# 3차원 그래픽 가속기의 지연 감소를 위한 개선된 래스터라이져 및 캐쉬 메모리 구조 제안 및 실험

박진홍*° 김일산* 박우찬** 한탁돈*
*연세대학교 컴퓨터과학과
**세종대학교 인터넷공학과
jhpark@kurene.yonsei.ac.kr

# The Advanced Rasterizer and Cache Memory Architecture For Latency Reduction Of 3D GPU

Jin-Hong Park*° Il-San Kim* Woo-Chan Park** Tack-Don Han*
*Dept. of Computer Science, Yonsei University
**Dept. of Internet Engineering, Sejong University

요 약

현재 3차원 그래픽 가속기에서 성능 향상에 대한 문제점으로 대두되고 있는 것은 실제 화면에 그려지는 정보가 저장되는 프레임버퍼에 대한 접근 지연이다. 따라서 본 논문은 기존 픽셀 캐쉬가 포함된 래스터라이져 구조에서 캐쉬 읽기 접근 실패 시 발생하는 패널티와 이에 따른 프레임버퍼에 대한 지연이 발생하는 문제점을 개선하고자, 기존 래스터라이져를 래스터라이져와 합성기로 구분하고 그 사이에 캐쉬 읽기 접근 실패 시 프레임 버퍼에서 정보를 읽어오지 않는 깊이 캐쉬와 색상 캐쉬가 쌍을 이룬 픽셀 캐쉬 메모리 시스템으로 구성된 개선된 3차원 그래픽 가속기 구조을 제안하고 실험을 수행하였다. 실험 결과 제안하는 3차원 그래픽 가속기 구조가 기존 구조에 비해 캐쉬 접근 실패율이 약 23% 감소하였으며, 평균 메모리 접근 사이클이 10%-13% 감소하였으며 이는 상당수의 프레임버퍼에 대한 접근 지연을 감소시킨 것이다. 합성기와 메모리 간의 대역폭은 약 10% 증가하지만 파이프라인의 작업에는 영향을 미치지는 않는다.

## 1. 서 론

3차원 컴퓨터 그래픽은 3차원 좌표에 표현되는 사물들을 출력 장치의 좌표계로 변환하여 볼 수 있게 한다. 일반적으로, TV와 모니터 등 대부분의 출력장치는 픽셀들로 구성된 2차원 화면이기 때문에 3차원 좌표로 표현된 사물들은 2차원으로 투영하여 출력되어진다.

3차원 컴퓨터 그래픽의 일반적인 처리 과정은 입력되는 정보 단위를 기준으로 하여 기하 처리 단계(Geometry Processing Stage)와 래스터라이제이션 단계(Rasterization Stage)로 나뉠 수 있다. 기하 처리 단계는 입력된 물체의 정점들에 대한 색상 계산 및 위치를 변환시키는 연산을 수행한다. 래스터라이제이션 단계는 기하 연산이 수행된 정점들의 정보를 가공하여 2차원 화면에 출력하기 위한 픽셀 정보로 변환한다. 픽셀 변환 작업은 크게 실제와 유사한 표면 질감을 표현하기 위한 텍스처 매핑 단계, 가시성 검사를 위한 깊이 검사 단계, 그리고 픽셀 정보를 화면에 출력하기 위한 픽셀 쓰기 단계로 구성된다[1][2]. 이 때, 각각의 처리단계들이 요구하는 메모리 대역폭 및 메모리 접근에 따른 파이프라인 지연이 성능저하의 큰 요인이 된다.

이러한 문제점들을 극복하기 위하여 다양한 메모리 구조가 연구되었는데, 텍스처 작업을 위한 텍스처 캐쉬[3] 및 텍스처 선 입력 구조[4], 픽셀 정보를 처리하기 위한 픽셀 캐쉬 구조[5]가 있다. 텍스처 매핑 작업의 경우, 텍스처 캐쉬 구조, 텍스처 캐쉬와 선 인출 구조를 결합한 텍스처 선 입력 구조 등을 사용하여 메모리 접근에 따른 대역폭 및 접근 지연 문제를 상당부분 감소시킬 수 있다. 하지만 픽셀 처리 단계는 선 입력 구조와 같은 기법을 적용하기 어렵고, 픽셀 캐쉬를 사용하여도 읽기 및 쓰기 작업에 따른 메모리 접근 지연이 발생한다.

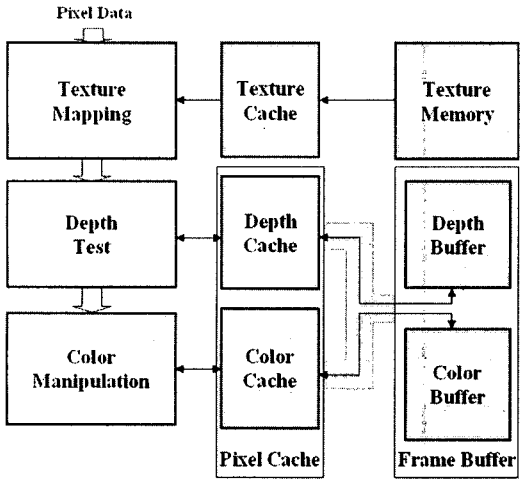
따라서 본 논문은 이 문제점을 해결하기 위하여 픽셀 캐쉬 읽기 작업 단계에서 발생하는 패널티를 제거하기 위한 개선된 구조를 제시하였다. 기존 래스터라이저를 변형하여 래스터라이져와 합성기(Compositor)로 구분하여 그 사이에 깊이 캐쉬와 색상 캐쉬가 쌍을 이루어 동작하는 픽셀 캐쉬를 둔 구조이다. 픽셀 캐쉬 읽기 접근 실패가 발생했을 때는 프레임 버퍼에서 정보를 읽어오는 단계를 두지 않고 캐쉬 쓰기 접근 실패가 발생했을 때 교체될 블록을 합성기로 전송하여 프레임버퍼 내의 정보와 비교하여 프레임 버퍼의 깊이 정보와 색상 정보를 업데이트하는 방식이다. 실험 결과, 평균 메모리 접근 사이클이 10%-13% 감소하였고 합성기와 메모리 간의 대역폭은 약 10% 증가하지만 파이프라인의 작업에는 영향을 미치지 않는다.

본 논문은 2장에서 관련 연구, 3장에서 제안하는 3차원 그래픽 가속기 시스템의 구조와 동작에 대한 설명, 4장에서 실험 환경 및 기존 캐쉬 메모리 시스템과 제안하는 3차원 그래픽 가속기 시스템에 대한 실험결과를 비교, 분석하였다. 마지막으로 5장에서는 결론을 서술하였다.

## 2. 관련 연구

3차원 그래픽 가속기는 화면에 원하는 색상을 나타내기 위해 각 색상 값들을 프레임 버퍼라고 하는 외부 메모리에 저장하게 된다. 또한, 픽셀 정보들이 어떤 순서로 화면에 배치되느냐에 따라서 프레임 버퍼에 저장되어 있는 픽셀 정보를 불러와서 처리하고 다시 저장하게 된다. 또한 현실감 있는 화면을 구성하기 위한 텍스처 정보 전송을 위한 텍스처 메모리가 필요하다.

복잡한 텍스처와 모델들로 고해상도의 화면을 구성한다면 텍스처 메모리와 프레임 버퍼 메모리의 전송량은 기하급수적으로 증가하게 되는데 메모리 대역폭은 한계가 있기 때문에 요구하는 정보를 빠르게 전송하기가 어렵다. 이러한 메모리 트래픽은 3차원 그래픽 가속기의 전체적인 성능향상에 심각한 제약 조건으로 대두되고 있다. 따라서 이러한 대역폭 문제를 해결하기 위해서 텍스처 캐쉬나 선 인출 기법을 사용하여 텍스처 전송효율을 높이는 방법과 프레임 버퍼의 대역폭 문제와 빠른 전송 위해 픽셀 캐쉬를 사용하고 있다. [그림 1]은 래스터라이져와 메모리 사이에 캐쉬가 삽입된 구조를 보여준다.




[그림 1] 래스터라이져와 메모리 사이에 캐쉬가 삽입된 기존 구조

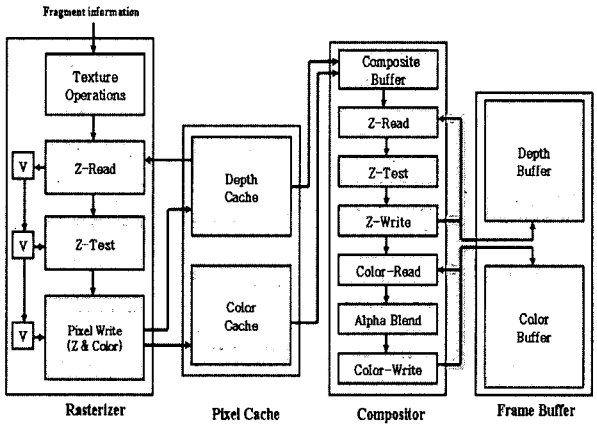
## 3. 제안하는 구조

제안하는 구조는 [그림 2]와 같다. 스펜 프로세싱 단계를 거친 픽셀 정보들을 처리하는 래스터라이져, 래스라이져를 통과한 깊이 값 정보와 색상 값 정보를 저장하는 깊이 캐쉬와 색상 캐쉬가 쌍을 이루어 동작하는 픽셀 캐쉬, 픽셀 캐쉬에서 교체된 정보 블록과 프레임 버퍼의 정보를 처리하여 다시 프레임 버퍼에 저장하는 합성기로 구성되어 있다.

래스터라이져에 픽셀 정보가 들어오면 먼저, 텍스처 매핑 등의 과정을 수행하고 캐쉬에 저장되어 있는 해당 좌표에 해당하는 깊이 정보를 읽어 깊이 검사 후 통과되면 픽셀 쓰기 단계에서 깊이 값 정보와 색상 값 정보를 동시에 각각의 깊이 캐쉬와 색상 캐쉬에 저장한다. 캐쉬 읽기 실패 시에는 프레임 버퍼에 정보를 요구하지 않는다. 따라서 캐쉬 읽기 패널티로 처리되지 않는다. 이때, 깊이 검사는 통과되는 것으로 처리하는데, 이를 위해 읽기 실패가 발생했다는 표시를 위한 유효 레지스터(V)를 두어 각 단계에서 이를 참조하여 처리한다.

픽셀 캐쉬는 깊이 캐쉬와 색상 캐쉬로 구성되어 있고, 같은 스크린 좌표를 가지는 깊이 값 정보와 색상 값 정보가 동시에 이동이 가능하므로, 깊이 캐쉬의 태그 비교만으로 색상 캐쉬와 쌍으로 작동하므로 비교 시간이 단축된다. 래스터라이져에서 픽셀 캐쉬에 쓰기 미스가 발생했을 때 교체될 블록이 합성기의 합성 버퍼(Composite Buffer)로 보내진다. 깊이 캐쉬는 읽고 쓰는 작업을 위한 2 포트의 SRAM, 색상 캐쉬는 쓰기 작업만 필요하므로 1 포트의 SRAM으로 구성한다.

합성기는 합성 버퍼에 저장되어 있는 캐쉬에서 교체된 블록 내의 픽셀들을 순서대로 같은 어드레스를 가진 프레임 버퍼의 깊이 정보를 읽어와 깊이 검사를 통과되면 프레임 버퍼내의 깊이 정보를 업데이트 하고, 프레임 버퍼 내의 색상 정보를 읽어 와서 색상을 혼합하는 단계인 알파 블랜딩(Alpha Blending)을 한 후, 생성된 색상 값을 프레임 버퍼에 쓴다.




[그림 2] 제안하는 구조

## 4. 실험 환경 및 결과

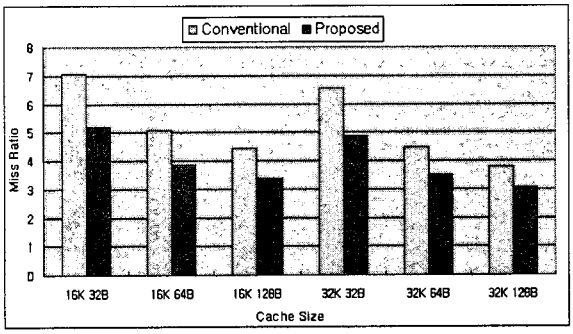
### 4.1 실험 환경

3차원 응용프로그램의 메모리 접근을 분석하기 위해서 본 논문에서는 RedHat Linux V8.0 운영체제와 OpenGL을 지원하는 Mesa 3D V3.4[6]를 이용하여 프레인 버퍼 내의 깊이 버퍼와 색상 버퍼 참조에 대한 트레이스를 생성하였다. 벤치마크로는 대표적인 3D 응용프로그램 중의 하나인 Id software의 Quake 3[7]를 선택하였고, 캐쉬 시뮬레이터로는 Dinero IV[8]를 사용하였다. 100 프레임의 벤치마크로부터 16KB, 32KB의 캐쉬 크기와 각각 32, 64, 128개의 블록 크기를 가지는 직접 매핑(Direct Mapped) 캐

쉬로 시뮬레이션 하였다. 또한 텍스처 캐쉬의 읽기는 100% 성공한다고 가정하였다.

### 4.2 실험 결과

Quake를 640x480 해상도에서에서 실행했을 때의 캐쉬 시뮬레이션 결과, [그림 3]과 같이 제안하는 구조가 기존 구조에 비해 캐쉬 접근 실패율이 약 23%감소하였다.




[그림3] Quake 3에서의 기존 구조와 제안하는 구조의 픽셀 캐쉬 접근 실패율 비교

제안하는 구조의 성능을 측정하기 위해서는 평균적인 메모리 접근 사이클(Average Memory Access Cycle, AMAC)[9]을 계산하여 기존 구조를 비교해 볼 필요가 있다. 평균적인 메모리 접근 사이클의 공식은

$$AMAC = T_c + m \times t_{main}$$

이다. $T_C$는 캐쉬의 접근 사이클이고, m은 캐쉬의 접근 실패율(Miss Ratio), 그리고 $t_{main}$은 캐쉬의 접근 실패 비용(Cache Miss Penalty)이다.

$t_{main}$은 다음과 같은 식으로 구할 수 있다.

$$t_{main} = L + B/TR,$$

L : 메모리 지연시간 (Memory Access Latency)
B : 캐쉬의 블록크기 (Cache Block Size)
TR : 메모리 전송률 (Memory Transfer Rate)

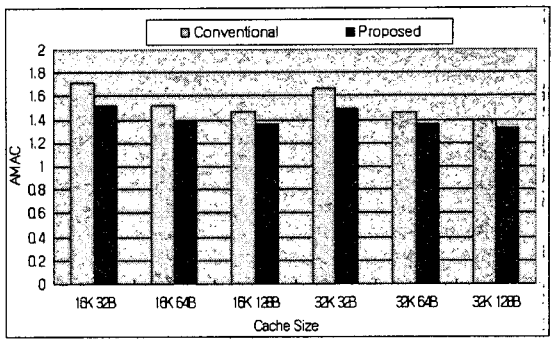
캐쉬의 접근 사이클은 1 cycle, 메모리 지연 시간은 10 cycle, 그리고 메모리 전송률은 256bit으로 가정하였다.

[그림 4]는 Quake를 640x480 해상도에서에서 실행했을 때 발생하는 기존 구조와 제안하는 구조의 평균 메모리 접근 사이클 비교 결과이다. 제안하는 구조가 기존 구조에 비해 평균 메모리 접근 사이클이 평균 13% 감소하였다.

합성기와 메모리 간의 대역폭은 약 10% 증가하는데, 이는 파이프라인의 작업에는 영향을 미치지는 않는다.

## 5. 결론

본 논문에서는 기존 래스터라이져 구조에서 캐쉬 읽기 접근 실패 시 발생하는 패널티와 이에 따른 프레임 버퍼에 대한 지연이 발생하는 문제점을 개선하기 위해 개선된 래스터라이져 및 캐쉬 메모리 시스템 구조를 제안하였다.




[그림 4] Quake 3에서의 기존 구조와 제안하는 구조의 평균 메모리 접근 사이클 비교

실험 결과 기존 구조에 비해 캐쉬 접근 실패율이 약 23% 감소하였으며, 이에 따른 평균 메모리 접근 사이클은 약 13% 감소하였다.

## 참고 문헌

[1] Tomas Akenine-Möller, Eric Haines, "Real-Time Rendering," A K PETERS, 2002

[2] Foley, van Dam, Feiner, Hughes, "Computer Graphics Principles and Practice," Addison & Wesley, 1996

[3] 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.

[4] Homan Igehy, Matthew Eldridge, Kekoa Proudfoot, "Prefetching in a Texture Cache Architecture," Proceedings of Eurographics / SIGGRAPH Workshop on Graphics Hardware 98.

[5] Ikedo, T., Ma, J.. "Pixel cache architecture with FIFO implemented within an ASIC", ASIC Conference and Exhibit, 1996. Proceedings., Ninth Annual IEEE International 23-27 pp 19-22 Sept. 1996

[6] Mesa Library, http://www.mesa3d.org/

[7] Quake 3, http://www.idsoftware.com/

[8] 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.

[9] B. Anderson, R.MacAulay, A Stewart, and T. Whitted, "Accommodating Memory Latency in a Low-Cost Rasterizer," 12th Eurographics Workshop on Graphics Hardware, pp. 97-101, Aug. 1997.