실시간 단일 패스 가시성 선별 기법 기반의 3차원 그래픽스 가속기 구조

주 지 원[†] · 최 문 희[††] · 김 신 덕[†††]

요 약

차폐 선별 기법은 가시성 선별 기법 중 하나로, 다른 물체에 가려서 보이지 않는 물체나 삼각형에 대한 연산을 제외시키는 기법이다. 이는 불필요한 연산량을 효과적으로 줄이기 때문에 복잡한 장면을 실시간으로 처리하기 위해 필수적이다. 하지만 기존의 차폐 선별 기법인 차폐 쿼리는 가시성 검사를 위해 물체 데이터를 하드웨어에 두 번 보내야 하며, 이로 인해 불필요한 연산이 발생한다. 또 다른 기존 하드웨어 차폐 선별 기법인 VCBP는 빠른 수행을 하지만 바운딩 볼륨의 검사를 지원하지 않으며 응용으로 그 결과를 보내는 기능이 없다. 본 논문에서는 이러한 문제점들을 해결한 가시성 선별과 렌더링을 한 번에 처리할 수 있는 단일 패스 알고리즘을 제안한다. 제안하는 기법은 일차적으로 3차원 가속 하드웨어의 초기 단계인 삼각형을 픽셀로 나누는 래스터화 단계에서 캐쉬를 이용하여 빠르게 가시성 선별을 수행한다. 그와 동시에 가시성 선별 과정에서는 각 프리미티브의 가시성 정보를 응용단계로 보낸다. 응용단계에서는 하드웨어로부터 받은 이전 프레임의 가시성 정보와 공간 계층 트리 구조를 이용하여 하드웨어로 보내는 보이지 않는 프리미티브를 위한 데이터량을 획기적으로 줄인다. 제안하는 구조는 하드웨어 차폐 선별 쿼리를 이용하는 기존 이중 패스 알고리즘 중 S&W 대비 최대 44%, 최저 14%의 성능이 향상되었고, CHC 대비 최대 25%, 최저 17%의 성능이 향상되었다.

키워드 : 실시간 3차원 그래픽스, 가시성 선별 기법, 차폐 선별 기법

A Real-time Single-Pass Visibility Culling Method Based on a 3D Graphics Accelerator Architecture

Choo Catherine ESTL[†] · Moon-Hee Choi[††] · Shin-Dug Kim[†††]

ABSTRACT

An occlusion culling method, one of visibility culling methods, excludes invisible objects or triangles which are covered by other objects. As it reduces computation quantity, occlusion culling is an effective method to handle complex scenes in real-time. But an existing common occlusion culling method, such as hardware occlusion query method, sends objects' data twice to GPU and this causes processing overheads once for occlusion culling test and the other is for rendering. And another existing hardware occlusion culling method, VCBP, can test objects' visibility quickly, but it neither test bounding volume nor return test result to application stage. In this paper, we propose a single pass occlusion culling method which uses temporal and spatial coherency, with effective occlusion culling hardware architecture. In our approach, the hardware performs occlusion culling test rapidly with cache on the rasterization stage where triangles are transformed into fragments. At the same time, hardware sends each primitive's visibility information to application stage. As a result, the application stage reduces data transmission quantity by excluding covered objects using the visibility information on previous frame and hierarchical spatial tree. Our proposed method improved maximum 44%, minimum 14% compared with S&W method based on hardware occlusion query. And the performance is increased 25% and 17% respectively, compared to maximum and minimum performance of CHC method which is based on occlusion culling method.

Keyword : Real-time 3D Ggraphics, Visibility Culling Method, Occlusion Culling Method

1. 서 론

삼차원 그래픽 하드웨어의 성능이 나날이 발전하고 있지만, 아직도 장면의 복잡성과 자원의 제약 때문에 이를 실시간으로 처리하는 데에는 많은 어려움이 따른다. 가시성 선별 기법은 삼차원 그래픽을 처리하기 위한 방대한 연산량을 효과적으로 줄일 수 있는 대표적인 방법으로 실시간 렌더링을 좀더 가능성 있게 만들어준다. 몇 가지 가시성 선별기법(visibility culling method) 중 차폐 선별(occlusion culling)

※ 이 논문은 2006년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2006-311-D00750).
† 정 회 원 : 삼성전자 DM 연구원
†† 정 회 원 : 삼성전자 TN 선임 연구원
††† 정 회 원 : 연세대학교 컴퓨터과학과 교수
논문접수 : 2007년 9월 11일, 심사완료 : 2008년 1월 3일

DOI: 10.3745/KIPSTA.2008.15-A.1.1

은 다른 물체에 가려져 화면에 보이지 않는 물체를 가려내는 기법인데, 이 기법은 깊이 복잡도가 큰 장면에 특히 효과적이다[1,2,3]. 다른 가시성 선별 기법과는 달리 차폐 선별은 복잡한 알고리즘을 필요로 하지만, 점점 복잡한 장면을 많이 사용하게 됨에 따라 그 효용이 커지면서 많은 연구가 진행되어왔다.

여러 차폐 선별 기법 중 지금 상용 그래픽스 엔진에 쓰이는 차폐 선별 기법은 하드웨어 차폐 선별 쿼리(hardware occlusion query)를 이용하는 방법이다[4,5,6]. 이 기법은 하드웨어에서 쿼리 모드로 한 번 렌더링을 하여 가시성 검사를 하고, 다시 한번 렌더링 모드로 실제 화면에 물체를 그리기 위해 렌더링한다. 즉, 이 기법은 하드웨어로 물체 정보를 두 번 보내어 처리하는 투 패스(two-pass)방식이다. (그림 1)의 (1)과 (2)는 이와 같이 2번 렌더링하는 투 패스방식을 그림으로 표현한 것이며, 이러한 투 패스 방식은 오버헤드가 크기 때문에 검사를 위한 쿼리의 수를 줄이려는 시도가 있었지만, 근본적인 문제인 쿼리 자체의 오버헤드는 줄이지 못하였다. 한편, 최근에 발표된VCBP(Visibility culling based on pixel cache block)는 래스터화(rasterization) 단계에서 빠르게 차폐 선별을 수행한다. 하지만 바운딩 볼륨(bounding volume)을 처리하는 기능이 없으며 각 물체나 바운딩 볼륨의 테스트 결과를 어플리케이션에 알려주는 기능도 없다. 따라서 이 하드웨어만을 사용해서는 화면의 공간적 일관성이나 시간적 일관성을 이용하여 응용단계에서 데이터를 줄일 수 없다.

우리의 목표는 실시간으로 불필요한 폴리곤에 대한 연산 시간을 줄이는 효과적인 렌더링 시스템의 구성이다. 이를 위하여 위에서 언급한 두 목표 기법인, 하드웨어 차폐 선별 쿼리와 VCBP의 문제점을 해결하면서 장점을 취하는 방식으로 연구를 진행하였다. 차폐 선별 쿼리는 최근의 상용 3D 그래픽스 하드웨어에서 지원하는 방식이며, VCBP는 최근에 발표된 단순하고 빠르게 차폐 선별을 수행하는 하드웨어라는 점에서 그 개선에 의의가 있다고 판단하였다.

본 연구의 주요 성과는 세 가지로 다음과 같다. 첫째, 차폐 선별 쿼리 방법의 문제인 2 pass 알고리즘을 1 pass 알고리즘으로 개선하였다. 둘째, VCBP의 기본 차폐 선별 구조를 응용하면서 차폐 선별 쿼리처럼 공간 트리 구조의 내부 노드, 즉 바운딩 볼륨의 가시성 검사도 가능하게 하였으며, 내부 노드의 검사 결과를 응용단계에 알려주도록 하였다. (그림 1)의 (2)와 (3)에서 래스터화 단계에서 응용으로 향하는 화살표가 검사 결과의 전송을 의미한다. 첫 번째와 두 번째 성과를 위하여 VCBP의 차폐 선별 방식을 응용하여 래스터화 단계에서 차폐 선별 검사를 수행하였으며, 간단한 하드웨어를 추가하였다. 마지막으로 응용단계에서는 공간 계층 트리 구조와 전 프레임에서의 노드 검사 결과를 바탕으로 응용에서 GPU로 전송하는 데이터의 양을 최소화 하였다.

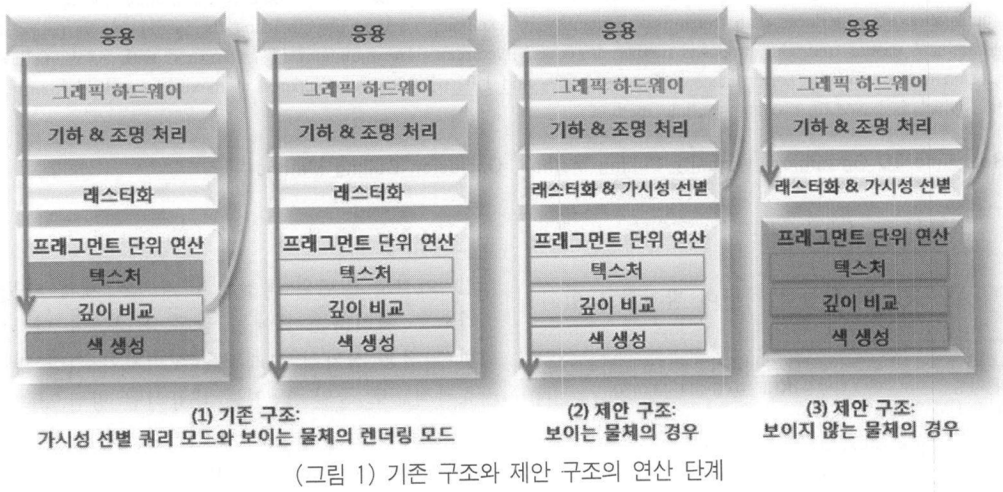
효율적인 연구를 위한 몇 가지 가정을 하였다. 투명한 물체가 있는 경우의 처리는 매우 복잡해지며, 본 연구의 목적은 빠른 수행을 위한 방법 연구이므로 화면에는 투명한 물체가 없는 경우만 고려하였다. 또한 연속된 프레임 간에는 배경이 되는 물체처럼, 물체의 움직임이 크지 않은 물체를 효과적으로 처리하기 위하여 시간적인 일관성을 이용하였다. 이는 일반적인 장면의 특징이며 이러한 가정이 성립하지 않는 경우에도 올바르게 작동하지만, 연속하는 프레임 간에 공통점이 많은 경우에 해당 알고리즘의 효과가 극대화된다.

본 논문은 다음과 같은 순서로 구성되어있다. 다음 절에서 기존 차폐 선별 기법을 소개하고, 3절에서 해결 방안인 단일 패스 가시성 선별 알고리즘에 대한 자세한 설명을 한 후, 4절에서 이를 위해 필요한 하드웨어 구조에 대해 논하겠다. 5절에서는 우리가 제안하는 구조의 성능 평가와 그 결과를 보이고, 6절에서 결론을 맺도록 하겠다.

2. 배경 및 관련 연구

2.1 계층 깊이 버퍼 (HZB: Hierarchical Z-Buffer)와 VCBP (Visibility Culling based on Pixel Cache)

HZB 기법은 하드웨어 차폐 선별 기법 중에서 가장 전통적이고 대표적인 방법으로 프레임 버퍼(frame buffer)의 깊이 버퍼를 다양한 해상도로 만든 깊이 피라미드(z-pyramid)



(그림 1) 기존 구조와 제안 구조의 연산 단계

를 이용한다. 각 물체는 8진 트리를 이용하여 깊이 피라미드의 값에 접근하며, 물체의 깊이 값이 깊이 피라미드의 최고 해상도 단계인 프레임 버퍼의 깊이 값보다 작으면 보이는 것으로 판단한다. 이 방법은 정확한 차폐 선별을 수행할 수 있지만, 하드웨어에서 실시간으로 수행하기엔 너무 복잡하고 메모리도 많이 필요하다는 단점이 있어서 실제로 하드웨어로 사용되지 않는다[1].

최근에 발표된 하드웨어 차폐 선별 기법인 VCBP는 rasterization 단계에서 픽셀 캐쉬에 저장된 깊이 대표값과 비교하여 차폐 선별을 수행하는 기법이다. Mid-texturing 기법을 이용하여 캐쉬 미스 패널티를 줄여 좋은 성능을 낸다. 하지만 공간 계층 구조를 사용할 수 있도록 바운딩 볼륨에 대한 처리는 할 수 없고, 각 물체에 대한 차폐 선별 검사 결과를 저장하거나 응용단계로 알려주는 기능이 없다. 따라서 응용단계에서 검사할 데이터를 최소화하기 위해 공간 계층 트리 구조를 응용한다거나 프레임간의 유사성을 이용하여 처리량을 줄이는 연산을 할 수 없다 [11].

2.2 하드웨어 차폐 선별 쿼리 (Hardware Occlusion Query)

하드웨어 차폐 선별 쿼리는 물체나 바운딩 볼륨이 다른 물체에 의해 가려지는지를 확인하기 위한 응용에서 GPU로 보내는 요청이다[7,8]. 이 쿼리를 받은 GPU는 쿼리 모드로 전환하여 라이팅(lighting)과 텍스처링(texturing)을 비롯한 색 관련 기능을 제외한 렌더링을 한다. 그리고 최종 렌더링 단계인 깊이 비교 단계에서 현 프레임 버퍼와 생성된 깊이 값을 비교한 후 프레임 버퍼에 쓰는 대신, 보이는 프래그먼트의 개수를 응용에 알려준다 [9]. 쿼리의 결과를 받은 응용단계는 렌더링 여부를 결정하여, 물체 정보를 다시 하드웨어로 보내어 렌더링 하게 된다. 이러한 2pass 방식은 물체를 두 번 하드웨어로 보내야 하기 때문에 연산량의 부담이 크다는 단점이 있는데, 이 문제점을 완화하기 위해 쿼리의 수를 줄이기 위한 S&W 알고리즘(Stop and Wait)과 CHC (Coherent Hierarchical Culling) 알고리즘이 제안되었다.

S&W 알고리즘은 공간 계층 트리 구조를 이용하여 이웃하는 물체들의 가시성 검사를 한 번에 할 수 있도록 하였다. 이 알고리즘은 트리의 내부 노드에 대해 차폐 선별 쿼리를 보내고, 그 결과를 기다렸다가, 결과가 보이는 것으로 나온 경우에 그 자손의 확인하는 방식으로 진행되었기 때문에, 멈추고 기다리는 (Stop and Wait) 방식이라고 불린다. 한편, CHC는 S&W를 발달시킨 방식으로 큐를 이용한 스케줄링과 장면간의 유사성을 이용하여 GPU로 보내는 차폐 선별 쿼리의 더욱 줄인 알고리즘이다. 이 두 알고리즘 모두 차폐 선별 쿼리의 수를 줄여 그로 인한 오버헤드를 절감시켰지만, 차폐 선별 쿼리 자체가 가지는 오버헤드는 여전히 해결되지 못한 채 근본적인 해결책을 제시하지 못하였다. 더욱이 쿼리의 수를 획기적으로 줄인 CHC 알고리즘은 이전 프레임에서 보였던 물체는 무조건 GPU로 보내 렌더링하기 때문에, 전체적으로 장면의 가시성 변화가 많은 경우에는 차폐 선별 검사의 효과를 거의 볼 수 없게 되는 문제점이 생긴다.

3. 제안하는 차폐 선별 방법

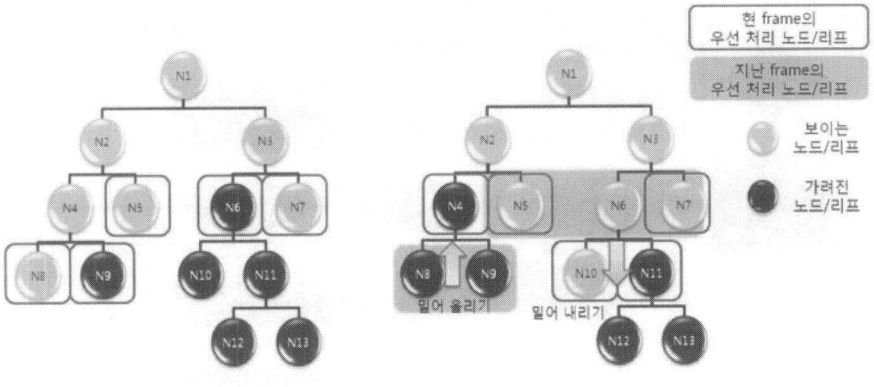
3.1 단일 패스 가시성 선별 알고리즘

제안하는 단일 패스 가시성 선별 알고리즘은 크게 시간적 일관성을 사용할 수 없는 첫 번째 프레임인 경우와 시간적 일관성을 이용할 수 있는 2번째 프레임 이후의 프레임인 경우로 나눌 수 있다. 우선 제안하는 알고리즘의 기본적인 요소인 시간적 일관성과 공간적 일관성에 대해 설명한 후, 첫 번째 프레임인 경우의 흐름과 두 번째 프레임 이후인 경우의 흐름에 대해 자세하게 설명하도록 하겠다.

3.1.1 공간적 일관성과 시간적 일관성

공간적 일관성은 공간 계층 트리를 사용하여 화면에서 비슷한 위치에 있는 물체들을 그룹으로 묶어 바운딩 볼륨으로 만들어 한 번에 처리할 수 있도록 하는 것을 말한다. 공간적 일관성을 이용하면 큰 물체에 가려 보이지 않는 비슷한 위치에 있는 여러 물체들을 한 번의 검사로 모두 잘라낼 수 있게 된다. 비슷한 위치에 있는 물체의 그룹은 공간 계층 트리에서 내부 노드로 저장되며, 각각의 물체는 가장 하단의 리프에 저장된다.

효과적이다. 응용단계에서는 이렇게 공통되는 부분의 판단을 통해 GPU로 보내는 데이터의 양을 효과적으로 줄일



K번 째 프레임 K+1번 째 프레임

(그림 2) 시간적 일관성을 이용하기 위한 우선 처리 노드의 유지

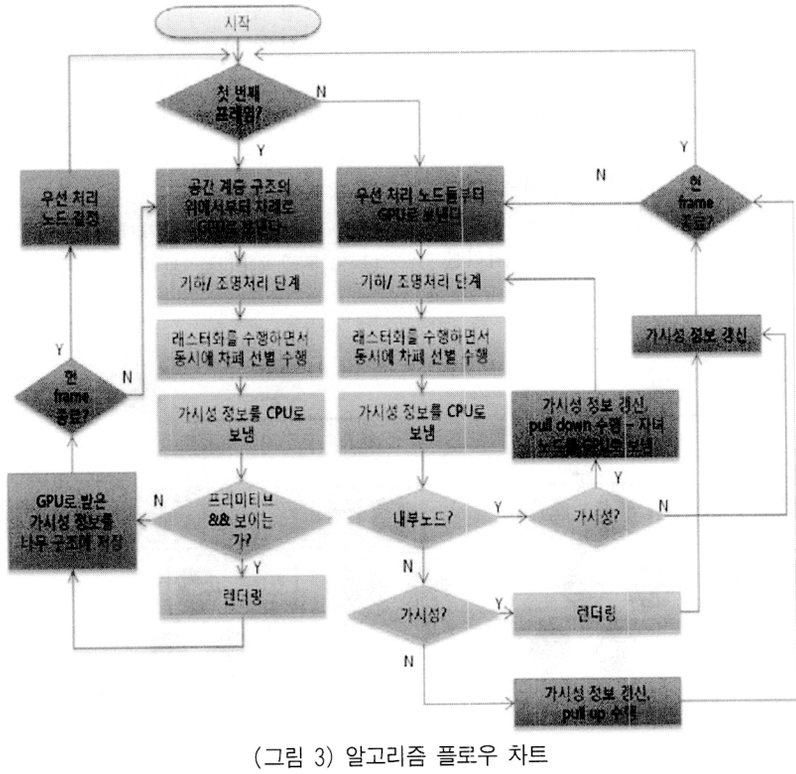
수 있다. 이를 위하여 우선 처리 노드를 정하는데, 우선 처리 노드는 다른 노드들 보다 먼저 GPU로 보내져 우선적으로 차폐 선별 검사를 받거나 렌더링 된다. 우선 처리 노드가 우선적으로 렌더링되는 경우, CHC에서는 보이지 않는 물체가 렌더링되는 경우가 발생하는데, 제안하는 방법에서는 하드웨어가 래스터화 단계에서 차폐 선별 검사를 하기 때문에 보이지 않는 물체는 연산량이 많은 픽셀 처리 단계를 거치지 않고 그 전에 걸러지게 된다. 우선 처리 노드를 유지하기 위하여 (그림 2)에서처럼 밀어 올리기 (pull up), 밀어 내리기 (pull down)의 방법을 사용한다. 이전 프레임에서 보이던 노드가 이번 프레임에서 보이지 않게 되면, 그 우선 순위가 부모로 이전되기 때문에 이 경우에 필요한 작업을 밀어 올리기라 하고, 반대의 경우에는 그 우선 순위가 자녀 노드로 이전되기 때문에 이 때 필요한 동작을 밀어 내리기라 한다. 즉 (그림 2)의 N8과 같이 k번째 프레임에서는 보였지만 k+1 번째 프레임에서는 보이지 않고, 형제 노드/리프도 모두 보이지 않는다면, GPU로 보낼 대상을 부모로 밀어 올린다. 또 N10과 같이 전 프레임에서는 보이지 않았던 물체가 다음 프레임에서는 보이게 된다면 밀어 내려서 다음 프레임에서 먼저 처리할 우선 처리 노드를 유지하도록 한다.

위와 같은 방법으로 보일 가능성이 높은 노드와 보이지 않을 가능성이 높은 노드 중 가장 상위 노드를 우선적으로 처리하여 GPU로 보내는 데이터의 양을 최소화 한다. 우선 처리 노드에 대한 정보와 해당 프레임에서의 가시성 정보는 각 프리미티브와 프리미티브 그룹에 대한 정보를 저장하고 있는 계층 트리 구조의 노드에 저장된다.

3.1.2 첫 번째 프레임

첫 번째 프레임의 경우는 참고할 이전 프레임이 존재하지 않기 때문에 3.1.1. 절에서 소개한 시간적 일관성을 사용할 수 없다. 즉 우선순위 노드를 먼저 GPU에서 처리할 수 없다. 따라서 이 경우에는 공간적 일관성만을 사용하여, 응용단계에서 GPU로 보내는 데이터의 양을 줄인다. 이때 응용단계에서의 동작은 기존의 차폐 선별 쿼리를 이용하는 CHC 알고리즘과 유사하지만, 제안하는 방법에서는 차폐 선별 쿼리의 오버헤드가 없기 때문에 이에 비해 효과적으로 수행될 수 있다. 여러 공간 계층 트리 중에서 우리는 시점을 기준으로 앞에서 뒤로 정렬하기 좋은 K-d 트리를 사용하며, 이 구조는 시작하기 전에 프리 프로세싱으로 구축하며 중간에 트리 구조의 변형은 없는 것으로 가정한다. 이 과정은 장면의 오브젝트의 개수에 따라 $O(n\log^2$의 시간이 소요된다.

(그림 3)의 왼쪽 부분은 첫 번째 프레임의 연산 흐름을 나타내고 있으며 응용단계에서 처리되는 부분은 노란색으로 표시하였고, GPU에서 처리되는 부분은 녹색으로 표기하였다. 응용단계에서 장면의 첫 프레임이라고 판단하면 응용은 너비 우선 탐색(breadth first search)을 하여 공간 계층 트리의 루트부터 차례로 GPU로 보낸다. 노드는 GPU에서 기하단계와 조명처리단계를 거친 후 폴리곤을 프래그먼트로 쪼개는 단계인 래스터화 단계로 넘어가게 된다. 래스터화 단계에서는 래스터화를 진행하며 동시에 가시성 차폐 선별을 수행한다. 하드웨어 차폐 선별 단계의 자세한 설명은 3.2절 하드웨어 구조에서 설명하도록 하겠다. 응용에서는 차폐 선별 검사의 정보를 정보를 계층 구조에 저장하고, 이를 기반으로 다음 프레임에서 이용한 우선 처리 노드를 선별한



(그림 3) 알고리즘 플로우 차트

다. 가시성 차폐 선별을 마친 GPU에서는 대상이 내부 노드가 아닌 프리미티브이고, 보이는 것으로 판별되면 렌더링을 계속하고, 그렇지 않은 경우에는 수행을 멈추고 다음 물체를 받아 수행한다.

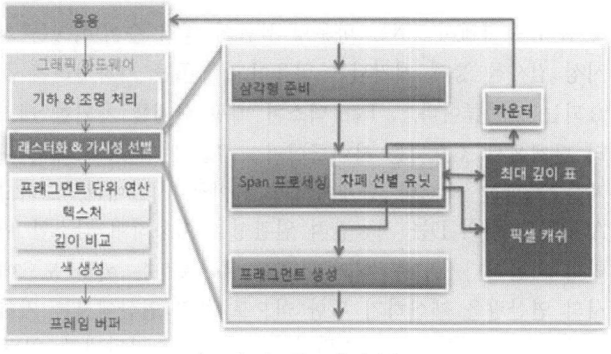
3.1.2 두 번째 이후의 프레임

첫 번째 프레임을 화면에 그리면서 응용단계는 계층 구조의 모든 노드의 가시성 정보를 얻게 되고, 이를 이용하여 우선 처리 노드를 선별할 수 있으므로, 두 번째 프레임 이후에는 시간적 일관성을 사용할 수 있게 된다. (그림 3)의 오른쪽 부분은 두 번째 이후의 프레임의 경우를 위한 플로우 차트이며, 노란색 상자는 응용단계에서의 수행, 녹색 상자는 GPU에서의 수행을 의미한다. 첫 번째 프레임의 경우와 다르게 응용단계는 우선 처리 노드들부터 GPU로 보내고, GPU는 같은 동작을 수행한다. 응용은 GPU로부터 해당 노드의 차폐 선별 처리 결과를 받으면, 그 결과를 기반으로 다음 프레임에 이용할 우선 처리 노드를 유지하기 위한 동작을 수행한다. 노드의 가시성 검사 결과가 보인다이고, 내부 노드이면 밀어 내리기(pull down)을 수행하여 해당 노드의 자식 노드를 GPU로 보낸다. 보이는 노드가 리프 노드, 즉 오브젝트인 경우에는 우선 처리 노드가 된다. 한편, GPU에 보낸 리프 노드가 가시성 검사를 통과하지 못했으면 (보이지 않으면), 밀어 올리기(pull down)을 수행하여 다음 프레임에서는 우선 처리 노드에서 제외한다.

3.2 하드웨어 구조

제안하는 구조의 하드웨어는 VCBP의 차폐 선별 구조를 기반으로 한다. (그림 4)에서 처럼 GPU로 들어온 공간 계층 구조 노드를 위한 바운딩 볼륨이나 프리미티브는 먼저 기하 조명처리 단계를 수행하고 래스터화 단계로 넘어온다[11,12]. (그림 4)의 오른쪽에 확대해 놓은 부분처럼 래스터화 단계에서는 먼저 삼각형을 프래그먼트 단위로 나누기 위한 준비 단계에서 삼각형의 세 빗변을 구한다. 이후, 삼각형을 span, 즉 삼각형을 프래그먼트 높이만큼씩 가로로 자른 단위에 대한 연산 단계를 수행한다. 이 단계에서는 span의 양 끝 점에 위치하는 프래그먼트 값을 기준으로 중간 값을 보간한다. Span 연산 단계를 수행하면서 span의 깊이 값을 기반으로 차폐 선별을 수행하는데, 이때 픽셀 캐쉬에서 깊이 값을 가져와 현재 프레임 버퍼에 그려져 있는 물체들과의 깊이 비교를 한다. 만약 이번에 처리하고 있는 물체의 깊이 값이 시점에서 더 가깝다면 보이는 것으로 판단하고, 카운터를 증가시킨다. 카운터에 저장된 값은 새로운 노드가 들어올 때, 응용단계로 보내고, 갱신한다. 처리하고 있는 노드가 리프 노드인 경우, 차폐 선별 유닛에서 보이는 것으로 판명난 프래그먼트만을 프래그먼트 생성 유닛으로 보낸다. 프리미티브가 아닌 경우에는 차폐 선별 유닛에서 이 후의 단계로 데이터를 보내지 않는다.

이러한 동작을 수행하는 하드웨어를 위해서는 처리하고 있는 노드가 내부 노드인지 리프 노드인지 알려주는 추가적인 1bit 데이터가 필요하다. 또한 응용에 보내기 위한 보이는 프래그먼트의 수를 세기 위해 기존의 차폐 선별 유닛에 카운터가 필요하며, 내부 노드인 경우 앞으로의 수행을 막기 위한 비교 유닛이 필요하다. 마지막으로 하드웨어 차폐 쿼리 의 이용처럼 프리미티브당 최소 몇 개의 보이는 프래그먼트가 있을 때만 보이는 것으로 판단하기 위해서는, 해당 장면의 렌더링 전에 응용에서 물체당 최소 프래그먼트 수를 하드웨어로 보내는 것이 필요하다. 이 경우 최소 프래그먼트 수가 크면, 보이는 부분이 작아 잘려나가는 물체의 비율이 높아져 빠르게 수행할 수 있지만, 화면에 그림이 올바르게 그려지지 않을 수 있으므로 이 값를 위해 unsigned 7bit를 할당하여 최소 프래그먼트 수를 최대 128까지 설정할 수 있으면 충분하다고 판단하였다. 앞서 언급한 노드와 프리미티브를 구별하기 위한 1bit는 응용에서 GPU로 보내는 노드마다 필요하지만, 효율을 위한 최소 프래그먼트 수는 초기화 할 때와 응용에서 변화를 줄 때만 선택적으로 보낼 수 있다.



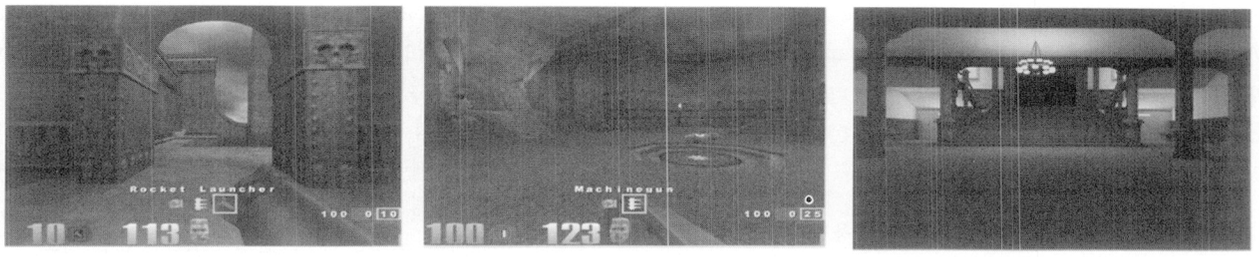
(그림 4) 하드웨어 구조

4. 성능 평가

연구의 성능평가는 세 가지의 벤치마크를 이용하였으며, 벤치마크를 이용한 실험을 통해 얻어진 데이터들을 기반으로 제안한 구조의 하드웨어 성능을 수식을 통해 측정하였다. (그림 5)는 성능평가를 위해 쓰인 벤치마크의 장면들이다. 퀘이크(Quake3)는 전형적인 OpenGL 기반의 비디오 게임으로, 3D 그래픽스 관련 논문의 시뮬레이션에서 자주 등장하는 일반적인 성능평가 벤치마크이다[13]. Lightscape는 SPECviewperf™에서 제공하는 벤치마크로, OpenGL 기반의 3D 렌더링 시스템의 성능 측정을 위한 산업 표준 벤치마크이다[14]. 모든 성능평가는 Mesa Library[15]를 기반으로 한 시뮬레이터 상에서 수행되었으며, 시뮬레이션은1.84GHz, 512RAM, RADEON® 9550, 128 RAM의 그래픽카드, 그리고 해상도 1024X768 환경에서 수행되었다. Mesa Library에 그래픽스 하드웨어에 따라 구동하는 부분이 있기 때문에 그래픽 카드도 환경으로 언급하였다.

4.1 성능 평가 방법

우리는 제안한 구조의 성능을 분석적으로 평가하기 위해 보이는 하나의 프래그먼트의 연산에 필요한 평균 사이클



(그림 5) 벤치마크 퀘이크3 데모1, 퀘이크3 데모2, Lightscape

(Average cycles per visible fragment: ACPF)을 계산하였다. 제안하는 알고리즘과의 성능비교를 위해 기존의 하드웨어 가시성 선별 쿼리를 응용하는 알고리즘인 S&W와 CHC의 ACPF를 구하였다.

$$ACPF_{s\&w} = \gamma \times (T_{geo} + T_{ras}) \div N_{BV} + (1-\gamma) \times ((T_{geolit} + T_{ras}) \div N_{frame} + T_{ras\_tot})$$

$$ACPF_{CHC} = \gamma \times (T_{geo} + T_{ras}) \div N_{BV} \times D + (1-\gamma) \times ((T_{geolit} + T_{ras}) \div N_{frame} \times D + T_{ras\_tot})$$

위의 수식에서 N는 바운딩 볼륨의 정점 수이며, γ는 가시성 검사를 통해 제거되는 비율이고, T_{ξ}는 기하처리에 소요되는 사이클이다. T는 텍스처 매핑 단계를 제외한 래스터 단계에서 소요되는 사이클이며, T_{ras}는 텍스처 매핑 단계를 포함한 래스터 단계에서의 사이클 시간이다. 그리고 $ACPF_{CHC}$ 식의 D는 시공간적 일관성을 이용하여 감소한 쿼리의 비율이다. 위 두 식의 $(T_{geo} + T_{r\xi}$ 부분은 쿼리 모드에서의 연산량을 계산하기 위한 식으로 정점 연산에서는 기하연산만을 수행한 후 래스터화하는 동작을 나타내고 있다. 렌더링 모드에서의 사이클 식은 $(T_{geolit} + T_{ras}) + T_{ras_t\xi}$으로 정점연산, 래스터화, 프래그먼트 연산을 의미한다.

한편, 제안하는 구조의 하드웨어는 VCBP의 하드웨어를 응용하므로 제안하는 구조의 ACPF를 구하기 위해서는 [11]에 나온 다음과 같은 $ACPF_{VCBP}$를 적용해야 한다.

$$ACPF_{VCBP} = (H_{pix} \times \gamma) \div N_{frag} + (H_{pix} \times (1-\gamma) \times (H1_{pix} \times H_{tex} + (1 - H1_{pix}) \times T1_{pix} + (1 - H_{tex}) \times T_{tex} + (1 - H_{pix}) \times ((1 - H_{tex}) \times T_{tex} + T1_{pix} \times (1 - \text{reduction}))$$

여기서 H_{ι}와 τ는 각각 픽셀 캐쉬와 텍스처 캐쉬의 히트 비율이고, N_{fi}는 하나의 블록에 포함되는 프래그먼트의 수 이다. $H1_{\iota}$는 캐쉬 미스가 일어나 완전히 캐쉬 미스 패널티를 받을 확률을 1에서 뺀 값이며, $T1_{\iota}$는 VCBP에서의 픽셀 캐쉬에서의 미스 패널티에 필요한 사이클이다. 그리고 reduction은 중간 텍스처링을 통해 얻는 미스 패널티 감소 비율이다.

마지막으로 위와 같은 VCBP의 ACPF를 이용하여 구한 제안한 구조의 ACPF는 다음과 같다.

$$ACPF_{proposed} = ACPF_{VCBP} + \frac{T_{geolit}}{N_{frame}} + \frac{D \times (ACPF_{VCBP} + T_{geolit}) \times (1 + N_{BV}) \times (N_{frame} - 1)}{N_{BV} \times N_{frame}}$$

$T_{ge\iota}$은 기하단계와 조명처리 단계를 수행하는데 필요한 사이클이다. 위의 식에서 $ACPF_{VCBP} + T_{geolit}/N_fram$부분은 GPU에 들어오는 모든 물체들이 기본적으로 수행하는 부분으로 처음의 기하 단계와 조명 단계와 차폐 선별을 수행하는 부분까지의 연산을 의미한다.

$$\left(D \times (ACPF_{VCBP} + T_{geolit}) \times (1 + N_{BV}) \times (N_{frame} - 1)\right) / (N_{BV} \times N_{frame})$$

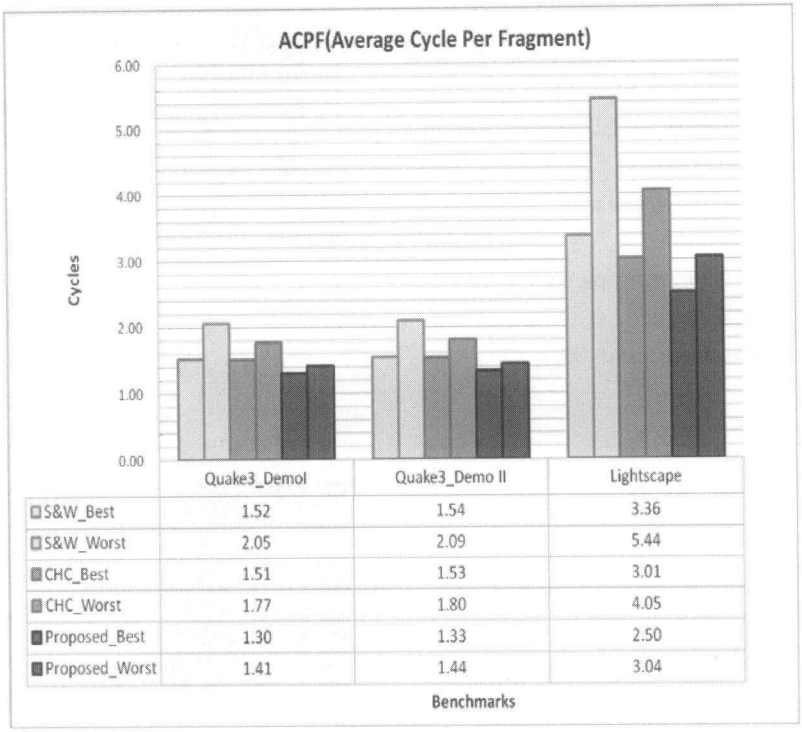
부분은 시공간적 일관성 이용으로 줄어든 검사 비율을 적용한 처리의 연산 수식이다.

실험 결과는 장면을 최고 성능인 경우와 최저 성능인 경우 두 가지로 나누어 각각의 경우를 그림6)과 같이 그래프로 표현하였다. 최고 성능인 경우는 시점과 물체가 가까워 화면을 구성하고 있는 삼각형의 크기가 커서 차폐 검사 기법이 처리하기 좋은 경우이며 최저 성능인 경우는 시점과 물체가 멀어 삼각형의 크기가 작아 차폐 검사에 불리한 경우이다. 최고 성능의 경우에는 그려진 삼각형의 평균 크기가 780 pixel이고, 최저 성능의 경우에는 그려진 삼각형의 평균 크기가 30 pixel이었다. 각각의 경우에 대해 구한 ACPF는 (그림 6)에 그래프로 표현하였다. 가장 옅은 막대는 차폐 쿼리 하드웨어를 사용하는 S&W 알고리즘의 최고 성능의 경우, 최저 성능의 경우이며, 중간의 막대는 차폐 쿼리 하드웨어를 사용하는 CHC 알고리즘의 최고, 최저 성능의 경우를 표현한다. 마지막으로 가장 짙은 막대는 제안하는 방법의 ACPF을 나타낸다. 정확한 ACPF값은 (그림 6) 그래프 아래의 표에 나와있다.

〈표 1〉 성능 향상 백분율

성능 향상(%)	Quake3 Demol	Quake3_Demo II	Lightscape
S&W Best	14%	14%	26%
S&W Worst	31%	31%	44%
CHC Best	13%	13%	17%
CHC Worst	20%	20%	25%

<표 1>은 위에서 구한 ACPF를 기반으로 각각 최상의 성능인 경우와 최저의 성능인 경우로 비교한 성능향상 백분율을 표현한 것이다. 이 결과 분석 표를 보면 제안한 구조는 하드웨어 가시성 선별 쿼리를 이용하는 이중 패스 알고리즘을 사용한 경우와 비교했을 때, S&W와 비교해서 최대 44%, CHC에 비해 최대 25%의 성능 향상을 보인 것을 알 수 있다. 특히 최저의 성능인 경우에 대한 성능 향상이 두드러졌으며, 최상의 성능인 경우에도 CHC대비 최저 13%의 성능향상을 보였다.



(그림 6) 벤치마크 별 ACPF

5. 결 론

우리는 본 논문에서 일반적으로 사용되는 기존의 하드웨어 가시성 선별 쿼리를 이용한 이중 패스 알고리즘을 단일 패스 알고리즘으로 만들어 동일한 효과를 좀더 효율적으로 얻을 수 있는 구조를 제안하였다. 응용단계에서는 하드웨어에서 받은 처리 결과를 기반으로 공간 계층 트리 구조와 시간적공간적 일관성을 사용하여 GPU로의 데이터량을 줄여 불필요한 연산을 최소화 하였다. 그래픽스 하드웨어 부분에서는 단일 패스를 가능케 하기 위하여 기존의 VCBP 하드웨어를 응용한 구조를 응용하면서 바운딩 볼륨의 가시성 검사도 가능하게 하였으며 하드웨어로 받은 트리 구조의 노드의 가시성 선별 결과를 응용 단계에서 알 수 있도록 하였다. 이와 같은 구조는 ACPF를 기준으로 측정하였을 때, 기존 이중 패스 알고리즘 중 S&W 대비 최대 44%, 최저 14%의 성능이 향상되었고, CHC 대비 최대 25%, 최저 17%의 성능이 향상되었다.

참 고 문 헌

[1] T. Akenine-Molloer, E. Haines, 리얼-타임 렌더링, 2판, pp.415-436, 2003.

[2] D. Cohen-Or, Y. Chrysanthou, C. Silva, "A Survey of Visibility for Walking through Applications," IEEE Trans. Visualization and Computer Graphics, Vol.9, No.3, pp. 412-431, Jul.-Sept., 2003.

[3] N. Greene, "Interactive Geometric Computing Using Graphics Hardware-Visibility Culling Using Graphics Hardware," Proc. SIGGRAPH '02 Tutorial Course #31. Jul., 2002.

[4] N. Greene, M. Kass, G. Miller, "Hierarchical Z-buffer visibility." Computer Graphics (Proc. SIGGRAPH '93), pp.231-238, 1993

[5] S. Morein, "ATI Radeon Hyper-Z Technology," proc. Hot3D, Graphics Hardware Workshop, 2000.

[6] H. Zhang, D. Manocha, T. Hudson, and K.K. Hoff III, "Visibility Culling Using Hierarchical Occlusion Maps," Computer Graphics (proc. SIGGRAPH '97), pp.77-88, 1997.

[7] J. Bittner, V. Haran, "Exploiting Coherence in Hierarchical Visibility Algorithms," Visualization and Computer Animation, pp.277-286, 2001.

[8] D. Bartz, M. Meiner, T. Hutter, "Extending Graphics hardware for Occlusion Queries in OpenGL," proc. Working Graphics Hardware '98, pp.97-104, 1998.

[9] D. Bartz, M.ael Meiner, T. Hutter, "OpenGL_assisted Occlusion Culling for Large Polygonal Models," Computer & Graphics, Vol.23, No. 5, pp.667-679, 1999.

[10] J. Bittner, M. Wimmer, H. Pringer, W. Purgathofer, "Coherent Hierarchical Occlusion Culling: Hardware Occlusion Queries Made Useful," proc. EUROGRAPHICS'04, Vol.23, No.3, 2004.

[11] M. H. Choi, W. C. Park, F. Neelamkavil, Tag-Don Han, Shin-Dug Kim, "An Effective Visibility Culling Method Based on Cache Block," IEEE Transactions on Computers,

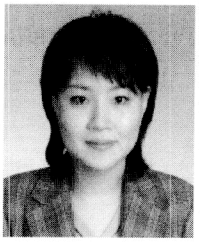
Vol. 55, No.8, 2006.

[12] J. McCormack, R. McNamara, C. Gianos, et al., "Neon: A (Big)(Fast) SingleChip 3D Workstation Graphics Accelerator," Research Report 98/1, Western Research Laboratory, Compaq Corp., 1998.

[13] Quake3, http://www.idsoftare.com/games/quake/quake3-arena, 2006.

[14] Specviewperf (Lightscape), http://www.spec.org/gpc/opc.static/viewperf711info.html, 2006.

[15] Mesa Library, http://www.mesa3d.org, 2006.



최 문 희

e-mail : mhchoi@yonsei.ac.kr

1999년 숙명여자대학교 컴퓨터공학과(학사)

2001년 연세대학교 대학원 컴퓨터과학과 (석사)

2007년 연세대학교 대학원 컴퓨터과학과 (박사)

2007년~현 재 삼성전자 TN 선임 연구원

관심분야 : 3D 렌더링 구조, 미디어 처리 시스템, 저전력 임베디드 시스템 등

주 지 원

e-mail : estl@yonsei.ac.kr

2006년 연세대학교 컴퓨터과학과(학사)

2008년 연세대학교 대학원 컴퓨터과학과 (석사)

2008년~현 재 삼성전자 DM 연구원

관심분야 : 3D 그래픽스 가속 구조, 3D 그래픽스 가속 알고리즘 등

김 신 덕

e-mail : sdkim@yonsei.ac.kr

1981년 연세대학교 전자공학과(학사)

1987년 University of Oklahoma 전기컴퓨터공학과(석사)

1991년 Purdue University 전기컴퓨터공학과(박사)

1995년~현 재 연세대학교 컴퓨터과학과 교수

관심분야 : 고성능 컴퓨터 구조, 지능형 캐쉬 메모리, 병렬처리 시스템, 그리드 컴퓨팅 등