3 차원 렌더링 프로세서를 위한 효과적인 가시성 선별 방법

최문희*, 박우찬**, 김신덕*
*연세대학교 컴퓨터과학과
**세종대학교 컴퓨터공학부
e-mail : mhchoi@cs.yonsei.ac.kr

An effective visibility culling method for 3D rendering processor

Moon-Hee Choi*, Woo-Chan Park** , Shin-Dug Kim*
*Dept. of Computer Science, Yonsei University
** School of Engineering, Sejong University

요 약

최근 3 차원 그래픽 영상의 복잡도가 점점 증가함에 따라, 가시성 선별에 관련된 연구는 3 차원 렌더링 프로세서 설계에 있어서 중요한 핵심 연구 중 하나가 되었다. 본 논문에서는 기존의 픽셀 캐쉬의 정보를 이용하여 가시성 선별을 수행하는 새로운 래스터라이제이션 파이프라인을 제안하고 있다. 제안 구조에서는 가시성 정보를 관리하기 위해서 계층적 z-버퍼 (*HZB*)와 같이 규모가 큰 별도의 하드웨어를 추가하지 않고, 픽셀 캐쉬에 저장되어 있는 데이터를 참조하여 주사 변환 과정에서 가시성 선별을 수행하고 있다. 캐쉬에서 접근 참조 실패된 프리미티브에 대해서는 픽셀 래스터라이제이션 파이프라인의 *z*-테스트 과정에서 은면 제거를 수행하도록 하였고, 선 인출 기법을 적용하여 픽셀 캐쉬의 접근 실패에 따른 손실을 줄여주었다. 실험 결과, 제안 구조는 일반 픽셀 파이프라인 구조에 비해 약 32%, *HZB* 구조에 비해 약 7%의 성능 향상을 보이고 있다.

1. 서론

현재 3 차원 그래픽 처리 기술은 상당히 발전하였으나 아직도 실시간 내로 현실과 같은 영상을 그려내는 데에는 한계가 있다. 가시성 선별은 이를 극복하기 위한 중요한 기법 중 하나이다. 은면 제거 (hidden surface removal) 방법으로 널리 알려진 가시성 선별은 실제 보이는 부분만을 화면에 그려줌으로써 필요 없는 데이터를 효과적으로 제거해주는 방법이다 [1, 2, 3, 4, 5, 6].

가시성 선별의 관련 연구 중 하드웨어로 구현하기 가장 적합한 분야로는 이미지 공간 선별 (image space culling)로, 이 분야는 현재 처리중인 객체들의 가려짐 여부 (occlusion determination)를 이미지 공간에서 수행한다 [5]. 대표적인 예로 계층적 z-버퍼 (hierarchical z-buffer: HZB) [2], 계층적 가림지도 (hierarchical occlusion map: HOM) [7], Bartz 의 *OpenGL assisted occlusion culling* [8], 그리고 HP 의 *VISULIZE-fx* [9] 등이 있다.

HZB 는 z-버퍼를 기반으로 z-피라미드 (z-pyramid)를 생성하여 보이지 않는 프리미티브들 (primitives)을 텍스쳐 매핑 과정 (texture mapping) 이전에 빠르고 효과적으로 제거하는 구조이다. 이 구조는 *conservative visibility* 을 수행하므로 보여진다고 판단되는 모든 프리미티브들은 항상 렌더링 (rendering)된다. *Conservative visibility set* 에는 보이는 프리미티브 뿐만 아니라 보일 것 같은 프리미티브까지도 포함된다. 그러나 HZB 는 z-피라미드를 생성하기 위한 규모가 큰 하드웨어를 필요로 하는 약점이 있다 [5, 6, 7].

HOM 구조는 전면의 가리개 (occluder)들을 혼합하여 가림지도를 생성하고, 이를 사용하여 가시성 선별을 수행하는 구조이다. 그러나 이는 낮은 선별 효율성을 가지고, *conservative visibility* 가 아니므로 보이는 프리미티브가 렌더링 되지 않고 버려질 수 있다는 단점을 가진다 [5, 6]. *OpenGL assisted occlusion culling* 과 *VISULIZE-fx* 는 가시성 정보를 구하기 위해 기존의 OpenGL 파이프라인 흐름에 특별한 모드를 추가한 구조이다. 이는 적용되는 하드웨어의 사

양에 따라 그 성능이 달라지고, 연산의 중복 수행 및 렌더링 파이프라인의 지연을 발생시키며, *conservative visibility* 가 아니라는 단점을 가지고 있다 [5, 6].

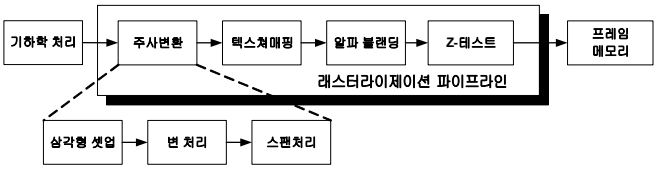
본 논문에서는 래스터라이제이션 상단에서 픽셀 캐쉬에 저장되어 있는 데이터를 참조하여 가시성 선별을 수행하는 구조를 제안한다. *HZB* 는 프레임의 전부 또는 일부를 z-피라미드에 유지함에 반하여, 제안 구조는 각 픽셀 캐쉬 블록에 속하는 픽셀의 깊이 값 중 최대 값만을 저장하는 별도의 하드웨어에 두어 가시성 선별을 수행한다. 이때 픽셀 캐쉬에서 접근 실패된 블록은 픽셀 래스터라이제이션 파이프라인의 z-버퍼링 과정에서 은면 제거를 수행하도록 하였고, [4]에서 제안한 선 인출기법을 적용하여 픽셀 캐쉬의 접근 실패에 따른 손실을 줄여주었다.

본 논문에서는 두 개의 벤치 마크인 *Quake3* [14]와 *Lightscape* [15]를 가지고 제안 구조의 동작 및 성능을 입증하였다. 래스터라이제이션 상단에서 픽셀 캐쉬의 접근 성공률은 Quake3 의 경우 72%, Lightscape 의 경우 66% 이었다. 이 실험 결과와 래스터라이제이션 상단에서 픽셀 캐쉬 접근 실패된 경우에 선 인출 기법을 적용한 결과를 가지고, 프래그먼트 당 필요 사이클 (average cycle per fragment: *ACPF*)를 구하여 제안 구조의 성능을 비교 분석하였다. 그 결과, 제안 구조는 일반 픽셀 래스터라이제이션 파이프라인에 비해 제안 구조는 *Quake3* 의 경우는 약 44%, Lightscape 는 약 21%의 성능 향상을 보이고 있다. 그리고, HZB 구조와 비교해 보니, 제안구조는 Quake3 의 경우는 약 9%의 성능 향상을 보이나 특유의 실행 동작을 가지는 Lightscape 의 경우에서는 8%의 성능 감소가 있었다. 여러 벤치마크 중에 Quake3 는 널리 알려진 게임엔진이자 다른 연구에서도 널리 사용하고 있는 벤치마크이므로 실험 결과 중에 Quake3 의 결과가 가장 중요한 결과라고 말할 수 있다.

본 논문의 구성은 다음과 같다. 2 장에서는 *HZB* 기법과 래스터라이제이션 파이프라인 구조에 대해서 기술하고, 3 장에서는 제안하는 구조의 특징에 대해 설명한다. 4 장에서는 구축한 실험 환경을 기반으로 하여 얻어진 실험 결과를 분석하고, 마지막으로 5 장에서는 결론에 대해 기술한다.

2. 관련 연구

일반적으로 3 차원 그래픽스 파이프라인은 [그림 1]에서 보는 바와 같이, 3 차원 데이터는 크게 기하학 처리와 래스터라이제이션 단를 거쳐 프레임 메모리로 보내진다. 일반적으로 래스터라이제이션 단계는 주사 변환 과정 (scan conversion stage)과 픽셀 처리 과정 (pixel processing stage)로 나뉘어진다. 주사 변환 과정은 입력된 프리미티브의 세 정점의 값을 가지고 내부의 모든 프레그먼트들을 구하는 작업을 하고, 픽셀 처리 과정에서는 구해진 프래그먼트에서 최종의 픽셀을 구하는 여러 과정인, 텍스쳐 맵핑 (texture mapping), 알파-블랜딩 (α-blending), z-테스트를 수행한다 [1, 3, 4].

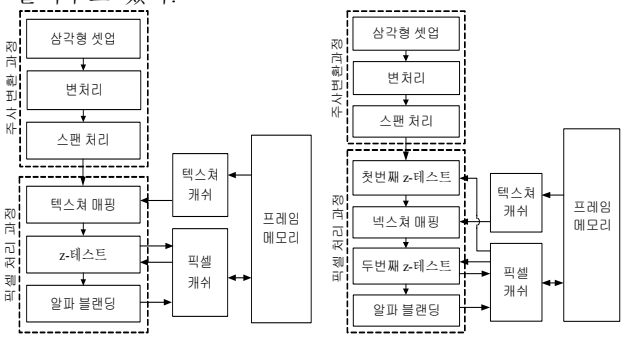


[그림 1] 그래픽 렌더링 파이프라인.

2.1 래스터라이제이션 파이프라인 구조

현재 고성능 그래픽 프로세서에서 일반적으로 사용되는 픽셀 래스터라이제이션 파이프라인은 텍스쳐 매핑이 z-테스트 이전에 수행되는 *pre-texturing* 구조 이다 ([그림 2]의 (가)) [4]. OpenGL 과 같은 표준 API 구성의 의미적 구조를 정확히 제공하는 이 구조는 실제 화면에 그려지지 않는 부분의 프래그먼트까지 텍스쳐 매핑을 수행해야 되는 단점을 가진다. 이와 반대로 z-테스트 이후에 텍스쳐 매핑을 수행하는 *post-texturing* 구조를 생각해 볼 수 있다. 그러나 이 구조는 z-테스트 결과를 프레임 메모리에 저장하지 못하고 텍스쳐 매핑 및 알파-블랜딩을 한 후에 저장하므로 프레임 메모리의 일관성을 유지 하는 데에 어렵다는 단점이 있다 [4].

이에 [4]에서는 [그림 2]의 (나) 구조와 같은 텍스트 매핑 전후에서 z-테스트를 두 번 수행 하여 위 두 구조의 단점을 극복하는 *mid-texturing* 구조를 제안하고 있다. 이는 *pre-texturing* 에서 발생하는 불필요한 프래그먼트들의 텍스쳐 매핑 연산을 제거하고, *post-texturing* 에서 발생하는 프레임 메모리의 일관성 문제를 해결하였다. 또한 이 구조에서는 선 인출 기법을 적용하여 픽셀 캐쉬의 접근 실패에 따른 손실을 줄여주고 있다.

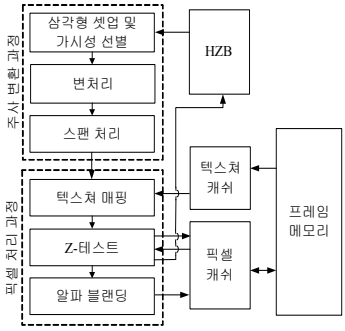


(가) *pre-texturing* **(나) *mid-texturing***

[그림 2] 래스터라이제이션 파이프라인 구조.

2.2 계층적 z-버퍼

Greene 에 의해 제안된 *HZB* 은 이미지 공간 일관성 (image-space coherence), 객체 공간 일관성 (object-space coherence), 그리고 시간적 일관성 (temporal coherence)을 모두 이용한 가시성 선별 기법이다 [2, 5, 6]. 특히 *HZB* 는 z-버퍼에 기반한 이미지 공간 선별을 수행하기 위해서 *z*-피라미드를 사용하여 보이지 않는 객체를 빠르게 제거하고 있다. *Z*-피라미드는 *z*-버퍼를 계층적으로 분할하여 구성한 것으로, 가장 하단에 $N \times N$ 인 기존의 *z*-버퍼의 데이터를 그대로 저장하고, 바로 다음 상단은 z-버퍼를 $N/2 \times N/2$ 표본화하여 저장하며, 해상도가 1×1 이 될 때까지 반복하여 저장한다 [2].



[그림 3] 계층적 z-버퍼 구조.

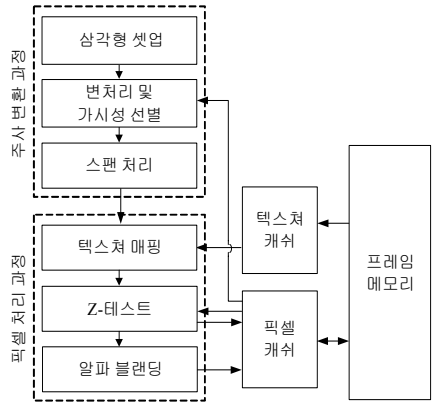
Z-피라미드를 생성하기 위해서 매우 큰 규모의 하드웨어와 메모리 트래픽이 요구되므로, *Green* 은 최적화된 계층적

z-버퍼 구조와 같이 적은 대역폭과 저장장치를 이용하여 이를 극복하고 있다. 현재 대표적 고성능의 그래픽 가속기인 ATI 의 *Radeon* 에서는 최적화된 *HZB* 기법을 적용하여 가시성 선별을 효과적으로 수행하고 있다 [5].

3. 제안하는 래스터라이제이션 파이프라인

이 장에서는 제안하는 새로운 구조인 선 인출 기법 가진 캐쉬 블록 별 가시성 선별의 수행 구조 (visibility culling per cache block with pre-fetch: *VCBP*) 에 대해 설명한다.

3.1 선 인출 기법 가진 캐쉬 블록 별 가시성 선별 구조



[그림 4] 제안하는 래스터라이제이션 파이프라인 구조.

[그림 4]에서 보듯이, 본 논문의 제안 구조는 변 처리 과정에서 가시성 선별을 수행한다. 따라서 *HZB* 와 유사하게 픽셀 처리 과정에서의 불필요한 파이프라인 점유는 발생하지 않는다. 제안 구조의 가시성 선별은 변 처리 과정 (edge walk stage)에서 생성된 스팬 블록들 (span block)의 대표 값과 이와 동일한 화면 주소를 갖고 픽셀 캐쉬에 존재하는 블록들의 대표 값 간의 비교를 통해 수행된다.

변 처리 과정에서의 생성된 각 스팬의 대표 값은 스팬 내부의 프리그먼트들 중 시점에서 제일 먼 프래그먼트의 z-값이다. 각 픽셀 캐쉬 블록의 대표 값은캐쉬 블록 내 픽셀의 z-값들 중 가장 큰 z-값이고, 최대 z 값 테이블에 저장된다. 이 테이블에 저장된 z 값은 프레임 메모리로부터 캐쉬 블록을 가져올 때와 z-테스트 과정이 끝난 후 그 결과 값을 픽셀 캐쉬에 쓸 때 이 값이 테이블에 저장된 z-값보다 큰 경우에 갱신된다.

가시성 선별 시 픽셀 캐쉬에 존재하는 블록의 경우는 제안한 가시성 선별 과정을 수행하면 된다. 그러지 않는 경우는 프레임 메모리로부터 해당 블록을 가져와야 한다. 이와 같은 캐쉬 접근 실패는 래스터라이제이션 파이프라인 지연을 야기함으로 성능에 큰 영향을 미칠 수 있다. 따라서 본 논문에서는 이러한 파이프라인 지연을 줄여주기 위해 [4]에서 제안한 선 인출 기법을 적용한다. 즉, 접근 실패된 블록을 프레임 메모리에서 가져 옴과 동시에, 접근 실패를 일으킨 스팬 내의 각 프래그먼트들은 래스터라이제이션 파이프라인의 하단 과정들을 수행도록 한다. 그리고 최종적으로 z-테스트 과정에서 픽셀 캐쉬에 접근하여 가시성 테스트를 수행 하도록 한다.

제안 구조는 일반 픽셀 래스터라이제이션 파이프라인 구조에 비교 유닛, 최대 z 값 생성기, 그리고 최대 z 값 테이블을 첨가하여 새로운 구조로 구성하였다. 비교 유닛은 변 처리 과정에서 가시성 선별을 수행하기 위해, 스팬 블록의 대표 값과 픽셀 캐쉬의 해당 블록의 대표 값간의 비교를 수행한다. 최대 z 값 생성기는 프레임 메모리로부터 가져오는 캐쉬 블록의 대표 값을 구하는 역할을 한다. 픽셀 캐쉬 크기가 16K bytes 이고, 블록 사이즈는 64 bytes 인 직접 사상 캐쉬라고 가정하면, 최대 z 값 테이블은 총 15 개의 32bit 비교기가 깊이 4 인 트리 구조로 구성될 수 있다.

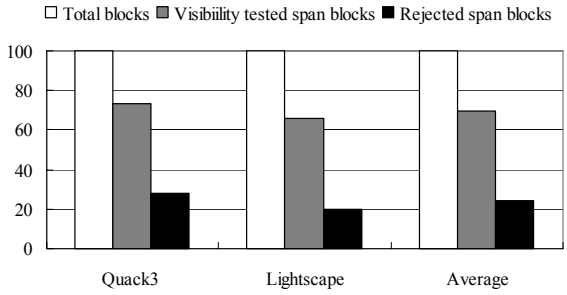
4. 실험 및 결과 분석

(가) Quake3 **(나) Lightscape**

[그림 5] 벤치마크.

본 논문에서는 제안하는 구조를 입증하기 위해 추적 기반의 시뮬레이터를 구축하였다. 실험에 사용된 환경은 레드햇 리눅스 *V*7.1 운영체제에서 OpenGL 명령어를 지원하는 Mesa 라이브러리 *V*3.4 를 사용하여 z-버퍼의 메모리 참조에 대한 주소를 생성하였다. 생성된 주소는 캐쉬 시뮬레이터인 *Dinero III* 를 사용하여 제안 구조를 분석할 수 있는 실험들을 수행하였다 [13]. [그림 5]는 실험에 사용한 벤치마크의 영상을 캡쳐한 것이다. *Quake3* 는 OpenGL 기반의 매우 유명한 게임 엔진으로서, 현존하는 상용 게임 소프트웨어 중 가장 완성도가 뛰어나다고 평가 받고 있고 현재 3 차원 그래픽 가속기의 성능을 측정하는데 공식적으로 사용되고 있다 [14]. *Lightscape* 는 *SPECviewperf*^{TM} 에서 제공하는 벤치마크로서, OpenGL 에서 수행되는 3 차원 렌더링 시스템의 성능을 측정하는데 기본적으로 사용되고 있다 [15].

4.1 제안구조에서의 가시성 선별의 효과



[그림 6] 제안된 VCBP 구조에서의 가시성 선별 비율.

[그림 6]는 변 처리 과정에서의 제안된 가시성 선별을 수행한 경우 그 효과에 대해 보여준다. 즉, 제안 구조의 가시성 선별 과정에서 제거되는 스팬 블록의 양, 즉 가시성 선별의 효과 정도를 알 수 있다.

여기서 가시성 테스트되는 스팬 브록 비율은 변 처리 과정에서 가시성 선별을 위해 픽셀 캐쉬의 접근 성공 비율 (H_{block})과 같다. 그리고 제거되는 스팬 블록은 가시성 선별을 수행한 후 제거되는 보이지 않는 스팬 블록을 의미한다.

그 결과, Quake3 의 경우는 가시성 테스트되는 스팬 브록 비율이 72%, 제거되는 스팬 블록의 비율이 28%이고, Lightscape 의 경우는 각각 66%, 20%이다.

4.2 제안구조에서의 가시성 선별의 효과

본 논문에서는 제안하는 구조의 성능을 비교 분석하기 위해서 프래그먼트 당 필요 사이클 (average cycle per fragment: *ACPF*)를 측정하였다. 제안 구조의 비교 대상으로는 2 장에서 설명된 *pre-texturing*, *mid-texturing*, 그리고 *HZB* 구조로 하였다. 본 논문에서는 각각의 구조의 ACPF 를 측정하기 위해서 픽셀 캐쉬와 텍스쳐 캐쉬의 접근 실패에 따른 손실을 10 사이클, 캐쉬와 프레임 메모리 간의 대역폭은 무제한적이라고 가정한다. [4]에서 정의된 $ACPF_{pre}$ 와 $ACPF_{mid}$ 를 토대로 하여, 다음과 같이 *HZB* 과 제안한 *VCBP* 의 각각의 *ACPF* 를 구하였다:

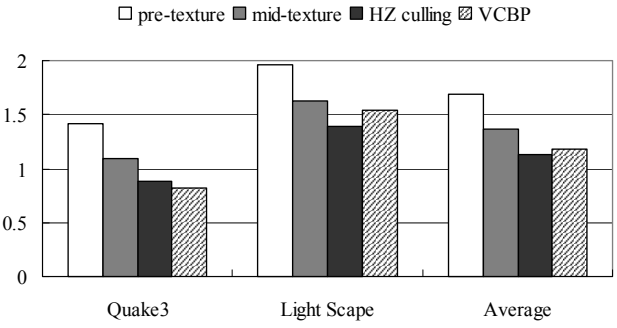
$$ACPF_{pre} = H_{pix} \times H_{tex} + (1 - H_{pix}) \times T_{pix} + (1 - H_{tex}) \times T_{tex},$$

$$ACPF_{mid} = (H_{pix} \times \gamma) + \{H_{pix} \times (1-\gamma) \times (H1_{pix} \times H_{tex} + (1 - H1_{pix}) \times T_{pix} + (1 - H_{tex}) \times T_{tex})\} +$$
$$\{(1 - H_{pix}) \times ((1 - H_{tex}) \times T_{tex} + T_{pix} \times (1 - reduction))\},$$

$$ACPF_{HZB} = (H_{pix} \times H_{tex} + (1 - H_{pix}) \times T_{pix} + (1 - H_{tex}) \times T_{tex}) \times (1 - \gamma),$$

$$ACPF_{propose} = O_{block} \times \gamma \div N_{frag} + O_{block} \times (1-\gamma) \times \left(\sum_{i=1}^{N_{frag}} (H1^{i}_{pix} \times H_{tex} + (1 - H1^{i}_{pix}) \times T1_{pix}) \div N_{frag} + (1 - H_{tex}) \times T_{tex}\right) +$$
$$(1 - O_{block}) \times ((1 - H_{tex}) \times T_{tex} + (1 - reduction) \times \sum_{i=1}^{N_{frag}-1} (1 - H^{i}_{pix})) \times T1_{pix} \div N_{frag}).$$

여기에서 H_{pix} 와 H_{tex} 는 각각 픽셀 캐쉬와 텍스쳐 캐쉬의 접근 성공율이고, T_{pix} 와 T_{tex} 는 각각 픽셀 캐쉬와 텍스쳐 캐쉬의 접근 실패에 따른 손실을 말한다. 그리고 $T1_{pix}$ 는 제안 구조에서 최대 *z* 값 생성기에서 발생되는 메모리 지연 시간이 포함된 픽셀 캐쉬의 접근 실패에 따른 손실이다. γ 는 전체 영상에서 가려지는 부분의 비율을 의미하고, O_{cache} 는 프리미티브 당 차지하는 픽셀 캐쉬의 블록의 점유율이다. *Reduction* 은 선 인출 기법을 적용한 구조에서 파이프라인 단계의 길이에 따라 픽셀 캐쉬 접근 실패에 따른 손실이 얼마나 감소 되는가를 의미한다 [4]. O_{block} 은 각 프리미티브가 가시성 선별을 수행하기 위하여 처음 픽셀 캐쉬를 참조할 때의 점유되어 있는 블록의 비율을 의미하고, N_{frag} 는 픽셀 캐쉬의 한 블록내의 프래그먼트의 수를 말한다. $H1_{pix} = 1 - M1_{pix}$ 로서, $M1_{pix}$ 는 *mid-texturing* 구조에서 첫 번째 z-테스트에서는 픽셀 캐쉬 접근 성공 되었으나, 두 번째 *z*-test 시 접근 실패되어 접근 실패에 따른 손실을 그대로 받게 되는 경우를 말한다. $H1^{i}_{pix} = 1 - M1^{i}_{pix}$ 로서, 제안 구조에서 $H1_{pix}$ 와 유사한 경우를 말한다.



[그림 7] 각 구조에서의 프래그먼트 당 필요 사이클 (*ACPF*).

[그림 7]에서 보면 알 수 있듯이, *Quake 3* 의 경우 제안 구조는 *pre-texturing* 에 비해 약 44%, *mid-texturing* 에 비해 약 27%, 그리고 *HZB* 에 비해 약 9%의 성능 향상을 보이고 있다. *Lightscape* 의 경우는 각각 약 21%, 약 7%의 성능 향상을 보이나, *HZB* 구조에서는 8%의 성능 감소가 있었다. 이는 *Lightscape* 가 *Quake3* 에 비해 매우 낮은 reduction 비율을 가지고 있기 때문이다.

5. 결론

본 논문에서는 효과적인 가시성 선별을 수행하기 위해서, *HZB* 와 달리 주사 변환 과정 에 적은 규모의 하드웨어를 추가하고 픽셀 캐쉬의 데이터를 참조하는 새로운 래스터라이제이션 파이프라인을 제안하고 있다. 또한 픽셀 캐쉬에서 접근 실패된 프리미티브에 대해서는 픽셀 래스터라이제이션 파이프라인의 *z*-테스트 과정에서 은면 제거를 수행하도록 하였으며, 선 인출 기법을 적용하여 픽셀 캐쉬의 접근 실패에 따른 손실을 줄여주었다. 두 벤치마크인 *Quake3* 와 *Lightscape* 로 실험한 결과, 제안 구조는 다른 래스터라이제이션 구조에 비해 좋은 성능을 가짐을 알 수 있었다.

참고 문헌

[1] J.D. Foley, A. Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics, Principles and Practice*, Second Edition, Addison-Wesley, Massachusetts, 1990.

[2] N. Greene, M. Kass, and G. Miller, “Hierarchical z-buffer visibility,” *Proc. SIGGRAPH ‘93*, pp. 231–238, Aug. 1993.

[3] S. Morein, “ATI Radeon - HyperZ technology,” *Hot3D Session 2000 SIGGRAPH/Eurographics Workshop Computer Graphics Hardware*, http://www.ibiblio.org/hwws/previous/www_2000/presentations/ATIHot3D.pdf, Aug. 2000.

[4] Woo-Chan Park, Kil-Whan Lee, Il-San Kim, Tack-Don Han, and Sung-Bong Yang, "An Effective Pixel Rasterization Pipeline Architecture for 3D Rendering Processors," *IEEE Transactions on Computers*, Vol. 52, No. 11, pp. 1501-1508, Nov. 2003.

[5] N. Greene, “Interactive Geometric Computing Using Graphics Hardware -Visibility Culling using Graphics Hardware,” *SIGGRAPH 2002 Tutorial Course #31*, http://gamma.cs.unc.edu/SIG02_COURSE/course31_2002.pdf, July, 2002.

[6] D. Cohen-Or, Y. Chrysanthou, and C. Silva, "A Survey of Visibility for Walkthrough Applications," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 9, No. 3, pp. 412-431, Jul.-Sep. 2003.

[7] H. Zhang, D. Manocha, T. Hudson and K. E. Hoff, “Visibility culling using hierarchical occlusion maps”, *Proc. SIGGRAPH’97*, pp. 77-88, July 1997.

[8] D. Bartz, M. Meißner and T. Hüttner, “Extending Graphics Hardware for Occlusion Queries in OpenGL,” *Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pp. 97-104, Aug. 1998.

[9] Scott N, Olsen D, Gannett E., “An Overview of the VISULIZE fx graphics accelerator hardware,” *The Hewlett-Packard Journal*, pp. 28-34, 1998.

[10] M. Meißner, D. Bartz, R. Günther and W. Strßer, “Visibility Driven Rasterization,” *Computer Graphics Forum*, vol. 20, no. 4, pp. 283-294, 2001.

[11] M. Woo, J. Neider, and T. Davis, *OpenGL programming guide*, Addison Wesley, 1996.

[12] M. F. Deering, S. A. Schlapp, and M. G. Lavelle, “Fbram: A new form of memory optimized for 3d graphics,” *Proc. SIGGRAPH’94*, pp. 167-174, 1994.

[13] L. Bishop, D. Eberly, T. Whitted, M. Finch, and M. Shantz, “Designing a PC Game Engine,” *IEEE Computer Graphics and Applications*, vol. 18, no. 1, pp. 46-53, Jan. 1998.

[14] http://www.idsoftware.com/games/quake/quake3-arena/, 2003.

[15] http://www.spec.org/gpc/opc.static/viewperf711info.html, 2003.