광원 트리를 사용한 간접 조명의 실시간 렌더링
(Real-Time Indirect Illumination using a Light Quad-Tree)

기 현 우 † 오 경 수 ††
(Hyunwoo Ki) (Kyoungsu Oh)

요 약 간접 조명은 사실적인 이미지 생성에서 매우 중요한 역할을 한다. 우리는 광원에 대한 사진트리(quadtree)를 사용한 확산 간접 조명의 실시간 렌더링 기법을 제안한다. 먼저, 수십만 개의 간접 광원 정보를 이미지의 각 픽셀에 저장하며, 이를 시점에 독립적인, 사진트리 형태의 이미지 피라미드로 만든다. 이 광원 트리를 하향식의 너비우선으로 탐색하며, 좋은 화질을 제공하는 적합한 광원 집합을 찾아 조명을 근사한다. 우리는 트리의 생성과 탐색 등의 모든 과정을 그래픽스 하드웨어 상에서 실시간에 처리한다. 결과 이미지들은 제안된 기법이 컬러 블리딩(color bleeding) 등의 확산 간접 조명 효과를 잘 연출하는 것을 보여준다. 우리의 기법은 복잡한 장면에서도 별도의 전처리과정없이 초당 수십~수백 프레임으로 렌더링할 수 있었다. 샘플링을 사용한 기존의 기법에 비하여 동급 화질에서 7배 가량 더 빠른 성능을 보였으며, 샘플링 노이즈를 피할 수 있었다.

키워드 : 간접 조명, 실시간 렌더링, 사진트리, GPGPU

Abstract *Indirect illumination plays an important role for realistic image synthesis. We present a novel realtime indirect illumination rendering technique using image pyramids. Hundreds of thousands of indirect point light sources are stored into images, and then they hierarchically clustered into quad-tree image pyramids. We also introduce a GPU based top-down and breadth-first traversal of the quad-trees to approximate the illumination with clusters (set of lights). All steps entirely run on the GPU in real-time. Result images demonstrate that our method represents diffuse interreflection, especially a color bleeding effect well. We achieved interactive frame rates of tens to hundreads, without any preprocessing. We can avoid artifacts caused by sampling, and our method is seven times faster than a recently proposed sampling based method.*

Key words : Indirect Illumination, Real-time Rendering, Quadtree, GPGPU

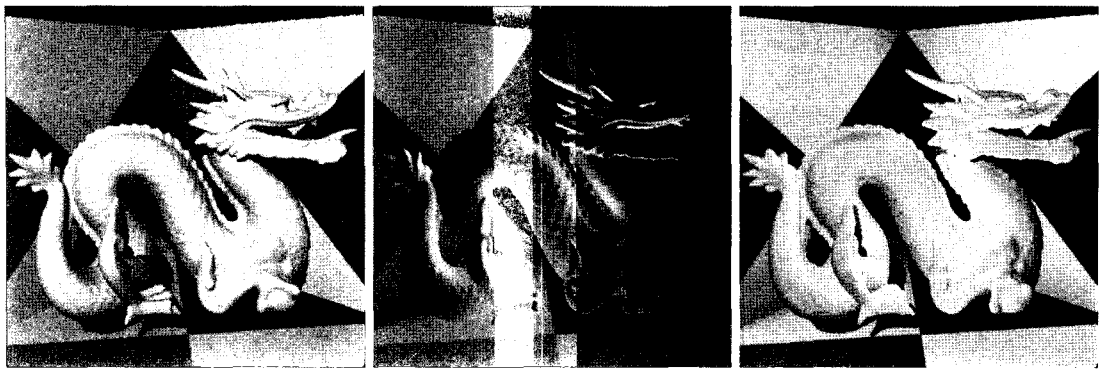
1. 서 론

지역 조명 모델은 광원으로부터 직접 받은 빛만을 조명 계산에 사용하기 때문에, 실세계와 비교하여 많이 다른 조명을 연출하게 된다. 예를 들어, 광원의 반대 방향에 있는 표면은 어둡게 표현된다. 우리는 환경광(ambient light), 라이트맵(light map) 등을 사용함으로써 이러한 문제를 완화할 수 있지만, 이들은 오직 정적인 조명만을 표현할 수 있으며, 컬러 블리딩(color bleeding)과 같은 중요한 간접 조명 효과를 연출하기 어렵다.

간접 조명을 표현하기 위해서, 우리는 경로 추적법(path tracing), 래디오시티(radiosity), 포톤 매핑(photon mapping) 등의 기법을 사용해야하지만, 이러한 기법들은 계산량이 매우 많아 실시간에 수행하기 어렵다.

전역 조명을 위한 기법 중의 일부는 간접 조명을 많은 수의 점 광원으로부터의 직접 조명으로 근사한다[1-3]. 이러한 경우 수많은 간접 광원을 다루어야하며, 우리는 수많은 광원을 효율적으로 처리하기 위하여 몬테 카를로(Monte Carlo) 샘플링이나 클러스터링(clustering) 기법 등을 사용할 수 있다. 몬테 카를로 샘플링은 대체로 효율적이지만 샘플 수가 불충분할 때에는 심각한 시각적 오류가 발생한다[1,4]. 클러스터링은 수만에서 수십만 개의 간접 광원을 수백에서 천 여개의 광원 집합으로 대체하여 조명을 적은 계산만으로 근사한다[5-9]. 특히, 광원 트리를 사용하는 방법은 많은 수의 점 광원을 트리로 만든 뒤, 루트(root)부터 리프(leaf)를 향하여 탐색해 나아가며 충분히 좋은 화질을 제공하는

· 본 연구는 학술진흥재단 중점연구소(KRF-2004-005-D00198) 지원으로 이루어졌습니다.
† 학생회원 : 숭실대학교 미디어학과
kih@ssu.ac.kr
†† 종신회원 : 숭실대학교 미디어학과 교수
oks@ssu.ac.kr
논문접수 : 2006년 8월 22일
심사완료 : 2007년 2월 12일



(a) 직접 조명 (b) 간접 조명 (c) 직접 조명 + 간접 조명

그림 1 확산 간접 조명은 용의 몸에 벽의 색이 반사되어 물드는 현상(컬러 블리딩; color bleeding)과 같은 중요한 간접 조명 효과를 연출한다.

광원 집합을 선정하고, 이것을 사용하여 빠르게 조명을 근사하는 전략이다[7-9]. 이러한 방법은 빠른 편이지만 트리와 같은 복잡한 자료구조와 트리 탐색 등의 복잡한 알고리즘으로 인하여 실시간 응용 프로그램 뿐만 아니라 GPU 기반 응용 프로그램에서 사용되기에 적합하지 않았다.

본 논문에서는 한 번 반사된 확산 간접 조명을 실시간에 렌더링하기 위하여 광원의 사진트리(quadtree)를 사용하는 GPU 기반의 기법을 제안한다. 우리는 이미지 상에 저장된 많은 광원들을 사진트리 구조의 이미지 피라미드(image pyramid)로 계층화하고, 광원 집합을 대표 광원으로 취급하여 간접 조명을 근사한다. 우리는 조명 계산을 근사할 광원 집합의 선정을 간단히하는 방법을 제안하며, 이 과정을 광원 트리를 생성할 때 수행한다(즉, 광원 트리가 시점에 독립적이다). 또한, 조명을 근사할 광원 집합을 찾기 위하여, 우리는 사진트리를 GPU 상에서 하향식의 너비우선 방식으로 탐색하는 새로운 방법을 소개한다.

모든 과정을 GPU 기반의 이미지 공간 상에서 실시간에 처리함으로써 복잡한 장면나 움직이는 물체가 있는 장면을 별도의 전처리과정없이 빠르게 렌더링할 수 있다.

2. 관련 연구

Keller는 장면 상에 가상의 점 광원들을 무작위로 생성하고, 이들 광원으로부터의 직접 조명으로써 간접 조명을 근사하는 방법을 제안하였다[3]. 하지만 조명 계산의 부하 때문에 많은 광원을 사용하지 못하였다.

Tabellion은 미리 계산해 놓은 래디오시티 맵(radiosity map)을 사용하여, 간접 조명의 계산을 가속화하는 방법과 함께, 한 번 반사된 간접 조명으로도 충분한 화질을 제공한다는 것을 보여주었다[10].

Jeppe는 포톤 매핑과 유사한 방식이지만, 모든 과정을 GPU를 사용하는 새로운 방식을 제안하였다[2]. 여기에서는 한 번 반사된 확산 간접 조명만을 다룬다. 광원의 시점에서 렌더링하여 직방사도 맵(direct radiance map)을 생성하고, 쉐이딩할 때, 포톤 매핑과 유사하게, 현재 표면 주변에 있는 픽셀 광원을 직방사도 맵으로부터 모아서 이를 간접 조명 계산에 사용한다. 또한, Jeppe는 방사도를 계산하기 위한 입체각(solid angle)의 계산을 여러 가지 방법으로 근사하였다. 하지만 매우 적은 개수(4~16 개)의 픽셀 광원만을 모음으로 인하여 어색한 결과를 낳았다.

Paquette는 많은 수의 점 광원이 있는 장면에 대해 효과적으로 광선 추적법을 수행하기 위하여, 광원들을 팔진트리(octree)로 계층화하고, 가려짐(occlusion)을 무시한다는 가정에서, 충분한 화질을 제공하는 광원 집합으로 조명 계산을 근사하는 기법을 제안하였다[7].

Jensen은 무수히 많은 표면 상의 조사도 샘플 (irradiance sample)들을 팔진트리로 만든 뒤, 각 복셀(voxel)의 입체각이 유저가 정의한 임계값보다 작을 경우, 해당 복셀을 사용하여 조명을 근사하였다[8].

Walter는 이진트리(binary tree) 형태의 광원 트리를 생성한 뒤, 대표 광원(더 밝은 광원)을 사용한 방사도와 광원 집합을 사용한 방사도의 차이가 작으면 해당 광원 집합을 이용하여 조명 계산을 근사하였다[9]. 이러한 기법들은 광원의 개수가 수만에서 수십만 개로 늘어나도 $O(\log N)$의 비용으로 렌더링할 수 있다(여기서 N은 광원의 개수이다). 비록 이들의 기법들은 빠르지만, 여전히 실시간에 수행될 수는 없다. 또한, 이 기법들은 매 픽셀마다 트리를 탐색하며 광원 집합을 사용한 조명 근사의 오류를 계산하여 현재 광원 집합을 조명 계산에 사용할 것인지 더 하위에 있는 광원 집합을 사용할 것인지를

선정한다. 하지만 이러한 오류 검사에 소요되는 시간은 전체 렌더링 시간의 20%나 차지하였다[9].

2.1 반사적 그림자 맵

우리의 기법은 반사적 그림자 맵(reflective shadow maps)[1]을 기반으로 한다. 이 기법은 광원 시점에서 렌더링하여 생성한 그림자 맵에 반사된 빛의 밝기, 위치, 법선 벡터 및 깊이를 저장하고, 각 픽셀을 간접 점 광원으로 사용하여 한 번 반사된 확산 간접 조명을 계산한다. 쉐이딩 시, 그림자 맵에 저장된 전체 간접 광원 중에서 그림자 맵에 투영된 표면 상의 점과 인접한 간접 광원들을 샘플링하여 간접 조명을 계산한다. 하지만, 실시간 렌더링을 위하여, 많은 전체 간접 광원 수(보통 512 × 512 = 262,144개)에 비하여 매우 적은 개수(수백~천 여 개)의 픽셀 광원만을 샘플링하였고, 이로 인하여 결과 이미지에 무늬(pattern)가 나타나는 시각적 오류가 발생하였다. 이 문제를 완화하려면 샘플의 개수를 많이 늘려야 하는데, 그러한 경우 표 3과 그림 11에서와 같이 처리 속도가 현저히 감소한다.

이 기법은 모든 간접 점 광원 정보를 이미지에 저장하기 때문에 GPU로 처리하기에 매우 적합하다. 뿐만 아니라, 이러한 점 광원은 광원 트리로 만들기에 매우 적합하다.

무수히 많은 광원과 렌더링할 표면 사이의 가시성 검사는 매우 많은 계산을 요구하기 때문에, 반사적 그림자 맵과 같은 실시간 렌더링 기법[1,2,11] 뿐만 아니라 일부 비실시간 기법들[5,7]은 이를 무시한다. 우리도 이들 연구처럼 가시성 검사를 무시한다.

3. 계층화를 통한 간접 조명의 근사

이번 절에서는 조명 근사에 사용할 광원 집합의 결정을 광원 트리를 만들 때 수행되도록 단순화한다(즉, 시점에 독립적인 광원 트리를 사용한다). 또한, 광원 집합을 사용하여 간접 조명의 계산을 근사하는 식을 유도한다.

먼저, a 방향에서 바라본 표면 상의 한 점 x의 방사도(radiance) L_s가 점 광원의 집합 s로부터 야기된 직접 조명일 때, 이는 표면의 재질(M: material), 표면과 광원 간의 기하정보(G: geometric), 가시성(V: visibility) 및 빛의 세기(I: intensity) 항목의 곱을 모두 합하여 계산할 수 있다[13].

$$L_s(x,\omega) = \sum_{i \in s} M_i(x,\omega) G_i(x) V_i(x) I_i. \tag{1}$$

광원과 표면 사이의 가시성 검사는 어렵기 때문에, 일부 연구들은 이를 무시한다[1,2,5,7]. 또한, 확산(diffuse) 광만을 고려한다[1,2,5-7]. 만일, 모든 재질을 확산 표면으로 가정하고, 항상 가려지지 않는다고 가정하면 식 (1)은 다음과 같이 간단해진다.

$$L_s(x,\omega) = \frac{\rho_d}{\pi} \sum_{i \in s} G_i(x) I_i. \tag{2}$$

정확한 방사도를 계산하기 위한 식 (1)은 광원의 개수에 비례하기 때문에, 계산량이 매우 많다. [9]와 같이, 방사도 L_c가 광원 집합, $C \subseteq S$, 로부터 야기된 직접 조명일 때, 만일 광원 집합의 위치와 방향이 각 구성 요소들의 위치와 방향과 서로 거의 일치한다면(즉, 평균값과 유사하다면), 위치와 방향에 의존하는 기하 정보를 광원 집합에 대한 항목으로 대체할 수 있다.

$$\begin{aligned} L_c(x) &= \frac{\rho_d}{\pi} \sum_{i \in C} G_i(x) I_i \\ &\approx \frac{\rho_d}{\pi} G_c(x) \sum_{i \in C} I_i \\ &= \frac{\rho_d}{\pi} G_c(x) I_c, \end{aligned} \tag{3}$$

여기서 M_c는 광원 집합의 재질, G_c는 광원 집합의 기하정보, I_c는 광원 집합의 밝기이며, 광원 집합의 밝기는 광원 집합의 구성 요소들의 밝기의 총 합을 사용한다. 우리는 광원 집합의 위치와 방향값으로 광원 집합을 구성하고 있는 각 구성 요소(개별 광원 또는 하위 광원 집합)들의 위치와 방향의 평균을 사용한다.

우리는 이렇게 광원 집합을 구성하는 각 구성 요소 위치와 방향이, 그들의 평균값과 서로 유사한 (즉, 서로 간의 차이값이 유저에 의해 정의된 임계값보다 작은) 광원 집합을 *유효한 광원 집합*이라고 부른다. 이로써 우리는 조명 계산 시, 오직 광원 집합이 유효한 지의 여부만을 확인하고 유효한 광원 집합을 사용하여 조명을 근사할 수 있다. 유효한 광원 집합은 오직 광원의 위치와 방향에 의존적이므로 시점이 변하더라도 계속 유효하다.

최종적으로, a 방향에서 바라본 표면 상의 한 점 x에서의 간접 조명은 모든 유효한 광원 집합으로부터의 직접 조명의 합으로 근사될 수 있다.

$$L(x,\omega) \approx \frac{\rho_d}{\pi} \sum_{clusters\,c} G_c(x) I_c. \tag{4}$$

광원 집합의 빛의 양, 위치 및 방향은 광원 집합을 구성하는 개별 광원 또는 하위 광원 집합들의 평균값을 통해 구할 수 있으며, 이들 값들은 모두 GPU에 의해 자동으로 밉맵된 값을 사용하므로 매우 쉽고 빠르게 계산할 수 있다.

확산 조명을 위한 빛의 세기는 램버트(Lambert) 모델을 사용하여 다음과 같이 계산된다.

$$I_c = \Phi_c max\{0,\ n_c \cdot (x_c - x)\}. \tag{5}$$

여기서, Φ_c는 광원 집합의 빛의 양(flux), x_c는 광원 집합의 위치, n_c는 광원 집합의 방향, x는 표면의 위치이다.

4. 알고리즘

알고리즘은 크게 세 단계로 이루어진다. 먼저, 광원을 다수의 점 광원으로 변환한다. 우리는 반사적 그림자 맵을 사용하기 때문에, 광원 시점에서 렌더링한 그림자 맵에 광원의 정보(반사된 빛의 세기, 월드 상의 위치, 법선 벡터 및 깊이)를 저장하고, 각 픽셀을 간접 점 광원으로 간주한다(그림 2.1). 둘째로, 광원 트리를 생성한다. 빛의 세기, 위치, 법선, 깊이는 GPU 상에서의 자동 밉맵을 사용하여 계층화한다. 또한, 픽셀 쉐이더를 사용하여 하위4개의 픽셀 광원과 이들의 평균값이 공간적 그리고 방향적으로 유사한 지 여부를 저장한 이미지 피라미드를 만든다(그림 2.2). 우리는 이것을 광원 집합 맵(light cluster maps)이라고 부른다.

쉐이딩 시에는 조명 근사에 사용할 광원 집합을 찾기 위하여, GPU를 사용하여 하향식의 너비우선 방식으로 광원 트리를 탐색한다. 만일 현재 레벨의 광원 집합이 유효하면, 이 광원 집합을 사용하여 조명을 근사하고, 그렇지 않으면 다음 레벨에서 다시 유효성을 검사한다(그림 2.3). 각 단계에 대한 상세한 설명은 하위 절에서 다룬다.

4.1 간접 광을 다수의 점 광원으로 표현

간접 조명에 광원 트리 기법을 적용하기 위해서는 간접 광을 다수의 점 광원으로 표현해야한다. 우리는 앞서

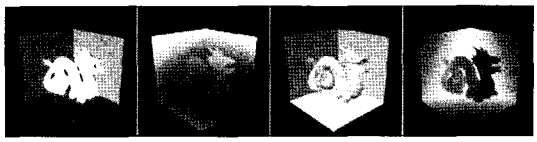


그림 3 간접 광원 정보(빛의 양, 월드 상의 위치, 법선 벡터 및 깊이)는 이차원 이미지에 저장된다.

설명한, 반사적 그림자 맵(RSM)[1]을 기반으로 한다. 그림 3과 같이, 간접 광원은 광원 시점에서의 렌더링한 이미지에 간접 광원 정보를 저장함으로써 생성된다.

4.2 광원 트리의 생성

우리는 이차원 이미지 상에 저장된 각 픽셀 점 광원들을 GPU를 사용하여 사진트리 형태의 이미지 피라미드로 만든다. 빛의 양, 위치 및 방향은 밉맵을 사용하여 트리로 만들고, 별도의 픽셀 쉐이더 작업을 통해 각 광원 집합의 유효 여부를 결정한다. 이 유효 여부는 차후 조명 근사에 사용할 광원 집합을 찾을 때 사용된다. 광원 집합의 기하정보는 이전 단계에서 생성한 이미지의 밉맵된 값 (평균값)을 사용한다.

보통의 밉맵에서 텍스처의 각 하위 레벨의 픽셀에는 바로 이전 레벨 (높은 해상도)에 해당하는 4개의 픽셀들의 평균을 저장한다. 우리는 이전 레벨에서의 4개의 픽셀이 서로 공간적 그리고 방향적으로 유사하면 이를 유

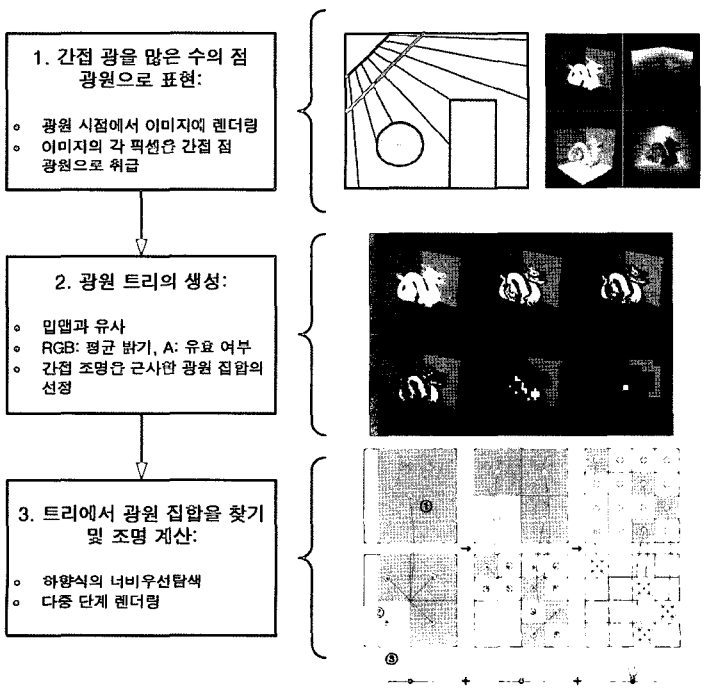


그림 2 알고리즘의 전체 프로세스 도식화

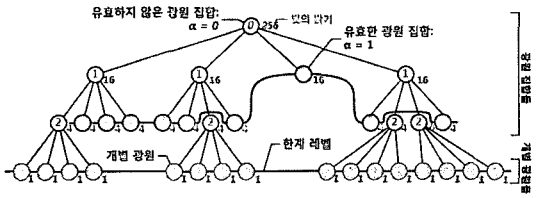


그림 4 광원 트리의 예

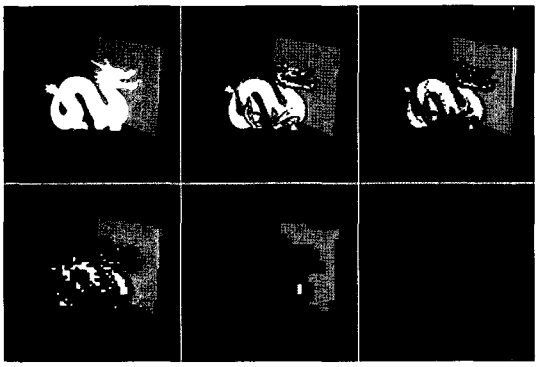


그림 5 광원 집합 맵(0~5 레벨). 외곽의 검정색 영역은 배경을 나타낸다. 루트 레벨로 갈수록 (그림상에서 우측하단으로 갈수록) 검정색 영역이 표면 내부로 확장되는데, 이것은 유효하지 않은 광원 집합을 나타낸다. 이는 루트 레벨로 갈수록 간접 광원 또는 하위 광원 집합 간의 유사도가 낮아지기 때문이다.

효한 광원 집합으로 취급한다.

그림자 맵 상에서 인접한 픽셀은 이를 역투영한 월드 공간 상에서도 인접하다. 하지만, 그림자 맵 상에서의 x와 y 좌표가 가깝더라도, 맵에 저장된 깊이(z) 값의 차이가 크다면, 월드 공간 상에서 멀리 떨어져 있게 된다. 따라서, 공간적 유사성의 확인으로 깊이맵에 저장된 4개의 픽셀의 깊이값과 이들의 평균 깊이값만을 비교한다. 한 편, 방향적 유사성 확인으로, 텍스처에 저장된 픽셀 광원들의 법선 벡터와 이들의 평균 법선 벡터의 내적으로 비교한다.

공간과 방향의 유사도에 대한 임계값은 유저가 임의로 정할 수 있다. 느슨하게 설정된 임계값을 사용하면 총 렌더링 시간이 감소하지만, 광원 집합에 의해 근사된 조명의 화질이 낮아지는 트레이드 오프(trade-off)를 가진다. 이에 대한 실험과 분석은 6절에서 다룬다.

이러한 광원 트리의 생성은 매우 빠르게 수행된다. 뿐만 아니라, 이 광원 트리는 시점에 독립적이기 때문에 광원이나 장면이 움직였을 때에만 수행할 수 있다.

4.2.1 기본의 광원 트리 기법과의 비교

Paquette [7]와 Jensen [8]은 광원을 하향식으로 접근하며 팔진트리로 만들지만, 우리는 Walter[13]처럼 상향식으로 접근하여 트리를 만든다. 하지만, 보다 GPU에 적합한 형태인 사진트리로 만든다. 우리는 그림자 맵에 저장된 광원을 트리로 만들기 때문에, 월드 공간상에서의 이웃 노드(픽셀)들간의 공간적 인접성이 보장되어 어떠한 이웃 광원 탐색(nearest searching) 절차없이 빠르게 계층화할 수 있다. 더욱이, 자동 밉맵 생성 등의 GPU의 기능을 최대한 이용하기 때문에 매우 빠르게 수행된다.

위에서 언급한 기존 연구들은 쉐이딩 시, 각 픽셀마다 어떤 광원 집합을 사용하여 조명을 계산할 지를 결정하는데, 이러한 *오류 상한(error bound)* 검사는 매우 복잡한 계산을 요구한다. 하지만 우리는 쉐이더의 명령어 개수를 줄여서 보다 빠르게 렌더링하기 위하여, 조명 계산에 사용할 광원 집합을 트리를 만들 때 미리 결정짓는, 시점에 독립적인 오류 상한을 사용한다. 우리의 오류 검사 기준은 유저가 정의한 임계값과의 비교를 사용는 Jensen[8]과 유사하며, 식 (4)에 의하여 광원 집합을 구성하는 요소들의 위치적 그리고 방향적 유사도만을 기준으로하는 간단한 방식을 사용한다.

4.3 트리 탐색과 조명 근사

쉐이딩할 때, 그림 2의 (3)과 같이 광원 트리에서 유효한 광원 집합을 찾아 간접 조명을 근사한다. 유효한 광원 집합을 찾기 위한 사진트리의 탐색은 너비 우선한 하향식 탐색 방식으로써, GPU를 사용한 기존의 트리 탐색 기법들[12,13]과는 달리, 자식 노드에 이웃한 노드들(siblings)도 탐색해야 한다.

우리는 하향식의 너비우선으로 자식 노드의 이웃 노드까지 탐색하기 위해서, 트리의 각 레벨별 탐색을 다중 단계(multi-pass)로 수행하는 새로운 방법을 소개한다. 각 레벨에 대하여, 그림 7에서와 같이 이전 레벨의 광원 집합이 유효하지 않은 광원 집합인 경우, 현재 레벨에서의 4개의 픽셀들(즉, 광원 집합을 구성하는 하위 레벨 픽셀들)에 대하여 각각이 유효한 지 여부를 확인하고, 해당 픽셀이 유효한 경우에만 조명 계산에 사용한다.

최악의 경우, 항상 리프 노드까지 탐색해야 한다. 실시간 처리를 위하여, 우리는 탐색의 *한계 레벨*을 두어, 한계 레벨에서의 광원 집합들은 항상 유효한 광원 집합으로 간주한다. 이로 인하여, 루트에 가까운 레벨을 한계 레벨로 설정하면 속도의 이득을 볼 수 있지만 화질을 손상시키게 된다. 이에 대한 분석은 5장에서 다룬다. 우리는 모든 개별 광원(클러스터링되지 않은 본래의 광원)들은 항상 유효한 광원 집합으로 취급하여, 유효성 검사를 하지 않고 바로 조명 계산에 사용한다.

유효한 광원 집합을 찾으면, 이를 사용하여 간접 조명을 계산한다(식 (5)). 각 패스에서 유효한 광원 집합들

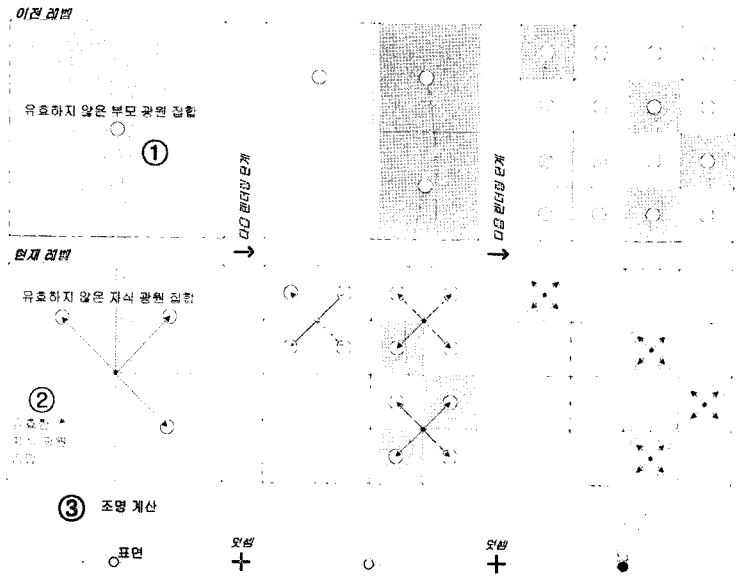


그림 6 유효한 광원 집합을 찾기 위하여, 우리는 트리의 각 레벨을 다중 단계 렌더링을 사용하여 직접 탐색하고(1과 2), 발견된 유효한 광원 집합들을 사용하여 조명을 근사한다(3).

을 사용하여 근사된 간접 조명 값은 높은 정밀도의 이미지에 렌더링한다. 최종적으로, 이렇게 렌더된 이미지들은 한 번의 추가적인 렌더링 단계에서 프레임 버퍼에 블렌딩된다. 이 때, 적합한 가중치(4^n, n은 레벨)를 곱한 뒤에, 이들의 합으로 간접 조명 값을 계산한다. 광원 집합 맵의 낮은 해상도의 레벨(상위 노드)에서 계산된 조명에는 보다 높은 가중치를 곱한다.

5. 구 현

우리는 Direct3D 9.0과 쉐이더 모델 3.0의 HLSL로 구현하였으며, 반사적 그림자 맵(reflective shadow maps; RSM)을 기반으로 하였다. 4장에서 설명한 광원 트리의 생성과 조명을 근사할 유효한 광원 집합을 찾는 과정이 추가되며, 식 (5)를 사용하여 쉐이딩한다. 이들 모든 과정은 GPU 상에서 실시간에 실행된다.

하향식의 너비우선으로 사진트리를 탐색하기 위하여, 광원 집합 맵의 각 레벨을 다중 단계로 렌더링한다. 정점 수가 많은 장면에 대해 다중 단계 렌더링(multi- pass rendering)을 수행하는 것은 매우 비효율적이다. 우리는 RSM처럼 지연 쉐이딩(deferred shading) 기법을 사용함으로써 이 문제를 해결한다[1]. 이 기법은 현재 화면상의 위치, 방향, 재질 등의 정보를 이미지에 렌더링 한 뒤, 이 정보를 사용하여 조명 계산 등의 주 작업을 추가적인 렌더링 단계에서 수행하는 방법이다. 이 기법은 장면을 한 번만 렌더링하기 때문에, 복잡한 장면에 대한 다중 단계 렌더링을 사용할 때에 매우 유용하다.

우리는 렌더링 횟수를 줄이기 위해서, 루트 노드 혹은 루트에 가까운 노드에 대한 패스를 생략하였다. 그림 5에서 보이는 바와 같이, 실험을 통해 이들 노드의 광원 집합들은 항상 무효하였음을 확인하였다. 따라서, 우리는 8×8의 해상도를 갖는 7레벨부터 하향식으로 진행하였다. 하지만, 최대 4,096 개의 노드를 탐색해야하는 4레벨 (64×64)부터는 픽셀 쉐이더의 루프 횟수 제한 (최대 256회) 문제에 부딪치게 된다. 이로 인하여, 우리는 4개로 쪼갠 뒤 각각을 4번의 렌더링 단계를 사용하여 탐색하였다. 우리는 비록 구현하지 않았지만, 16,384개의 노드를 가지는 3레벨(128×128) 이하에 대한 탐색도 가능할 것이다.

각 단계에서 렌더링한 결과는 컴포넌트 당 16비트 정수형 타입을 갖는 64비트 이미지에 렌더링하고, 최종적으로 프레임 버퍼에 블렌딩한다.

6. 실험 결과

이번 절에서, 우리는 우리의 기법을 몇 개의 장면에 적용하고 여러 가지 속성값을 다양하게 변화시키며 렌더링한 결과를 보여준다. 그래픽 카드로는 ATI X1900 GTX 512MB를 사용하였다. 모든 과정은 GPU 상에서 수행되기 때문에 PC의 사양은 알고리즘의 성능에 영향을 주지 않는다. 화면과 버퍼들(간접 광원 맵, 광원 집합 맵 및 지연 쉐이딩 버퍼 및 각 레벨별 버퍼)의 해상도는 512×512를 사용했다. 렌더될 모든 픽셀에서 조명 계산을 하는 것이 비효율적이기 때문에, RSM에와 같이,

작은 해상도(64×64)의 텍스처에 렌더링한 뒤, 보간으로 인한 오류가 적은 영역은 최종 해상도에 보간한다(sub-sampling).

먼저, 한계 레벨을 6(16 × 16; ≪ 256), 5 (32 × 32; ≪ 1,024), 4 (64 × 64; ≪ 4,096)로 다양하게 바꿔가며 코넬 상자(Cornell Box)를 렌더링하였다(그림 7). 깊이 임계값은 0.05, 그리고 법선 임계값은 0.9로 고정하였다.

표 1 한계 레벨에 따른 초당 프레임율

한계 레벨	6 (16×16)	5 (32×32)	4 (64×64)
버퍼 사용	204.20	73.71	20.57
버퍼 미사용	107.58	47.96	14.42

표 1에서와 같이, 광원 집합 맵의 한계 레벨이 낮아짐에 따라 초당 프레임율은 떨어진다. 그림 7은 6 레벨(최악의 경우, 총 16 × 16 = 256 번의 조명 계산을 수행)만으로도 충분한 화질을 제공하는 것을 보여준다. 하지만, 동적인 장면을 연출할 때에는 5레벨(최악의 경우, 32 × 32 = 1,024번의 조명 계산) 이하로 설정해야 튀는 현상을 줄이고 보다 깨끗한 영상을 얻을 수 있었다. 여기서 *버퍼 사용*은 장면과 광원이 고정되었을 때를 의미하며, 이는 이미 생성한 광원 집합 맵, 지연 쉐이딩 버퍼 및 보간에 사용할 저해상도의 버퍼를 재사용하기 때문에, 보다 빠르게 수행될 수 있다. 반면에, *버퍼 미사용*은 장면과 광원이 움직일 때를 의미한다.

다음으로, 한계 레벨을 5로 고정하고 깊이와 방향 임계값을 조정해보았다. 코넬 상자와, 24,975개의 정점과 50,000개의 삼각형을 가진 부처(Happy Buddha) 모델을 렌더링해보았다.

표 2에서와 같이, 임계값들이 낮을수록 더 높은 프레임율을 보이는 것을 확인할 수 있다. 하지만 임계값이 낮으면 공간적 그리고 방향적인 유사성이 낮더라도 클러스터링되기 때문에 화질이 떨어진다(그림 8). 비록 우

표 2 깊이(공간)와 법선(방향)의 임계값에 따른 초당 프레임율(버퍼사용)

임계값	깊이: 0.45 법선: 0.3	깊이: 0.25 법선: 0.6	깊이: 0.05 법선: 0.9
코넬 상자	82.76	76.03	73.71
부처	46.50	46.45	40.70

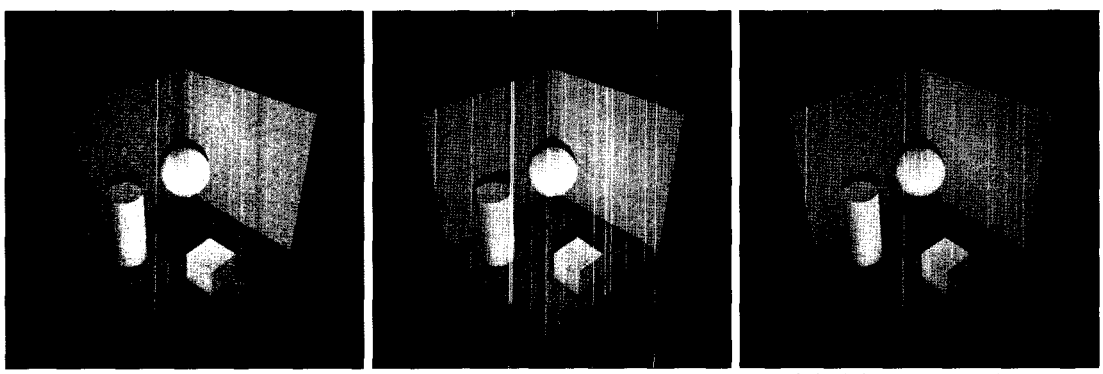
리의 기법이 대부분 이미지 공간 상에서 수행되지만, 삼각형의 개수가 늘어나면(즉, 장면이 복잡해지면) 유효한 광원 집합의 상한이 낮아지기 때문에 프레임율이 감소하게 된다. 또한, RSM에서와 동일하게, 화면 공간에서 보간할 수 없는 픽셀이 증가하여 초당 프레임율이 감소하게 된다. 그림 1과 10은 텍스처 매핑을 적용한 결과이다.

다음으로, 우리는 샘플링을 사용하는 RSM 기법과 우리의 기법을 비교해보았다(표 3). RSM에서는 Possion 샘플링을 사용하여, 렌더된 이미지에 얼룩과 같은 무늬가 나타났다. 이러한 무늬는 샘플 개수가 부족할 때와, 특히, 광원이 이동할 때 매우 심각하게 드러난다. 이러한 시각정 오류를 완화하기 위해서는 매우 많은 샘플을 필요로 한다(그림 9).

표 3 우리의 기법과 샘플링을 사용한 기존의 반사적 그림자 맵과의 성능 비교

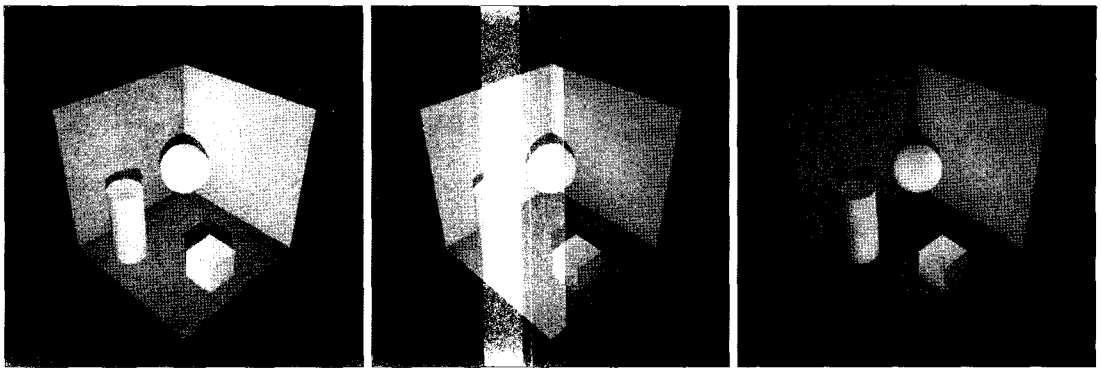
	우리의 기법		RSM	
한계 레벨 / 샘플 수	6 (16×16)	5 (32×32)	256	2,040
버퍼 사용	204.20	73.71	183.92	26.58
버퍼미사용	107.58	47.96	95.38	15.88

우리의 기법은 광원 클러스터링과 트리 탐색 등의 프로세스를 필요로 하지만, 광원 집합에 의한 근사로 조명 계산 횟수를 현저하게 줄이기 때문에 훨씬 빠른 렌더링 성능을 보인다. 그림 9는 포톤 매핑에 다중 반사와 파이널 개더링(final gathering) 수행한 결과 (a), 반사적 그



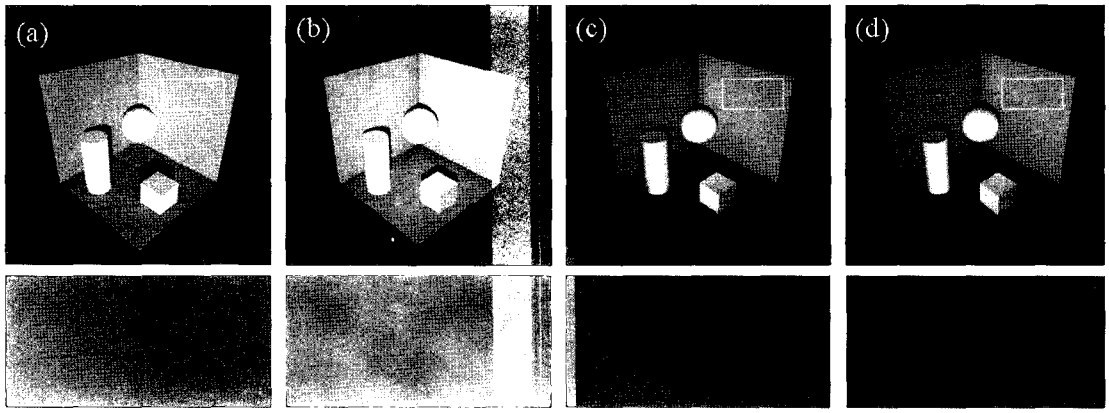
(a) 한계 레벨 6: 316.75 fps / 107.58 fps (b) 한계 레벨 5: 73.71 fps / 47.96 fps (c) 한계 레벨 4: 20.57 fps / 14.42 fps

그림 7 한계 레벨에 따른 화질과 속도 (버퍼사용 / 버퍼 미사용)



(a) 깊이: 0.45, 법선: 0.3; 82.76 fps (b) 깊이: 0.25, 법선: 0.6; 76.03 fps (c) 깊이: 0.05, 법선: 0.9; 73.71 fps

그림 8 공간적/방향적 임계값에 따른 화질과 속도(버퍼 사용). 임계값이 낮으면 속도는 빠르지만, 화질이 떨어진다.



(a) 포톤 매핑
포톤 수: 25,000
1분 02초

(b) 반사적 그림자 맵
샘플 수: 216
203.91 fps / 107.64 fps

(c) 반사적 그림자 맵
샘플 수: 2,040
26.58 fps / 15.88 fps

(d) 우리의 기법
광원 집합: 147
204.70 fps / 108.34 fps

그림 9 (a)는 포톤 매핑, (b)는 우리의 기법과 같은 속도에서의 반사적 그림자 맵, (c)는 같은 화질에서의 반사적 그림자 맵, (d)는 우리의 기법이다. fps는 버퍼 사용 / 버퍼 미사용 시의 초당 프레임율(frames per second)을 나타낸다. 우리의 기법과 같은 속도에서 렌더된 (b)에서는 얼룩과 같은 무늬가 나타나며, 샘플 개수를 늘려 얼룩을 제거한 (c)는 느리다. 하단 이미지는 차이 비교를 용이하게 하기위하여, 대비(contrast) 값을 증가시킨 이미지이다.

림자 맵 (b, c) 및 우리의 기법 (d)을 비교한 것이다. 반사적 그림자 맵을 사용하는 9.b와 9.c 및 우리의 기법 (d)는 포톤 매핑에 비하여 벽의 구석 부분이 어둡게 나타난다[2]. 하지만, 이렇게 가려짐이 적은 장면에서는 전반적으로 눈으로 보기에 충분한 화질을 제공한다. 그림 9의 샘플링을 사용한 (c)에 비하여, 광원 트리를 사용한 우리의 기법 (d)가 비슷한 화질에서 7배 가량 더 빠름을 확인할 수 있다.

그림 9의 하단 이미지는 벽의 일부분을 확대한 뒤, 대조 (contrast)값을 높임으로써, 서로간의 차이를 쉽게 눈으로 파악할 수 있도록 한 것이다. 포톤 매핑은 벽의 끝부분(이미지 상의 우측 상단 영역)이 어둡게 나타난다. 반면에, 우리의 기법은 끝부분이 오히려 더 밝게 나타나는 오류가 발생한다. 하지만, 같은 속도에서, 샘플링을 사용하는 (b)는 오류가 눈에 띄지만, 우리의 기법 (d)의 오류는 상대적으로 눈에 덜 민감하게 나타난다.

그림 10은 보다 복잡한 장면인 로마 건축물(Sponza Atrium)을 렌더링한 이미지로, 복잡한 장면과 텍스처를 사용함에 따라 하위 샘플링(sub-sampling)을 하지 않았다. 눈에 드러나는 무늬와 같은 오류없이 깨끗한 간접조명 효과를 연출할 수 있었으며, 초당 30~50 프레임으로 렌더링할 수 있었다.

7. 토 론

우리의 트리 탐색은, 이미 조명 근사에 사용된 노드인 경우에도, 때로는 그들의 자식 노드들까지 하위 레벨에

그림 10 우리의 기법을 보다 복잡한 장면인 로마 건축물에 적용한 이미지이다. 한계 레벨 6을 사용하여 렌더링하였고, 큰 이미지는 직접 조명과 간접 조명을 사용하여 렌더된 것이며, 작은 이미지는 직접 조명만을 사용한 것이다.

서 재방문하여 조명 근사에 사용했는지 여부를 확인해야 한다. 이것은 GPU 프로그래밍의 특성상 방문 여부를 저장할 수 없기 때문이다. 트리의 탐색 경로를 전처리를 통해 테이블화하는 방법[12,13] 등은 CPU 상에서의 복잡한 작업을 요구한다. 우리는 모든 과정에서 상호작용성(interaction)을 제공하는 것이 목표였기 때문에, 그러한 작업을 피하는 대신 재방문을 채택하였다.

Paquette[7]는 많은 비용을 소비하는 비실시간 접근이었지만, 픽셀마다 서로 매우 다른 레벨에서의 광원 집합을 사용하여 조명을 계산함에 따라 시각적 오류가 심각하였다. Walter[9]는 인지와 대표 광원 기반 방식을 사용함으로써 이러한 문제를 해결하였다. 하지만, 이러한 방법은 광원 집합의 기하 정보로써 평균값을 사용하는 것이 아니기 때문에, 밉맵에 기반한 계층화를 사용할 수 없으며 무작위 결정을 사용함에 따라 GPU 환경에서 적합하지 않다(GPU 프로그래밍에서 난수를 생성하는 기능은 제공되지 않는다). 비록 그들의 엄격한 오류 상한 검사는 화질을 향상시키고 다양한 BRDF를 지원하지만, 많은 명령어를 요구하기 때문에 GPU 기반의 실시간 환경에는 적합하지 않다.

광원 트리를 사용하는 알고리즘들은 대체로 시간적 일관성을 유지하지 못한다. 이로 인하여, 매우 동적인 장면에서는 깜박임이 발생한다. 이것은 임계값을 좀 더 엄격하게 설정(즉, 보다 하위 레벨의 광원 집합을 조명 계산에 사용)함으로써 완화할 수 있다. 우리의 광원 트리는 시점에 독립적이기 때문에, 빛의 이동이나 장면의 움직임 없이 시점만 변화할 때에는 이러한 문제가 전혀 발생하지 않는다. 중간 노드부터의 트리 탐색 및 하한 레벨은 속도의 향상 뿐만 아니라 서로 다른 레벨의 광원 집합을 사용함에 따른 오류[7]를 감소시키는 효과를 가져온다.

가시성을 고려하는 것은 매우 어려운 문제이다. 이전 연구들 또한 가시성을 종종 무시해왔다. 이로 인한 오류가 존재하지만, 확산된 조명에서는 상대적으로 덜 중요하다. 또한, 가려짐이 심하지 않은 장면에서는 눈으로 볼 때에는 보기 좋은 화질을 제공할 수 있었다. 비록 구현하지는 않았지만, 우리의 기법을 동적인 환경 가려짐(dynamic ambient occlusion)[11] 기법과 접목하면 가시성을 무시함에 따른 시각적 오류를 보다 완화할 수 있을 것이다.

우리의 구현에서는 하나의 스폿(spot) 광원만을 사용하여, 한 번 반사된 확산 간접 조명 렌더링만을 보여주었다. 하지만, 6개의 광원 트리를 사용하여 전방향(omni-directional) 광원에도 적용할 수 있을 것이다. 비록 구현하지는 않았지만, Walter[9]의 경우처럼, 면광원이나 환경맵에도 적용할 수 있을 것으로 기대한다.

8. 결론과 향후 연구

제안된 GPU 기반의 간접 광원 트리를 사용한 확산 간접 조명의 렌더링 기법은 수십만 개의 간접 광원을 수백 개의 광원 집합만으로 근사함으로써 실시간에 수행될 수 있었다. 기존의 CPU 기반 기법에 비하여 월등히 향상된 속도를 보였고, 기존의 GPU 기반의 샘플링 기법에 비하여 화질과 속도에서 모두 앞선 결과를 보였다. 또한, 우리는 조명 계산을 근사할 광원 집합의 선정을 간단화하고, 이를 광원 트리의 생성 단계에 포함시킴으로써 렌더링 속도를 가속화하였다. 우리의 기법에서는 별도의 전처리과정이 필요없으며, 모든 과정이 이미지 공간 상에서 수행되기 때문에, 장면의 복잡도에 영향을 적게 받는다.

GPU에서 효율적으로 수행할 수 있도록, 우리는 광원 트리의 생성을 밉맵과 유사한 방식을 사용하여 수행하

였다. 이렇게 생성한 광원 트리를 하향식의 너비우선 접근으로 탐색하며 조명 근사에 사용할 광원 집합을 찾기 위하여, 다중 단계 렌더링을 사용하여 탐색하는 새로운 방법을 소개하였다.

제안된 확산 간접 조명의 실시간 렌더링 기법은 비디오 게임, 가상현실 등의 실시간 렌더링 시스템에 적용하여 보다 사실적인 세계를 표현하는데에 기여할 것이다. 또한, 트리를 생성하고 탐색하는 기법은, GPGPU 관점에서 컴퓨터 그래픽스가 아닌 다른 분야의 알고리즘에서 활용될 수 있을 것으로 기대한다. 우리는 환경맵(environment map), 면광원(area light)과 같은 다양한 형태의 광원에서도 수행될 수 있도록 개선할 것이다.

참 고 문 헌

[1] Dachsbacher, C., and Stamminger, M. 2005. "Reflective shadow maps," In SI3D 05: Proceedings of the 2005 symposium on Interactive 3D graphics and games, ACM Press, New York, NY, USA, 203-231.

[2] Jeppe, R. F., Rasmus, R. F., Niels, J. C., and Peter, F. "Scene independent real-time indirect illumination," Proceedings of Computer Graphics International 2005, pp. 185-190.

[3] Keller, A. "Instant radiosity," In Proceedings of SIGGRAPH 97, Computer Graphics Proceedings, Annual Conference Series, 49-56.

[4] Shirley, P., Wang, C., and Zimmerman, K. "Monte carlo techniques for direct lighting calculations," ACM Transactions on Graphics 15, 1 (Jan.), 1-36.

[5] Hanrahan, P and D. Salzman, "A rapid hierarchical radiosity algorithm for unoccluded environments," In Proceedings Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics, (Rennes, France), pp. 151-71, (June 1990).

[6] Hanrahan, P., Salzman, D., and Aupperle, L. "A rapid hierarchical radiosity algorithm," In Computer Graphics (Proceedings of SIGGRAPH 91), vol. 25, 197-206.

[7] Paquette, E., Poulin, P., and Drettakis, G. "A light hierarchy for fast rendering of scenes with many lights," Computer Graphics Forum 17, 3, 63-74.

[8] Jensen, H. W, and Buhler. J. "A rapid hierarchical rendering technique for translucent materials," In Proceedings of the 29th Annual Conference on Computer Graphics and interactive Techniques, SIGGRAPH '02. ACM Press, New York, NY, 576-581.

[9] Walter, B., Fernandez, S., Arbee, A., Bala, K, Donikian, D, and Greenberg, D. P. "Lightcuts: A Scalable Approach to Illumination," In Proceedings of ACM SIGGRAPH 2005, pp 1098-1107.

[10] Tabellion, E., and Lamorlette, A. "An approximate global illumination system for computer generated films," ACM Tran. on Graphics 23, 3 (Aug.), 469-476.

[11] Bunnell, M. "Dynamic ambient occlusion and indirect lighting," GPU Gems 2 - Programming Techniques for High-Performance Graphics and General-Purpose Computation, Chapter 14.

[12] Sylvain, L., Samuel, H., and Fabrice, N. "Octree textures on the GPU," GPU Gems 2 -Programming Techniques for High-Performance Graphics and General-Purpose Computation, Chapter 36.

[13] Tim, F., and Jeremy, S. "KD-tree acceleration structures for a GPU raytracer," Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware 2005.

기 현 우

2005년 8월 숭실대학교 미디어학부(학사) 2005년 9월~현재 숭실대학교 미디어학과(석사). 관심분야는 외양 모델링, 사실적/예술적 렌더링, 시각 효과, 인터랙티브 프로그램

오 경 수

1994년 서울대학교 계산통계학과(학사) 1996년 서울대학교 전산과학과(석사). 2001년 서울대학교 전기 컴퓨터 공학부(박사) 2001년~2002년 ㈜조이멘트 개발팀장. 2003년~현재 숭실대학교 미디어학부 조교수 관심분야는 실시간 컴퓨터 그래픽스, 게임