3차원 그래픽 가속기의 효율적인 파이프라인 설계

우 현재, 정 종철, 이 문기
연세대학교 전자공학과
전화 : 02-2123-4731 / 핸드폰 : 017-708-3339

An efficient pipelined architecture for 3D graphics accelerator

Hyun-Jae Woo , Jong-chul Jeong, Moon-Key Lee
Dept. of Electrical and Electronic, Yonsei University
E-mail : woohj@spark,yonsei.ac.kr

Abstract

This paper is proposed about an efficient pipelined architecture for 3D graphics accelerator to reduce Cache miss ratio. Because cache miss takes a considerable time, about 20~30 cycle, we reduce cache miss ratio to use pre-fetch. As a result of simulation, we figure out that the miss ratio of cache depends on the size of tile, cache memory and auxiliary cache memory. We can save 6.6% cache miss ratio maximumly .

I. 서론

컴퓨터의 성능의 향상과 가격의 인하로 이제는 컴퓨터가 더 이상 업무만을 위한 것이 아니라 여가생활을 위한 도구로도 많이 사용되어지고 있다. 그 대표적인 예가 컴퓨터 게임이고 그 외에도 박물관 및 관광지등을 컴퓨터를 통하여 체험할 수 있게 해주며 조만간 가상현실 시스템의 상용화가 이루어 질것이다.

이런 것들을 실현시키기 위해서는 효율적인 3차원영상의 처리가 요구된다. 지금까지는 3차원영상처리를 CPU가 단독으로 처리를 하였으나 최근에는 고성능을 위해 대용량의 데이터 연산을 요구하며 그 연산 과정 또한 점점 복잡해지고 있어 CPU가 단독으로 처리하기엔 그 부담이 너무 크다. 그래서 3차원 영상만을 전담하여 그래픽을 가속하는 Co-Processor가 요구되며, 이러한 연산들을 수행하기 위해서 파이프라인화가 사용된다.

본 논문은 기존의 파이프라인 중 메모리의 access시간이 긴 텍스쳐 매핑 단의 해결 및 미리 메모리를 fetch할 수 있게 설계한 새로운 파이프라인을 구현하였다.

제2장에서는 3차원 그래픽 가속기 파이프라인의 이해를 위한 장으로 각 단계에서 이루어지는 일과 절차에 대해 기술하였고, 3장에서는 제안하는 Pre-fetch식 3차원 그래픽 가속기 파이프라인 그리고 C 모델링 결과 및 성능비교를 하며 마지막장은 본 논문의 결과로 이루어진다.

II. 3차원 그래픽 가속기 파이프라인

3차원 그래픽 가속기는 크게 기하학 연산 처리기와 렌더링 처리기로 이루어지며, 기하학 처리기는 3차원 영상 데이터의 크기 및 위치 변화와 그에 따른 빛의 변화를 처리하는 부분으로 본 논문에서는 CPU가 전담한다. 다음으로 렌더링 처리기는 삼각형 내부 픽셀의 색상을 결정하는 단계로 색상을 계산하고 텍스쳐를 입히는 처리를 한다. 렌더링처리기는 기하학처리기로부터 입력된 삼각형 데이터의 기울기 및 색상 정보의 증가분을 구하는 삼각형 셋업 과정, 변에 대한 정보를 구하는 변 처리기, 픽셀에 대한 색상정보를 구하는 픽셀 처리기, 텍스쳐를 입히는 텍스쳐 매핑으로 구성된다[1][2][4].

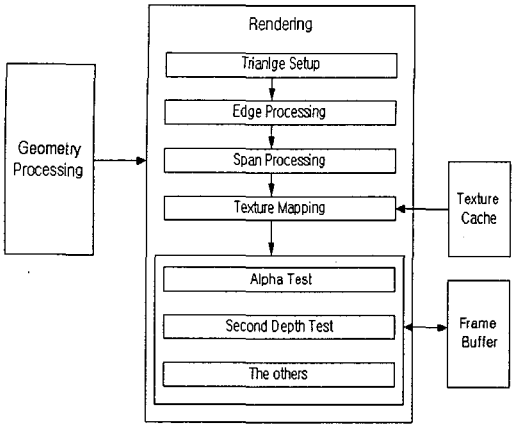

그림 3. 3차원 그래픽 가속기 파이프라인

우선 삼각형 셋업은 각 삼각형 당 한번만 수행되며 삼각형의 각 변의 기울기와 컬러 값, 깊이 값 및 텍스쳐 맵핑 관련 정보들의 증가분을 구하는 단계로 변처리 와 스팬 처리에 사용되는 증가분들의 계산을 수행한다.

다음 단계인 변 처리는 삼각형에 속하는 주사선 당 한번씩 일어나며 주사선과 삼각형의 양변과 교차하는 지점의 좌표 값을 구하고, 매개 변수의 값(R, G, B, u, v, d, w)을 y축에 대한 증분을 이용하여 구한다.

스팬 처리는 각 주사선에 있는 픽셀들의 색상 정보를 구하는 과정으로 스팬 내의 모든 픽셀 위치의 값을 구하는 것으로 이전 픽셀 위치의 R, G, B, u, v, d, w 값에 삼각형 셋업 단에서 구한 x축에 대한 증분 값들을 이용하여 보간 하는 과정으로 얻는다.

텍스쳐 맵핑은 텍스쳐 이미지를 화면내의 객체의 표면에 투영하는 것으로써 텍스쳐 데이터를 각 픽셀에 매핑하는 것을 말한다. 시점과의 거리에 따라 객체의 크기가 변하기 때문에 거리에 따른 텍스쳐의 크기도 변하여야 한다. 즉 텍스쳐링이란 이미지와 function, 또는 다른 데이터 셋을 이용하여 각 위치의 표면을 얻고 그것의 외형을 수정하는 것을 말한다. 즉 표면의 개체를 효과적으로 모델링하는 기술을 말 한다.

그림2의 텍스쳐 파이프라인을 보면 표면의 위치(x, y, z)를 얻은 후(1'stage) parameter space(u, v)로 projecting (2'stage)해 준다(3차원 좌표에서 2차원 자표(u, v)로).

실시간 처리에서는 프로젝터 평션은 모델링 단에서 수행하고 그 프로젝션의 결과를 정점(vertices)에 저장한다.

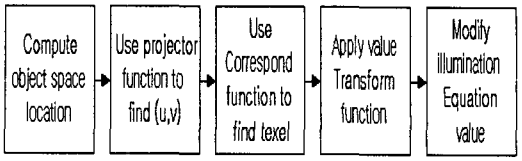

그림 4 Generalized texture pipeline

다음 단계로 텍셀을 찾기 위해 일치화를 수행한다 (x,y좌표 값에서 u, v 값으로 바꾸어 준다). 텍셀이란 표면의 질감을 느낄 수 있도록 3차원 개체를 2차원 질감 묘사 map으로 변환하였을 때의 최소 그래픽 구성 요소를 말한다. 즉 projection에서 얻은 u, v 값에 해상도를 곱해주는데 u, v값은 0~1사이의 값이다. 그러기 위해 일치화기(corresponder)는 wrap (repeat or tile), mirror, clamp, border등의 함수를 가진다. 텍스쳐의 값이 정정 되어진 후 그 값들은 RGB 값으로 변환 된다.

마지막으로 modify illumination equation value 단에서는 일반적으로 대체 와 수정를 수행하는데 대체란 텍스쳐 자신의 것을 제외한 모든 것(lightings)을 제거하는 방법을 말한다. 또 수정 방법은 표면의 색에 텍스쳐의 색을 곱해주는 것을 말하며 만약 원래의 표면이 흰색이고 그 뒤 빛 이 비추어 진다면 텍스쳐의 색을 수정하여 명암처리된 텍스쳐드 표면을 만드는 것을 말한다.

이 외에도 사실감을 높여 주기 위한 기법들로 Anti-aliasing, α-blending등의 과정이 있는데 Anti-aliasing이란 객체의 경계면에 생기는 계단현상을 없애는 과정으로서 화면의 해상도와 비교해서 객체의 픽셀의 수가 많지 않으면 경계면에 계단모양이 생길 수 있다. 즉 객체의 경계면 주의의 픽셀들을 양쪽 중간 색상을 가지도록 처리하여 계단 현상을 없애는 방법을 말하며 이런 방법으로 Super-sampling, accumulation buffer 방법 등이 있다.

마지막으로 α-blending 이란 원래의 칼라에 투명도 값을 석어 주는 것을 말하며 아래의 수식과 같다[3][5].

$$Co=\alpha Cs+(1-\alpha)Cd$$

(Cs: 투명한 물체가 갖는 색, Cd: 혼합전의 색
α :주어진 픽셀의 불투명 정도의 값.)

III. 제안하는 효율적인 3차원 그래픽 가속기 파이프라인 고찰

3.1제안하는 Pre-fetch식 3차원 그래픽 가속기 파이프라인.

앞에서 설명한 파이프라인 단계들 중 텍스쳐 메핑 단에서 파이프라인의 정체현상이 빈번히 일어난다. 그것은 대부분 텍스쳐 케쉬의 시 미스가 일어나는 것에 기인한다.

이러한 미스는 케쉬 메모리에 텍스쳐 메핑에서 원하는 텍스쳐가 없을 경우 일어나며 이런 상황에선 주 메모리를 읽어야 하는데 그 수행 시간이 20~30 cycle 정도가 걸리게 되어 정체 현상을 일으킨다. 보통 케쉬의 읽는 시간이 1cycle이라고 볼 때 이 시간은 매우긴 시간으로 본 논문에서는 이러한 미스가 날 확률을 줄일 수 있는 파이프라인을 제안하였다.

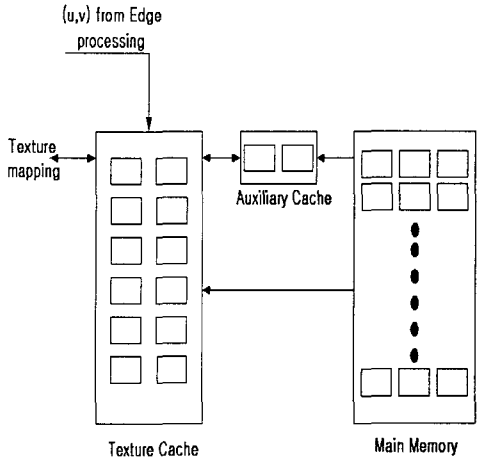

그림 5 제안하는 Pre-Fetch System

그림3은 본 논문에서 제안하는 새로운 파이프라인 시스템 이다. 그림 1의 텍스쳐 케쉬 뒤에 보조 케쉬를 추가하여 변 처리의 결과 중 u, v값을 pre-fetch하여 미리 텍스쳐 케쉬에 그 값이 있는지 확인을 하고 없을 시에 보조 케쉬에 에 있는 그 값을 저장해 놓는다.

또다른 중요한 점은 그림 4와 같이 동일 수행시간 동안 변 처리의 결과가 스팬 처리의 각 픽셀들의 처리보다 많다는 점이다. 보통 변 처리기 와 스팬 처리기 사이에 버퍼를 두어 버퍼의 사이즈만큼 변의 결과가 저장되고 버퍼가 차게 되면 정체 현상이 일어난다. 변 처리 과정과 스팬 처리 과정은 바로 이전 픽셀의 값에 증분 값을 더해 주어 현재의 값을 얻어주는 연산을 반복수행 한다. 변의 결과 와 스팬의 픽셀 결과의 throughput이 1이 되게 하기위하여 본 논문에서는 [2]에서와 같이 7개(매개 변수의 수)의 반복기를 두었다고 가정하였다. 반복기의 구조는 두개의 입력을 가진 하나의 가산기의 두개의 곱셈기, 그리고 두개의 레지스터로 구성되며 파이프라이닝 한 구조로 하나당 3 사이클의 지연이 걸린다[2][4].

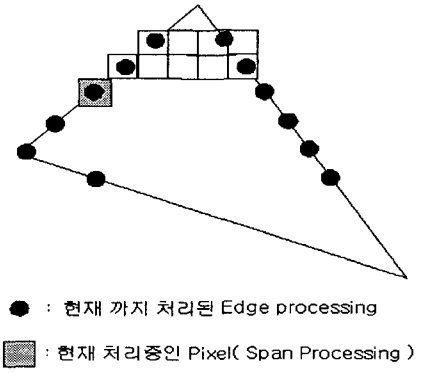

그림 6 Edge and Span Processing

3.2 동작원리 및 결과.

제안하는 pre-fetch 시스템은 다음의 순서로 수행한다.

①처음에는 텍스쳐 케쉬가 비어있기 때문에 케쉬의 미스가 발생하게 된다. 케쉬가 미스 되면 주 메모리에서 pre-fetch된 값 또는 스팬의 결과로 산출되는 값을 가져오게 된다.

(주 메모리에서 데이터를 가져올 때에는 하나의 텍셀씩을 가져오는 것이 아니라 stream이나 tile방식으로 값을 가져오게 되는데 tile 방식은 그 텍셀을 포함한 사각형의 값을, stream 방식은 그 텍셀로부터 일정 길이의 값을 가져오는 방법을 말하며 최근에는 tiling 방식이 많이 사용 되고 있다. 메모리에서 텍셀을 가지고 오는 방법 및 한번에 가져올 수 있는 크기 또한 케쉬의 미스률에 영향을 준다[3][5].)

②케쉬가 채워진 후 pre-fetch된 값(변 처리에서 받은 u, v값)이 텍스쳐 케쉬에 있는지 검사를 하게 되며 없을시 주 메모리로부터 그 값을 보조 케쉬로 가져오게 된다. 검사할 때 1 cycle이 소요되며 보조 케쉬에서 텍

스쳐 캐쉬로의 이동도 1 cycle에 가능하다고 가정한다. 메인 메모리에서 읽을 때에는 20~30 cycle이 사용된다.

③텍스쳐 매핑 처리시 원하는 값이 텍스쳐 캐쉬에 없고 보조 캐쉬에 있을 때 그 값을 텍스쳐 캐쉬에 저장한다. (1cycle)

④변 처리로부터 Pre-fetch에 의한 보조 캐쉬의 주 메모리 읽기와 스판 처리에 의한 텍스쳐 캐쉬의 메인 메모리 읽기가 동시에 일어날 경우 그 우선순위를 텍스쳐 캐쉬의 읽기에게 준다.

⑤위의 ②~④를 반복한다.

본 논문에서는 제안하는 3D 그래픽 가속기 파이프라인의 구조를 C언어를 사용하여 각 경우에 대해 구현하였고 삼각형 셋업 단에 각 경우에 대해 동일한 삼각형의 입력을 주어 캐쉬의 미스률을 비교하였다.

표1은 이 각각의 경우에 대한 시뮬레이션 결과를 보여준다.

	Cache size	size of tile	Auxiliary Cache	Miss Ratio
Non pre-fetch system	256	32	non	14%
	512	32	non	12%
	256	64	non	11%
	512	**64**	**non**	**9%**
Pre-fetch system	256	32	64	10%
	256	32	128	8%
	512	32	64	9.4%
	512	32	128	7%
	256	64	64	6.6%
	256	**64**	**128**	**3.7%**
	512	64	64	3.2%
	512	64	128	**1.4%**

[표1] 성능비교 (단 위 : Byte)

동일 조건에서 cache의 크기를 256 byte와 512 byte로 시뮬레이션 하였을 때 Non pre-fetch에선 3%의 향상을 그리고 pre-fetch에서는 0.6~3.4%의 성능의 향상을 보였고 사용된 캐쉬의 크기 면에서 비교를 하면, 64byte의 타일크기를 가질 때 Non pre-fetch중 512 byte 일때와 pre-fetch 시스템의 384 byte(256(캐쉬) + 128(보조 캐쉬))일때를 비교하면 128 byte나 캐쉬의 크기가 작은 pre-fetch 시스템이 5.3%의 미스률의 감소를 가져오는 것을 알 수 있다.

또 표의 결과를 보면 미스률에 가장 큰 영향을 주는 것은 주 메모리에서 캐쉬로 한번에 얼마나 많은 데이터를 전송할 수 있는가, 즉 tile의 크기 이다. 다음으로 보조 캐쉬의 크기 그리고 캐쉬의 크기가 미스률에 영향을 주는 것을 알 수 있었다.

제안하는 프리펫치 시스템을 사용함으로써 우리는 최대 6.6%의 cache miss를 절약할 수 있었다.

Ⅳ. 결론

반도체 공정의 향상으로 보조 Cache 및 pre-fetch 회로에 의한 약간의 Chip size 증가보다 성능의 향상이 중요한 요소로 떠오를 것으로 예상된다.

앞에서 살펴 본 것과 같이 Cache의 miss는 전체 pipeline에 큰 영향을 미치는 것으로 Shading의 방법 및 버퍼의 크기, 그리고 메모리에서 Cache로 읽어드리는 크기와 방법, 보조 Cache의 크기에 따라 달라진다.

제안하는 프리펫치 시스템을 사용함으로써 우리는 최대 6.6%의 cache miss를 절약할 수 있었다.

Reference

[1]J.D Foley, A. Dam, S. K. Feiner and J. F. Hughes, computer Graphics principles and practice 2th Ed, Addson-Wesley, Massachusetts, 1997.

[2] A.Kugler, "The setup for Triangle Rasterization, " 11th Euro-graphics Workshop in Computer Graphics Hardware, August 1996.

[3] Tomas Mőller and Eric · Haines "Real-Time Rendering. p99~120: Texturing

[4] COMPAQ, "Neon: A (Big) (Fast) Single-Chip 3D Workstation Graphics Accelerator." http://www.research.compaq.com/wrl/admin/publications.html

[5] Intel. "Intel740 Graphics Accelerator Software Developer's Manual," Feb. 1998, http://support.intel.com/support/graphics/intel740/.