한국게임학회 논문지
제7권 제3호 2007년 9월

GPU 기반의 실시간 인터렉티브 광선추적법 구현

배성민[0], 홍현기
중앙대학교 첨단영상대학원 첨단영상학과
misslink@hanmail.net[0], honghk@cau.ac.kr

Implementation of Real-time Interactive Ray Tracing on GPU

Sungmin Bae[0], Hyunki Hong
Dept. of Image, GSAIM Chung-Ang Univ.

요약

광선추적법(ray tracing)은 빛의 반사, 투과 등을 사실적으로 표현할 수 있는 대표적인 전역조명(global illumination) 기술이지만, 복잡한 계산과정으로 인해 실시간 활용에는 많은 제약이 존재한다. 이런 문제를 해결하기 위해 최근에는 GPU(Graphics Processing Unit) 기반의 광선추적법 알고리즘이 활발하게 개발되고 있으며, 본 논문에서는 J. Purcell 등이 제안한 광선추적법 기법을 구현하였다. 그리고 구현된 알고리즘을 인터렉티브 응용분야에 활용하기 위해 렌더링 성능을 개선하는 두가지 방법을 적용하였다. 먼저, 그래픽스 하드웨어에서 지원하는 래스터라이제이션(rasterization)을 적용해 초기 광선의 교차점을 효과적으로 구했다. 또한 대상 물체를 가속화(acceleration) 구조로 구성하여 광선과 물체간의 교차연산에 소요되는 계산시간을 단축하였다. GPU 기반의 광선추적법 렌더링에서 다양한 성능 개선 알고리즘을 적용하여 향상된 렌더링 결과를 구체적으로 분석한 기존 연구가 비교적 적었으며, 본 논문에서는 각 과정에 따른 개선 결과를 제시하였다. 구현된 렌더러와 GPU 기반의 환경 맵을 비교하였으며 이동형 개인 컴퓨터와 무선 센싱 장비를 이용한 무선 원격 렌더링 시스템을 구현하였다. 제안된 시스템은 실시간 합성, 증강현실(augmented reality), 가상현실 등의 다양한 분야에서 활용될 것으로 기대된다.

색인어 : 광선추적법, 실시간 렌더링, 전역조명, GPU

Abstract.

Ray tracing is one of the classical global illumination methods to generate a photo-realistic rendering image with various lighting effects such as reflection and refraction. However, there are some restrictions on real-time applications because of its computation load. In order to overcome these limitations, many researches of the ray tracing based on GPU (Graphics Processing Unit) have been presented up to now. In this paper, we implement the ray tracing algorithm by J. Purcell and combine it with two methods in order to improve the rendering performance for interactive applications. First, intersection points of the primary ray are determined efficiently using rasterization on graphics hardware. We then construct the acceleration structure of 3D objects to improve the rendering performance. There are few researches on a detail analysis of improved performance by these considerations in ray tracing rendering. We compare the rendering system with environment mapping based on GPU and implement the wireless remote rendering system. This system is useful for interactive applications such as the realtime composition, augmented reality and virtual reality.

key-words: ray tracing, real-time rendering, global illumination, GPU

* 본 연구는 서울시 산학협력사업으로 구축된 서울 미래형콘텐츠컨버전스클러스터, 문화콘텐츠진흥원의 CT연구소 지원사업으로 수행되었습니다.

1. 서론

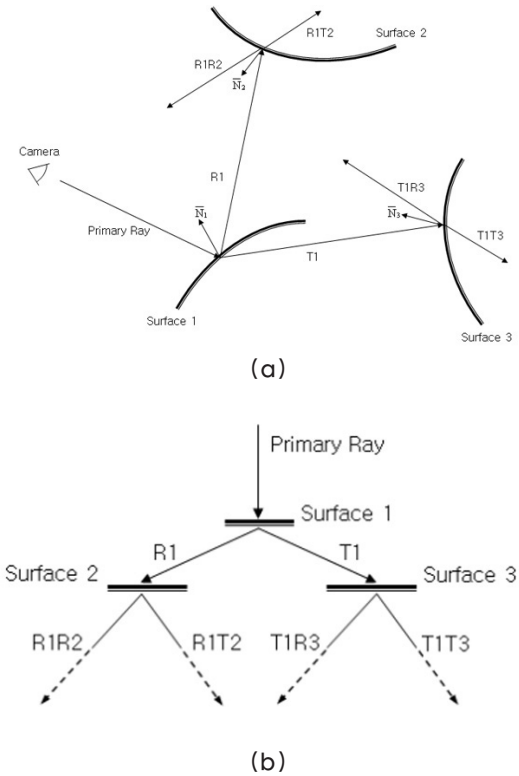
컴퓨터 그래픽스 분야에서 사실적인 영상의 생성은 중요한 연구 분야 중 하나이다. 전역조명모델(global illumination)은 물체들 간의 관계에서 빛의 상호작용을 모델링하며, 광선추적법(ray tracing)은 가장 기본적인 알고리즘이다[1]. 광선추적법은 시점으로부터 3차원 공간상으로 방출되는 광선을 추적하기 때문에 사실적인 렌더링 영상을 표현할 수 있는 장점이 있지만, 계산량이 많다는 단점을 지니고 있다. 최근 컴퓨터 산업이 비약적으로 발전함에 따라 그래픽 계산을 위한 전용 프로세서(GPU: Graphics Processing Unit)가 개발되었으며, 이를 활용한 실시간 광선추적법 연구가 활발히 진행되고 있다[2 ~ 4].

본 논문에서는 J. Purcell 등이 제안한 그래픽스 하드웨어를 이용한 광선추적법[2]을 구현하였으며, 렌더링 성능을 개선하기 위해 래스터라이제이션(rasterization)과 대상 물체들에 대한 가속화 구조(acceleration structure) 기법을 적용하였다. 먼저, 카메라의 시점에서 바라본 장면의 래스터라이제이션 기능을 활용하여 초기(primary) 광선과 물체 표면 간의 교차(intersection) 연산을 효과적으로 처리하였다. 기존 광선의 경로 추적과정에서는 3차원 공간의 물체들을 모두 검색(search)하기 때문에 많은 계산이 필요하며, 이를 줄이기 위해 물체들을 가속화 구조로 구성하였다[5]. 또한 효율적인 렌더링을 위해 추가적으로 적용한 방법을 통해 얻어진 시스템의 성능을 시뮬레이션을 통해 비교 및 분석하였으며, 무선 원격 렌더링 시스템을 구현하여 실시간 응용분야[13,14]에서 다양하게 활용될 수 있음을 확인하였다.

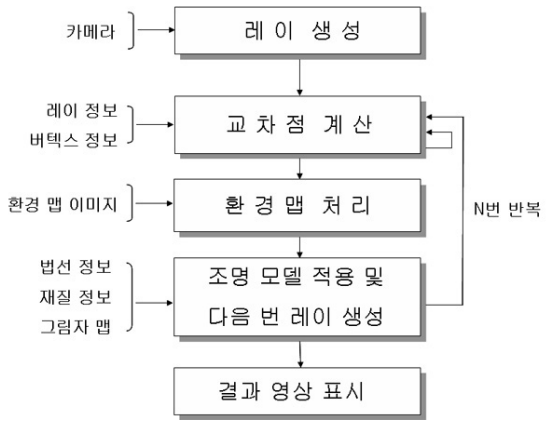
본 논문의 구성은, GPU 기반의 광선추적법을 2장에서 소개하며, 광선추적법의 실제 구현 및 가속화를 위해 적용된 방법을 3장에서 기술한다. 그리고 4장에서는 구현된 기존 알고리즘과 가속화 방법에 의해 개선된 렌더링 성능을 분석하고, GPU 기반의 환경매핑(environment mapping)과의 비교 및 구현된 무선 원격 렌더링 시스템 등을 기술하였으며, 마지막으로 5장에서 결론 및 향후 연구방향을 제시한다.

2. GPU기반 광선추적법

광선추적법 방법은 카메라의 시점으로부터 광선을 방출하며, 그림 1 (a)와 같이 하나의 광선은 물체표면의 특성에 따라 반사(R) 및 굴절(T) 광선으로 구분되고 각각의 광선은 같은 방법으로 계속 분기된다. 그림 1 (b)에서 하나의 광선에 대한 밝기값을 결정하기 위해서는 광선 트리의 아래에서부터 위쪽으로 재귀적으로 값을 처리해야 한다[1]. 그러나 연속된 데이터의 배열을 병렬로 처리하도록 설계된 GPU 등의 스트림 프로세서(stream processor)에서는 기존 광선추적법 연산의 재귀적 구조가 적합하지 않다. 따라서 그래픽스 하드웨어의 특성을 최대한 활용하기 위해서는 스트림 프로그래밍 방법을 사용해야 하며, 계산과 관련된 명령어들을 모아서 커널(kernel)이라는 집합을 만들고 입력과 출력으로 스트림 데이터를 사용한다.



[그림 1] 재귀적 광선추적법 방법 (a) 물체와 반사 및 투과 광선 (b) 광선의 트리구조.



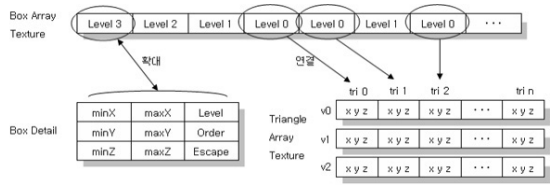
[그림 2] 전체 알고리즘 구조.

기존 광선추적법의 전체 구조를 주요 모듈별로 구성하여 커널을 만들고, 그림 2와 같이 각 커널의 결과가 다음 커널에 입력되도록 설계한다. 그리고 커널의 입력 및 출력의 형태는 그래픽 하드웨어에서 사용할 수 있는 텍스처 형태로 전환하여 사용한다. 광선의 생성 커널에서는 데이터 입력과 초기 광선의 위치 및 방향 등을 사전에 생성하며, 교차점 계산 커널은 주어진 광선과 물체간의 교차점을 계산하여 다음 커널에 전달한다. 환경맵 처리 커널은 3차원 공간상의 물체표면과 교차되지 않고 무한으로 진행하는 광선을 처리하며, 조명모델 커널은 조명 모델을 적용시켜 해당 교차점에서의 색상을 결정한다. 그리고 그 교차점에서 다음 광선의 방향을 계산하고 광선의 반사 반복횟수만큼 처리한 다음, 마지막 커널에서 결과 렌더링 영상을 디스플레이한다.

3. GPU 기반의 광선추적법 구현

3.1 자료구조 및 교차점 계산

광선추적법 알고리즘을 그래픽 하드웨어에서 구현하기 위해서는 모든 데이터를 텍스처 형태로 전환한다. 렌더링의 기본 단위인 삼각형(triangle)의 정점들에 대한 정보를 텍스처로 만들고, 처리속도를 효율적으로 개선하기 위해 박스(box)를 기본으로 하는 BVH(Bounding Volume Hierarchies)의 트리 배열 텍스처로 재구성한다. 그림 3에서 박스는 기본 속성 외 레벨(level), 오더(order)와 탈출(escape) 인덱스(index)를 가지며, 레벨은 트리에서의 높이를 의미한다. 즉, 레벨 0은 트리 구성에서 가장 끝단을 의미하는 잎(leaf) 박스를 나타내며, 레벨 0인 박스 당 하나의 삼각형이 연결된다. 오더는 해당 레벨에서의 순서를 나타내고, 탈출 인덱스는 광선과 박스 표면과의 교차점이 존재하지 않을 때 다음 교차 테스트에 해당하는 박스의 위치를 나타낸다.

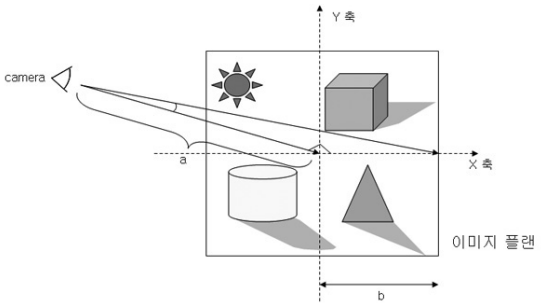


[그림 3] 주요 데이터들의 자료구조.

광선과 물체 표면을 구성하는 삼각형과의 교차점을 구하는 과정이 기존 광선추적법에서 가장 많은 계산시간이 소요된다. 따라서 광선과 삼각형의 교차점을 구하는 알고리즘의 선택은 렌더링 성능에 많은 영향을 미친다. 본 논문에서는 그래픽스 하드웨어의 벡터 연산의 장점을 이용하는 Moller 등의 교차점 알고리즘[5]을 구현하였다. 또한 교차점 계산 과정에서 발생하는 무게중심 좌표(barycentric coordinate)의 u, v 파라미터를 이용하여 퐁(Phong) 쉐이딩의 접선 보간법을 처리하였다.

3.2 가속화 알고리즘

광선추적법에서 계산량의 비중이 높은 부분을 나누어 보면(2 패스일 경우) 크게 네 개 부분으로 나눌 수 있다. 초기 광선의 교차점을 찾는 부분, 해당 교차점에서 그림자 필러(feeler)를 방출하여 그림자에 포함되는지를 판별하는 부분, 두 번째 광선의 교차점을 찾는 부분, 그 교차점에서 그림자 필러를 방출하여 그림자에 포함되는지를 판별하는 부분으로 구분된다. 본 논문에서는 초기 광선의 교차점을 찾는 부분에 대해 그래픽 파이프 라인의 레스터라이제이션 방법을, 두 번째 광선의 교차점을 찾는 부분에 대해서는 3차원 공간 및 물체의 가속화 구조를 구성하여 이용하는 방법을, 그림자 필러에 대해서는 그림자 매핑[7] 방법을 각각 적용하였다.



[그림 4] 레스터라이제이션의 사영 각도.

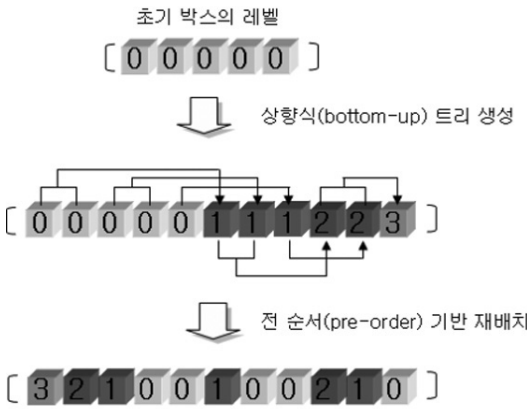
3.2.1 래스터라이제이션

광선추적법에서는 3차원 공간상에서 광선과 물체 표면과의 교차점을 찾기 위해 많은 연산을 수행해야 한다. 초기 광선에 대한 교차점 테스트를 효율적으로 처리하기 위해 본 논문에서는 OpenGL 그래픽 파이프 라인의 래스터라이제이션 방법[4]을 사용한다. 기존 광선추적법에서의 초기 교차점과 래스터라이제이션을 통해 얻은 결과가 서로 일치하도록 하기 위해 사영(projection) 설정을 다음과 같이 조정한다. 즉, 기존 광선추적법에서 대상 장면과 카메라 및 이미지 평면(image plane)으로의 투시 사영(perspective projection) 관계로부터 레스터라이제이션의 사영 각도를 식 (1)와 같이 계산한다. 그리고 교차점과 함께 조명 모델에 적용시키기 위한 대상 물체의 구성정보도 구한다.

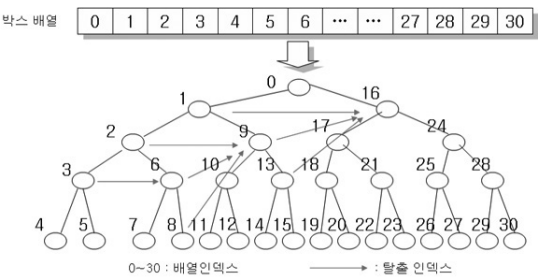
$$\text{사영각도} = 2 \times \tan^{-1}(b/a) \qquad (1)$$

3.2.2 가속화 구조 및 그림자 매핑

기존 광선추적법에서는 3차원 공간상에서 광선과 물체와의 교차연산에 가장 많은 시간이 소요되며, 이를 개선하기 위해 Grid, KD-Trees, BVH 등의 가속화 구조 연구가 진행되었다. 그래픽 하드웨어에서 다양한 가속화 구조의 성능을 비교한 참고문헌[5]에서 BVH 알고리즘이 구조적으로 간단하고 가장 뛰어난 성능을 보였다. 따라서 본 논문에서는 그래픽 하드웨어 상에서 가장 적합한 BVH 구조로 대상 공간 및 물체 데이터를 구성하였다.



[그림 5] 1차원 BVH 구조 배열.

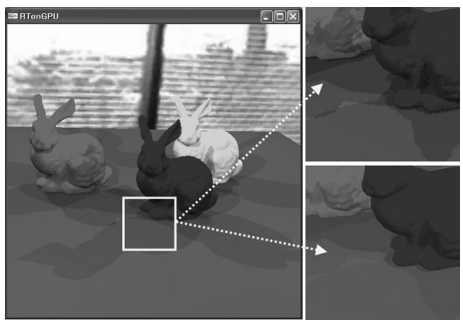


[그림 6] BVH 구조 배열와 탈출 인덱스.

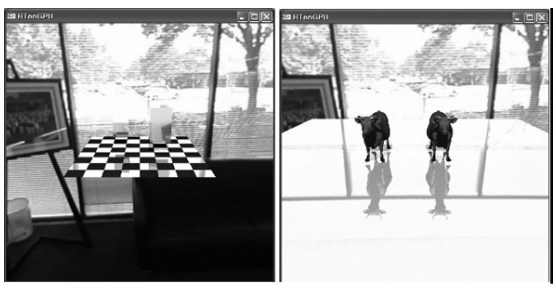
데이터 구조를 BVH로 구성하기 위해 먼저 1차원 배열에 순차적으로 저장된 정점(vertex) 정보를 완전트리(complete tree) 형태로 변환한다. 그림 5와 같이 초기 정점 정보들을 사용하여 상향식 방법(bottom-up)으로 트리 구조를 구성한 다음, 텍스처를 처음부터 다시 검색하기 위해 전 순서(pre-order) 기반으로 이를 재배치한다. 각 상위 레벨 노드들은 하위 레벨 노드들의 경계(boundary)를 포함하는 범위를 지닌다. 즉, 상위 노드가 하위 노드의 범위를 포함하기 때문에 하나의 노드에서 교차점을 테스트해 교차점이 없는 경우, 하위 노드들 역시 교차점이 없음을 의미한다. 따라서 하위 노드들에서 교차점을 테스트하는 대신에 다음 노드로 이동하며, 여기서 탈출 인덱스는 다음에 고려될 노드를 지시한다. 그림 6와 같이 탈출 인덱스는 미 교차점으로 고려된 노드에서 배열을 기준으로 왼쪽으로 한 칸 이동한 상태에서 대상 노드의 포화(full) 트리만큼 합한 값이 되며, 이 관계를 식 (2)에 나타내었다.

탈출 인덱스 = (대상 노드 인덱스 -1) + 2 << (대상 노드 레벨 + 1) (2)

고려되는 지점과 광원 사이에 물체가 존재하면 그림자가 일반적으로 발생하며, 따라서 렌더링에서 그림자의 판별은 해당 위치와 광원 사이에 가려지는 물체의 유무에 따라 결정된다. 그림자 매핑의 기본 원리는 광원에서 바라보는 장면인 그림자 맵을 생성하여 렌더링하려는 위치와 그림자 맵에 저장된 거리 값을 비교하여 판별한다. 그러나 그림자 맵이 불연속적인 텍스처 값으로 저장되어 렌더링 결과 영상에 계단 현상이 발생하며, 박스 필터를 적용하여 이러한 에일리어싱(aliasing) 문제를 그림 7과 같이 해결하였다.



[그림 7] 박스 필터 적용 전(오른쪽 위)과 박스 필터 적용 후(오른쪽 아래) 비교.



(a) (b)

(c) (d)

[그림 10] GPU기반 환경맵과 광선추적법의 렌더링 영상 비교. (a) 기존 환경 매핑 (b) 상호반사 환경 매핑 (c) GPU 기반 광선추적법.

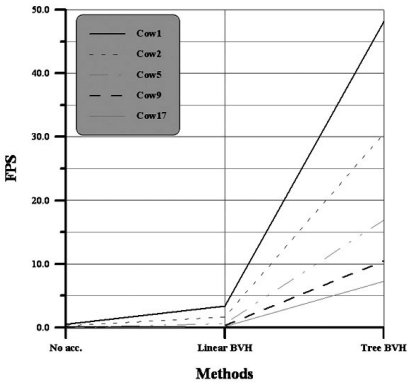
4. 실시간 렌더링 시스템의 구현

4.1 개선된 렌더링 성능 분석

본 논문에서는 NVIDIA FX8800 그래픽 하드웨어를 장착한 Intel CPU 3.2 GHz, 4GB RAM의 개인용 컴퓨터상에서 시뮬레이션을 진행하였다. 결과 렌더링 영상의 크기는 512×512 픽셀이며 각 알고리즘의 렌더링 속도를 비교하기 위해 fps(frame per second) 단위를 사용하였다. 그림 8은 렌더링된 결과 영상이며, 각 방법에 의한 렌더링 성능을 표 1과 그림 9에 나타내었다.

장면 (삼각형 수)	No Acceleration	선형 BVH (박스 크기: 8)	트리 BVH
Box (153)	174.652	24.315	3.407
Cow1 (5837)	12937.57	1615.989	49.053
Cow2 (11641)	26524.78	3211.53	68.417
Cow5 (29053)	Overload	8007.67	138.311
Cow9 (52269)	Overload	14018.847	238.444
Cow17 (98701)	Overload	20760.26	361.564

[표 1] 가속화 구조를 이용한 히트 수의 비교 (2 패스).



[그림 9] 가속화 구조를 이용한 렌더링 성능 비교 (2 패스).

장면 (삼각형 수)		트리 BVH (1 패스)	래스터라이제이션 + 트리 BVH (1 패스)	트리 BVH (2 패스)	래스터라이제이션 + 트리 BVH (2 패스)
Box (153)	FPS	443.34	646.73	268.77	339.27
	Hit numbers	3.056	0.541	3.407	1.044
Cow1 (5837)	FPS	85.71	153.54	48.22	16.89
	Hit numbers	35.055	12.846	49.053	26.845
Cow2 (11641)	FPS	51.23	85.62	30.54	41.66
	Hit numbers	47.079	18.660	68.417	39.998
Cow5 (29053)	FPS	32.48	37.43	16.93	19.10
	Hit numbers	87.545	36.332	138.311	87.099
Cow9 (52269)	FPS	18.42	22.63	10.50	12.42
	Hit numbers	160.157	69.270	238.444	147.558
Cow17 (98701)	FPS	12.24	14.53	7.31	8.59
	Hit numbers	248.820	99.822	361.564	212.567

[표 2] 래스터라이제이션를 이용한 렌더링 성능 비교 (1, 2 패스)

표 1에서 가속화 방법을 사용하지 않은 경우, 장면의 모든 삼각형에 대해 교차 검사를 하기 때문에 삼각형의 개수에 따라 렌더링 성능이 선형적으로 비례하여 감소하며, 세 번째 장면인 Cow3의 경우, 그래픽 하드웨어에서 처리할 수 있는 명령어의 개수를 초과하여 렌더링을 진행할 수 없었다. 박스 하나당 8개의 삼각형을 포함하는 선형적 BVH는 상대적인 성능향상을 보여 실시간 렌더링은 불가능하지만, BVH 트리 구조를 사용한 경우는 대상 장면의 복잡도에 따라 실시간 렌더링이 가능함을 확인하였다. 광선추적법에서 패스는 하나의 광선이 출발하여 표면에 부딪칠 때까지를 의미한다. 즉, 1패스는 첫 번째 교차지점까지를 나타내며 2패스는 초기 광선으로부터 두 번째 교차점까지를 각각 나타낸다. 본 실험에서 각 광선당 패스는 2로 설정하였으며, 초기 광선 + 2 × 쉐도우 광선 + 반사 또는 굴절 광선으로 각각 구성하였다. 표 1에서 Hit numbers는 각 광선당 교차 테스트의 횟수를 의미한다.

표 2에서는 BVH 구조와 래스터라이제이션 과정을 결합한 시스템의 렌더링 성능을 비교하였으며, 정확한 분석을 위해 1, 2개의 패스에 대해 각각 실험을 하였다. 한 개의 패스는 초기 광선과 새도우 광선을 고려한 경우이다. 표 2에서 패스가 증가함에 따라 렌더링 성능향상의 비율이 변화함을 확인하였다. 즉, 한 번의 패스에서는 래스터라이제이션을 사용했을 때와 사용하지 않았을 때 약 1.6배의 성능향상을 보였지만, 두 번의 패스에서는 1.2배 정도의 성능향상을 보였다. 따라서 래스터라이제이션 방법은 초기 광선의 교차검사에서 속도 향상을 얻을 수 있기 때문에 고려하는 광선의 수가 많아지면 초기 광선의 비중이 줄어들어 상대적인 성능향상의 비율이 감소함을 확인하였다.

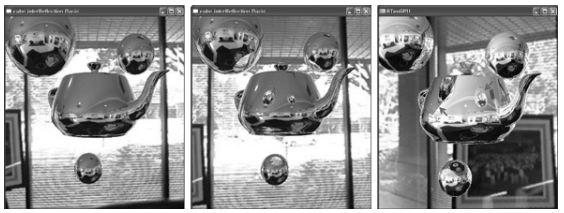
4.2 환경 매핑과 영상기반 라이팅(image-based lighting)

환경 매핑은 텍스쳐 매핑의 형태로 주변 환경의 반사를 근사화하여 표현하는 고전적 렌더링 방법[9]으로 광선추적법에 비해 물체의 반사현상 등을 매우 빠르게 렌더링할 수 있지만, 광선의 정확한 반사, 물체 간 상호반사(inter-reflection) 및 자기 반사(self-reflection) 등을 표현하기 어렵다. 또한 한 시점에서 취득된 환경맵을 이용하기 때문에 기준점을 중심으로 입사각도에 따라 텍스처 매핑되기 때문에 대상 물체가 이동할 때 시차 문제(parallax) 등이 발생한다. 따라서 이를 해결하기 위해 많은 연구[9 ~ 12]가 진행되고 있지만, 추가적인 고려와 특정 조건하에서의 적용 등과 같은 다양한 제한이 존재한다.

본 논문에서 구현된 GPU기반의 광선추적법 알고리즘은 짧은 렌더링 시간 내에 실제 광선의 반사, 투과 등을 표현할 수 있기 때문에 기존의 환경 매핑이 가진 단점들을 보완할 수 있다. 먼저, 광선 탐색과정의 속도를 높이기 위해 3차원 공간과 물체를 BVH로 구성하였으며, 최상위 단계의 박스는 전체 장면을 포함하는 공간이 되므로 전체의 환경 공간은 그 크기의 이상이 되도록 설정한다. 그리고 시차 문제를 해결하기 위해 환경 매핑 방법 대신에 광선과 환경 공간 큐브와의 직접적인 교차점을 구하였으며, 그림 2의 환경맵 처리 커널에서 이를 처리한다.

광선 추적과정에서 발생할 수 있는 예외상황을 다음과 같이 세 종류로 나누어 처리한다. 광선과 물체의 교차가 발생하지 않은 경우는 광선 속성을 -1로 설정하며, 구성된 환경맵 커널에서 환경 공간 큐브와의 교차점을 구하여 해당 위치의 색상을 가져온다. -1의 속성을 가진 광선에 대해 환경맵 처리까지 한 광선을 -2로 설정하여 처리 완료를 나타내며, -3은 광선과 교차된 물체 표면의 속성이 디퓨즈 성분이 강하기 때문에 더 이상 반사, 굴절 등이 계산될 필요가 없는 광선을 나타낸다. 할당된 속성에 따라 각 광선의 이후 처리 과정이 진행된다.

GPU기반의 환경 매핑으로 대상 물체들의 반사현상을 렌더링한 결과를 그림 10 (a)에 나타내었다. 표 3의 결과에서 환경 매핑은 높은 렌더링 성능을 보이지만, 물체간의 반사, 물체 자신의 자기반사, 그림자 등을 정확하게 표현하고 시차 문제를 해결하기 위해서는 추가로 많은 보완이 요구된다. 그림 10 (b)는 물체간의 상호 반사 등을 표현하기 위해 다중 맵을 사용하기 때문에 렌더링 속도가 크게 감소하였다. 본 논문에서 구현한 광선추적법으로 얻어진 렌더링 영상을 그림 10 (c)에 나타내었으며, 실시간 활용 등에 적용될 수 있는 정도의 렌더링 속도 내에서 대상 물체들의 상호 및 자기 반사 등을 정확하게 표현됨을 확인하였다. 본 실험에서 장면의 삼각형 개수는 25,254개이며 광선추적법에서 정확한 상호 반사, 자기반사 및 그림자를 표현하기 위해 다섯 번의 추적경로를 적용하여 실험하였다.



(a) (b) (c)

[그림 10] GPU기반 환경맵과 광선추적법의 렌더링 영상비교. (a) 기존 환경 매핑 (b) 상호반사 환경매핑 (c) GPU 기반 광선추적법.

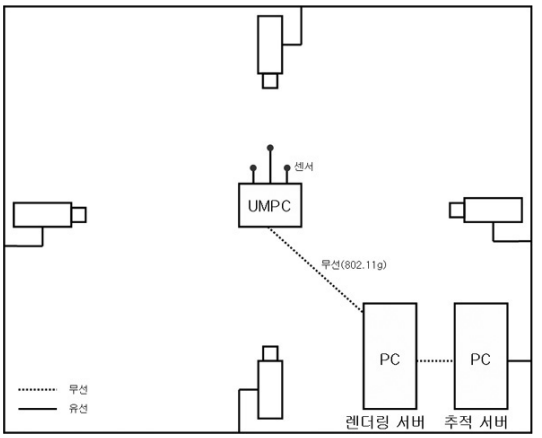
	환경 매핑 (기본)	상호반사 환경 매핑	구현된 GPU 기반 광선추적법
FPS	101.74	3.46	12.79
표현되는 효과	근사된 반사	상호반사	상호반사, 자기반사, 그림자

[표 3] 구현된 광선추적법과 환경 매핑과의 비교.

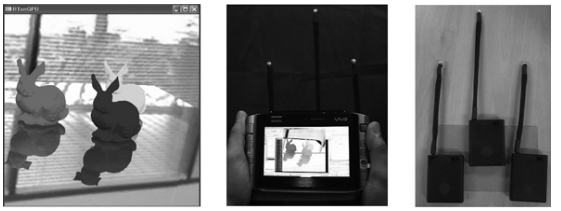
4.3 무선 원격 렌더링 시스템

급격하게 발전하는 컴퓨팅 기술을 바탕으로 렌더링 분야는 컴퓨터비전, 증강현실 등의 여러 분야와의 접목이 활발히 진행되고 있다. 본 논문에서는 구성된 렌더링 시스템을 기반으로 이동형(mobile) 단말기를 이용한 무선 원격 렌더링 시스템을 그림 11와 같이 구성하였다. 구현된 무선 원격 렌더링 시스템에서는 사용자는 UMPC(Ultra Mobile PC)등의 이동형 단말장치를 사용하여 위치 및 시선방향에 따라 적합하게 합성된 렌더링 영상을 인터렉티브하게 렌더링 서버로부터 서비스 받을 수 있다.본 시스템은, GPU기반의 광선추적법을 구현한 렌더링 서버, 단말기의 위치를 실시간에 추적하기 위한 추적 서버, 그리고 디스플레이를 위한 이동형 단말기로 구성된다. 개인용 컴퓨터를 렌더링 서버로, 소니사의 UMPC VGN-UX27LN을 이동형 단말기로 사용했으며, 단말기를 실시간에 추적하기 위해 Worldviz사의 PPT(Precision Position Tracker)를 사용하였다. 그리고 동시에 3개의 센서(그림 11 (c))를 이용, 각 센서가 이루고 있는 평면의 3차원 법선 벡터의 움직임으로 측정하여 단말기의 시선각도를 측정하였다.

렌더링 서버에서 렌더링된 영상과 이동 단말기에 디스플레이되는 장면을 각각 그림 12 (a)와 (b)에 나타내었다. (a)는 55,000개의 삼각형에 2 패스, 네 개의 광원을 적용시킨 결과영상이며, 렌더링 서버에서 512×512 해상도로 렌더링 속도는 11fps이다. 또한 구현된 시스템은 대상 물체가 비교적 단순한 장면에서 30fps 이상의 실시간 렌더링 성능을 보임을 표 1에서 확인하였다. 그러나 본 시스템에서 이용된 무선 네트웍 방식은 802.11g로써 실제 전송되는 최대 용량은 20~30Mbps이기 때문에 현재 512×512, 256×256 해상도의 결과 렌더링 영상의 전송율은 약 3.8fps 및 15.2fps이다. 그러나 무선 네트웍 방식의 새로운 표준인 802.11n을 사용하면 최대 108~320Mbps까지 지원 가능하며, 실제 서버에서의 렌더링 영상이 지연 없이 실시간으로 이동형 단말기에 전송될 수 있다.



[그림 11] 이동형 단말기를 이용한 무선 원격 렌더링 시스템.



(a) (b) (c)

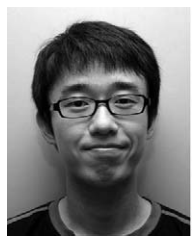
[그림 12] (a) 렌더링 영상 (b) 이동형 단말기에 전송된 영상(512×512) (c) 3개의 추적용 센서.

5. 결론

본 논문에서는 GPU 기반의 광선추적법 알고리즘을 바탕으로 초기 광선의 교차점 계산과정에서 래스터라이제이션을 적용하고 대상 물체의 교차점 검색과정에서 가속화 구조를 구현하였으며, 향상된 렌더링 성능을 비교 및 분석하였다. 또한 환경 매핑과의 비교 및 이동형 단말기를 이용한 무선 원격 렌더링 시스템 구현을 통해 실시간 렌더링 응용분야에 GPU기반의 광선추적법 방법이 활용될 수 있음을 확인하였다. 이후로는 그래픽 하드웨어의 특성을 이용해 좀 더 사실적인 렌더링 영상을 생성하는 방법을 제안하고 대화식(interactive) 실시간 렌더링 시스템의 개발에 활용할 예정이다.

참고문헌

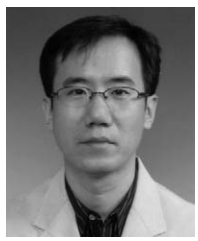
[1] T. Whitted, "An Improved Illumination Model for Shaded Display," Communications of ACM, vol. 23, no. 6, pp. 343-349, 1980.
[2] T. J. Purcell, I. Buck, W. R. Mark, and P. Hanrahan, "Ray Tracing on Programmable Graphics Hardware," ACM Trans. on Graphics, vol. 21, no. 3, pp. 703-712, 2002.
[3] N. Thrane and L. O. Simonsen, "A Comparison of Acceleration Structures for GPU Assisted Ray Tracing," Master's thesis, Univ. of Aarhus, Denmark, August 2005.
[4] I. Wald, T. J. Purcell, J. Schmittler, C. Benthin, and P. Slusallek, "Realtime Ray Tracing and its use for Interactive Global Illumination," Eurographics State of the Art Reports, 2003.
[5] E. Reinhard, B. Smits, and C. Hansen, "Dynamic Acceleration Structures for Interactive Ray Tracing," Proc. of Eurographics Workshop on Rendering, vol. 11, pp. 299-306, 2000.
[6] W. R. Mark, R. S. Glanvile, K. Akeley, and M, J, Kilgard, "Cg: A System for Programming Graphics Hardware in a C-like Language", ACM Trans. on Graphics, vol. 22, no. 3, pp. 896-907, 2003.
[7] R. Ferando and M. J. Kilqard, The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics, nVIDIA, 2003.
[8] R. Ferando, GPUGems: Programming Techniques, Tips, and Tricks for Real-time Graphics, nVIDIA, 2004
[9] K. H. Nielsen and N. J. Christensen, "Real-Time Recursive Specular Reflections on Planar and Curved Surfaces using Graphics Hardware," Journal of WSCG, vol. 10, no. 3, pp. 91-98, 2002.
[10] D. Roger and N. Holzschuch, "Accurate Specular Reflections in Real-Time," Computer Graphics Forum, vol. 25, no. 3, pp. 293-302, 2006.
[11] V. Popescu, C. Mei, J. Dauble, and E. Sacks, "Reflected-Scene Impostors for Realistic Reflections at Interactive Rates", Computer Graphics Forum, vol. 25, no. 3, pp. 313-322, 2006.
[12] Z. S. Hakura, J. M. Snyder, and J. E. Lengyel, "Parameterized Environment Maps," Proc. of Symposium on Interactive 3D Graphics, pp. 203-208, 2001.
[13] A. Pomi, G. Marmitt, I. Wald, and P. Slusalek, "Streaming Video Textures for Mixed Reality Applications in Interactive Ray Tracing Environments," Proc. of Virtual Reality, Modelling and Visualization, pp. 261-269, 2003.
[14] B. Reitinger, C. Zach, and D. Schmalstieg, "Augmented Reality Scouting for Interactive 3D Reconstruction," Proc. of IEEE Virtual Reality, pp. 219-222, 2007



배 성 민 (Sungmin Bae)

2007년 2월 중앙대학교 컴퓨터공학과 졸업
2007년 3월 ~ 현재 중앙대학교 첨단영상대학원 영상공학과 석사과정

관심분야 : 컴퓨터그래픽스



홍 현 기 (Hyunki Hong)

1993, 1995, 1998년 중앙대학교 전자공학과 학사, 석사, 박사 졸업
1998년 9월 ~ 1999년 8월 서울대학교 자동제어특화연구센터 연구원.
1999년 9월 ~ 2000년 2월 중앙대학교 정보통신연구소 연구교수
2002년 2월 ~ 2003년 1월 Univ. of Colorado at Denver 방문연구원.
2000년 3월 ~ 현재 중앙대학교 첨단영상대학원 영상공학과 부교수 재직 중.

관심분야 : 컴퓨터그래픽스, 컴퓨터비전 등