영상 기반 실시간 재조명 렌더링 시스템

Image-Based Relighting Rendering System

김순현, Soonhyun Kim*, 이주행, Joo-Haeng Lee**, 경민호, Min-Ho Kyung***

요약 재조명(relighting) 렌더링은 장면 내에 새로운 광원의 추가 또는 기존 광원 속성의 변경으로 인한 영상의 변화를 효율적으로 계산하는 과정을 말한다. 본 논문에서는 쉐이딩(shading) 계산에서 광원에 독립적인 파라메터를 미리 텍스쳐 이미지 형태로 캐시화하여 재조명 렌더링 과정에서의 계산량을 줄이는 방법을 사용하였다. 이러한 쉐이딩 파라메터들의 캐시 이미지들은 사용자가 카메라 시점을 바꾸고자 할 경우 새로 생성을 하여야 하는데, 이러한 캐시 이미지 생성에는 많은 시간이 소요된다. 본 논문에서는 새로운 시점에서의 캐시 이미지들을 영상 기반 렌더링(image-based rendering) 기법을 이용하여 실시간에 구하는 방법을 제시한다. 이 방법은 먼저 여러 개의 지정된 카메라 시점에 대한 캐시 이미지들을 미리 생성해 둔다. 다음 원하는 시점의 캐시 이미지는 각 픽셀에 투영되는 3차원 표면점을 역시점변환(inverse viewing transform)을 통해 구하고, 이 점을 지정된 카메라 시점으로 다시 투영하여 캐시 이미지에서의 대응 픽셀을 찾는다. 대응 픽셀의 파라메터 값들을 평균하여 새 캐시 이미지에 써준다. 이 과정들은 하드웨어 그래픽 가속기의 단편 쉐이더(fragment shader)를 이용하여 실시간으로 수행된다.

Abstract We develop an interactive relighting renderer allowing camera view changes based on a deep-frame buffer approach. The renderer first caches the rendering parameters for a given 3D scene in an auxiliary buffer with the same size of the output image. The rendering parameters independent from light changes are selected from the shading models used for shading pixels. Next, as the user interactively edits one light at one time, the relighting renderer instantly re-shades each pixel by updating the contribution of the changed light with the shading parameters cached in the deep-frame buffer. When the camera moves, the cache values should be re-computed because the currently cached values become obsolete. We present a novel method to synthesize them quickly from the cache images of the user specified cameras by using an image-based technique. This computations are all performed on GPU to achieve real-time performance.

핵심어: rendering, relighting, shading, image-based rendering

본 논문은 정보통신연구진흥원의 대학 IT연구센터 육성 및 지원사업의 지원과 선도기반기술개발지원사업의 기능확장형 초고속 렌더링 기술개발 과제 지원에 의하여 연구되었음.

*주저자 : 아주대학교 정보통신전문대학원 박사과정; e-mail: kkubs@ajou.ac.kr

**공동저자 : 전자통신연구원(ETRI) 연구원

***교신저자 : 아주대학교 미디어학과 교수; e-mail: kyung@ajou.ac.kr

1. 서론

3차원 애니메이션 제작에서 각 장면(scene)의 조명 설정은 많은 시간과 노력을 요구한다. 가장 큰 이유는 광원을 조정한 결과를 정확하게 확인하기 위한 피드백(feedback)에 많은 시간이 걸리기 때문이다. 일반적으로 많이 사용하는 저해상도, 저품질의 프리뷰 렌더링은 시간이 적게 걸리는 반면 영상 품질이 낮아서 조명 결과를 정확하게 확인하기 어렵다. 반면에 고품질 렌더링은 한 프레임 렌더링에 수 시간 이상의 시간이 소요되기 때문에 잦은 결과 확인에는 사용할 수 없다.

본 논문에서는 2005년에 SIGGRAPH에서 발표되었던 픽사(Pixar)의 Lpics 시스템[7]을 기반으로 한 재조명 렌더링 시스템을 제안한다. 재조명 렌더링은 조명이 변할 경우 이에 따른 새로운 렌더링 결과의 계산을 빠르게 수행하는 렌더링 과정을 말한다. 본 논문에서는 Lpics 시스템과 마찬가지로 딥프레임버퍼(deep-frame buffer)를 사용한다. 먼저 각 픽셀의 쉐이딩 계산에 필요한 파라메터를 미리 추출하여 이미지 형태로 저장한다. 재조명 렌더링 과정에서는 이 캐시 이미지에 저장된 파라메터 값을 이용하여 광원 별로 각 픽셀의 쉐이딩 계산을 수행한다. 광원 별로 계산된 각 픽셀의 컬러를 모두 더하면 최종 이미지가 얻어진다. 딥프레임버퍼 방식의 장점은 재조명 렌더링에 걸리는 시간이 장면 복잡도(scene complexity)가 아닌 이미지 해상도에 따라 결정된다는 점이다. 즉, 아무리 복잡한 장면에 대한 재조명 렌더링도 지정된 이미지 크기에 따라 계산시간이 결정 된다.

하지만 이러한 접근 방법은 카메라가 움직이지 않는다는 가정 하에서만 빠른 렌더링이 가능하다. 카메라를 움직일 경우에는 시간이 많이 걸리는 캐시 이미지를 매번 새로 계산해야 하기 때문에 실시간 렌더링이 불가능하다. 본 논문에서는 이 문제를 해결하기 위해 영상 기반 렌더링(image-based rendering) 기법[2]에 의한 방법을 제안한다. 기본 개념은 미리 지정한 카메라 시점들의 캐시 이미지들을 이용하여 새로운 시점에서의 캐시 이미지를 픽셀 기반으로 합성해 내는 것이다. 즉, 사용자가 이동시킨 카메라의 새로운 시점에서 각 픽셀에 대하여 다른 카메라 시점에서 대응되는 픽셀 위치를 역시점변환(inverse viewing transform)을 통해 구하고, 이 위치에 캐시된 쉐이딩 파라메터를 가져와 새로운 캐시 이미지를 합성하는 것이다. 이 과정들은 하드웨어 그래픽 가속기의 단편 쉐이더(fragment shader)를 이용하여 실시간 처리되도록 구현된다.

2. 관련 연구

컴퓨터 그래픽스 분야에서 렌더링을 가속하기 위한 노력은 꾸준히 진행되어 왔다. 광선추적기법(ray tracing)에서 광선의 확산 경로를 트리 구조로 각 픽셀에 저장하여 재렌더링에서의 계산 시간을 단축하거나[3], 기하학적 데이터를 별도의 버퍼에 저장하여 렌더링을 가속하는 방법[4], 그리고 이를 발전시켜 렌더링에 필요한 쉐이딩 파라메터들을 저장하는 딥프레임버퍼를 만들어 하드웨어 가속을 하는 렌더링 방법 등이 연구되었다[5,6,7]. 이 외에 카메라 또는 3차원 객체의 변화에 따라 광선추적 렌더링을 실시간으로 업데이트하는 방법으로 렌더 캐시(render cache)[8]와 쉐이딩 캐시(shading cache)[1] 등이 제안되었다. 이러한 기존 연구들은 정상적인 렌더링에 비해 계산 시간을 큰 폭으로 단축시킬 수 있음을 보여주었지만, 3D 애니메이션 영화 제작에서 사용되는 높은 복잡도를 가진 장면 데이터를 처리하는 데는 실제로 적용하기 어려운 한계를 가지고 있다.

픽사의 Lpics 시스템[7] 역시 딥프레임버퍼를 이용하여 렌더링을 가속하였다. Lpics는 재조명 렌더링 가속을 목적으로 쉐이딩 파라메터를 모두 캐시화하여 이미지 형태로 저장하고, 픽셀당 쉐이딩 계산 시간을 단축하기 위해 복잡한 쉐이딩 모델을 단순화하는 방법을 사용하고 있다. 이를 통해 영화 The Cars의 제작에 실제 적용할 수 있는 수준의 성능과 안정성을 얻을 수 있었다. 하지만 이 시스템은 캐시 이미지 계산을 새로 하는 것을 피하기 위해, 고정된 카메라 시점에서의 렌더링만이 가능하다.

3. 재조명 렌더링 시스템

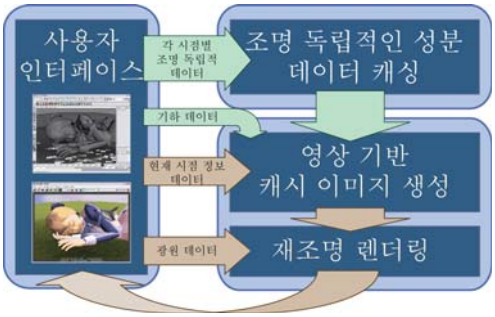

그림 1. 캐시를 이용한 고품질 렌더링 개요

재조명 렌더링 시스템의 전체적인 구성은 그림 1과 같다. 전체적으로 세 모듈로 구성되어 있는데, 조명 독립적 성분(light-independent parameter) 캐시화, 현재 시점에서의 캐시 이미지 합성, 그리고 마지막으로 재조명 렌더링 모듈이다. 조명 독립적 성분 캐시화에서는 장면 내에 미리 지정된 몇 개의 카메라 시점들에서 광원 속성과 관계없는 쉐이딩 파라메터들의 값을 구하여 캐시 이미지를 생성한다. 이 이미지들은 시작 단계에서 한 번만 계산되어 텍스쳐 이미지 형태로 그래픽 하드웨어의 내부 메모리에 올린다.

다음 단계로 현재 카메라 시점의 캐시 이미지가 이미 존재한다면 그대로 재조명 렌더링 모듈로 진행하고, 그렇지 않을

경우 새로운 캐시 이미지를 만들어야 한다. 새로운 캐시 이미지들은 영상 기반 렌더링 기법을 이용하여 기존 캐시 이미지들로부터 합성된다. 마지막으로 이 캐시 이미지들을 받아서 재조명 렌더링 모듈에서는 광원별로 렌더링을 한 후 그 결과를 합성하여 화면에 보여준다.

다음 절에서 각 모듈들에 대해 자세히 설명한다.

3.1 조명 독립적 성분 캐시화

캐시화되는 쉐이딩 파라메터들은 광원 속성에 관계없는 파라메터들이 선택되는데, 이러한 파라메터들은 쉐이딩 모델에 따라서 결정된다. 쉐이딩 모델은 3차원 표면에서 반사광의 색상을 결정하는데 사용되는 계산 모델을 말한다. 쉐이딩 모델에는 단순한 퐁(Phong) 쉐이딩 모델부터 복잡한 BRDF(bidirectional reflectance distribution function), BSSRDF(bidirectional surface scattering reflectance distribution function) 모델까지 다양하게 연구되어 왔다 [9]. 최근 렌더링 분야에서는 대부분의 쉐이딩 모델을 포괄할 수 있는 BRDF와 BSSRDF 모델로 통일되어 가고 있는데, 전역적 조명 효과까지 포함하여 물리적으로 사실적인 반사 효과를 표현할 수 있다. 하지만 다양한 BRDF와 BSSRDF 모델을 실시간에 가속하는 일반적인 방법을 개발하기가 쉽지 않고 디자이너가 사용하기에도 어려운 단점이 있다. 또한 실제 애니메이션 제작에서는 물리적으로 정확한 이미지 보다는 오히려 비사실적 렌더링 스타일이 더 선호되기 때문에 전역 조명 효과의 계산 외에는 BRDF와 BSSRDF 모델을 거의 사용하지 않고 있다. 이러한 이유로 본 논문에서는 이러한 모델들 대신 실제 애니메이션 제작에 참여하는 전문 디자이너들의 요구 사항을 반영하여 선택된 몇 가지 쉐이딩 모델을 사용하기로 한다. 여기에는 램버트(Lambert) 모델, 퐁(Phong) 모델, 그리고 블린(Blinn) 모델이 포함된다.

그림 2. 쉐이딩 파라메터들의 캐시 이미지. 왼쪽 위부터 표면 컬러, 쉐이더 타입, 주변광에 따른 표면 컬러, 난반사 표면 컬러, 전반사 표면 컬러, 법선 벡터, 깊이값, 전반사율이다.

쉐이딩 모델(Lambert, Phong, Blinn 등)에서 광원 속성을 제외한 성분들이 캐시 이미지에 저장되는 파라메터들이 된다. 이러한 쉐이딩 파라메터들은 모두 8가지인데, 여기에는 기본 표면 컬러, 쉐이딩 모델 ID, 주변광(ambient light) 표면 컬러, 난반사(diffuse reflection) 표면컬러, 전반사(specular reflection) 표면 컬러, 법선 벡터, 깊이 값, 전반사율(roughness)이 들어간다 (그림 2 참조).

각 파라메터에 대한 캐시 이미지 생성은 기존 소프트웨어 렌더러를 그대로 이용하여 쉽게 얻을 수 있다. 본 연구에서는 파라메터 값을 그대로 출력 컬러로 보내주도록 소프트웨어 렌더러의 쉐이더를 수정하여 구현하였다. 따라서 한 번에 하나의 파라메터 값만을 얻을 수 있기 때문에 8개의 파라메터에 대한 캐시 이미지를 얻기 위해 8번의 렌더링을 수행하게 된다. 이 캐시 이미지들은 하나의 카메라 시점에 대해서만 유효하기 때문에 n개의 카메라 시점이 지정되면 $n \times 8$개의 이미지가 필요하다. 캐시 렌더링에 걸리는 총 시간은 소프트웨어 렌더러로 1장의 완전한 이미지를 만드는데 걸리는 시간보다 훨씬 많이 걸린다. 하지만, 캐시 이미지들은 각각 따로 계산될 수 있기 때문에 렌더팜(render farm)을 이용하여 시간을 단축할 수 있고, 또한 조명 작업 시작 단계에서 한 번만 만들어 두면 되기 때문에 큰 문제가 되지 않는다.

장면에 범프 맵핑이 사용된 표면이 있을 경우 법선 벡터의 캐시 이미지 생성이 간단하지 않을 수 있다. 소프트웨어 렌더러에 따라 사용자가 현재 픽셀의 3차원 표면점에 대한 법선 벡터를 접근할 수 있는 경우와 그렇지 못한 경우가 있기 때문이다. 후자인 경우라면 캐시 이미지를 만든 후 범프 맵핑에 의한 법선 벡터의 수정이 필요하다. 하지만 이 과정은 소프트웨어 렌더러 마다 사용하는 범프 맵핑의 구현 방법을 알지 못하면 그 결과가 정확하게 일치하지 않을 수 있다 [10]. 우리는 이러한 문제를 피하기 위해 각각 x, y, z 축과 일치하는 3개의 방향성 광원(directional light)을 가지고 렌더링하는 방법을 사용하였다. 먼저 3차원 객체 표면의 난반사 표면 컬러를 모두 1.0으로 설정하고, 쉐이딩 모델을 램버트로 설정한다. 그리고 각각 x, y, z 축을 향하는 밝기가 적색(R), 녹색(G), 청색(B)인 세 개의 방향성 광원을 켜고 소프트웨어 렌더러로 렌더링을 한다. 이 때 계산되는 픽셀의 색상은

$$\begin{aligned}(r,g,b) = {} & (1,0,0) \otimes (\boldsymbol{N} \bullet (1,0,0)) + \\ & (0,1,0) \otimes (\boldsymbol{N} \bullet (0,1,0)) + \\ & (0,0,1) \otimes (\boldsymbol{N} \bullet (0,0,1)) = \boldsymbol{N}\end{aligned} \quad (1)$$

즉 법선 벡터와 같아진다($\otimes$ 는 색상 성분별 스칼라 곱). 색상은 음수가 없기 때문에 법선 벡터에 음수가 있는 경우 0으로 채워진다. 이 문제를 해결하기 위해 -x, -y, -z 축을 향하는 방향성 광원들을 가지고 다시 렌더링을 한 후 구해진 이미지를 앞의 이미지와 합성한다.

3.2 영상 기반 캐시 이미지 생성

카메라 시점이 미리 지정된 시점이 아닌 경우 캐시 이미지를 새로 생성하여야 한다. 소프트웨어 렌더러를 이용해 캐시

이미지를 만들 경우 실시간 렌더링이 어렵기 때문에 우리는 영상 기반 렌더링 방법을 통해 캐시 이미지를 합성하는 방법을 사용한다. 영상 기반 렌더링의 기본 아이디어는 기존에 만들어진 다른 카메라 시점에서의 캐시 이미지에 저장된 정보를 재구성하여 현재 카메라 시점에 대한 캐시 이미지를 합성하는 것이다. 이 과정은 GPU를 이용해 구현 가능하기 때문에 실시간에 캐시 이미지를 만들어낼 수 있다.

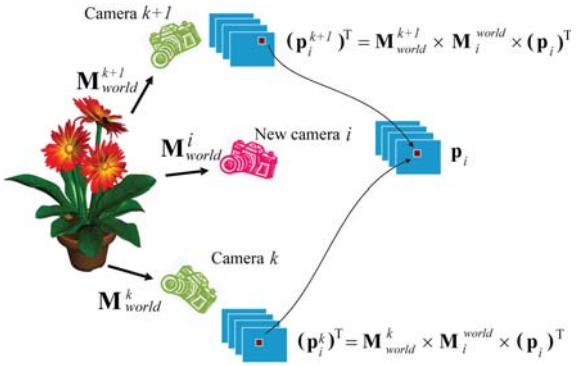

그림 3. 카메라 시점 k 와 $k+1$ 의 캐시 이미지들로부터 새로운 카메라 시점 i 에서의 캐시 이미지 생성.

먼저 새로운 시점에서 각 픽셀에 투영되는 3차원 표면점의 위치를 구하기 위해 GPU를 이용하여 깊이 버퍼를 렌더링한다. 픽셀 좌표 $\mathbf{p}^i$ 와 여기서 구해진 깊이 값 d_i 에 카메라 투영 행렬의 역행렬 $\mathbf{M}_i^{world}$ 을 곱하면 전역좌표계에서의 3차원 위치가 나오게 된다. 이 위치에 카메라 시점 k 의 투영 행렬 $\mathbf{M}_{world}^k$ 을 곱하면 시점 k에서의 대응되는 픽셀 위치 $\mathbf{p}_i^k$ 와 깊이값 d_i^k 가 구해진다.

$$\begin{bmatrix} \mathbf{p}_i^k \\ d_i^k \end{bmatrix} = \mathbf{M}_{world}^k \cdot \mathbf{M}_i^{world} \cdot \begin{bmatrix} \mathbf{p}_i \\ d_i \end{bmatrix} \quad (2)$$

다음으로 $\mathbf{p}_i^k$ 의 가시성 검사(visibility test)가 필요하다. 시점 k에서 $\mathbf{p}_i$ 가 보이지 않는 점일 수 있기 때문이다. 이 검사는 간단히 d_i^k와 시점 k 의 깊이 이미지(depth image)의 값을 비교하는 것으로 할 수 있다. $\mathbf{p}_i$ 가 시점 k 에서 보이는 점이라면 $\mathbf{p}_i^k$에 기록된 캐시 값들을 새로운 캐시 이미지의 $\mathbf{p}_i$ 위치에 써준다. 만일 여러 개의 시점에서 $\mathbf{p}_i$가 보일 경우 이 시점들에서 가져온 캐시 값들을 평균하여 써 주게 된다. 이를 종합하면

$$C_j(\mathbf{p}_i) = \frac{1}{\sum_{i=1}^{n} v(d_i^k)} \sum_{k=1}^{n} v(d_i^k) C_j^k(\mathbf{p}_i^k) \quad (3)$$

이 된다. 여기서, $v(d_i^k)$ 는 d_i^k 와 깊이 값을 비교하여 가시성을 판단하는 함수이고, C_j 는 파라메터 j의 캐시 값을 나타낸다.

영상 기반 렌더링은 2차원 이미지로부터 얻을 수 있는 제한된 정보만을 사용하기 때문에 필연적으로 정확하게 채울 수 없는 픽셀들(holes)이 나타난다. 특히, 캐시된 카메라 시점에서 많이 벗어나 있거나 3차원 장면의 복잡도가 높을 경우 채울 수 없는 픽셀들이 많이 남아있게 된다. 이러한 픽셀들을 그대로 두면 보기에 좋지 않기 때문에 정확하지는 않지만 주위의 픽셀값을 그대로 복사하여 채워준다. 대부분의 이미지 영역들이 공간적 응집성(coherency)을 가지고 있는 사실을 볼 때, 이 방식은 합리적인 휴리스틱이라고 할 수 있다.

영상 기반 캐시 생성은 GPU를 이용한 멀티패스(multi-pass) 렌더링으로 구현된다. 수식(2)와 수식(3)은 GPU의 단편 쉐이더(fragment shader)로 구현되어 한 패스에 한 쉐이딩 파라메터의 캐시 이미지를 합성 한다. 다음 채우지 못하고 내버려 둔 픽셀들을 주변 픽셀값으로 채우는 과정이 이후 렌더링 패스에서 차례로 처리된다.

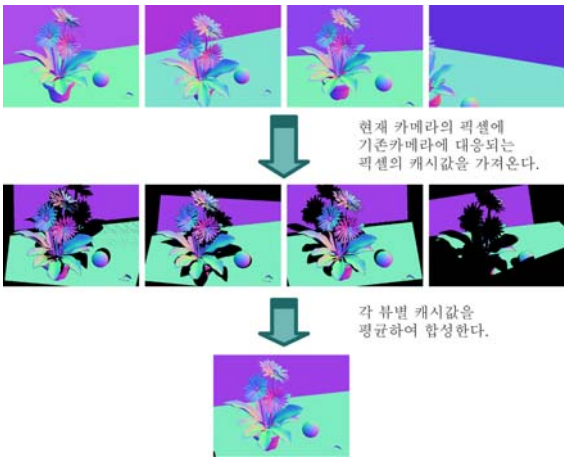

그림 4. 네 개의 카메라 시점에서 가져온 캐시 이미지로부터 합성된 새로운 시점에서의 캐시를 합성하여 얻어진 그림

그림 4는 영상 기반 캐시 생성의 예를 보여주고 있는데, 첫 번째 줄이 캐시된 카메라에서 법선벡터 이미지들이다. 두 번째 줄은 현재의 시점에서 각각의 캐시된 카메라로부터 합성한 캐시 이미지들이다. 맨 밑의 그림은 이 이미지들을 통합하여 얻은 최종 결과이다.

3.3 재조명 렌더링

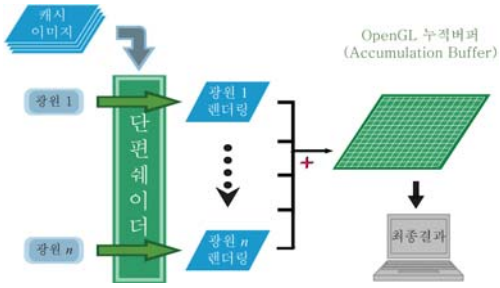

그림 5. 재조명 렌더링 구조. 캐시 이미지를 이용하여 각 광원별로 렌더링한 후 그 결과를 누적버퍼(accumulation buffer)에서 합산함.

재조명 렌더링 단계에서는 앞에서 구한 쉐이딩 파라메터들의 캐시 이미지들을 이용하여 최종 이미지를 렌더링한다. 이 렌더링 방식은 기존의 소프트웨어 렌더러에서 사용해 온 주사선 렌더링이나 광선추적법, 그리고 GPU기반의 렌더링 파이프라인 구조와는 상이한 구조를 가지고 있다. 이 구조는 오직 조명에 관련된 속성의 수정만을 허용하는 대신 렌더링 결과 갱신을 빠르게 하도록 최적화되어 있다. 재조명 렌더링에서는 캐시 이미지에 저장된 렌더링 파라메터 값을 이용하여 장면 내의 광원별로 쉐이딩 모델을 계산하여 픽셀의 색상을 구한다. 광원별로 구해진 결과는 누적버퍼(accumulation buffer)에 반영되어 최종 이미지로 완성된다. (그림 5 참조).

이러한 구조는 확장성과 최적화에 용이하다. 일반적으로 애니메이션 제작에 사용되는 광원의 수는 수십에서 수백개에 이르고, 조명 작업 과정에 수시로 삭제와 추가가 일어난다. 본 재조명 렌더링 구조에서는 이러한 동적인 조명 데이터를 다루기가 매우 쉽다. 또한 사용자는 보통 한번에 하나씩의 광원만 수정하기 때문에 나머지 수정되지 않는 광원들의 광원별 렌더링한 결과는 매번 새로 계산할 필요가 없다. 따라서 이러한 광원들의 결과는 한 번 계산하여 별도의 그래픽 버퍼에 누적하여 보관한 후 재사용하게 된다. 이 경우 평균 렌더링 시간을 광원 수에 관계없이 일정하게 유지할 수 있다.

광원에 의한 그림자의 실시간 렌더링 기법은 그 동안 많은 연구가 되어 왔다. 일반적으로 그림자맵(shadow map)과 그림자볼륨(shadow volume)의 두 가지 접근 방법이 있는데, 많은 다각형으로 구성된 복잡한 3차원 장면에서는 그림자볼륨보다는 품질이 떨어지기는 하지만 그림자맵이 성능과 안정성면에서 유리하다. 이러한 이유로 그림자맵 방식을 선택하여 광원별 렌더링에 그림자 렌더링 효과를 추가하였다. 그림자맵은 광원 이동에 따라 새로 만들어 주어야 하기 때문에 실시간으로 GPU를 이용해 OpenGL을 이용해 렌더링하였다.

4. 결과

본 연구 결과는 인텔 듀얼코어 3.0Ghz, 2GB 메모리, Nvidia QuadroFX 5500시스템에서 OpenGL을 이용하여 구현되었다. 본 재조명 렌더링 시스템의 사용자 인터페이스는 별도로 개발하는 대신 실제 디자이너들이 익숙한 애니메이션 시스템인 Autodesk Maya 7.0의 사용자 인터페이스를 그대로 사용하도록 하였다.

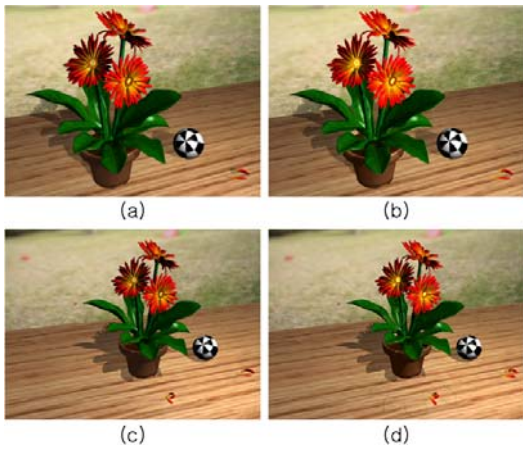

그림 6. (a) 고정된 시점에서 소프트웨어 렌더러로 렌더링한 이미지. (b) 그림2의 캐시 이미지들을 이용하여 렌더링한 이미지. (c) 이동한 시점에서 소프트웨어 렌더러로 렌더링한 이미지 (d) 이동한 시점에서 영상 기반으로 실시간 생성된 캐시를 이용하여 렌더링한 이미지

첫 번째 테스트는 186,143개의 삼각형으로 구성된 화분 예제이다. 비교적 가벼운 예제로 Maya의 소프트웨어 렌더러로 640x480크기로 렌더링할 경우 20초가 걸리는 반면, 재조명 렌더링을 할 경우 0.03초가 걸려서 667배의 속도향상이 있음을 알 수 있다. 고정 시점이 아니고 시점을 바꿀 경우 캐시 생성에 걸리는 시간으로 인해 약간 느린 0.05초가 걸렸다. 그림 6을 보면 소프트웨어 렌더러로 렌더링 하여 얻어진 이미지(a)와 재조명 렌더링을 통해 얻어진 이미지(b)의 차이가 거의 없음을 알 수 있다. (c)는 시점 이동한 시점에서 소프트웨어 렌더러로 렌더링한 이미지이며, (d)는 이동한 시점에서 영상 기반으로 실시간으로 생성된 캐시를 이용하여 재조명 렌더링 한 이미지이다.

두 번째 테스트는 극장용으로 제작되는 실제 애니메이션 장면을 예제로 하여 수행하였다. 이 장면의 다각형 수는 1,404,052개이고 총 크기 473MB의 421개의 텍스쳐 이미지들이 사용되는 비교적 무거운 예제이다. 이 장면을 Maya 소프트웨어 렌더러로 렌더링 하는 데는 16분 54초가 소요되었다. 재조명 렌더링을 할 경우 0.05~0.0625초가 걸려서 약 16,000배 이상의 속도 향상을 보였다. 이 예제의 경우 일반 GPU의 렌더링 파이프라인을 이용해 OpenGL 렌더링을 시도해보았으나 GPU의 처리용량을 넘어서서 렌더링에 실패했다. 이 사실은 재조명 렌더링이 실제 애니메이션 제작에 있어서도 얼마나 효과적으로 사용될 수 있는지를 말해준다. 예를 들어 한 시간 정도의 조명 작업 동안에 기존 소프트웨어 렌

더러를 사용할 경우 최대 11번 정도의 피드백이 가능했는데, 재조명 렌더링에서는 이러한 피드백 횟수의 제한이 없다. 이 예제의 결과 이미지는 저작권으로 인해 논문에 포함시키지 못했다.

5. 결론 및 향후 계획

본 논문에서는 딥프레임버퍼를 이용한 재조명 렌더링 시스템을 구현하였다. 이 시스템은 쉐이딩 파라메터를 캐시 이미지로 만들고, 이를 이용하여 GPU의 단편 쉐이더에서 쉐이딩 계산을 하도록 하였다. 본 연구에서는 카메라가 이동할 경우 캐시 생성을 효율적으로 하기 위해 영상 기반 렌더링 기법을 도입하였다. 이 방법은 카메라 이동 중에 캐시 이미지를 미리 캐시화한 몇 개의 시점으로부터 실시간으로 합성할 수 있다. 따라서 기존 재조명 연구 결과들이 가졌던 고정 카메라의 한계를 해결하였다.

제안된 재조명 시스템은 실제 애니메이션 제작 과정에서 장면 내의 조명 작업에 실험적으로 적용되어 실용성을 확인하였다. 특히 대용량의 데이터를 가지는 장면에서 매번 조명 결과 확인에 수분에서 수십분이 걸리던 시간을 절약하여 제작 시간 단축에 큰 도움이 됨을 확인하였다.

현재 재조명 렌더링된 이미지는 소프트웨어 렌더러로 얻어진 이미지와 유사한 품질을 보이지만 앨리어싱을 피할 수 없는 문제가 있다. 물론 캐시 이미지 생성에서 안티앨리어싱을 적용할 수 있지만, 이 경우에 오히려 재조명 렌더링의 결과가 부정확해지는 문제가 발생한다. 예를 들어 법선벡터의 캐시 이미지를 안티앨리어싱하여 생성할 경우 서로 다른 표면 위의 법선벡터들이 두 표면의 경계 부분에서 서로 혼합되어 전혀 의미 없는 중간벡터가 나오게 되기 때문이다. 이것은 깊이값의 캐시 이미지에서도 마찬가지이다. 따라서 캐시 이미지에 안티앨리어싱을 적용할 수 없고, 결과적으로 재조명 렌더링 이미지에서의 앨리어싱을 피할 수 없다. 이를 해결할 수 있는 한 가지 방법은 캐시 이미지의 해상도를 높이는 것인데, 이는 그래픽 하드웨어 메모리의 한계와 렌더링 속도 저하가 문제이다.

재조명 시스템의 또 다른 한계는 쉐이딩 파라메터가 고정된다는 점이다. 일반적으로는 애니메이션 제작의 조명 작업이 3차원 객체 표면의 재질 속성을 정하는 작업과 완전히 분리되어 있지 않다. 따라서 광원 수정 과정에서 수시로 표면 재질 속성의 수정도 같이 이루어지게 된다. 만일 쉐이딩 파라메터에 영향을 주는 재질 속성이 수정될 경우 이를 반영하기 위해서는 해당되는 쉐이딩 파라메터의 캐시 이미지를 완전히 새로 생성해 주어야 한다. 여기에는 많은 시간이 소요되기 때문에 이를 효과적으로 처리할 수 있는 방법이 연구되어야 한다.

반사, 굴절, 간접광 등의 전역 조명 효과는 아직 재조명 렌더링에서 사용할 수 없다. 애니메이션 제작에서 전역 조명 효과의 비중이 높아지고 있는 현실에서 이것은 큰 제한점이라고 할 수 있다. 하지만 반사와 굴절은 광선추적 단계를 두 단계로 제한한다면 캐시 이미지를 추가함으로써 근사적으로 구현하는 것이 가능하다. 이에 대한 연구는 현재 진행 중에 있다. 그 외에 일반적인 간접광의 경우는 최근에 Hasan등 [11]에 의해 발표된 연구 결과를 적용함으로써 해결하는 것이 가능하다. 이들은 간접광을 효율적으로 계산하기 위해 전계산(precomputation) 단계에서 장면 내에서 샘플링된 표면점 간의 광전달(light transport) 관계를 선형시스템으로 표현하고 이 선형 시스템을 웨이블릿(wavelet) 압축을 통해 크기를 줄여 GPU에서 실시간 렌더링이 가능하도록 하였다.

감사의 글

애니메이션 제작의 조명 작업에 관한 많은 조언과 테스트용 예제 장면을 제공해 주신 임유상 교수에게 감사드립니다.

본 연구는 정보통신연구진흥원의 대학 IT연구센터 육성 및 지원사업의 지원과 선도기반기술개발지원사업의 기능확장형 초고속 렌더링 기술개발 과제에서 지원을 얻어 수행되었습니다.

참고문헌

[1] Parag Tole, Fabio Pellacini, Bruce Walter, and Donald P. Greenberg, "Interactive Global Illumination in Dynamic Scenes", In Proceedings of ACM SIGGRAPH 2002, pp. 537~546, 2002.

[2] Leonard McMillan, and Gary Bishop, "Plenoptic Modeling: An Image-Based Rendering System", In Proceedings of SIGGRAPH 1995, pp. 39~46, 1995.

[3] C. H. Sequin and E. K. Smyrl, "Parameterized raytracing", ACM SIGGRAPH Computer Graphics, Vol. 23, No. 3, pp 307~314, 1989.

[4] T. Saito and T. Takahashi, "Comprehensible rendering of 3-D shapes", ACM SIGGRAPH Computer Graphics, Vol. 24, No. 4, pp. 197~206, 1990.

[5] R. Gershbein and P. M. Hanrahan, "A fast relighting engine for interactive cinematic lighting design" in Proceedings of ACM SIGGRAPH 2000, pp. 353~358, 2000.

[6] J. M. Ragan-Kelley, "Practical interactive lighting design for RenderMan scenes". Undergraduate thesis, Stanford University, 2004.

[7] F. Pellacini, K. Vvidimce, A. Lefohn, A. Mohr, M. Leone, and J. Warren, "Lpics: a hybrid

hardware-accelerated relighting engine for computer cinematography", in Proceedings of ACM SIGGRAPH 2005, pp. 464~470, 2005.

[8] Bruce Walter, George Drettakis, and Steven Parker, "Interactive rendering using the render cache", in Proceedings of the 10th Eurographics Workshop on Rendering, pp. 235~246, 1999.

[9] Matt Pharr, and Greg Humphreys, "Physical Based Rendering: From Theory to Implementation", Morgan Kaufmann, 2004.

[10] Marco Tarini, Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno, "Real Time, Accurate, Multi-Featured Rendering of Bump Mapped Surfaces", Computer Graphics Forum (Eurographics 2000), Vol. 19, No. 3, 2000.

[11] M. Hasan, F. Pellacini, K. Bala, "Direct-to-Indirect Transfer for Cinematic Relighting", ACM Transaction on Graphics (SIGGRAPH 2006), Vol. 25, No. 3, pp. 1089~1097, 2006.

[12] M. Eissele, D. Weiskopf, and T. Ertl. "The G2-Buffer Framework", In Proceedings of SimVis 2004, pp. 287~298, 2004.

[13] Ronen Barzel, "Lighting Controls for Computer Cinematography", Journal of Graphics Tools, Vol. 2, No. 1, pp. 1~20, 1997.

김 순 현

2001년 3월 ~ 2005년 2월 아주대학교 미디어학부 졸업(미디어학학사). 2005년 3월 ~ 2007년 2월 아주대학교 일반대학원 미디어학과 졸업(미디어학석사). 2007년 2월 ~현재 아주대학교 정보통신전문대학원 박사과정. 관심분야는 컴퓨터 그래픽스임

이 주 행

1990년 3월 ~ 1994년 2월 포항공과대학교 전자계산학과 졸업(공학사). 1994년 3월 ~ 1996년 2월 포항공과대학교 전자계산학과 졸업(공학석사). 1996년 3월 ~ 1999년 2월 퍼듀대학교 전자계산학과 졸업(공학박사). 1999년 3월 ~ 현재 전자통신연구원 디지털콘텐츠연구단 선임연구원. 관심분야는 컴퓨터 그래픽스, CAD/CAM임

경 민 호

1989년 3월 ~ 1993년 2월 포항공과대학교 전자계산학과 졸업(공학사). 1993년 3월 ~ 1995년 2월 포항공과대학교 전자계산학과 졸업(공학석사). 1997년 8월 ~ 2001년 7월 퍼듀대학교 전자계산학과 졸업(공학박사). 2002년 8월 ~ 현재 아주대학교 미디어학과 교수. 관심분야는 컴퓨터 그래픽스, CAD/CAM임.