3D 그래픽 프로세서 검증을 위한 래스터라이저 설계

이미경 · 장영조

한국기술교육대학교

A Design on Rasterizer for the verification in a 3D Graphic Processor

Mi-Kyoung, Lee and Young Jo, Jang

Korea University of Technology and Education

E-mail : jeje410@kut.ac.kr

요 약

고차원적인 멀티미디어 컨텐츠를 처리하는 그래픽 가속기를 설계함에 있어서 쉽고 정확한 하드웨어 검증 환경과 임베디드 장치에서의 성능 평가가 필요하다. 이를 해결하기 위해 시뮬레이션파형 분석을 통한 검증이 아니라 실제 연산된 그래픽 이미지를 확인할 수 있는 소프트웨어 래스터라이저를 설계하였다. 설계한 래스터라이저는 윈도우 기반의 환경에서 C언어를 이용하여 래스터화 각 단계 별로 함수로 구현하고 정점 데이터를 입력하여 결과를 검증하였다.

ABSTRACT

When the graphics accelerator for high-quality multimedia content design, hardware verification environment, easy and accurate performance evaluation in an embedded device is required. To work around this is not verified through the simulation waveform analysis to determine the actual calculated graphic images has designed a software rasterizer. Rasterizer is designed for Windows-based environment using the C language implementation of rasterization has a function at each step. Vertex data is entered and the results were verified.

키워드

3차원 그래픽, 래스터화, 래스터라이저, 렌더링

Ⅰ. 서 론

최근 3차원 그래픽 기반의 디지털 콘텐츠가 많이 제작되고 휴대용 기기(embedded device, 휴대용 정보기기 시스템)에서도 이를 이용한 더욱 현실감 있는 3차원 그래픽 영상표현에 대한 관심이 증가하고 있다. 사실적인 고품질의 3차원 그래픽 영상을 얻기 위해서는 많은 양의 연산이 필요하고 그 연산 또한 복잡하기 때문에 제한이 많은 휴대용 기기에 적용하는 것은 어려운 일이다. 이를 극복하기위해 SoC 기술을 이용한 고성능의 기하변환 하드웨어 엔진 연구가 활발히 진행되고 있다.

본 논문은 고성능 그래픽 처리 하드웨어의 연산 결과를 디스플레이 장치를 이용하여 검증하는 환경을 구축하기 위해 래스터라이저를 설계하고, 기하변환 연산 결과를 입력으로 받아 윈도우 환경에서 모니터로 출력하는 동작을 검증하였다.

Ⅱ. 3차원 그래픽 렌더링 파이프라인

3차원 그래픽 처리는 그림 1에서 보는 바와 같이 5개의 추상적 단계들로 나눌 수 있다. 이 단계들은 응용단계, 기하단계, 셋업단계, 래스터화 단계, 그래픽 버퍼단계라 하고 순차적으로 수행된다.

그림 1. 3차원 그래픽 렌더링 파이프라인

응용(application) 단계는 모델을 통해 만들어진 데이터베이스를 3차원 그래픽 렌더링 과정에서 처리할 수 있는 점, 선, 삼각형과 같은 렌더링 기본요소를 전송하는 과정이며, 일반적으로 호스트 CPU로 처리한다.

기하(geometry)단계는 응용 단계에서 생성된 데이터를 기하학적으로 변환시키는 과정으로 크게 변환(transform)부분과 빛 처리(lighting)부분으로 분류된다. 변환부분은 모델/시야변환, 투영, 클리핑, 화면 매핑으로 다시 분류된다. 빛 처리 부분은 실제 빛 처리를 하기 전에 빛 처리에 필요한 요소들을 미리 계산하는 부분과 실제 빛 처리를 하는 부분으로 분류된다.

셋업(setup)단계는 기하단계에서 처리된 부동소수점 데이터를 정수 데이터로 변환하고 삼각형의 기울기를 구하게 된다.

래스터화(rasterization)단계는 셋업 단계에서 구해진 삼각형 데이터들을 트라이앵글 셋업, 변 처리, 스팬 처리, 텍스쳐 매핑, 투명도 처리, 안개처리, 깊이 비교 등의 과정을 거쳐 색상 정보를 가진 픽셀들을 이루게 된다. 이 픽셀들을 그래픽 버퍼에 갱신한다[1].

그래픽 버퍼 단계는 그래픽 버퍼의 픽셀들을 디스플레이 장치에 보내 한 영상을 이루어 화면에 보여 진다.

3차원 그래픽의 실시간 렌더링에서는 속도를 향상시키고, 화질을 개선하는 방법을 제공하는 것이 목표이다. 이 목표를 달성하기 위해서는 많은 양의 연산이 필요한 기하 처리의 부동소수점 연산 부분과 래스터화 과정의 효율적인 구조 설계에 대한 연구가 필요하다.

III. 래스터화 파이프라인

래스터화 단계는 크게 기하단계에서 전송된 정점 단위의 기하 요소들을 화소 단위로 변환시키는 스캔 변환(scan conversion) 단과 깊이 비교를 하는 z-버퍼 단, 그리고 텍스쳐 맵핑을 위한 텍스쳐 단으로 구성되어져 있다. 그림 2는 래스터화 단계를 전반적으로 나타낸 그림이다[2].

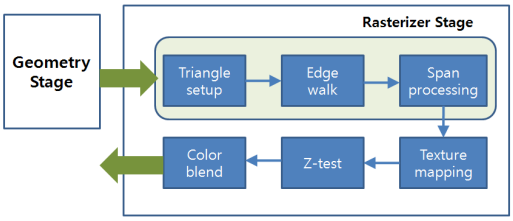

그림 2. 래스터화 파이프라인

3.1 스캔변환

3.1.1 삼각형 셋업

기하 단계를 거친 후 래스터화 단계로 전송되어지는 데이터는 다각형 또는 선분의 정점이다. 삼각형 셋업 과정은 기하 단계에서 전송되어지는 정점을 화면에 맞추어 화소로 사상시키는 것이다. 정점을 화소로 사상시키는 삼각형 셋업과정을 거친 이후의 과정부터는 정점 단위가 아닌 화소 단위로 계산이 이루어진다.

기하 단계에서 전송되어진 세 개의 정점으로 이루어진 삼각형을 나타내고[3], 래스터화 단계로 전송되어진 정점 단위의 삼각형은 스캔 변환 과정중 제일 첫 번째 과정인 삼각형 셋업 과정을 거치게 된다. 삼각형 셋업 과정을 거친 화소들은 각각의 화소들마다 고유의 주소 값, 깊이 값 그리고 색상 값을 가지게 된다.

3.1.2 에지 워크

삼각형 셋업 과정을 거친 후에는 에지 워크 과정을 거치게 된다. 에지 워크 과정은 삼각형 셋업 과정에서 생성된 화소들 사이에 선분을 이어주는 과정으로 보간법을 통하여 이어준다. 에지 워크 과정을 거친 후에는 그림 3에서처럼 선분으로 표현되는 화소들이 생성된다. 생성된 화소들은 삼각형 셋업 과정과 마찬가지로 화소마다 고유의 주소 값, 깊이 값, 색상 값을 가지게 된다[3].

3.1.3. 스팬 프로세싱

에지 워크 과정을 거친 이후에는 스팬 프로세싱 과정을 거치게 된다. 스팬 프로세싱 과정은 에지 워크 과정에서 생성된 선분 사이에 비워진 화소들을 채워주는 과정이다. 스팬 프로세싱 과정은 선분을 표현하는 화소들 중 같은 높이를 갖는 화소 사이를 보간법을 통하여 보간 한다. 생성된 화소들은 에지 워크 과정과 마찬가지로 화소마다 고유의 주소 값, 깊이 값, 색상 값을 가지게 된다. 그림 3은 스캔 변환에 따른 예를 나타낸다[3].

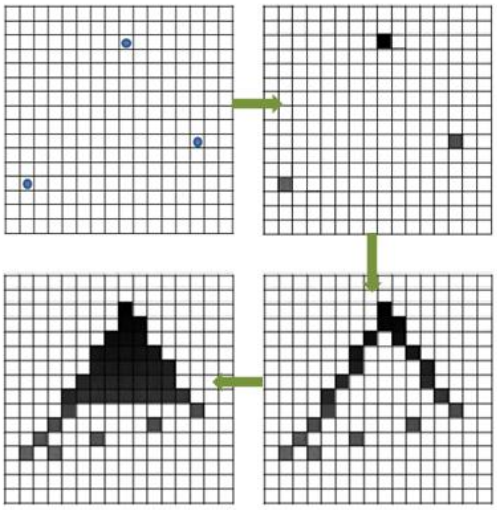

그림 3. 스캔 변환 처리 단계별 결과

3.1.4 래스터화 후단부로의 전송

삼각형 셋업 과정, 에지 워크 가정, 스팬 프로세싱 과정으로 구성되어진 스캔 변환 과정을 거치면 세 개의 정점으로 전송된 삼각형은 완전히 화소로 구성된 삼각형으로 변하게 된다. 화소로 변화된 삼각형은 래스터화 후단부에게 삼각형 데이터를 화소 단위로 전송하게 된다. 래스터화 후단부는 크게 z-버퍼 과정과 텍스쳐 가정으로 구성되어져 있으며, z-버퍼 과정은 가시성 판별을 위한 과정이고, 텍스쳐 과정은 텍스쳐 매핑을 위한 과정이다[2,3,5].

3.2 z-버퍼 과정

z-버퍼 과정은 중첩되는 물체를 올바르게 화면에 표시하기 위해서 화소들의 가시성을 판별하기 위한 과정이다. 가시성 판별을 위한 알고리즘으로 화소들의 깊이 값을 저장하는 z-버퍼를 이용한 z-버퍼 알고리즘을 많이 사용한다. 일반적으로 래스터화 단계에서 수행되는 일반적인 z-버퍼 알고리즘은 그림에 나타나있다. z-버퍼링을 위해서는 화소의 깊이 값이 깊이 값을 저장해주는 z-버퍼와 화소의 색상 값을 저장 하는 프레임 버퍼 그리고 깊이 값을 비교해주는 비교기가 필요하다.

Ⅳ. 구현 및 검증

래스터화 과정은 그림 4와 같이 지오메트리 처리 결과를 입력으로 받아 트라이앵글 셋업, 에지 워크, 스팬 프로세싱 단계를 수행한다. 화려하고 현실감있는 그래픽 가속 프로세서를 설계하기 위해서는 지오메트리 연산기와 래스터라이제이션을 효율적으로 설계하는 것이 중요하다. 하드웨어로 구현한 기하 변환 엔진을 시뮬레이션 파형으로 검증하기는 어렵다[6]. 이를 해결하고자 기하 변환 결과를 텍스트 파일로 저장하고, 파일 입출력을 이용하여 래스터라이저에 전송하여 화면을 통해 3차원 그래픽 이미지를 정확히 검증하는 소프트웨어 래스트라이저를 구현하였다.

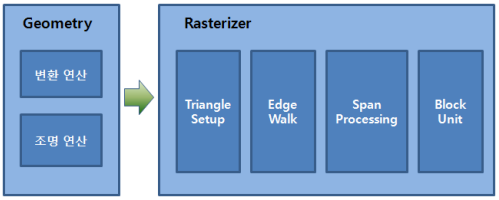

그림 4. 기하변환과 래스터라이저 수행단계

삼각형 셋업 단계에서는 입력된 세 개의 정점이 정점 정렬과 정의 과정에서 Y축을 기준으로 Vmin(y) <= Vmid(y) <= Vmax(y) 순으로 Vmax, Vmid, Vmin으로 정의되고 각 정점으로 정의된 면이 화면에 출력되는 방향인지, 화면영역 안에 위치하는지 판단한다.

에지 워크 단계에서는 그림 5와 같이 Vmax와 Vmin을 연결하는 major edge Emaj, Vmin과 Vmid를 연결하는 bottom edge Ebot, Vmid와 Vtop을 연결하는 Etop의 각각의 정점을 계산하여 스팬 프로세서로 넘겨준다.

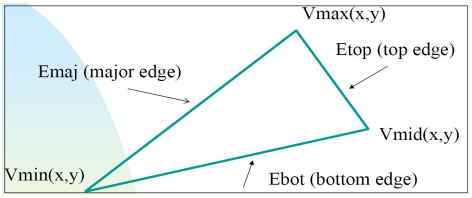

그림 5. 정점 정렬과 정의

계산 되어진 각각의 정점 값은 스팬 첫 점과 끝점이 된다. 하나의 삼각형을 모두 처리한 후 종료 정보를 트라이앵글 셋업으로 보낸다.

스팬 프로세서 단계에서는 첫 점, 끝 점의 좌표와 색상을 입력받아 하나의 주사선을 그린다[7].

설계한 래스터라이저는 윈도우 기반의 환경에서 C언어를 이용하여 래스터라이저 각 단계 별로 함수로 구현하고 정점 데이터를 입력하여 결과를 검증하였다.

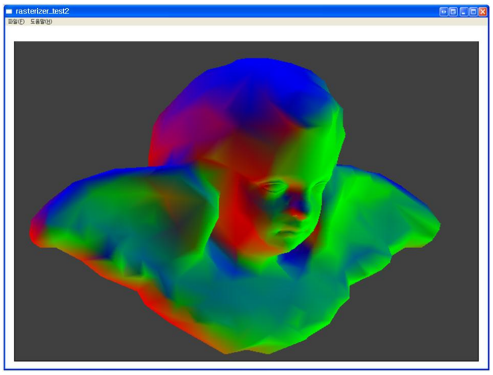

그림 6. 데모 결과

Ⅴ. 결 론

본 논문에서는 그래픽 가속 프로세서 설계 단계에서 정점 데이터를 소프트웨어 래스터라이저에 입력함으로써 윈도우 기반에서 쉽고 빠르게 기능을 검증할 수 있는 래스터라이저를 구현하였다.

참고문헌

[1] Tomas Akenine-Moller, Eric Haines, "Real-time rendering," A K PETERS, 2002

[2] W.C Park, K. W. Lee, I. S. KIM, T. D. Han, S. B. Yang, "AnEffective Pixcel Rasterization Pipeline Architecture for 3D Rendering Processors, " IEEE Transactions on computers, vol.52, no.11, Nowvember 2003.

[3] Andrew Wolfe, Derek B. Noonburg, "A Superscalar 3D Graphics Engine," MICRO-32. Proceeding, 1999.

[4] 주우석, OpenGL로 배우는 컴퓨터 그래픽스, 한빛미디어, 2006

[5] Bor-Sung Liang, Yuan-Chung Lee, Wen-Chang Yeh Chein-Wei Jen, "Index Rendering : Hardware-Efficient Architecture for 3D Graphics in Multimedia System, " IEEE Transactions on multimedea, vol.4, no.2, June 2002

[6] 정형기, 곽재창, 이광엽, " C 인터페이스를 이용한 3D Graphics Shader HDL 검증 시스템 구현"

[7] http://www.pldworld.com/_hdl/1/erc.msstate.edu/www/~reese/EE4993/lectures/triangle/sld001.htm