# 깊이 검사 결과에 의한 선택적 적재 방법을 가지는 픽셀 캐쉬 구조

## (A Pixel Cache Architecture with Selective Loading Scheme based on Z-test)

이 길 환 † 박 우 찬 †† 김 일 산 † 한 탁 돈 †††
(Kil-Whan Lee) (Woo-Chan Park) (Il-San Kim) (Tack-Don Han)

**요 약** 최근 대부분의 3차원 그래픽 렌더링 프로세서는 메모리 접근 지연 및 대역폭을 줄이기 위해 깊이 데이타와 색상 데이타를 가지고 있는 픽셀 캐쉬(pixel cache)를 사용한다. 본 논문에서는 렌더링 프로세서의 성능을 향상시키기 위한 효과적인 픽셀 캐쉬 구조를 제시한다. 제안하는 픽셀 캐쉬 시스템은 기본적으로 3차원 렌더링 프로세서에서 수행되는 깊이 검사(Z-test) 결과에 의해 선택적으로 깊이 데이타를 캐쉬에 저장하며, 색상 데이타는 별도의 버퍼에 저장한다. 시뮬레이션 결과에 의하면 16K바이트의 제안하는 캐쉬 시스템이 32K바이트의 일반적인 캐쉬보다 더 나은 성능을 보인다.

**키워드** : 3차원 그래픽스 하드웨어, 픽셀 캐쉬, 깊이 검사

***Abstract*** Recently most of 3D graphics rendering processors have the pixel cache storing depth data and color data to reduce the memory latency and the bandwidth requirement. In this paper, we propose the effective pixel cache for improving the performance of a rendering processor. The proposed cache system stores the depth data selectively based on the result of Z-test and the color data are stored into the auxiliary buffer. Simulation results show that the 16Kbyte proposed cache system provides better performance than the 32Kbyte conventional cache.

**Key words** : 3D Graphics Hardware, Pixel Cache, Depth Test

## 1. 서 론

최근 고성능 렌더링(Rendering) 프로세서들이 거의 모든 PC에 도입되고 있으며, 이에 따른 렌더링 프로세서의 성능은 급격히 발전하였다. 그러나, 메모리 성능은 렌더링 프로세서의 처리 능력을 따라오기에는 부족하기 때문에 3차원 가속기 성능 제약의 주요한 원인이 되고 있다[1-5].

최근 대부분의 렌더링 프로세서는 메모리 접근 지연 및 대역폭을 줄이기 위해 깊이 데이타와 색상 데이타를 가지고 있는 픽셀 캐쉬(Pixel Cache)와 텍스쳐(Texture) 데이타를 가지고 있는 텍스쳐 캐쉬를 사용하고 있다[1,3,10,11]. 텍스쳐 캐쉬에 대한 연구로써 텍스쳐 캐쉬의 효율성 분석 및 텍스쳐 선인출 등 텍스쳐 데이타에 대한 접근 지연 및 대역폭을 줄이기 위한 연구 등이 있었다[1,2]. 그러나 픽셀 캐쉬의 경우에는 현재까지 구체적인 구조가 제시된 연구가 발표되지 않았다.

본 논문에서는 3차원 그래픽 벤치마크를 가지고 다양한 캐쉬 시뮬레이션을 수행하였고 이를 분석하여 효과적인 픽셀 캐쉬 구조를 제시한다. 시뮬레이션에 의하면, 픽셀 캐쉬의 특성은 마이크로프로세서에서 쓰이는 일반적인 캐쉬와는 다른 특성을 보인다[5,8]. 픽셀 캐쉬의 경우에 캐쉬 접근 실패율이 마이크로프로세서에서의 캐쉬에 비해서 높은 편이며, 높은 공간적 지역성을 가진다.

깊이 데이타의 경우에는 공간적 지역성이 높을 뿐만 아니라, 깊이 검사 시 성공한 데이타가 실패한 데이타에 비해 높은 재사용률을 가진다. 색상 데이타의 경우에는 시간적 지역성이 거의 없고 주로 공간적 지역성만을 가진다. 이에 제안하는 픽셀 구조는 기본적으로 수행되는 깊이 검사 결과에 의해 선택적으로 깊이 데이타를 캐쉬

· 이 논문은 과학기술부 국가지정연구실(NRL)사업의 지원에 의해 연구되었음
† 비 회 원 : 연세대학교 컴퓨터과학과
kiwh@kurene.yonsei.ac.kr
sany@kurene.yonsei.ac.kr
†† 비 회 원 : 세종대학교 컴퓨터공학부 조교수
pwchan@korea.com
††† 종신회원 : 연세대학교 컴퓨터과학과 교수
hantack@kurene.yonsei.ac.kr
논문접수 : 2003년 3월 17일
심사완료 : 2003년 6월 25일

에 적재하도록 구성하였다. 즉, 깊이 검사 시 성공한 깊이 데이타는 캐쉬에 적재하고, 실패한 깊이 데이타와 색상 데이타는 별도의 버퍼에 적재한다. 실패한 깊이 데이타가 재사용될 경우에는 버퍼에서 캐쉬로 적재한다. 이와 같이 실패한 깊이 데이타와 색상 데이타를 별도의 버퍼에 적재함으로써 캐쉬 오염(Cache Pollution)을 줄일 수 있다. 색상 데이타는 별도의 캐쉬 없이 적은 엔트리를 가지는 작은 버퍼만을 사용함으로써 하드웨어 비용 및 전력 소모를 줄일 수 있다.

성능 평가 결과에 의하면, 시뮬레이션에 사용된 모든 벤치마크들에서 같은 크기를 가지는 일반적인 캐쉬 구조에 비해 평균적으로 20%정도 향상된 평균 메모리 접근 사이클(Average Memory Access Cycle)을 보일 뿐만 아니라 두 배 큰 크기를 가지는 일반적인 캐쉬 구조보다도 10.7%정도 향상된 성능을 보인다.

다음 장에서는 전형적인 픽셀 래스터라이제이션 파이프라인(Rasterization Pipeline)에 대해서 소개하고, 관련 연구를 제시한다. 3장에서는 제안한 픽셀 캐쉬에 대해서 설명한다. 4장에서는 제안한 캐쉬와 일반적인 캐쉬에 대한 성능 평가 결과를 제시한다. 마지막으로 5장에서 결론을 제시한다.

## 2. 연구 배경 및 관련 연구

전형적인 픽셀 래스터라이제이션 파이프라인은 그림 1과 같다[5-7]. 입력되는 프래그먼트(fragment) 정보는 보간을 통하여 생성된 색상 데이타, 3차원 위치 좌표인 (x,y,z), 텍스쳐 좌표 등이 포함된다. 처음 두 개의 파이프라인 단계에서는 해당 텍스쳐 좌표에 대하여 4개 혹은 8개의 텍셀(Texel)을 텍스쳐 캐쉬로부터 읽기 연산을 하고 필터링을 수행하여 한 개의 텍셀을 생성하고, 이를 입력되는 프래그먼트 정보의 일부인 색상 데이타와 혼합한다[1]. 다음으로 프래그먼트 정보의 알파 데이타와 필터링된 텍셀의 알파 데이타를 검사한다. 여기서 알파 데이타는 투명도를 나타낸다. 평가의 결과가 성공이면 다음 파이프라인으로 계속 진행이 되고 실패이면 다음 파이프라인으로 진행되지 않고 현재의 프래그먼트 정보는 버려진다.

다음의 두 개의 파이프라인 단계에서는 픽셀 캐쉬로부터 깊이 데이타에 대한 읽기 연산을 수행하고 현재의 프래그먼트에 대한 깊이 데이타와 비교한다. 이 두 가지 과정을 각각 깊이 읽기와 깊이 검사라고 한다. 만약 검사의 결과가 실패이면, 즉 현재의 프래그먼트가 이전에 처리된 영상으로 인하여 안보이면, 현재의 프래그먼트는 파이프라인에서 버려지게 된다. 만약 성공이면 현재의 프래그먼트의 깊이 데이타가 픽셀 캐쉬에 쓰여지게 된다. 마지막 세 개의 단계에서는 픽셀 캐쉬로부터 색상

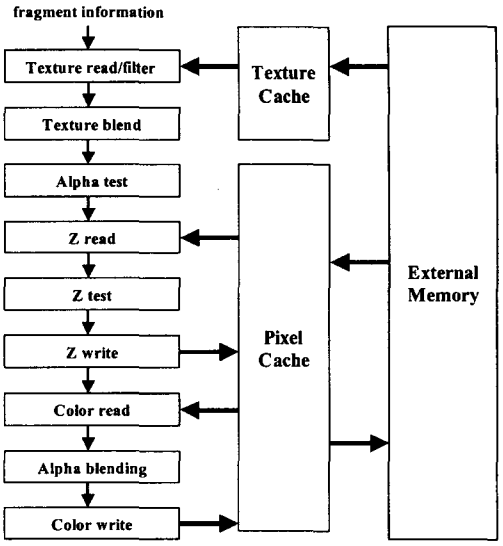


그림 1 전형적인 픽셀 래스터라이제이션 파이프라인

데이타에 대한 읽기 연산이 수행되고, 이 데이타와 텍스쳐 혼합이 수행된 데이타에 대하여 알파 혼합이 수행되고, 최종적인 색상 데이타가 픽셀 캐쉬에 쓰여지게 된다.

래스터라이제이션 처리단계에서의 전체 메모리 전송량의 대부분은 픽셀 처리 파이프라인 단계의 텍스쳐 데이타 접근과 프레임 버퍼 메모리 접근에서 일어난다. 3차원 그래픽 하드웨어에서 메모리 문제는 성능향상에 영향을 미치는 중요한 요소 중 하나이지만 메모리 데이타나 접근 성향에 대한 분석과 메모리 구조등에 대한 연구는 상당히 적은 편이었다. [4]에서 동적 데이타(*dynamic workload*)에 대한 분석 중 래스터라이제이션 단계 내의 텍스쳐 전송량과 메모리 뱅크 활용도에 대하여 언급한 정도이고, 텍스쳐 데이타로 인한 문제를 해결하기 위하여 텍스쳐 캐쉬나 텍스쳐 선인출 기법 등과 같은 연구들[1, 2]이 발표되었다. 텍스쳐 데이타의 경우에는 텍스쳐 선인출 기법을 통하여 메모리 접근 지연을 90%까지 줄임으로써 메모리 문제를 상당량 해결할 수 있었다[2]. 그러나 픽셀 캐쉬에 대한 효율성 및 구체적인 구조에 대한 연구는 발표된 경우가 없었다.

## 3. 제안하는 픽셀 캐쉬

이 장에서는 제안하는 픽셀 캐쉬의 구조 및 동작방법을 제시한다. 3.1절에서는 제안한 픽셀 캐쉬의 구조를 제시하고, 3.2절에서는 동작 방법에 대해서 자세히 설명한다.

### 3.1 제안하는 픽셀 캐쉬의 구조

제안하는 픽셀 캐쉬 구조는 그림 2에서 보여지는 것처럼 주 캐쉬와 보조 버퍼로 구성된다. 깊이 검사 시 성

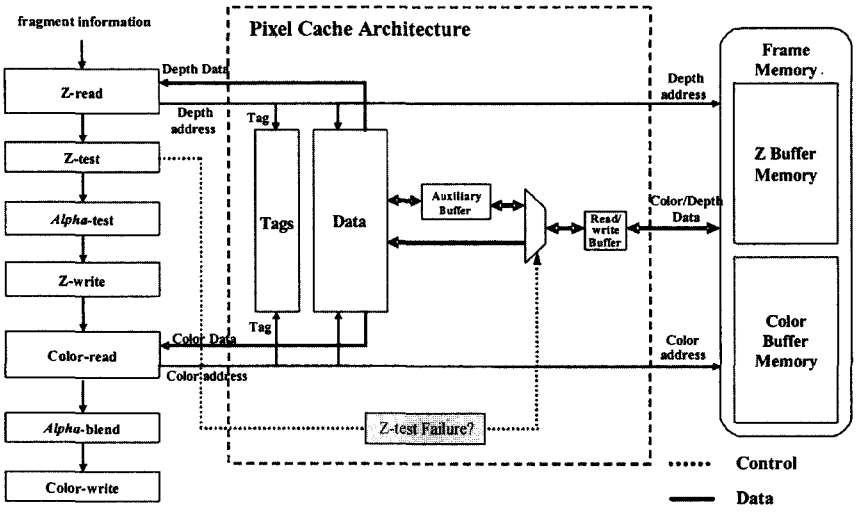


그림 2 제안하는 픽셀 캐쉬 구조

공한 깊이 데이타의 해당 캐쉬 블록은 주 캐쉬에 저장하고, 실패한 깊이 데이타의 해당 캐쉬 블록과 색상 데이타는 보조 버퍼에 저장한다.

주 캐쉬는 일반적인 캐쉬 구조와 같게 구성되며, [5]에서 제시된 일반적인 픽셀 캐쉬 구조의 접근 실패율을 바탕으로 블록 크기는 64바이트, 연관도(Associativity)는 1로 구성하였다. 일반적인 픽셀 캐쉬 구조에서 블록 크기가 64바이트인 경우에 캐쉬 접근 실패율과 대역폭을 고려할 때 가장 좋은 성능을 보였다[5]. 주 캐쉬의 연관도는 빠른 접근과 전력 소모를 줄이기 위하여 1로 구성하였다. 보조 버퍼에는 깊이 데이타뿐만 아니라 시간적 지역성이 거의 없고 높은 공간적 지역성만을 가지는 색상 데이타가 주로 저장되기 때문에 주 캐쉬보다 큰 블록 크기인 128바이트로 만들어지고, 연관도는 완전 연관도(Fully-Associativity)를 갖는다. 보조 버퍼의 연관도는 버퍼의 크기가 작게 구성되고, 시간적 지역성이 적은 색상 데이타가 주로 적재되기 때문에 직접 사상이나 집합 연관으로 구성되면 잦은 충돌 접근 실패가 일어날 수 있다. 이를 없애기 위해서 완전 연관도로 구성하였다. 보조 버퍼의 최적의 엔트리 수는 시뮬레이션 결과에 의해 4엔트리로 결정하였다. 엔트리 수에 대한 시뮬레이션 결과는 4.2절에서 제시한다.

보조 버퍼가 완전 연관으로 구성되지만, 적은 수의 엔트리로 구성되기 때문에 1 사이클에 접근이 가능하다. 또한, 색상 데이타에 대한 접근이 보조 버퍼에서만 이루어지기 때문에 기존의 캐쉬를 사용하는 경우에 비해서 전력 소모를 상당량 줄일 수 있다.

### 3.2 동작 방법

그림 3은 제안하는 픽셀 캐쉬에 대한 동작 방법을 보인다. 깊이 데이타 접근 시에 주 캐쉬와 보조 버퍼를 동시에 접근한다. 필요로 하는 깊이 데이타가 주 캐쉬에 있는 경우에는 일반적인 캐쉬 동작과 같다. 원하는 깊이 데이타가 주 캐쉬에 없고 보조 버퍼에 있는 경우에는 우선 원하는 데이타를 보조 버퍼로부터 가져와서 렌더링 파이프라인에서 처리한다. 그 후에 깊이 검사 결과에 따라서 접근한 보조 버퍼에 있는 데이타의 해당 캐쉬 블록을 주 캐쉬에 저장할 지를 결정한다. 깊이 검사가 성공한 경우에는 해당 캐쉬 블록을 주 캐쉬에 저장하고 주 캐쉬에서 대체되는 블록은 보조 버퍼에 저장한다. 실패한 경우에는 보조 버퍼에 저장된 상태로 유지한다.

주 캐쉬와 보조 버퍼에도 원하는 데이타가 없을 시 즉, 캐쉬 접근 실패 시에는 프레임 메모리로부터 원하는 블록을 가져와서 읽기 버퍼(Read Buffer)에 저장하고 깊이 검사 결과에 따라서 읽기 버퍼에 있는 블록을 주 캐쉬와 보조 버퍼 중에 어디에 저장할 지를 결정한다. 깊이 검사가 성공인 경우에는 주 캐쉬에 저장하고 실패인 경우에는 보조 버퍼에 저장한다. 읽기 버퍼는 일반적으로 캐쉬에서 사용되는 라인 필 버퍼(Line Fill Buffer)이다. 이와 같은 동작 방법으로 수행됨으로써 깊이 검사 결과가 실패한 데이타에 의한 캐쉬 오염을 줄이면서 재사용될 가능성이 많은 데이타를 주 캐쉬에 저장함으로써 픽셀 캐쉬의 효율성을 높일 수 있다.

색상 데이타 접근 시에는 보조 버퍼만을 접근한다. 색상 데이타의 경우에는 시간적지역성이 거의 없고, 주로 공간적 지역성만을 가지는 특징을 보이기 때문에 블록 크기가 큰 보조 버퍼에만 적재하도록 함으로서 주 캐쉬의 오염(Pollution)을 없애고자 하였다. 필요로 하는 데이타가 보조 버퍼에 있는 경우에는 렌더링 파이프라인

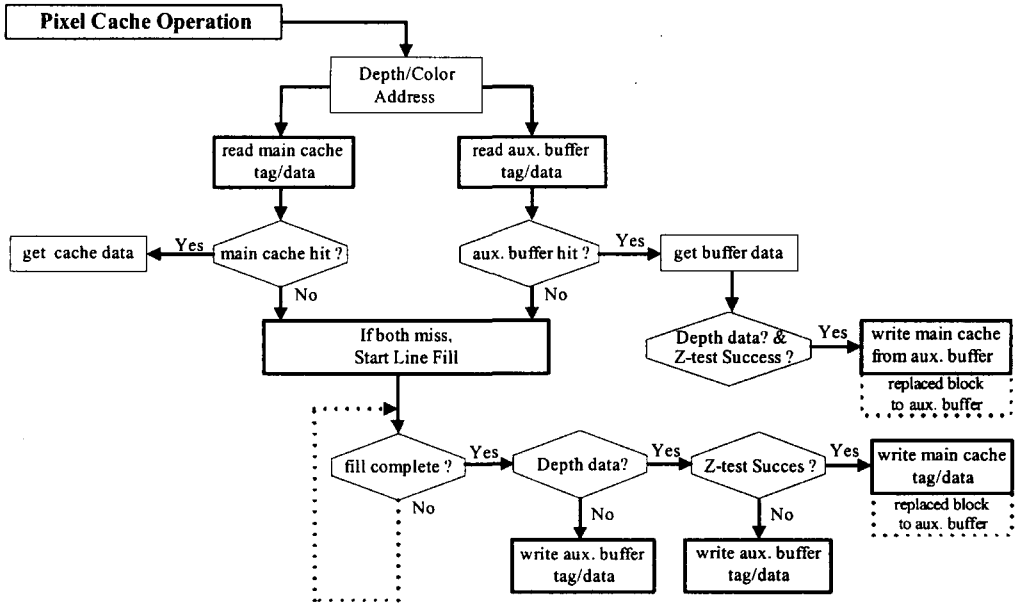


그림 3 제안하는 픽셀 캐쉬의 동작 방법

에서 처리하고, 보조 버퍼에 원하는 데이타가 없을 시, 즉 접근 실패 시에는 프레임 메모리로부터 데이타를 가져와서 보조 버퍼에만 저장한다.

## 4. 실험적인 시뮬레이션 결과

이 장에서는 제안된 캐쉬 구조에 대해서 성능 평가 및 보조 버퍼의 크기를 결정하기 위한 시뮬레이션 결과를 제시한다. 4.1절에서는 사용된 벤치마크 및 시뮬레이션 방법에 대해서 설명한다. 4.2절에서는 보조 버퍼의 크기 변화에 따른 시뮬레이션 결과를 제시한다. 4.3절에서는 캐쉬 접근 실패율을 제시하고, 4.4절에서는 제안한 캐쉬 시스템의 성능 평가 결과를 제시한다.

### 4.1 시뮬레이션 방법론

제안된 픽셀 캐쉬 구조를 평가하기 위해서 트래이스 구동 시뮬레이션(Trace-driven simulation)을 수행하였다. 트래이스(Trace)는 800 * 600 해상도에서 Quake3 demo I, Quake3 demo II와 lightscape 벤치마크들에 대해서 Mesa OpenGL compatible API를 수정하는 것에 의해서 생성하였다. 각 벤치마크마다 트래이스를 생성하기 위해 100 프레임이 사용되었다. 픽셀 캐쉬 시뮬레이션은 일반적인 캐쉬 구조와 제안한 캐쉬 구조에 대해서 DineroIII 캐쉬 시뮬레이터[9]를 수정하여 수행하였다. 일반적인 캐쉬 구조의 경우에는 DineroIII 캐쉬 시뮬레이터를 그대로 이용하였고, 제안한 구조의 경우에는 DineroIII 캐쉬 시뮬레이터에 버퍼 구조를 추가하고 제안한 동작 방법을 적용하여 수정하였다.

그림 4는 벤치마크들에 대해서 생성한 장면을 보여준다. Quake3는 전형적인 비디오 게임들 중의 하나이며 다른 관련된 연구에서 자주 사용될 뿐만 아니라 3차원 그래픽 가속기의 성능을 측정하기 위한 벤치마크 중 하

(a) Quake3 Demo I (b) Quake3 Demo II (c) Lightscape

그림 4 사용된 벤치마크

나이다[12]. Lightscape는 SPEC에서 제시하는 SPEC-viewperf 벤치마크들 중 하나이며 OpenGL을 기반으로 수행되는 3차원 렌더링 시스템의 성능을 측정하기 위한 산업체 표준 벤치마크이다[13].

### 4.2 보조 버퍼 크기

그림 5는 보조 버퍼의 블록 크기가 128바이트일 경우, 보조 버퍼의 크기를 256바이트부터 2048바이트까지 변화시킬 때의 캐쉬 접근 실패율을 보여준다. 즉, 보조 버퍼의 엔트리 수를 2~16으로 변화시킬 때의 시뮬레이션 결과이다.

그림 5에서 볼 수 있듯이 모든 벤치마크의 경우에 있어서 4개 엔트리 이상이 되면 거의 성능 변화가 없음을 알 수 있다. 따라서 제안한 픽셀 캐쉬 구조에서는 보조 버퍼 엔트리를 4개로 설정하여 제안한 픽셀 구조에 대한 성능 평가를 수행하였다.

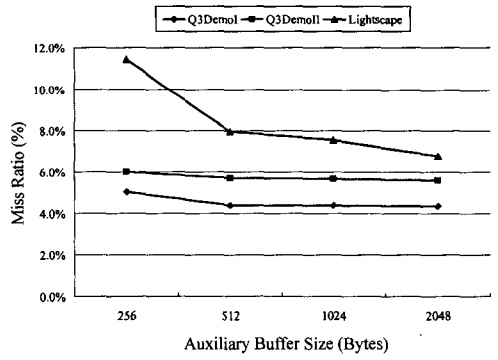


그림 5 버퍼 크기에 따른 캐쉬 접근 실패율

### 4.3 캐쉬 접근 실패율

그림 6은 캐쉬 접근 실패율을 보여준다. 성능 비교를 위한 일반적인 캐쉬 구조는 [7]에서 제시된 시뮬레이션 결과를 바탕으로 블록 크기는 64 와 128바이트, 연관도는 1인 경우에 대해서 시뮬레이션을 수행하였다. 픽셀 캐쉬의 경우에 연관도에 의해서는 접근 실패율의 변화가 거의 없고, 블록 크기 증가에 의한 영향을 많이 받는다. 픽셀 데이타가 시간적 지역성에 비해 공간적 지역성이 많기 때문이다[5,8].

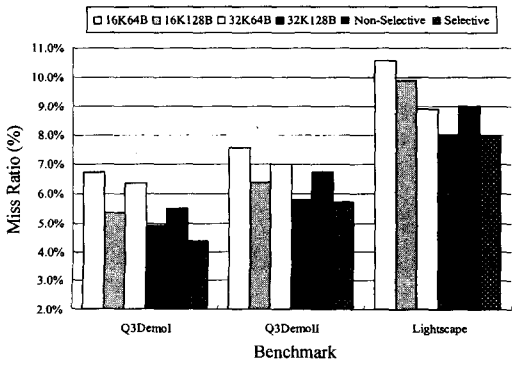


그림 6 캐쉬 접근 실패율

그림 6에서 "16K64B"는 16K바이트 캐쉬 크기, 64바이트 블록 크기, 직접 사상인 일반적인 캐쉬를 의미한다. "Non-Selective"는 본 논문에서 제안한 캐쉬와 구조는 같지만, 적재 방법에 있어서 깊이 검사 결과에 의해 깊이 데이타를 선택적으로 적재하지 않는다. 단지 깊이 데이타는 주 캐쉬에, 색상 데이타는 보조 버퍼에 저장하는 캐쉬를 의미한다. 이 구조와의 비교를 통해서 깊이 검사 결과에 의한 선택적 적재에 따른 성능 향상을 볼 수 있다. "Selective"는 제안한 구조를 의미한다. 제안한 구조에서 주 캐쉬의 크기는 16K바이트이고, 보조 버퍼의 크기는 512바이트이다.

그림 6에서 볼 수 있듯이 제안한 캐쉬 구조가 일반적인 캐쉬의 크기를 두 배 한 경우보다 모든 벤치마크에서 더 낮은 캐쉬 접근 실패율을 보인다. 보조 버퍼를 이용하여 재사용될 가능성이 많은 데이타만을 주 캐쉬에 저장하고, 그렇지 않은 데이타는 보조버퍼에 저장함으로써 충돌 접근 실패 및 캐쉬 오염을 상당량 줄일 수 있다. 이로 인해 두 배 크기의 일반적인 캐쉬보다 더 낮은 캐쉬 접근 실패율을 보일 수 있었다. Q3DemoII와 Lightscape의 경우에는 "32K128B"와 유사한 캐쉬 접근 실패율을 보이지만, 32K128B인 경우에는 접근 실패 시에 128바이트를 항상 가져와야 하기 때문에 제안한 캐쉬 구조에 비해 더 많은 대역폭을 요구한다. 이와 달리 제안한 캐쉬 구조에서는 깊이 데이타의 경우에는 64바이트이고, 색상 데이타의 경우에만 128바이트를 요구하기 때문에 대역폭을 상당량 줄임으로써 더 나은 성능을 보인다.

### 4.4 평균 메모리 접근 사이클에 의한 성능 평가

제안하는 구조의 성능 평가를 위하여 캐쉬 접근 실패율 및 대역폭을 고려한 성능 척도인 평균 메모리 접근 사이클(*Average Memory Access Cycle : AMAC*)을 사용하였다. *AMAC*는 캐쉬 접근 성공 시간(Cache Hit Time)과 평균적인 캐쉬 접근 실패 처리 시간(Cache Miss Penalty)의 합으로 계산될 수 있다. 일반적인 캐쉬 구조와 제안한 캐쉬 구조에 대한 *AMAC*인 $AMAC_c$와 $AMAC_p$는 다음과 같다.

$$AMAC_c = HT + MR_c \times (LA + \frac{BL_1}{TR})$$

$$AMAC_p = HT + MR_{pd} \times (LA + \frac{BL_1}{TR}) + MR_{pc} \times (LA + \frac{BL_2}{TR})$$

$HT$는 캐쉬 접근 성공 시간, $MR_c$은 일반적인 캐쉬의

접근 실패율, $MR_{pd}$는 깊이 데이타의 접근 실패율, $MR_{pc}$는 색상 데이타의 접근 실패율, $BL_1$은 주 캐쉬의 블록 크기, $BL_2$는 보조 버퍼의 블록 크기, $LA$는 메모리 접근 지연, $TR$은 전송률을 의미한다. 본 시뮬레이션에서는 현재 기술을 고려하여 주 캐쉬 및 보조 버퍼의 접근 성공 시간을 1사이클, 메모리 접근 지연을 10사이클, 전송률을 16바이트/사이클로 가정하였다. 이는 [5]에서의 가정과 동일하다.

그림 7은 *AMAC*를 보여준다. 그림 7에서 볼 수 있듯이 *AMAC*는 그림 6의 캐쉬 접근 실패율과는 다른 경향을 보인다. 일반적인 캐쉬의 경우들에 있어서는 블록 크기가 128바이트인 경우보다 64바이트가 더 나은 성능을 보임을 알 수 있다. 이는 블록 크기가 64바이트인 경우에 접근 실패율에 의한 성능 감소보다 대역폭에 의한 성능 향상이 더 크기 때문이다.

그림 7의 결과에 의하면, 16K바이트 크기인 제안하는 픽셀 캐쉬인 "Selective"의 $AMAC_p$가 모든 벤치마크에서 가장 나은 성능을 보인다. 제안하는 구조와 같은 크기를 가지는 일반적인 캐쉬인 "16K64B"와 "16K128B"의 $AMAC_c$보다 평균적으로 각각 20%, 30.7%정도 향상된 성능을 보이고, 제안하는 구조보다 두 배 큰 "32K64B"와 "32K128B"의 $AMAC_c$에 비해서는 평균적으로 각각 10.7%, 16.9%정도 향상된 성능을 보인다. 제안한 픽셀 캐쉬와 구조는 같지만 깊이 데이타에 대해서 선택적인 적재를 수행하지 않는 "Non-Selective"의 $AMAC_p$에 비해서는 평균적으로 12.3% 정도 향상된 성능을 보인다.

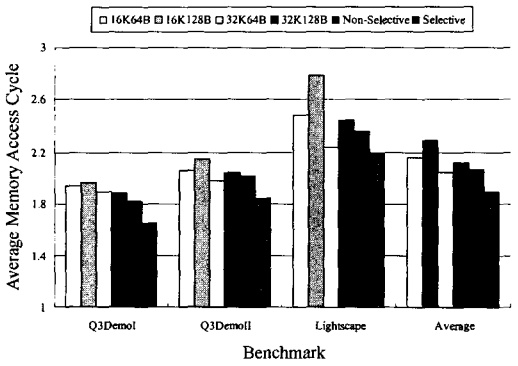


그림 7 평균 메모리 접근 사이클

## 5. 결 론

본 논문에서 픽셀 데이타의 특징에 맞게 캐쉬 구조를 설계함으로써 캐쉬 오염을 줄이고 이로써 픽셀 캐쉬의 성능을 향상시키고자 하였다. 제안한 픽셀 캐쉬 구조는 보조 버퍼를 사용하여 깊이 데이타와 색상 데이타에 대해 다르게 적재함과 동시에 렌더링 파이프라인에서 수행되는 깊이 검사 결과에 의해 깊이 데이타를 선택적으로 저장하도록 설계하였다. 보조 버퍼는 적은 엔트리를 가지는 완전 연관 캐쉬로 구성된다. 색상 데이타의 경우에는 보조 버퍼에 대한 접근만을 수행한다.

시뮬레이션 결과에 의하면, 16K바이트의 제안한 캐쉬 구조가 캐쉬 접근 실패율과 평균 메모리 접근 사이클의 성능 척도들에서 32K바이트의 일반적인 캐쉬 구조보다 더 나은 성능을 보였다.

## 참 고 문 헌

[1] Z.S. Hakura and A. Gupta, "The design and analysis of a cache architecture for texture mapping," *Proceedings of the 24th International Symposium on Computer Architecture*, pp. 108-120, June 1997.

[2] H. Igehy, M. Eldridge, and K. Proudfoot, "Prefetching in a texture cache architecture," *Proceedings of 1998 SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pp. 133-142, Aug. 1998.

[3] nVidia, "Technical briefs: NVIDIA lightspeed memory architecture II," nVidia Corporation, available at http://www.nvidia.com/docs/lo/1452/SUPP/LMAII.2.02.pdf

[4] Tulika Mitra and Tzi-cker Chiueh, "Dynamic 3d graphics workload characterization and the architectural implications," *32nd Annual ACM/IEEE International Symposium on Microarchitecture*, pp. 62-71, Nov. 1999.

[5] W. C. Park, K. W. Lee, I. S. Kim, T. D. Han, and S. B. Yang, "An effective pixel rasterization pipeline architecture for 3d rendering processors," *IEEE Transactions on Computers*, accepted to appear, available at http://msl.yonsei.ac.kr/user-web/chan/thesis/TC114724.pdf.

[6] G. Lawton, "The wild world of 3D graphics chips," *IEEE Computer*, vol. 33, no. 9, pp.12-16, Sept. 2000.

[7] M. Woo, J. Neider, T. Davis, and D. Shreiner, *OpenGL programming guide*, Addison-Wesley, Third edition, 1999.

[8] M. Deering, S.A.Schlapp, and M.G.Lavelle, "Fbram : A new form of memory optimized for 3d graphics," In *Proceeddings of SIGGRAPH 1994*, pp.167-174, July 1994.

[9] M. D. Hill, J. R. Larus, A. R. Lebeck, M. Talluri, and D. A. Wood, "Wisconsin architectural research tool set," *ACM SIGARCH Computer Architecture News*, vol. 21, pp. 8-10, Sep. 1993.

[10] ATI, "Stunning 2d & 3d performance," ATI Technologies INC., available at http://www.ati.

com/technology/hardware/rage128pro/index.html
[11] ATI, "RADEON™ 8500LE : specification," ATI Technologies INC., available at http://www.ati.com/products/pc/radeon8500le/specs.html
[12] http://www.idsoftware.com/games/quake/quake3-arena/
[13] http://www.spec.org/gpc/opc.static/opcview70.html

이 길 환

1996년 연세대학교 이과대학 컴퓨터과학과 졸업(학사). 1998년 연세대학교 대학원 컴퓨터과학과(공학석사). 1998년~현재 연세대학교 대학원 컴퓨터과학과 박사과정. 관심분야는 3차원 그래픽 가속기, 메모리 시스템, ASIC 설계, 고성능 컴퓨터 구조, 내장형 시스템

박 우 찬

정보과학회논문지 : 시스템 및 이론
제 30 권 제 4 호 참조

김 일 산

1996년 연세대학교 문리대학 통계학과 졸업(학사). 2002년 연세대학교 대학원 컴퓨터과학과(공학석사). 2002년~현재 연세대학교 대학원 컴퓨터과학과 박사과정. 관심분야는 3차원 그래픽 가속기, 메모리 시스템

한 탁 돈

정보과학회논문지 : 시스템 및 이론
제 30 권 제 4 호 참조