중첩 라이팅 방식을 이용한 기하학 프로세서

이승엽[O] 홍원기 김신덕
연세대학교 컴퓨터과학과
{sylee, wkhong, sdkim}@kurene.yonsei.ac.kr

Geometry processor using overlapped lighting method

Seung-Yup Lee[O] Won-Kee Hong Shin-Dug Kim
Dept. of Computer Science, Yonsei University

요 약

3차원 그래픽 가속기는 기하학 처리(geometry processing) 단계와 래스터라이제이션(rasterization) 단계로 구성되어 있다. 기존의 기하학 처리 방식에서는 꼭지점의 좌표계산과 빛의 효과를 계산하는 일련의 단계들이 순차적으로 수행되었는데 이는 많은 양의 폴리곤 처리가 요구되는 현재의 어플리케이션 환경에서 상당한 오버헤드로 작용한다. 본 연구에서는 기하학 처리 파이프라인을 보다 고속으로 처리하기 위해 라이팅 단계를 다른 단계들과 병렬적으로 수행할 수 있는 구조를 제안한다. 실험결과 제안하는 중첩 라이팅 방식의 기하학 처리기(Overlapped lighting geometry processor, OLGP) 는 기존의 순차적인 기하학 처리기(Sequential geometry processor, SeqGP) 에 비해 최대 21% 의 수행 성능 향상을 보였다.

1. 서론

3차원 애니메이션, 게임, CAD, GIS 등의 각종 어플리케이션 분야에서 멀티미디어의 요구 확대와 함께 3D 그래픽스는 그 활용 범위와 요구되는 성능이 급격히 증가되고 있다. 이러한 3D 그래픽스 처리량의 폭발적인 증가는 CPU 와 그래픽 카드의 부담을 가중시킨다.

3D 그래픽스 시스템은 기본적으로 데이터베이스 트레버설(database traversal), 기하학 처리와 래스터라이제이션이 순차적으로 수행되는 3가지 단계들로 이루어진다 [그림 1]. 데이터베이스 트레버설 단계는 모델링으로 생성된 객체의 자료 구조를 그래픽스 파이프라인에서 처리할 수 있는 데이터로 전달하고 기하학 처리 단계에서 3차원 데이터를 2차원 화면의 좌표로 기하학적인 변환과 빛의 효과를 계산한다. 래스터라이제이션 단계는 기하학 처리에 의해 변환된 점, 선, 다각형 등과 같은 프리미티브 (primitive)들을 프레임 버퍼 (frame buffer)에 픽셀 데이터로 변환시킨다. 이 과정에서 각종 객체들이 모니터에 표시될 수 있도록 각 픽셀들의 색깔값으로 변환하여 프레임 버퍼에 로드되며 일반적으로 이를 렌더링 (rendering)이라고 한다. 기하학 처리 단계는 모델뷰 변환, 라이팅, 클리핑, 프로젝션, 뷰포트 변환 단계로 구성된다. 이와 같은 기하학 처리를 위해서 부동 소수점 연산기가 주로 사용되는데, 특히 라이팅 단계에서 사용되는 제곱근, 나눗셈, 거듭 제곱 연산들로 인해 기하학 처리 파이프라인의 다른 단계들에 비해 수행 시간이 상당히 지연된다.

본 논문에서는 3D 그래픽스 파이프라인 중에서 CPU 의 부담이 큰 기하학 처리를 고속으로 수행하기 위한 방법으로서, 중첩 라이팅 방식을 이용한 기하학 프로세서 구조를 제안한다. 이 구조에서는 라이팅 단계와 클리핑, 프로젝션, 뷰포트 변환 단계가 동시에 수행될 수 있다. 실험결과 제안하는 구조는 응용프로그램이 요구하는 폴리곤 처리량을 충족시키며 기존의 기하학 파이프라인에 비해 수행 시간을 21% 향상시킬 수 있었다.

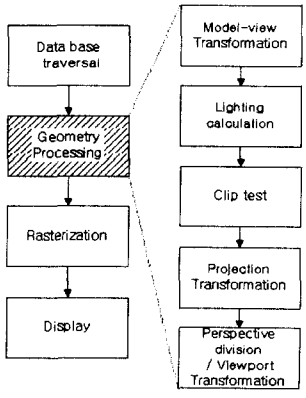

그림 1. 3D 그래픽스 파이프라인.

본 논문의 구성은 다음과 같다. 2 장에서는 3 차원 그래픽 가속기에서의 기하학 처리부의 각 단계별 역할과 기존의 순차적인 처리방식에서 문제점을 지적하고 3 장에서는 기존의 문제점을 해결할 수 있는 중첩 라이팅 방식의 기하학 처리기의 구조를 제안한다. 4 장에서는 기존의 방식과 중첩 라이팅 방식을 비교 실험한 결과를 분석하고 5 장에서 결론을 맺는다.

2. 기하학 파이프라인

기존의 기하학 처리는 다음과 같은 일련의 단계를 통해 순차

적으로 수행된다(순차적인 기하학 처리, SeqGP) [5, 7].

1) 모델뷰 변환(model-view transformation): 데이터 베이스 트레버설을 통해 기하학 파이프라인으로 넘어온 꼭지점 데이터의 객체 좌표계를 시점 좌표계로 변환한다. 이때 이동 변환(translation transformation) 의 계산을 단순화하기 위해 동차 좌표계(homogeneous coordinate)로 표현된다.
2) 라이팅(li ghting): 광원, 광원 모델, 물질 파라미터에 따라 각 꼭지점의 빛의 강도를 계산한다. 이때 각 꼭지점의 법선 벡터를 정규화하고, 이 정규화 된 법선 벡터, 시점, 광원 벡터를 이용하여 계산한다.
3) 클리핑(clipping): 프리미티브가 뷰 볼륨(view volume)을 벗어나는 부분이 있으면 제거한다. 클리핑의 목적은 우선 스크린 상의 임의의 윈도우가 다른 윈도우의 픽셀에 영향을 주지 않도록 하고, 시점보다 뒤쪽에 있거나 아주 멀리 있는 프리미티브들에 대한 계산에서 오버플로우나 언더플로우가 발생하지 않도록 하기 위한 단계이다.
4) 프로젝션 변환(projection transformation): 클리핑된 시점 좌표계의 모든 꼭지점을 뷰플레인(view plane)으로 프로젝션시킨다.
5) 퍼스펙티브 나눗셈(perspective division): 4차원의 동차 좌표계를 w 좌표로 나눔으로써 정규화된 장치 좌표계로 변환한다.
6) 뷰포트 변환(viewport transformation): 이전 단계에서의 좌표계상의 꼭지점들을 화면 좌표계로 매핑시킨다.

이와 같은 기하학 처리는 데이터의 특성상 부동 소수점 연산이 주를 이루는데 이는 상당한 하드웨어 비용과 지연시간(latency)이 요구되는 문제점을 안고 있다. 특히 라이팅 단계에서는 식(1)과 같은 조명 공식을 수행하게 되는데 이때 사용되는 연산들은 제곱근, 나눗셈, 거듭 제곱 연산 등이 있으며 이들은 상당한 지연시간을 소비한다.

$$I_R = I_{aR}k_aO_{dR} + \sum_{i=0}^{n-1}\left[\frac{1}{d_L^2}I_{pR}\left\{k_dO_{dR}(\vec{N}\bullet\vec{L}) + k_sO_{sR}(\vec{R}\bullet\vec{V})^n\right\}\right] \quad (1)$$

일반적인 파이프라인형태의 구조에서 알 수 있듯이 전체 수행시간은 지연시간이 가장 긴 단계에 의해 그 시간이 좌우되므로 라이팅 단계는 기하학 처리 단계에서 가장 큰 오버헤드로 작용한다. 이와 같은 이유로 대부분의 저가형 3D 그래픽 시스템에서는 CPU 에서 기하학 처리를 전담하고 3D 그래픽 가속기에서는 래스터라이제이션을 주로 수행한다. 그러나 어플리케이션에서 초당 처리해야 하는 폴리곤의 수가 지속적으로 증가함에 따라 3D 그래픽스 처리에 필요한 부동 소수점 연산을 처리하는 CPU 의 부담이 늘어나고 있다. 또한 실시간 고감도 3 차원 그래픽을 실현하기 위해서 저가형 그래픽 가속기에서도 고속의 독립된 기하학 처리부의 필요성이 더욱 절실해질 전망이다.

2.1 기하학 처리를 가속화 하는 다양한 방법

기하학 처리 성능을 향상시키기 위해 기존에 제안된 방법을 들면 다음과 같다. 고속의 기능 유닛(functional unit)를 설계하거나 복잡한 부동 소수점 계산을 미리 계산하여 테이블로 저장해 두는 방식으로 근사화하는 방법이 있다. 3D 그래픽스 파이프라인에서 병렬성을 활용하는 방법은 크게 하나의 프리미티브 내의 꼭지점들 간의 데이터 병렬성을 이용하거나, 꼭지점을 처리함에 있어서 명령어들 간의 병렬성 (Instruction Level Parallelism, ILP)을 이용하는 방법과 기하학 파이프라인의 단계들 간에 병렬적으로 수행할 수 있는 부분을 찾는 방법이 있다.

2.1.1 데이터 병렬성을 이용한 구조

기하학 계산은 서로 다른 꼭지점들에 대하여 같은 연산을 수행하기 때문에 데이터 병렬성을 쉽게 활용할 수 있다. 이러한 병렬성을 이용하여 AMD, 인텔, 모토롤라 등과 같은 업체들은 3D 그래픽스의 기하학 연산을 효과적으로 지원할 수 있는 부동 소수점 연산을 위한 SIMD 명령어 구조를 적용하고 있다 [1]. 소니의 Emotion Engine 은 2개의 부동 소수점 벡터 유닛를 사용하여 기하학 처리를 수행하고 있다 [3]. 이러한 구조에서는 그래픽 데이터의 특성상, 풍부한 병렬성 때문에 기능 유닛들을 더 추가할수록 선형적인 성능향상을 기대할 수 있다. 그러나 각 기능 유닛에 데이터를 분배하는 과정이 필요하며, 분배되는 데이터의 종류가 같은 형태가 아니면, 프로세서들이 각각 다른 종류의 데이터를 처리하는 메커니즘이 없기 때문에 한 개의 프로세서를 사용할 때보다 성능이 더 저하된다.

2.1.2 명령어 병렬성을 이용한 구조

기하학 프로세서에서 하나의 꼭지점 데이터를 처리하는 연산들의 대부분은 좌표계 변환이다. 이러한 좌표 변환은 행렬 곱셈을 통해서 이루어 진다. 이때 사용되는 부동 소수점 곱셈과 부동 소수점 덧셈 명령어들은 서로 독립적으로 수행될 수 있는 병렬성이 상당히 많다. AMD 3DNow! Exten sion [1]은 이러한 ILP를 수퍼스칼라 방식으로 처리하는 예이다.

3. 중첩 라이팅 방식 (OLGP)

본 논문에서는 2 장에서 살펴본 바와 같이 기하학 처리에 있어서 가장 큰 오버헤드로 작용하는 라이팅 단계를 다른 단계들과 병렬로 수행함으로써 지연시간을 감소시킬 수 있는 방법을

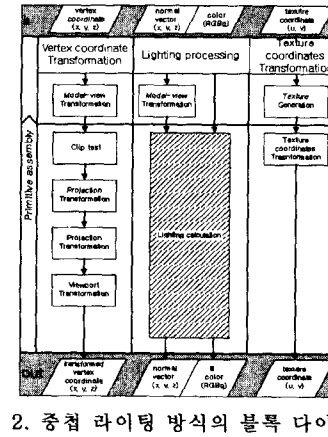

그림 2. 중첩 라이팅 방식의 블록 다이어그램.

제안한다. 라이팅 계산을 위한 패스에서 사용되는 데이터를 살펴보면, 입력은 꼭지점 법선 벡터, 꼭지점 좌표, 칼라 값이고, 그밖에 몇 가지 파라미터(광원, 광원 모델, 물질 파라미터)들이 사용된다. 반면 꼭지점 좌표변환을 위한 패스에 속하는 모든 단계에서는 꼭지점 좌표만을 입력으로 받아들여 연산을 수행하고 변환된 꼭지점 좌표를 래스터라이제이션 단계로 전송한다. 꼭지점 좌표를 제외한 다른 데이터들은 클리핑 이후의 단계들에 대해 영향을 주지 않기 때문에 두 가지 패스에서 동시에 사용될 수 있다. 또한 라이팅 패스에서 사용되는 꼭지점의

좌표계는 클리핑이나 좌표 변환에 의해 영향을 받지 않도록 변환 전의 좌표와 변환 후의 좌표가 따로 저장되는 데이터 구조를 가진다. 본 논문은 상대적으로 긴 지연시간을 보이는 라이팅 단계와 클리핑, 프로젝션, 퍼스펙티브 나눗셈, 뷰포트 변환 단계를 병렬적으로 수행하는 중첩 라이팅 방식을 이용한 기하학 처리 모델 (OLGP)을 제안한다 [그림 2]. 여기에서 좌표 변환 패스와 라이팅 패스의 서로 다른 연산들을 동시에 처리할 수 있는 명령어 구조로서 VLIW (Very Long Instruction Word) 구조를 채택하였다. VLIW는 컴파일러에 의해 명령어들의 병렬성을 충분히 활용할 수 있고 하드웨어 요구가 적으므로 OLGP와 같이 모델에 매우 적합하다.

4. 실험 및 결과 분석

기존의 순차적인 기하학 처리 방식과 중첩 라이팅 방식을 비교하기 위해 각각 SeqGP 수행 모델과 OLGP 수행 모델을 그림 3과 같이 구성하였고, 두 가지 모델에 대한 비교의 공정성을 기하기 위하여 표 1과 같이 하드웨어 모델에서 기능 유닛의 개수를 동일하게 가정하였다.

표 1. 가정된 기능 유닛의 특성.

	number of FU	latency	thruput
FMUL	2	4	1
FADD	2	4	1
FDIV	1	7	7
SQRT	1	7	7
POW	1	4	4

실험 환경은 [1] 구조에서 paired-single 명령어를 사용했을 경우와 사용하지 않았을 경우 수퍼스칼라 프로세서에서의 기하학 처리 성능을 비교했던 Chia-Lin Yang의 방법 [6]을 사용하여 실험하였으며 OpenGL API 의 공개 라이브러리인 Mesa [9]를 사용하였다. 사용된 명령어 세트는 4 슬롯의 VLIW 명령어를 사용하였으며 각 모델에 대해 어셈블리 명령어 단계에서 수작업에 의해 최적화를 수행했다. 벤치마크 프로그램은 SPECviewperf의 5 가지 데이터 세트를 사용하였으며 각 데이터 세트는 SeqGP와 OLGP 시뮬레이터에서 각 단계를 통과하면서 수행시간이 산출된다.

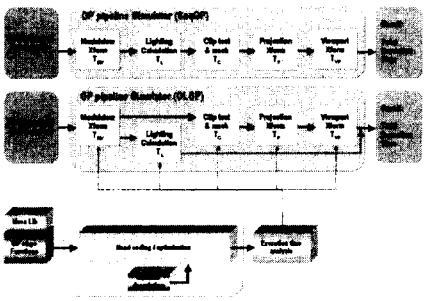

그림 3. 실험 환경.

그림 5 에서는 본 연구를 통해 작성된 기하학 파이프라인 시뮬레이터를 이용하여 3D 영상의 단계별 수행 시간에 대한 분포를 프로파일링한 결과를 나타낸다. 이 프로파일에서 전체 기하학 처리 수행시간 중에 라이팅 단계가 평균 57% 차지하고 있다.

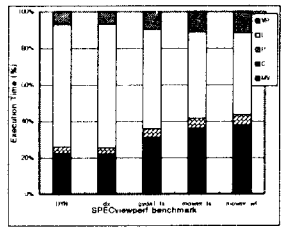

그림 4. 기하학 단계별 수행시간 분포.

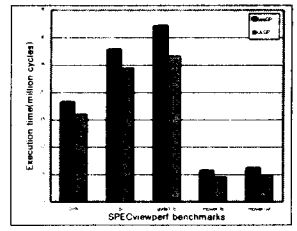

그림 5. SeqGP와 OLGP의 수행시간 비교.

그림 5 에서는 SPECviewperf 의 5 가지 데이터 세트에 대하여 각 모델별 수행시간을 비교한 것이다. 이와 같은 자료에 의해서, OLGP 에 의한 기하학 처리 성능향상 정도는 최대 12 ~ 21% 향상됨을 알 수 있었다. 이 결과는 완벽하게 최적화된 결과는 아니므로 더욱 향상될 수 있는 여지가 있다. 앞으로 진행될 실험에서는 OLGP 에 최적화될 수 있는 하드웨어 및 VLIW 구조를 제안함으로써 기하학 처리부의 오버헤드를 해결할 수 있을 것이다.

5. 결론

본 연구에서 제안하는 OLGP 구조는 기하학 처리부에서 처리 시간이 가장 긴 라이팅 단계를 다른 후속되는 단계들과 중첩시켜 수행하는 구조이다. 수행 모델은 많은 부동소수점 연산을 처리하는 두 가지 패스의 명령어들간의 병렬성을 충분히 활용할 수 있는 VLIW 구조를 기반으로 구성하였다. 실험 결과 기하학 처리를 위한 수행시간을 최대 21% 향상시킬 수 있었고, 이에 따라 전반적인 3D 그래픽스 시스템 성능이 대폭 향상될 것으로 기대된다.

6. 참고 문헌

[1] Advanced Micro Devices, Inc., " 3Dnow! ™ Technology: Architecture and implementations," Mar. 1999.

[2] K.Akeley, and T.Jermoluk, " High-Peformance Polygon Rendering," Computer Graphics, Vol. 22, pp 239-249, Aug. 1988.

[3] M.Oka, and M.Suzuoki, " Designing and Programming the Emotion Engine," Micro, Nov. 1999.

[4] Kurt Akeley, " RealityEngine Graphics, " In proceeding of SIGGRAPH' 93, pp.109-116.

[5] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes, " Computer Graphics - Principles and practice," 1995.

[6] Chia -Lin Yang, Barton Sano, and Alvin R.Lebeck, "Exploiting instruction level parallelism in geometry processing for three dimensional graphics applications," Micro in 1998.

[7] M.Segal and K.Akeley, " The OpenGL ™ Graphics System: A Specification," Mar, 23, 1998.

[8] Makoto Awaga, Tatsushi Ohtsuka, Hideki Yohizawa Shigeru Sasaki, "3D graphics processor chip set", Micro in Dec. 1995.

[9] Mesa library, http://www.mesa3d.org