효율적인 지역 프레임버퍼를 위한 병렬 래스터라이저의 설계

박 재 성○, 김 신 덕
연세대학교 컴퓨터과학과 병렬처리시스템 연구실

Design of Parallel Rasterizer for effective LFB memory

Jae-Seong Park and Shin-Dug Kim
Dept. of Computer Science, Yonsei University

요 약

폴리곤-렌더링을 위한 영상 합성 구조는 지역 프레임버퍼 메모리 비용이 큰 문제점을 가진다. 이를 개선하기 위해 화면-분할 방법과 가상 지역 프레임버퍼 방법이 도입되었으나 이 방법들 역시 상당한 메모리 비용이 요구된다. 본 논문에서는 지역 프레임버퍼 메모리 비용 측면에서 효율적이고, 영상 합성에 필요한 하드웨어를 제거하며, 동시에 영상 합성 시간을 숨길 수 있는 렌더링 시스템과 이에 필요한 병렬 래스터라이저를 설계한다.

1. 서론

고속 그래픽 가속기의 설계를 위해 지금까지 고려된 아키텍처는 크게 파이프라이닝(pipelining)과 병렬성(parallelism)이다. 일반적으로 3차원 그래픽 처리 과정에서의 연산 형태는 두 가지로 구분된다. 프리미티브(primitive)들을 원래의 좌표계에서 스크린 좌표계로 변환시키는 기하학 처리(geometry processing)와 기하학 정보를 픽셀 정보로 변환시키는 래스터라이제이션(rasterization)이다. 기하학 처리는 여러 개의 프리미티브들을 병렬로 처리함으로써, 그리고 래스터라이제이션은 여러 개의 픽셀이나 여러 개의 프리미티브들을 병렬로 처리함으로써 각각 병렬화 될 수 있다 [7].

폴리곤-렌더링을 위한 아키텍처로는 영상 합성 구조(image composition architecture)가 있다 [8]. 각 프로세서마다 전체 화면에 해당하는 픽셀 정보를 저장할 수 있는 크기의 지역 프레임버퍼를 가진다. 따라서 지역 프레임버퍼에 필요한 메모리 비용이 큰 문제가 된다. 이러한 지역 프레임버퍼 비용을 줄이는 방안으로 화면-분할 방법을 이용하거나[3] 가상 지역 프레임버퍼를 사용하였다 [9].

PixelFlow 시스템[3]에서 이용된 화면-분할 방법 [2, 3]은 화면을 일정한 영역으로 분할하여 각 영역을 렌더러(renderer)에 할당하고, 각 지역 프레임버퍼는 한 영역에 해당하는 픽셀 데이터들만을 저장하기 때문에 지역 프레임버퍼의 크기는 화면을 분할하는 만큼 줄일 수 있는 장점이 있으나, 부하 불균형을 초래하기 쉽고, 클리핑(clipping) 수행시간이 증가되는 단점이 있다. VC-1 시스템[9]에서 이용된 가상 지역 프레임버퍼 방법은 요구-페이징(demand-paging) 기법을 이용하여 지역 프레임버퍼를 운용하기 때문에 보다 적은 메모리를 사용하지만, 가상 지역 프레임버퍼에서 오버플로우(overflow)가 발생했을 때 처리해야 하는 오버헤드가 크고, 가상 지역 프레임버퍼 역시 상당한 양의 메모리를 요구한다.

본 논문에서는 3D-RAM [5, 6]을 전역 프레임버퍼로 사용한 객체 기반 렌더링 시스템[10]에 필요한 병렬 래스터라이저를 설계하고 그 성능을 분석한다. 제안 시스템은 객체 기반 렌더링을 이용하므로 화면-분할 방법의 문제점을 극복할 수 있고, 지역 프레임버퍼의 메모리 비용을 줄이며, 영상 합성에 필요한 하드웨어를 제거하고, 또한 영상 합성 시간을 숨길 수 있다.

본 논문의 2장에서는 관련연구에 대해 기술하고, 3장에서는 제안 시스템의 구조에 대해 설명한다. 4장에서는 제안 시스템의 성능을 비교 분석하고, 마지막으로 5장에서는 결론을 기술한다.

2. 관련 연구

영상 합성 구조는 각 렌더리마다 생성한 부분 영상을 저장할, 전체 화면을 저장할 수 있는 크기의 지역 프레임버퍼를 가진다. 렌더링이 끝난 후, 모든 지역 프레임버퍼의 내용은 영상 병합기(image merger)에 의해 병합된다. 이 구조의 가장 큰 단점은 지역 프레임버퍼의 메모리 비용에 있다. 프레임버퍼의 대역폭은 그래픽 시스템의 중요한 요소 중 하나로서 프레임버퍼 메모리는 속도가 빠를수록 유리하다. 그러나 렌더러의 수가 많을 경우 각 프레임버퍼마다 고속 메모리를 사용하는 것은 비실용적이다. 따라서 저속 메모리가 사용되며 이는 시스템 성능을 저하시키는 요인이 된다.

이 문제를 극복할 수 있는 방법 중 하나가 PixelFlow 시스템[3]에서 이용된 화면-분할 방법이다. 화면-분할 방법에서는 화면이 여러 개의 부분 영역으로 나뉘어지고, 각 지역 프레임버퍼는 하나의 영역에 해당하는 픽셀 정보만을 저장한다. 전체 화면의 픽셀 정보를 저장하기 위해 영상 병합기와 CRT사이에 전역 프레임버퍼가 존재한다. 래스터라이제이션에 앞서 각 렌더러는 기하학 처리를 수행하고 xy-버킷(bucket)을 이용하여 폴리곤들을 영역에 따라 분류한다. 이후 폴리곤들은 래스터되어 한 영역에 해당하는 영상들이 지역 프레임버퍼에 생성된다. 이 영상들은 병합되어 전역 프레임버퍼의 적절한 위치에 저장된다. 이 방법은 지역 프레임버퍼의 메모리 요구량을 분할 영역의 수에 반비례하여 줄일 수 있다. 그러나 폴리곤들이 하나의 영역에 집중되어 존재할 경우 부하 불균형을 초래하여 전체적인 시스템 성능을 저하시키며, 화면 경계뿐만 아니라 분할 영역의 경계에 위치하는 폴리곤들에 대해 클리핑을 수행하여야 하므로 클리핑 시간이 증가되는 단점이 있다

지역 프레임버퍼의 비용 문제를 극복하는 다른 방법으로

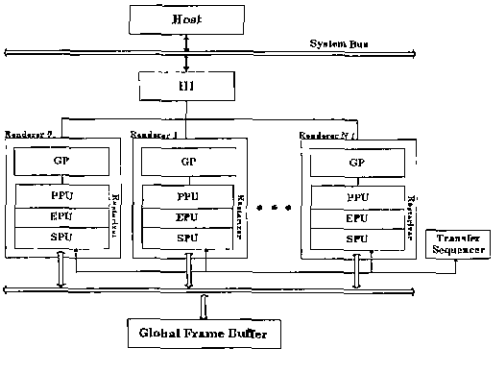
VC-1 시스템[9]에서 사용한 가상 지역 프레임버퍼 방법이 있다. 이 방법은 지역 프레임버퍼의 접근이 공간 지역성을 가지는 특성을 이용하여 메모리 요구량을 줄였고, 지역 프레임버퍼가 전체 화면의 픽셀 정보를 가상적으로 가지도록 요구-페이징 기법을 사용하였다. 화면은 동일한 크기의 영역인 패치(patch)들로 나뉘어지며, 지역 프레임버퍼는 영상 메모리와 패치 테이블이라는 두 가지 형태의 메모리로 구성된다. 영상 메모리는 픽셀 정보를 저장하며, 패치 테이블은 각 패치들의 영상 메모리 주소와 접근 여부를 유지한다. 지역 프레임버퍼의 접근시 공간 지역성이 약할 때, 영상 메모리가 부족한 경우가 발생하게 되는데 이를 지역 프레임버퍼 오버플로우라 한다. 이 경우, 지역 프레임버퍼의 픽셀 정보를 전역 프레임버퍼로 전송하고, 필요한 픽셀 정보를 전역 프레임버퍼로부터 가져와야 하는 오버헤드가 발생한다. VC-1 시스템은 16개의 렌더러로 구성된 경우 전체 화면 용량의 1/8 크기의 지역 프레임버퍼로 큰 성능 저하 없이 렌더링을 수행할 수 있다 [9].

3. 객체 기반 렌더링 시스템과 병렬 래스터라이저

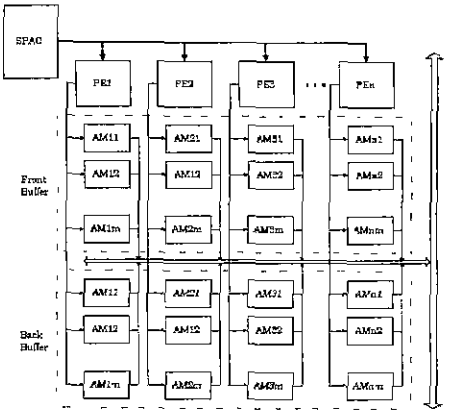
객체 기반 렌더링 시스템[10]의 구조는 [그림 1]과 같이 호스트 인터페이스(HI: Host Interface)와 렌더러, 전송순서 제어기(TS: Transfer Sequencer), 그리고 전역 프레임버퍼로 구성된다. HI는 호스트의 요구를 받아 원시 데이터를 각 렌더러에 부하 균형 정책에 따라 분배하는 역할을 한다. 부하 균형 정책은 본 논문의 초점을 벗어나므로 고려하지 않는다. 렌더러는 기하학 처리기(GP)와 래스터라이저(Rasterizer)로 구성된다. TS는 각 렌더러의 지역 메모리가 여유 공간이 없을 경우 발생하는 전역 프레임버퍼로의 전송요구를 제어한다. 4-way interleaved 100 MHz 3D-RAM [5,6]으로 구성된 전역 프레임버퍼는 렌더러로부터 전송된 픽셀 정보에 대해 초당 4억 개의 픽셀을 z-버퍼링한다.

래스터라이저는 스캔-컨버젼을 수행하는 PPU (Polygon Processing Unit)와 EPU (Edge Processing Unit), 그리고 구로우 쉐이딩 (Gouraud Shading) 방법에 따라 각 픽셀의 색깔과 깊이 정보를 계산하고, 이를 저장하는 SPU (Span Processing Unit)로 구성된다. SPU는 [그림 2]와 같이 32-비트 ALU로 구성된 n개의 PE와 각 PE마다 2 · m개의 지역 메모리들로 구성된다. 각 n · m개의 지역 메모리들은 선버퍼와 후버퍼로 더블 버퍼링을 지원한다 PE들에 픽셀들이 m번 매핑되어 픽셀 정보가 계산되면, 전송순서 제어기로 전송요구 신호를 보내고, 전송순서 제어기로부터 전송허가 신호가 오면 이들을 전역 프레임버퍼로 전송한다.

제안 시스템이 정상적으로 동작하기 위해서는 전역 프레임버퍼에서 병목현상이 발생하지 않는다고 가정할 때, 하나의 렌더



[그림 1] OBR 시스템의 구조



[그림 2] SPU의 구조

러가 렌더링을 수행하는 동안 다른 렌더러들은 픽셀 정보들을 전역 프레임버퍼로 전송하여 z-버퍼링할 수 있어야 한다. 즉 n · m개의 지역 메모리에 계산된 픽셀 정보가 저장되는 동안 다른 지역 프레임버퍼들의 픽셀 정보 전송이 가능해야 한다. 즉 아래의 식 (1)이 성립해야 한다.

$$T_{renderLFB} \geq T_{Zbuffer} \quad (1)$$

위의 식에서 좌변은 하나의 지역 프레임버퍼로 렌더링된 데이터가 저장되는 시간이고, 우변은 모든 지역 프레임버퍼들이 전역 프레임버퍼로 픽셀 정보를 전송하여 z-버퍼링하는 시간이다. $T_{renderLFB}$은 픽셀들이 PE에 매핑되어 렌더링을 마치는데 까지 걸리는 시간과 픽셀들이 PE에 매핑되는 수 (m)의 곱, 즉 $T_{renderLFB} = T_{render} \times m$으로 표현된다. T_{render}는 아래의 식 (2)와 같이 하나의 폴리곤을 렌더링하는 시간을 폴리곤 당 PE에 매핑되는 수로 나눈 값으로 정의한다. 식 (2)의 우변의 각 항은 다시 식 (3)과 같이 표현된다.

$$T_{render} = T^{render}_{polygon} / N^{raster}_{polygon} \quad (2)$$

$$\begin{aligned} T^{render}_{polygon} &= \max(T_{GP}, T_{rasterization}) \\ T_{rasterization} &= \max(T_{PPU}, T_{EPU}, T_{SPU}) \\ N^{raster}_{polygon} &= \left\lceil \frac{N^{polygon}_{pixels}}{n} \right\rceil \end{aligned} \quad (3)$$

식 (1)의 우변은 식 (4)와 같이 각 지역 프레임버퍼의 픽셀 데이터를 z-버퍼링하는 시간과 렌더러 수의 곱으로 정의한다 각 지역 프레임버퍼의 픽셀 데이터를 z-버퍼링하는 시간은 버스를 통해 전역 프레임버퍼로 전송하는 시간과 전송순서 제어기에 의해 제어되는 시간, 그리고 전역 프레임버퍼에서 z-버퍼링을 수행하는 시간의 합으로 표현된다. 식 (5)의 $T_{sequencer}$는 전송순서 제어기에 의하여 제어되는 시간이다

$$T_{Zbuffer} = N_{renderer} \times T_{ZbufferLFB} \quad (4)$$

$$T_{ZbufferLFB} = T_{transferLFB} + T_{sequencer} + T_{ZbufferGFB} \quad (5)$$

$$T_{transferLFB} = \frac{n \times m \times D_{pixel}}{R_{transfer}} \quad (6)$$

$T_{transferLFB}$는 다시 식 (6)와 같이 정의되며, D_{pixel}은 하나의 픽셀에 필요한 정보로서 80비트이다. 구로우 쉐이딩의 경우 한 픽셀 당 64비트가 필요하나 제안 시스템의 경우 지역 프레임버퍼가 화면상의 위치 정보를 가지지 않으므로, 화면상의 위치 정보(x, y)로 16비트가 추가로 필요하다. $R_{transfer}$는 지역 프레임버퍼와 전역 프레임버퍼를 연결하는 버스의 전송률이다.

<표 1> 각 실험 영상의 특성

실험 영상	총 폴리곤 수	평균 스팬수	평균 픽셀수
pool	3,678	7	50
pool1	17,152	3	25
pool2	68,608	2	10
room1	3,096	11	140
room2	10,560	5	45
room3	42,240	3	14
car	2,230	27	966
woodchair	4,195	5	18
zip	10,992	2	6
sofa	2,124	4	75

<표 2> 실험에 이용된 가정들

GP	i860
Rasterizer	100 MHz
Bus	256-bits at 100 MHz
GFB	4 way-interleaved 100 MHz 3D-RAM

4. 제안 구조의 특징 및 성능 평가

3D-RAM을 전역 프레임버퍼로 이용하는 제안 시스템은 아래와 같은 특징을 가진다.

4.1 지역 프레임버퍼 메모리 비용의 절감

실험에 사용된 영상들은 <표 1>과 같은 특성을 가진다. 이 실험 영상들에 대하여, <표 2>의 가정하에 SPU에 포함되는 PE의 수(n)와 지역 프레임버퍼의 크기($2 \cdot n \cdot m \cdot 80$ 비트), 그리고 렌더러의 수를 결정하였다.

이 실험 영상들에 대하여 렌더러의 수가 변함에 따라 제안 시스템이 이상적으로 동작하는 지역 프레임버퍼의 크기는 [그림 3]과 같다. [그림 3]은 시스템이 16개의 렌더러로 구성된 경우 각 SPU는 128개의 PE로 구성할 수 있으며, 지역 프레임버퍼는 10KB의 크기면 충분함을 보여준다.

<표 3>은 1280×1024의 화면 크기와, 구로우 쉐이딩을 이용한다고 가정했을 때, 16개의 렌더러로 구성된 각 시스템들에 필요한 지역 프레임버퍼의 크기를 비교한 것이다. 제안 시스템의 경우 다른 시스템과 비교하여 메모리 비용 면에서 매우 효율적임을 알 수 있다.

4.2 영상 합성에 필요한 하드웨어의 제거

제안 시스템은 Z-버퍼링이 3D-RAM으로 구성된 전역 프레임버퍼에서 수행되므로 영상 합성에 필요한 부가적인 하드웨어를 요구하지 않는다. 영상 합성을 이진 트리 형태로 구성하였을 경우, 렌더러의 수가 $N(=2^n)$이면, N-1개의 영상 합성 유닛이 필요하다.

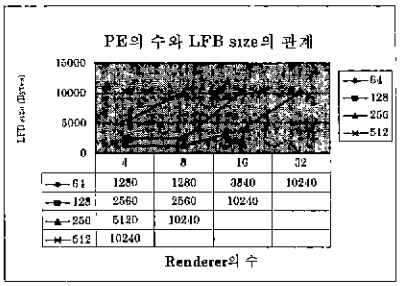
4.3 영상 합성 시간과 렌더링 시간의 중첩

제안 시스템은 지역 프레임버퍼의 픽셀 정보를 전역 프레임버퍼로 전송하는 시간과 렌더링 시간을 중첩시킬 수 있으므로 데이터 전송시간을 숨길 수 있는 장점이 있다. 영상 합성 구조의 경우, 모든 렌더러가 렌더링이 끝났을 때, 동기적으로 지역 프레임버퍼의 픽셀 정보를 합성하지만, 제안 시스템의 경우 픽셀 정보의 합성이 비동기적으로 수행된다.

그러나 제안 시스템은 확장성 측면에서 문제점을 가진다. 이는 지역 프레임버퍼와 전역 프레임버퍼를 연결하는 버스의 불충분한 대역폭 때문이다. 하지만 프로세서의 연산능력이 발전함에 따라 무한한 확장성을 요구할 필요는 없다

<표 3> 각 시스템에 요구되는 지역 프레임버퍼의 크기

렌더링 시스템	VC-1	화면-분할방법	제안 시스템
지역 프레임버퍼의 크기	240 MB	120 MB	120 KB



[그림 3] PE의 수와 지역 프레임버퍼의 크기

5. 결론

본 논문에서는 영상 합성 구조가 가지는 단점인 지역 프레임버퍼 메모리 비용의 문제를 해결하기 위한 렌더링 시스템을 제안하고 병렬 래스터라이저의 구조를 설계하였다. 제안 시스템은 영상 합성 구조를 기반으로 하는 다른 시스템들과 비교하여 메모리 비용 측면에서 매우 효율적이며, 영상 합성에 요구되는 하드웨어를 제거할 수 있고, 지역 프레임버퍼의 픽셀 정보를 전역 프레임버퍼로 전송하는 것과 렌더링을 중첩시킬 수 있어 데이터 전송시간을 줄일 수 있다.

참고 문헌

[1] K. Akeley and Jermoluk, "High-Performance Polygon Rendering," *Proceedings of SIGGRAPH '88* (August 1988), p.p.239-246.

[2] H. Fuchs, J. Poulton, J. Eyles, S. Molnar, et. al, "Pixel-Planes5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories," *Proceedings of SIGGRAPH '89* (July 1989), p.p.79-88

[3] S. Molnar, J. Eyles, J. Poulton, "PixelFlow: High-Speed Rendering Using Image Composition," *Proceedings of SIGGRAPH '92* (July 1992), p.p.231-240.

[4] K. Akeley, "RealityEngine Graphics," *Proceedings of SIGGRAPH '93* (August 1993), p.p.109-116.

[5] Michael F. Deering, Stephen A. Schlapp and Michael G. Lavelle, "FBRAM: A new Form of Memory Optimized for 3D Graphics," *Proceedings of SIGGRAPH '94* (July 1994), p.p.167-174.

[6] Kazunari Inoue, Hisashi Nakamura and Hiroyuki Kawai, "A 10 Mb Frame Buffer Memory with Z-Compare and A-Blend Units," *IEEE Journal of Solid-Circuits*, Vol 30, No. 12, December 1995, p.p.1563-1568.

[7] S. Molnar, "Image Composition Architectures for Real-Time Image Generation," *Ph.D. dissertation. UNC*, 1991

[8] S. Molnar, "Combining Z-buffer Engines for Higher-Speed Rendering," *Advances in Computer Graphics Hardware III*, Eurographics Seminars, 1988, p.p.171-182.

[9] Satoshi Nishimura, Tosiyasu L. Kunii, "VC-1: A Scalable Graphics Computer with Virtual Local Frame Buffers," *Proceedings of SIGGRAPH '96* (August 1996), p.p.365-372.

[10] 오인홍, 박재성, 김신덕, "개체기반 렌더링을 위한 그래픽 가속기 설계," *한국 정보과학회 가을 학술발표논문집 (1997)*, Vol. 24, No. 2. p.p.409-412.