3차원 그래픽 래스터라이제이션에서의 텍스쳐/픽셀 캐쉬 성능분석

김일산[0] 박기호 이길환 박우찬 한탁돈
연세대학교 미디어시스템 연구실
(sany[0], ghpark, kiwh, chan, hantack)@kurene.yonsei.ac.kr

Performance Analysis of Texture / Pixel Cache in 3D Graphics Rasterization

Il-San Kim[0], Gi-Ho Park, Kil-Whan Lee, Woo-Chan Park, Tack-Don Han
Dept. of Computer Science, Yonsei University

요 약

본 논문에서는 3차원 그래픽의 래스터라이제이션 단계에서 발생하는 메모리 트래픽 문제를 해결하기 위해 사용되는 텍스쳐 및 픽셀 캐쉬에 대한 성능을 분석하였다. 이를 위해 화면의 해상도, 컬러 정보, 깊이 정보 및 캐쉬 구성의 변화에 따른 이들 캐쉬의 성능변화를 살펴보았으며 실험결과 텍스쳐 캐쉬와 픽셀 캐쉬의 설계 시에 블록 크기에 의한 영향이 매우 중요함을 알 수 있었다. 특히 픽셀 캐쉬의 경우에는 시간적 지역성은 거의 없으며 매우 큰 공간적 지역성을 보이므로 이를 잘 반영할 수 있는 캐쉬 구조가 필요하다.

1. 서 론

최근 급속한 하드웨어의 발전으로 PC급에서도 실시간 렌더링이 가능해지면서 여러 부문에서 3차원 그래픽의 활용이 증대되고 있고 특히 게임과 같은 엔터테인먼트 분야에서 많이 사용되고 있다. 하지만 사진수준의 사실적인 화면을 초당 30 프레임 이상으로 구현하기에는 아직 하드웨어적인 제약이 많은 편이며, 그 중 메모리 대역폭 문제[1,2,4,5]는 지속적으로 그 중요성이 증대되고 있다. 이러한 대역폭 문제를 해결하기 위하여 최근의 3차원 그래픽 가속기들은 픽셀 캐쉬[1]와 텍스쳐 캐쉬[2]를 사용하고 있다.

본 논문에서는 3차원 그래픽에서 래스터라이제이션 단계의 픽셀처리 과정에서 사용되는 이들 텍스쳐 및 픽셀 캐쉬의 성능을 다양한 벤치마크를 통하여 분석하였다. 이러한 분석은 캐쉬용량, 블록크기, 연관도 등의 캐쉬 설계시의 기본 설계 변수뿐만 아니라, 화면의 해상도, 컬러 및 깊이 정보, 밉맵 시 필터링 방법 등의 변화에 따른 캐쉬 메모리의 성능변화를 살펴보았다. 이러한 분석에 따라 텍스쳐 캐쉬의 경우에는 연관도나 용량 보다는 블록크기에 의한 성능변화가 컸으며, 필터링 기법의 변화에 따른 전송량의 변화가 큼을 알 수 있었다.

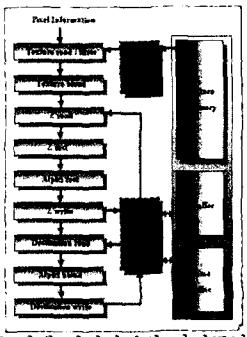
픽셀 캐쉬의 경우에는 매우 큰 공간적 지역성(spatial locality)으로 인해 큰 블록크기를 갖는 캐쉬구조가 우수한 성능을 보였다. 일반적인 캐쉬와는 달리 용량이나 연관도에 의한 성능의 변화는 상당히 적었다.

본 논문은 2장에서는 3차원 그래픽 가속기의 처리단계에 대하여 개략적으로 설명하고, 3장에서는 실험환경 및 실험결과를, 4장에서는 결론 및 향후 연구방향을 제시하였다.

2. 연구 배경 및 관련 연구

3차원 그래픽 처리과정은 크게 지오메트리(geometry) 처리단계와 래스터라이제이션(rasterization) 처리단계로 나뉘어진다. 그 중 래스터라이제이션 단계는 지오메트리 단계에서 처리된 정점들을 화면에 그리는 역할을 하는데 크게 스캔 컨버젼(scan conversion)과 픽셀 처리작업(pixel processing)으로 나눌 수 있다. 스캔 컨버젼은 삼각형 처리(triangle setup), 모서리 처리(edge processing) 및 선 처리(span processing)등의 작업을 통해 기울기 및 여러 속성값을 계산한다. 스캔 컨버젼이 완료되면 계산된 속성값을 픽셀 단위로 텍스쳐 및 알파 값과 혼합해 프레임 버퍼에 저장하는데 이를 픽셀 처리작업이라 한다. OpenGL에서 픽셀 처리작업의 파이프라인 과정은 그림 1와 같다(픽셀 캐쉬는 깊이 정보와 컬러 정보를 모두 저장한다고 가정한다).

스캔 컨버젼에서 처리된 픽셀의 정보가 입력되면, 텍스쳐 메모리에서 데이터가 읽혀지고 깊이 정보를 참조하여 현재 픽셀의 사용 유무가 결정된다. 그 결과에 따라 투명도 검사(alpha test)를 수행하고 깊이 정보는 깊이 버퍼에 갱신된다. 또한 투명도 결과에 따라 프레임 버퍼에서 동일한 위치의 컬러 값을 읽어 혼합(alpha blending)한 후 다시 프레임 버퍼를 갱신한다.



[그림 1] 픽셀 처리작업의 파이프라인 과정

메모리 병목현상의 대부분은 텍스쳐 메모리, 깊이 버퍼 및

프레임 버퍼 접근에서 발생한다. 대부분의 가속기들은 효과적인 텍스쳐 매핑을 위해서 바이리니어(bilinear) 또는 트라이리니어(trilinear) 방식의 밉맵 기법[3]을 사용하는데, 이것은 4개 또는 8개의 텍스쳐 데이터 전송을 필요로 한다. 또한 깊이 버퍼와 프레임 버퍼에서는 읽기 및 쓰기작업이 모두 수행되는데 이러한 처리과정은 픽셀단위로 처리되기 때문에 해상도가 증가하면 전송량 또한 증가하게 된다. 표 1은 화면 구성요소 변화에 따른 전송량의 변화를 보이고 있다.

[표 1] 응용프로그램에 따른 전송량

	응용프로그램 1	응용프로그램 2
해상도 및 컬러비트	640×480, 16비트	1024×768, 32비트
깊이비트 및 화면복잡도	16비트, 2	32비트, 3
밉맵 방법	바이리니어	트라이리니어
전송 요구량	9.4 MB/sec	108MB/sec

이때 초당 60프레임의 처리성능이 필요하다면 응용프로그램 1은 약 562.5MB/sec, 응용프로그램 2는 약 6.5GB/sec의 대역폭이 요구되는데, 최근에 개발 중인 게임 등과 같은 프로그램들은 고해상도에서 많은 폴리곤 및 높은 프레임 수를 요구함으로 전송량은 점점 더 증가하는 추세이다.

위의 예에서도 알 수 있듯이 3차원 가속기의 성능에는 메모리 대역폭이 대단히 중요한 역할을 차지한다. 기존 연구로는 3차원 그래픽 가속기 구조에서 텍스쳐 캐쉬의 유용성 여부를 제시한 연구[2]와 깊은 파이프라인 구조의 특성을 이용한 선인출(prefetch) 기법[5]등에 대한 연구가 있다. 그리고 Mitra와 Chiueh의 논문[4]에서는 3차원 가속기 구조의 래스터라이제이션단계에서 처리허야 할 연산의 양, 텍스쳐 대역폭, 픽셀 스탬프(pixelstamp), 파이프라인 단계의 변화 등에 대하여 분석하였다. 하지만 메모리 전송량과 참조특성에 대한 자세한 분석이 부족하였고, 사용된 *Viewperf*의 벤치마크들은 게임과 같은 실제 응용프로그램의 특성을 반영하기에는 미비한 점이 있었다. 이러한 점을 고려하여 본 논문에서는 *Viewperf*뿐만 아니라 게임등과 같은 실제 응용프로그램들도 이용하였으며, 화면의 해상도, 깊이 및 컬러 정보, 밉맵의 필터링 방법 등의 변화에 따른 텍스쳐 캐쉬 및 픽셀 캐쉬의 성능을 분석하였다.

3. 실험환경 및 성능분석

3.1 실험 환경

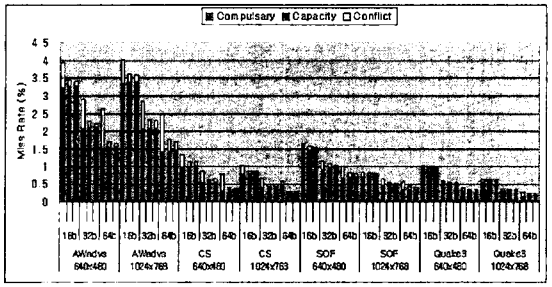
3차원 응용프로그램의 메모리 접근을 분석하기 위해서 본 논문에서는 RedHat Linux V7.1 운영체제와 OpenGL을 지원하는 Mesa 3D V3.4를 이용하여 텍스쳐 및 픽셀 메모리 참조에 대한 트레이스를 생성하였다. 표 2는 시뮬레이션을 수행한 캐쉬 구성 및 벤치마크들을 제시하고 있으며, 캐쉬 시뮬레이터로는 Dinero IV를 사용하였다. 모든 벤치마크들은 동일하게 50프레임을 수행하였으며, 화면의 해상도, 컬러 정보 및 밉맵시 필터링 방법 등을 변경하면서 트레이스를 생성하였다. 이에 따라 *Quake3*의 경우에는 최소 1억 개에서 최대 약 15억 개 이상의 메모리 참조를 트레이스로 생성하였다.

[표 2] 캐쉬 실험환경 및 벤치마크 리스트

		텍스쳐 캐쉬	픽셀 캐쉬
캐쉬 구성	캐쉬크기	8 ~ 128K바이트	2 ~ 64K바이트
	블록크기	8 ~ 128바이트	
	연관도	1, 2, 4, 8, 완전연관	
벤치마크		Tunnel, Teapot, AWadvs, Quake3, Crystal Space(CS), Soldier of Fortune(SOF)	Teapot, AWadvs, Lightscape(Light), Quake3

3.2 텍스쳐 캐쉬 성능분석

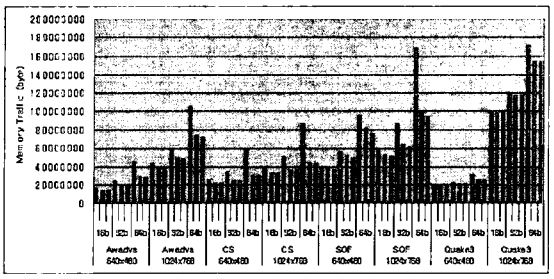
텍스쳐 데이터의 전송은 대부분의 경우 전체 메모리 대역폭의 상당부분을 차지하는데, 특히 3차원 게임들과 같이 고해상도의 텍스쳐 데이터를 많이 사용하는 경우는 그 비중이 더 크다. 그림 2는 캐쉬의 크기가 16K바이트일 때 텍스쳐 캐쉬의 접근실패율을 보이고 있으며, 같은 블록크기 내에서는 왼쪽부터 1, 2, 4의 연관도를 갖는 캐쉬의 경우를 나타낸다.



[그림 2] 텍스쳐 캐쉬 참조실패율

텍스쳐 캐쉬의 접근실패율을 살펴보면 연관도가 1에서 2로 증가하면 충돌접근실패가 크게 감소하지만 그 이상의 연관도에서는 큰 차이를 보이지 않는다. 해상도에 따른 접근실패율은 벤치마크에 따라 다른 성향을 보이는데 *CS, SOF, Quake3*에서는 해상도가 증가함에 따라 접근실패율이 감소하였고 *Tunnel, Teapot, AWadvs*에서는 해상도가 증가하면서 접근실패율이 증가하였다.

그림 3은 16K바이트의 캐쉬크기에서의 텍스쳐 메모리 전송량을 보이고 있다.

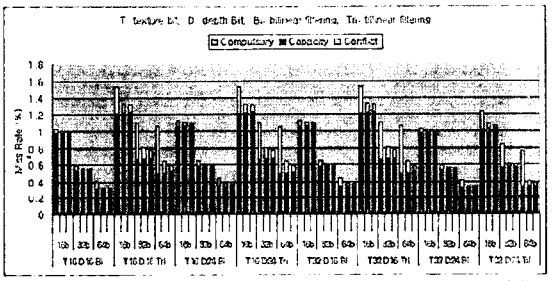


[그림 3] 텍스쳐 메모리 전송량

메모리 전송량은 50프레임을 수행하면서 캐쉬의 접근실패로 인해 발생한 전송량을 누적한 것이다. 메모리 전송량은 연관도가 1에서 2로 증가하면서 큰 폭으로 감소하였고 그 이상의 연관도에서는 큰 차이가 나타나지 않았다. 블록 크기가 증가함에 따라 전송량도 증가하였다. 그런데 한가지 특이한 점은 Quake3의 경우 640x480의 해상도에서 상당히 낮은 메모리 전송량을 보였는데, 이것은 벤치마크가 자체적으로 화면 구성을 최적화하기 때문이고 컬러 정보나 깊이 정보, 해상도가 증가하면 전송량도 점차적으로 증가하였다.

전송량과 캐쉬의 접근실패율의 관계를 살펴 보면, 텍스쳐 캐쉬에서 블록의 크기가 증가하면서 접근실패율은 감소하였지만 메모리 전송량은 증가하는 성향을 보였다. 따라서 접근실패율과 전송량을 고려하여 블록 크기를 결정해야 하는데, 텍스쳐 데이터 전송의 경우에는 블록의 크기를 줄여 전송량을 줄이는 것이 보다 효과적인 성능을 보일 것이다. 이는 선인출 기법[5]을 통해 접근지연 문제는 상당 부분 해소가 가능하기 때문이다.

그림 4는 *Quake3*에서 640×480의 고정된 해상도와 16K바이트의 용량을 갖는 캐쉬에서 필터링 방법과 컬러 정보, 깊이 정보를 변화시킬 때의 접근실패율을 보이고 있다.

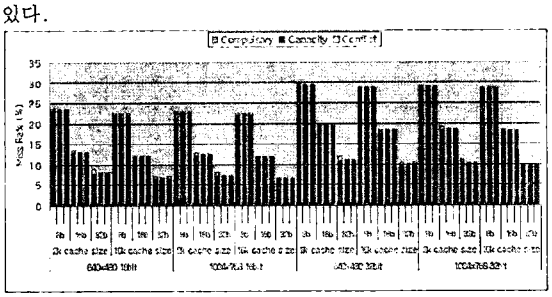


[그림 4] *Quake3*의 화면구성요소 변화에 따른 접근실패율

그림에서 T16D24Tri는 16비트의 텍스쳐 데이터, 24비트의 깊이 정보, 그리고 트라이리니어 필터링 방법을 의미한다. 바이리니어에서 트라이리니어로 필터링 방법이 바뀌는 경우에는 접근실패율이 증가하지만 컬러 정보나 깊이 정보가 변하는 경우에는 접근실패율의 변화는 일정하지 않으며 큰 차이를 나타내지 않는다. 본 논문에 제시하지는 않았으나 메모리 전송량의 결과 역시 동일한 성향을 보인다.

3.3 픽셀 캐쉬 성능분석

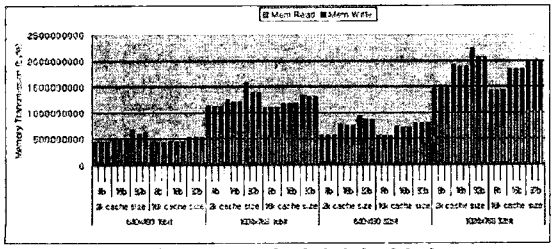
픽셀 캐쉬에 대한 시뮬레이션은 픽셀 캐쉬가 깊이 버퍼와 프레임 버퍼의 정보를 모두 저장하도록 하고 수행되었으며 그림 5는 *Quake3*에 대한 픽셀 캐쉬의 접근실패율을 보이고 있다.



[그림 5] *Quake3*의 픽셀캐쉬 접근실패율

위의 결과에서 보면 접근실패율은 캐쉬의 크기나 해상도, 연관도에는 별다른 영향을 받지 않지만 블록의 크기가 2배로 증가하면 32바이트보다 작은 경우에는 40%~50%, 32바이트보다 큰 경우에는 10~20%정도 감소한다. 이것은 한번 처리된 픽셀은 짧은 시간 내에 재사용되지 않는 픽셀 처리작업의 특징 때문이다. 따라서 해상도의 변화도 픽셀 캐쉬의 접근실패율에 거의 영향을 미치지 않는다. 논문에서는 *Quake3*에 대한 자료만 제시하였으나 이러한 성향은 모든 벤치마크들에서 공통적으로 보인다.

메모리 전송량의 경우는 해상도와 컬러 정보가 증가함에 따라 늘어난 픽셀의 수에 비례하여 증가하는 현상을 보인다. 그림 6은 *Quake3*에서의 메모리 전송량을 나타낸 것이다.



[그림 6] *Quake3*의 픽셀캐쉬 전송량

위 그림에서 나타나듯이 데이터 전송량의 경우도 접근실패율의 경우와 같이 캐쉬의 크기나 해상도, 연관도 등에 별다른 영향을 받지 않는다. 하지만 블록 크기의 증가에 따라 40~50% 정도 접근실패율이 감소하기 때문에 전송량의 증가는 상대적으로 크지 않다. 픽셀 처리단계에서는 대역폭 문제와 더불어 접근지연 역시 중요한 고려사항이기 때문에, 픽셀 캐쉬의 경우는 작은 용량의 큰 블록 크기를 갖는 캐쉬를 사용하는 것이 효과적인 방법이라 할 수 있다.

4. 결 론

본 논문에서는 3차원 그래픽 가속기의 래스터라이제이션 단계에서 텍스쳐 캐쉬와 픽셀 캐쉬의 성능을 분석하였다.

텍스쳐 캐쉬의 경우는 연관도가 1에서 2로 증가하면 접근실패율이 상당히 감소하지만 그 이상의 연관도에서는 큰 차이를 보이지 않는다. 블록의 크기가 증가함에 따라 접근 실패율은 감소하지만 전송량은 상당히 증가하는 성향을 보였는데, 텍스쳐 처리과정에서는 접근지연시간보다 전송량이 더 큰 영향을 미치기 때문에 작은 블록의 크기를 갖는 텍스쳐 캐쉬가 보다 바람직하다고 하겠다.

픽셀 캐쉬는 상당히 흥미로운 결과를 볼 수 있었다. 일반적으로 캐쉬의 크기 및 연관도가 증가하면 접근실패율은 감소하는 성향을 보이는데, 픽셀 캐쉬에서는 이들의 변화가 접근실패율에는 별다른 영향을 주지 못했다. 하지만 블록 크기의 증가에는 매우 민감한 반응을 보였으며, 블록 크기가 2배로 증가하는 경우 접근실패율이 40~50%까지 감소하였다. 따라서 픽셀 처리작업에서는 전송량과 함께 접근지연시간이 성능에 큰 영향을 미치기 때문에 큰 크기의 블록을 갖는 픽셀 캐쉬가 유용함을 알 수 있었다.

향후에는 응용프로그램의 화면구성과 텍스쳐 데이터 접근과의 연관성에 대한 분석을 수행하려 한다. 또한 픽셀 캐쉬의 실험결과에서 보인 매우 큰 공간적 지역성을 효과적으로 이용할 수 있는 캐쉬 구조에 관한 연구를 수행할 예정이다.

5. 참 고 문 헌

1. NVIDIA Corp, "GeForce4: Lightspeed II Memory Architecture," Technical Brief, 2002
2. Ziyad S. Hakura, Anoop Gupta, "The Design and Analysis of a Cache Architecture for Texture Mapping," Proceedings of the 24th International Symposium on Computer Architecture, 1997
3. L. Williams, "Pyramidal Parametrics," Computer Graphics(SIGGRAPH' 83 Proceedings), vol. 17, p.p. 1-11, July, 1983
4. T. Mitra, T. Chiueh, "Dynamic 3D Graphics Workload Characterization and the Architectural Implications," International Symposium on Micro-architecture, 1999
5. Homan Igehy, Matthew Eldridge, Kekoa Proudfoot, "Prefetching in a Texture Cache Architecture," Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware 98