산 과정이 간단하기는 하지만, 이 방법 역시 많은 계산 과정을 거쳐야만 한다.

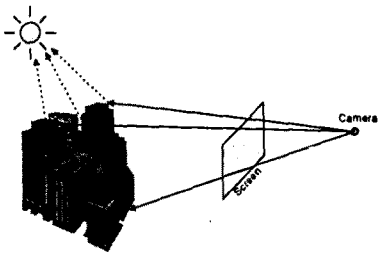


Fig. 1 역방향 Ray Tracing

Fig 2.에서와 같이 Camera에서 출발한 각각의 시선은 벡터로서 표시된다. Ray Tracing은 각각의 벡터가 만나는 물체의 표면을 찾아내고 그 표면이 받는 빛의 양을 판별하여 색상을 계산하여 각 시선에 해당하는 pixel에 색상의 값을 저장한다.

3. Ray Tracing의 구현

3.1 Ray Tracing의 블록 다이어그램

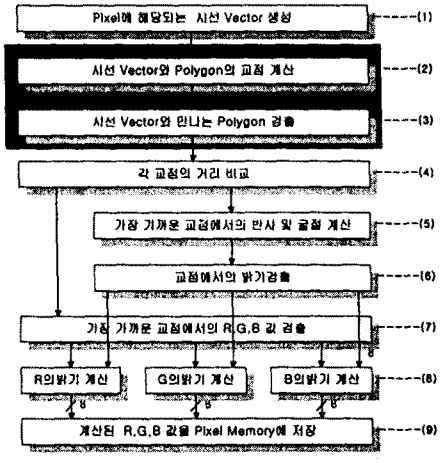


Fig. 2 Ray Tracing의 블록 다이어그램

3.2 Ray Tracing의 시선 벡터

Ray Tracing에서는 영상을 구현하기 위한 최소 단위로서 시선 벡터를 사용한다. 시선 벡터는 사용자의 시야로부터 화면의 각 픽셀 포인트까지의 위치벡터를 이용한다. 식 (1)은 시선 벡터를 구하기 위한 식이다.

$$\hat{I} = \frac{\hat{P} - \hat{O}}{|\hat{P} - \hat{O}|} = \frac{\hat{P}}{|\hat{P}|} \qquad (1)$$

I는 시선 Vector, P는 화면상의 Pixel의 위치 Vector, O는 카메라의 위치 Vector로서 원점을 사용한다.

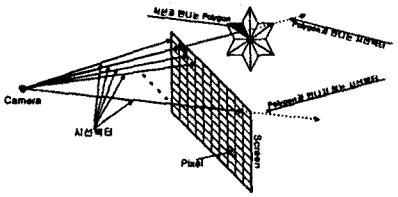


Fig. 3 Ray Tracing의 시선벡터

3.3. Hit-Test Unit

Fig 3.에서 회색 박스로 처리된 (2),(3)번 과정에 해당한다. Polygon과 시선벡터간의 교점을 계산하는 방법은 다음과 같은 벡터식으로 표현된다.

일반적으로 Polygon과 시선 벡터의 교점을 구하기 위해서는 평면의 방정식과 직선의 방정식을 사용한다.

$$a\hat{x} + b\hat{y} + c\hat{z} = d \qquad (2)$$

$$\begin{aligned} x &= t \cdot d_x + p_x \\ y &= t \cdot d_y + p_y \\ z &= t \cdot d_z + p_z \end{aligned} \qquad (3)$$

$$t = \frac{(a \cdot p_x + b \cdot p_y + c \cdot p_z)}{(a \cdot d_x + b \cdot d_y + c \cdot d_z)} \qquad (4)$$

식 (2)는 Polygon을 포함하는 평면의 벡터이고, 식 (3), (4)는 일반적인 평면과 직선간의 교점을 찾아내는 수식이다. 이 경우 9번의 곱셈과 7회의 덧셈, 1회의 나눗셈을 요한다. 본 논문에서는 위의 방법 대신 아래와 같은 방법을 택하였다.

$$\hat{H} = ai + bj + ck \qquad (5)$$

$$r_k = \sqrt{a^2 + b^2 + c^2} \qquad (6)$$

$$\hat{C} = \left(\frac{\frac{d}{r_k}}{\frac{\hat{I} \cdot \hat{H}}{r_k}} \right) \hat{I} = \left(\frac{d}{\hat{I} \cdot \hat{H}} \right) \hat{I} \qquad (7)$$

식 (5)은 Polygon의 법선 벡터를 나타낸다. 식 (6)는 법선 벡터의 크기를 나타낸다. 식 (7)는 시선 벡터와 Polygon과의 교점을 나타내는 벡터이다. 이 벡터식을 사용할 경우 6번의 곱셈, 세 번의 덧셈, 1번의 나눗셈 과정만으로 교점의 계산이 가능하다. 본 논문에서는 Hit-Test Unit을 구현하기 위해서 식 (7)를 사용하였다.

Ray Tracing에서는 하나의 시선벡터와 각각의 Polygon마다 모두 교점을 계산해야 하기 때문에 연산시간이 길어지게 된다. 따라서 Ray Tracing 중 연산시간이 가장 긴 부분이 바로 이 부분이다. 본 논문에서는 이 부분을 하드웨어로 구현하여, 전체 연산 시간을 줄이는 방법을 사용하였다. 현재 하드웨어의 설계 및 검증이 끝난 상태이며, 현재 하드웨어와 소프트웨어간의 Interface 부분을 개발하고 있다.

3.4 병렬 구조의 제안

설계된 Hit-Test Unit은 2개 이상의 하드웨어를 조합하여 병렬구조를 가지는 시스템의 구성이 가능하도록 설계되었다. 이 경우 여러 개의 Pixel을 동시에 처리가능 하기 때문에 보다 빠른 속도로 3차원 영상을 합성해 낼 수 있게 된다. 많은 수의 Hit-Test Unit을 조합할 경우 동영상의 실시간 합성도 가능해진다.

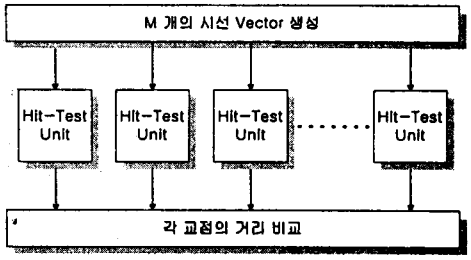


Fig. 4 병렬 처리를 위한 구조

3.4 연산장치의 구조

Hit-Test Unit은 연산의 반복에 의해 연산시간이 오래 걸리는 만큼 하드웨어의 성능이 연산을 직접 수행하는 곱셈기와 덧셈기의 성능에 크게 의존한다.

곱셈기는 수정된 Booth 알고리즘을 사용하여 부분곱의 수를 반으로 줄이고, Baugh-Wooley알고리즘을 사용하여 곱셈에 필요한 덧셈의 단수를 반으로 줄였다. 또한 병렬 곱셈기를 이용하여 연산속도를 증가 시켰다. 덧셈기의 경우에는 Carry Save Adder와 개선된 CLA(Carry Look Ahead)구조를 이용하며, 이와 함께 CSA(Carry Select Adder)를 이용하여 최적화[4] 시켜 고속의 덧셈을 구현하였다.

4. 하드웨어의 설계 및 Simulation

4.1 하드웨어의 설계

본 논문에서는 Altera사의 FPGA Tool을 이용하여 검증하였다. 본 연구에서 설계된 Hit-Test Unit은 병렬구조로의 응용이 가능하기 때문에 일정량 이상의 Hit-Test Unit칩을 사용할 경우 3차원 동영상의 실시간 합성이 가능하다.

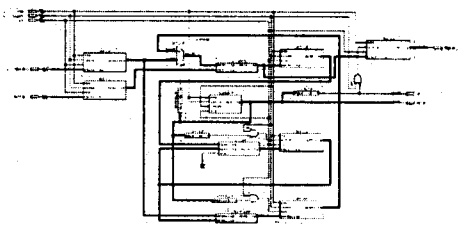
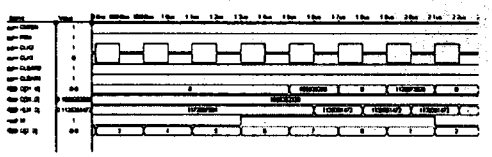


Fig. 5 Hit-Test Unit 의 Logic Diagram

Fig. 6은 Hit-Test Unit을 FPGA를 이용하여 설계한 Logic Diagram이다. 1개의 곱셈기, 1개의 덧셈기, 1개의 나눗셈기로 구성되어 있다. 각각의 연산장치는 32-bit Floating Point 연산을 지원하도록 설계되었다.

4.2 Simulation

Simulation 역시 Altera사의 FPGA Tool인 Max+Plus II를 이용하여 수행하였다. Simulation에서 사용한 동작 주파수는 10MHz 이고 Fig. 7에 보인 화면은 알아보기 쉽게 하기 위하여 5MHz로 동작시킨 화면이다. O[31..0] 항목은 Hit-Test Unit의 최종 결과 값이고, C[31..0] 와 H[31..0] 항목은 Input Signal이다. 한 개의 시선벡터와 한 개의 Polygon과의 교점계산을 수행하는 데에는 8clock이 소요된다.



5. 결론

Ray Tracing의 모든 과정을 소프트웨어로만 구현할 경우 320 * 200의 화면크기에서 약 10여 개의 Polygon을 사용하여 화면을 구성할 경우 소요되는 시간은 대략 Pentium 133MHz 시스템에서 약 10분이 소요된다. 하지만 본 논문에서 설계한 10MHz로 동작하는 Hit-Test Unit을 사용하여 Ray Tracing을 구현할 경우에는 약 40개정도의 Hit-Test Unit으로 320 * 200 의 크기에서 30 Frame/s를 갖는 3차원 동영상의 합성이 가능해진다. 기존의 경우 DSP를 이용하여 Ray Tracing을 구현한 사례[5]가 있었지만, 이 경우 512개의 Processor를 병렬로 연결하여 사용하였기 때문에 시스템으로 구현하는 데에는 어려움이 있다.

차후 Full Custom 또는 ASIC으로 칩을 구현할 경우에는 현재 FPGA를 통하여 구현가능한 속도보다 빠른 속도로 동작이 가능하므로, 10개 이하의 Hit-Test Unit만으로도 동영상의 실시간 처리가 가능해 질 수 있다.

참고문헌

[1] A. Watt, 3D Computer Graphics, Addison-Wesley, 1993

[2] 신영수 외, 3차원 그래픽-C언어로 배우는 알고리즘, 가남사, 1995

[3] R. Pulleyblank, "The Feasibility of a VLSI Chip for Ray Tracing Bicubic Patches", IEEE CG&A, March 1987, pp. 33-44.

[4] N. Ohkubo et al., "A 4.4 ns CMOS 54 × 54-b Multiplier Using Pass-Transistor Multiplexor", IEEE J. Solid-State Circuits, vol. 30, pp. 251-257, March 1995

[5] C. Scott Ananian, "The TigerSHARK Architecture", 1996