실사 영상물 제작을 위한 렌더링 기술 동향

The Recent Trends of Photo-Realistic Rendering Technologies

융합 시대를 주도할 디지털콘텐츠 기술 특집

장호욱 (H.W. Jang) 렌더링기술연구팀 선임연구원
이주행 (J.H. Lee) 렌더링기술연구팀 선임연구원
정재숙 (J.S. Cheong) 렌더링기술연구팀 선임연구원
이재호 (J.H. Lee) 렌더링기술연구팀 선임연구원
최진성 (J.S. Choi) 렌더링기술연구팀 팀장

목 차

* 본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업의 일환으로 수행하였음. [2006-S-045-01, 기능 확장형 초고속 렌더러 개발]

렌더링은 컴퓨터 그래픽스 장면의 3차원 description(기하 모델, 동작, 카메라, 텍스처, 조명 정보 등)을 2차원 영상으로 생성하면서 사실감을 부여하는 그래픽스 파이프라인의 최종 과정을 말하며, 사람이 직접 그린듯한 느낌의 영상을 표현하는 비사실적 렌더링 기술(non-photorealistic rendering)과 실사 수준의 영상을 표현하는 사실적 렌더링 기술(photorealistic rendering)로 나눌 수 있다. 비사실적 렌더링 기술은 영상의 주요 특징을 잘 표현하여 의미와 느낌을 효과적으로 전달하는 데 목표가 있으며 기술적인 일러스트레이션이나 의료분야 등에 많이 활용되고 있다. 사실적 렌더링 기술 분야는 컴퓨터 그래픽스의 역사와 함께 발전해 오면서 전반적으로 기술의 성숙도가 높은 분야이지만, 사실적인 장면 표현을 위해서는 3차원 질감 표현과 전역조명 처리를 필요로 하여 막대한 처리 시간이 소요되고 있으며, 사용자들의 새로운 요구에 대응하는 기술의 발전이 지속적으로 요구되어 최근에도 활발한 연구가 계속 진행되고 있다. 본 고에서는 사실적 렌더링 분야의 핵심 기술들의 현황과 발전 전망에 대해 살펴보기로 한다.

I. 서론

컴퓨터 그래픽스는 인간이 상상할 수 있는 장면이나 대상물을 표현하거나 실세계에서 실현이 불가능한 것을 디지털로 처리하기 위한 기술로 가상 객체 및 장면의 형상을 만드는 모델링 기술, 가상 객체 및 장면을 현실 세계에서 보이는 모습으로 보이게 하는 렌더링 기술, 물체의 자연스러운 움직임을 생성하는 애니메이션 기술로 구성되어 있다.

렌더링은 수작업에 의한 품질보완이 용이한 모델링이나 애니메이션과는 달리 수작업 대체가 불가능하여 어떤 렌더링 기술을 사용하는가에 따라 성능이 크게 좌우되며 이에 따라 생성되는 영상의 실재감과 품질이 결정된다고 할 수 있다. 과거에 비해 전역조명(global illumination) 및 셰이딩(shading) 기술이 발전하면서 극사실적인 영상의 렌더링이 가능하게 되었으나 이를 위해서는 아직도 많은 계산 시간을 필요로 하고 있다. 렌더링 기술의 발달과 렌더링을 계산하는 컴퓨터 속도의 빠른 발전에도 불구하고 요구하는 영상 품질에 대한 요구도 함께 높아지고 있어 최종 렌더링 소요 시간은 단축되지 않는 상황이다. 실제 하나의 장면에 대하여 렌더링 결과를 얻어내는 데 현재까지도 적게는 몇 분에서 많게는 몇 일까지도 걸리는 이유는 바로 요구되는 영상의 품질이 대폭적으로 증가되었기 때문이다.

사실적 렌더링 분야의 기술 개발은 극사실 표현과 렌더링 고속화로 크게 분류할 수 있다. 극사실 표현 부분은 컴퓨터 그래픽스의 태동과 함께 시작되어 일정한 수준까지 기술적 성숙도가 높아졌으나, 계속적인 새로운 장면 표현에 대한 요구사항의 대두와 영상품질을 높이기 위한 여러 번의 시행착오를 거친 리-렌더링(re-rendering)이 필요하다는 기술적 한계가 아직도 존재하고 있어 이를 해결하기 위한 연구가 계속되고 있다. 렌더링 고속화 부문에서는 셰이더 및 라이팅 디자인 시에 그 정확한 효과를 확인하는 데 상당한 시간을 기다려야 하는 문제를 해결하기 위해 프리뷰 렌더링(preview rendering) 고속화에 대한 연구가 시도되고 있다. 상호작용이 가능한 수준의 속도로 장면의 실제 렌더링 결과를 확인할 수 있는 프리뷰 기능이 제공된다면 전반적인 작업 효율의 향상 및 비용의 절감을 기대할 수 있다.

본 고에서는 사실적 렌더링을 위해 대표적으로 사용되고 있는 핵심 렌더링 기능들을 살펴보고, 콘텐츠 제작자들의 작업 효율을 향상시킬 수 있는 고속 프리뷰 기술 동향과 앞으로의 기술발전 전망에 대해 살펴보기로 한다.

● 용 어 해 설 ●

전역조명: 렌더링 과정에서 장면 내의 오브젝트 상호간의 관계도 모두 고려하는 방법으로 오브젝트간의 상호반사, 굴절, 그림자 효과 등을 재현할 수 있으며, 사실적인 영상을 생성할 수 있으나 많은 렌더링 시간이 걸리는 단점을 가지고 있다.

셰이딩: 3차원 오브젝트의 표면을 처리할 때 조명의 위치와 물체의 기울기, 색깔, 밝기에 반응하여 오브젝트에 음영을 주는 것이다.

Ⅱ. 핵심 렌더링 기술

사실적 영상 렌더링 분야에서는 고품질과 고속화의 두 가지 목적을 한꺼번에 얻기 위한 노력이 계속되고 있으며, 이를 위한 여러 방법들이 개발되어 왔다. 극사실적 렌더링을 위해서는 광학, 물리학, 시각인식, 수학 등 다양한 분야의 깊은 지식과 소프트웨어 및 하드웨어에 대한 종합적인 기술이 필요하다. 이번 장에서는 사실적 영상 렌더링을 위해 대표적으로 사용되는 주요 기법들에 대해 분석한다.

1. 스캔라인 렌더링

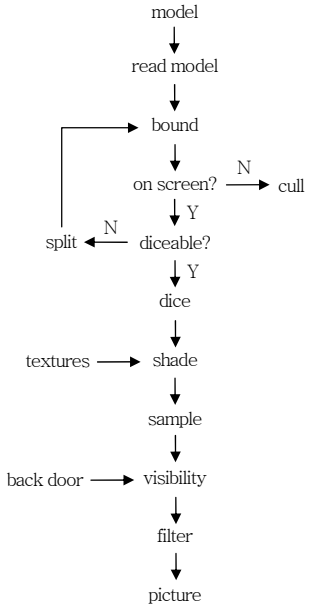
스캔라인(scanline) 알고리듬은 1967년 Wylie 등에 의해 최초의 아이디어가 소개되었다[1]. 기본적으로는 장면 내 객체의 가시성 검사를 위한 알고리듬이지만, 스캔라인 단위의 처리로 인해 계산 속도가 빠르고 메모리 요구량이 적어 초기의 여러 그래픽스 알고리듬과 결합하여 렌더링에 널리 사용되게 되었으며 이러한 렌더링 방식을 넓게 스캔라인

렌더링이라고 한다.

대표적으로 널리 알려진 스캔라인 렌더링 방법은 픽사의 PRMan에서 사용하고 있는 Reyes 렌더링 구조로, Reyes 렌더링 구조는 영화제작에 사용되는 복잡한 장면을 고품질로 비교적 빠른 시간에 처리하는 목표를 가지고 마이크로 폴리곤(micro polygon) 단위 처리를 하는 과정을 수행하며, 1987년 Cook [2] 등에 의해 SIGGRAPH에서 발표되었으나 실제로는 이미 1982년에 "스타트랙 II: 칸의 분노"의 제네시스 특수효과에 사용된 바 있다.

(그림 1)의 Reyes 기반 렌더링 파이프라인의 동작은 다음과 같이 요약될 수 있다.

(1) Bound: 각 기하 요소의 bounding volume을 이용하여 절두체(frustum) 안에 들어오는지 판별한다.
(2) Split: 크기가 큰 기하 요소를 충분히 작은 크기로 분할한다.(Dice가 가능한 수준으로)
(3) Dice: 분할된 기하 요소를 마이크로 폴리곤 망으로 변환한다. 이때 한 픽셀에 들어오는 마이크로 폴리곤의 수가 명시된 수만큼 되도록 분할(split)되어 있어야 한다.



(그림 1) Reyes 기반 렌더링 파이프라인

(4) Shade: 각 마이크로 폴리곤의 꼭지점에서 셰이더를 호출하여 셰이딩 계산을 수행하고 그 값을 꼭지점에 할당한다.
(5) Visibility: 마이크로 폴리곤 망을 개별 마이크로 폴리곤들로 분리하고, 각각에 대해 가시성 조사를 수행한다.
(6) Filter: 스크린 공간에서 픽셀 단위로 마이크로 폴리곤을 샘플링하고, 이들을 혼합하여 최종 픽셀 값을 계산한다.

마이크로 폴리곤은 일반적인 평면 사각형 형태를 가지며, Reyes 렌더링 구조는 기하요소의 종류(다각형, 스플라인 곡면, 블라비, 곡선, 입자 등)에 상관없이 마이크로 폴리곤 단위로 셰이딩을 할 수 있다는 단순성과 일관성을 특징으로 갖는다. 또한 여러 마이크로 폴리곤들에 대한 셰이딩이 벡터 형식으로 병렬 처리될 수 있기 때문에, 최근에는 멀티 코어 CPU의 등장으로 더욱 성능이 향상되고 있다.

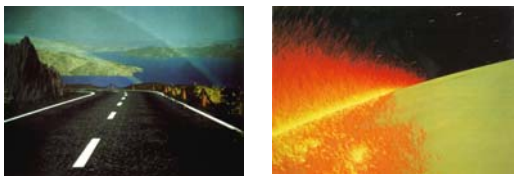
Reyes 렌더링 구조에서 셰이딩 계산 이후에 가시성 조사를 하는 이유는 변위 셰이더의 적용 후에 가시성이 변화될 수 있기 때문이다. 정교한 곡면 표현을 위해 필수적인 변위 맵(displacement map) 렌더링에 있어서 PRMan이 가장 우수하다고 평가받는 것도 이러한 구조에 기인한다. Reyes 구조의 또 다른 장점의 하나는 모션 블러(motion blur)와 피사계 심도(depth of field)의 계산 효율성을 들 수 있는데, hide 단계에서 픽셀 단위로 시간과 렌즈 위치를 몬테카를로 방식으로 추계적(stochastic) 샘플링하기 때문에, 예를 들어 모션 블러가 없는 경우보다 렌더링 시간이 크게 증가하지 않는다[3].

Reyes 렌더링 구조는 처음부터 영화 제작에 사용되기 위해 1980년 중반임에도 불구하고 다음과 같은 고사양의 조건을 만족시키도록 설계되었다.

(1) 100,000개 이상의 장면 요소 지원
(2) 프랙탈, 파티클 시스템, 절차적 모델을 포함하는 다양한 고급 장면 요소의 지원
(3) 단순한 이미지 텍스처 방식의 셰이딩이 아닌 사용자 프로그램 방식의 절차적 셰이딩 지원

(4) 개발 당시의 컴퓨팅 한계를 고려하여 광선 추적법에 대한 최소 지원
(5) 화면 이상(artifact)을 발생시킬 수 있는 알고리듬의 제외
(6) 추후에 개발될 좋은 알고리듬을 추가할 수 있는 구조적 유연성 제공

이와 같이 Reyes 렌더링 구조는 절차적 셰이더, 텍스처 매핑 기법들과 결합하여 광선 추적 및 전역 조명 기법들이 없이도 다양한 효과(그림자, 굴절 등)를 어느 정도 표현할 수 있었으며, PRMan은 업계 최고의 속도, 효율, 안정성으로 복잡한 장면을 표현하며 수많은 애니메이션과 실사 영화에 사용되어 왔다((그림 2) 참조). 픽사가 직접 제작하는 장편 컴퓨터 그래픽스 애니메이션은 출시될 때마다 전작의 표현의 한계를 넘고 있는데, 이때 사용된 기술은 PRMan 새 버전에서의 기능 향상으로 이어지는 순환 구조를 가지고 있다. PRMan과 경쟁 관계에 있는 Mental Image사의 mental ray에서는 스캔라인 렌더링 기능을 제공하지 않고 있다가 렌더링 속도를 높이기 위하여 3.2 버전부터 새로운 방식의 스캔라인 가속기인 rasterizer를 탑재하게 되었다.

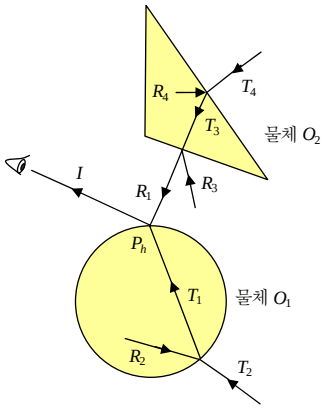


(a) Reyes로 렌더링된 Point Reyes 장면　(b) 스타트랙 II 장면

(그림 2) Reyes 렌더링 결과의 예

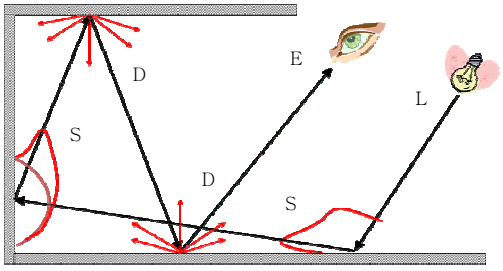
2. 광선 추적법

광선 추적법은 카메라에 보이는 점에서의 색깔에 빛의 직간접 효과들을 반영하는 전역조명의 한 방법으로, 눈에서부터 화면을 향해 광선(ray)을 방출한 다음 이 광선의 굴절, 반사 등을 계산해서 광선이 시작되었던 조명에 이를 때까지의 경로를 역추적하여 각 화면 픽셀의 전역조명 효과를 반영한다. 광선을 추적하는 과정에서 반사와 굴절이 되풀이하여 일어난다고 하여 재귀적 광선 추적법(recursive ray-tracing)이라고도 부른다. Whitted가 제안한 광선 추적법[4]은 그림자광선, 반사광선, 굴절광선이 눈에서 방출된 광선 하나 당 하나씩 가는데((그림 3) 참조) 계산량이 상대적으로 적지만, 아주 이상적인 반사, 굴절과 경계가 뚜렷한 그림자가 얻어져서 그다지 사실적이지 않다.



(그림 3) Whitted 광선 추적법

실제 일어나는 흐릿한 반사와 굴절, 부드러운 그림자들에 대한 사실적인 표현을 위해 분산 광선 추적법(distributed ray tracing)이 Cook[5] 등에 의해 제안되었는데, 모션 블러나 피사계 심도같은 고급 효과도 표현이 가능하다. 분산 광선 추적법은 눈에서 광선을 쏘아 물체와의 교차점에서 반사, 굴절, 빛 방향의 주변으로 여러 광선을 쏘아((그림 4) 참조) 그 광선들을 따라 들어온 효과들을 평균하여 교차점의 색깔 값을 정하는 방법이다.



(그림 4) 분산 광선 추적법

광선 추적법은 전역조명의 효과를 사실적으로 표현해주는 대신 수행 시간이 스캔라인 렌더링에 비하여 많이 걸린다. 또한 렌더링 이미지의 품질은 방출하는 광선의 수에 비례하고, 렌더링 시간은 광선의 수에 비례하기 때문에 많은 렌더링 시간을 필요로 한다.

품질을 저하시키지 않고 광선 추적 시간을 단축하기 위한 몇 가지 방법이 제안되었는데, 그 중 대표적인 방법들로는 효율적인 광선-물체 교차점 찾는 방법, 동일한 위치에서 반사, 굴절을 하는 광선들을 한 다발로 묶어서 처리하는 방법 등을 들 수 있다.

효율적인 광선-물체 교차점 찾기 방법은 우선 반사광선, 굴절광선 등 이차광선이 부딪히는 물체를 찾게 되는데, 이차광선은 어느 물체와 만날지 모르기 때문에, 주어진 광선의 교차 물체를 효율적으로 찾기 위한 자료구조가 필요하며, 이를 위해 대부분 Kd-tree를 쓰는 것이 기본 추세이다. Kd-tree를 사용하면 물체의 메시들을 Kd-tree에 넣어 정리하는 선처리 시간과 메모리 자원이 많이 필요하기는 하지만, 광선 교차점을 자주 찾아야 하는 많은 장면에서는 렌더링 시간이 크게 단축된다. n개의 삼각형이 있는 경우, Kd-tree 만드는 시간은 보통 $O(n \log^2 n)$에서 많게는 $O(n^2)$까지 걸리는데, 보이는 면적을 고려하여 빨리 만드는 SAH를 사용하면 Kd-tree 만드는 시간을 $O(n \log n)$으로 단축할 수 있다.

광선 개수를 줄이지 않고, 광선 추적을 가속하는 방법으로는 다단계 광선 추적법(multi-level ray tracing)[6]이 있다. 모든 광선들은 교차점을 찾기 위해 Kd-tree를 탐색해야 하는데, 각 광선마다 처음부터 탐색하는 대신, 비슷한 곳으로 가는 광선들을 다발로 묶어서 Kd-tree를 탐색시, 그 다발이 함께 갈 수 있는 곳까지는 함께 가고, 필요한 곳에서 개별적으로 탐색하도록 하여 탐색시간을 줄이도록 하는 방법이다.

3. 포톤 매핑

포톤 매핑은 광선 추적법과 달리 빛이 미치는 간접영향을 빛으로부터 시작하여 물체들에 부딪히고 눈에 도달하는 경로를 따라가며 그 효과들을 계산하는 방법으로, 장면의 포톤 값들을 저장하는 선처리를 한 후, 눈에서 광선을 쏜 점에서의 색깔을 저장된 빛의 여러 효과 값들을 참조하여 결정하게 된다.

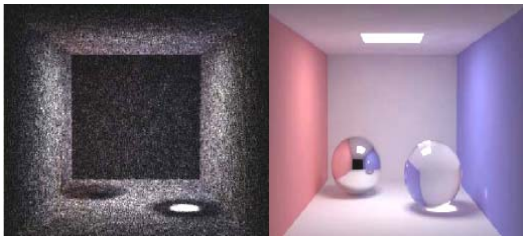
포톤 매핑의 강점은 코스틱(caustic) 현상, 간접조명 효과 등을 표현하는 데 효과적이라는 점을 들 수 있는데, 코스틱 현상은 빛이 유리나 물 같이 투명한 물체를 통과하며 굴절된 빛이 난반사 표면에 모여 무늬를 만드는 현상이다. 간접조명 효과는 빛이 직접 비추지 않는 부분도 다른 물체들에 반사된 빛 때문에 밝게 보이는 현상을 가리킨다. (그림 5)는 이 두 현상들을 보여주고 있다. 이 현상을 광선 추적법으로 표현하려면 아주 많은 광선을 쏘아야 하지만 포톤맵을 이용하면 효과적, 효율적으로 표현이 가능하며, 효율성을 위해 간접조명을 위한 포톤맵, 코스틱을 위한 포톤맵을 따로 만들어 사용하는 것이 일반적이다.

(그림 5) Caustic 현상과 간접조명 효과

구체적으로 포톤 매핑 과정은 다음 세 단계로 나누어진다[7]. 첫째, 광원에서 포톤들을 방출하여 퍼뜨리기, 둘째, 포톤들을 포톤맵이라는 구조에 저장하기, 셋째, 포톤맵의 자료들을 참조하여 렌더링 하기이다.

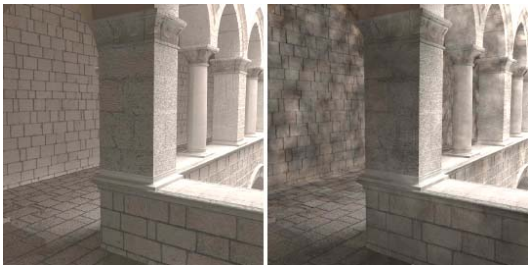
우선, 광원에서 일정한 광량을 가진 광자(photon)들을 사용자가 원하는 수만큼 방출하여 물체들에 부딪히면서 반사, 굴절되는 과정들을 거치면서 광자들은 물체 표면에 붙거나 다른 곳으로 튕겨 나가거나(반사) 꺾여 통과한다(굴절). 이런 과정이 끝나면, 어느 표면에 광자들이 붙어 있는지 쉽게 찾기 위해 공간 인덱스 구조, 주로 Kd-tree에 광자들의

자료를 저장하는데, 이 공간 인덱스 구조를 포톤맵이라고 부른다. 렌더링 할 때에는 눈에서 광선을 쏘아서 부딪힌 물체의 주변에서 지정된 반경 안에 있는 지정된 수의 광자들을 고려하여 교차점의 색깔 값을 계산한다. (그림 6)은 포톤맵과 그것을 이용한 렌더링 결과 이미지를 보여준다.



(그림 6) 포톤맵과 렌더링 결과

포톤 매핑만을 사용하여 렌더링하는 경우 얼룩덜룩한 영상이 생성될 수 있는데, 이를 보완하기 위해 파이널 게더(final gather)를 함께 사용하는 경우가 많다. 파이널 게더란 눈에서 방출한 광선과 물체의 교차점에서 여러 방향으로 이차 광선을 다시 쏘아 그 이차 광선과 물체의 교차점에서의 irradiance 값을 가지고 오는 방식이다. 포톤 매핑과 파이널 게더를 함께 사용하여 렌더링하게 되면 포톤 매핑만 사용하여 렌더링한 경우보다 고품질의 이미지를 얻을 수 있다. 물론 파이널 게더에 사용되는 시간이 상당한 만큼 전체 렌더링 시간은 크게 늘어나게 된다. (그림 7)은 포톤 매핑과 파이널 게더를 함께 사용하여 렌더링하는 경우와 포톤 매핑만 사용하여 렌더링하는 경우의 결과 이미지를 함께 보여준다.



(a) 파이널 게더 적용 결과 (b) 포톤 매핑만 적용 결과

(그림 7) 파이널 게더를 적용한 영상품질 향상

많은 계산 시간을 필요로 하는 파이널 게더의 가속을 위한 여러 방법들이 제안되고 있는데, 미리 정해진 점들에서의 irradiance 값들을 캐시로 저장하여 파이널 게더 과정에서 필요할 때 사용하여 가속하거나[8] 거꾸로 계산하여 파이널 게더 탐색시간을 단축하여[9] 가속한다.

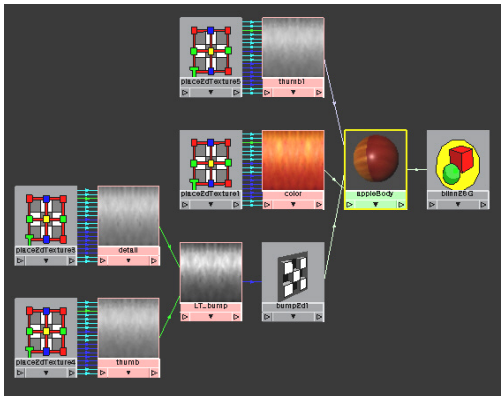
4. 셰이더 지원

렌더링에서 셰이더는 물체 외관의 표현 방식을 기술하고 제어하는 작은 프로그램이며, 렌더러의 중요한 기능적 구성요소이다. 예로 Reyes 구조 기반의 렌더러에서는 마이크로 폴리곤에 대해서 셰이딩 계산을 하게 되는데, 이때 마이크로 폴리곤의 각 꼭지점에서 원래 물체에 할당된 셰이더를 호출하여 색상 값을 결정하게 된다.

렌더링에 의해 객체의 외관이 얼마나 물리적으로 정확히 표현될 수 있는지, 얼마나 창조적이고 예술적인 표현을 유연하게 할 수 있는지는 셰이더의 표현 능력에 의해 결정되며, 이는 렌더러 자체의 성능 평가의 기준이 된다.

현대적인 셰이더는 Cook의 1984년도 논문인 “Shade Tree”에 의해서 출발하였다[10]. 이전의 셰이더들은 각각 매우 큰 독립적인 모듈이었으나, 이 논문은 매우 작은 기본 연산들을 사용자가 조합하여 새로운 셰이더를 쉽게 구성할 수 있는 방법을 제시하였다.

이를 통해, 새로운 조명을 만들거나, 텍스처 이미지에 곡면의 법선 벡터를 기록하여 입력하거나, 최종 이미지에 색상대신 원하는 어떠한 값(곡면의 법선, 깊이 값 등)도 출력할 수 있게 되는 등 새로운 기능 추가가 가능하게 되어 더욱 다양한 표현이 가능하게 되었다. 현대의 렌더러들은 모두 shade tree와 유사한 방식의 셰이더 조합을 지원하며, 대부분의 모델링 도구에서는 이를 위한 GUI 방식의 저작 도구를 제공한다. (그림 8)은 Maya에서 제공하는 hyper shade를 적용하여 사과의 표면을 표현하는 셰이더를 만든 예이다.



(그림 8) Maya의 Hyper Shade 적용 예

셰이더는 그 기능에 따라 구분이 가능한데, 이를 "셰이더 콘텍스트(shader context)"라고도 하며 렌더러에 따라 달라지게 된다. PRMan의 경우 다음과 같이 여섯 가지 종류의 셰이더를 지원하고 있다.

(1) 변환(transform) 셰이더: 기하 정보에 대한 임의의 변환을 제공
(2) 변위(displacement) 셰이더: 정교한 곡면 표면을 위해 곡면 자체를 변형할 수 있는 셰이더
(3) 표면(surface) 셰이더: 표면의 색상과 투명도를 결정
(4) 조명(light) 셰이더: 조명에 할당하여 표면 셰이더에서 한 점에서의 광량을 계산할 때 사용
(5) 볼륨(volume) 셰이더: 안개, 연기, 먼지와 같은 공간감을 표현하기 위한 셰이더
(6) 이미저(imager) 셰이더: 이미지의 픽셀 값이 결정되는 단계에서 호출되는 셰이더. 합성 등에 이용 가능

시장에서 널리 사용되고 있는 렌더러들은 셰이더 언어를 통해 고급 셰이더를 저작할 수 있도록 하고 있는데, PRMan의 RSL이 대표적인 셰이더 언어이며, 사실상 업계 표준으로 여겨지고 있다.

셰이더 언어는 셰이더 컴파일러에 의해 해석되어 셰이더 엔진에서 호출 가능한 기본 연산들로 변환된다. 셰이더 언어가 구현되는 방식은 크게 두 가지이다. 첫번째는 크로스 컴파일 방식이다. 이 방식은 셰이더를 C/C++ 언어로 변환하여 렌더러에 직접 연동되도록 하며, 이 방식은 계산이 효율적일 수 있지만 최적화 컴파일러가 제공되어야 하는 부담이 있다. 두번째 방식은 해석기(interpreter) 방식으로, 구현에 따라 크로스 컴파일러 방식에 비해 더 효율적일 수 있는데, 이를 위해 컴파일러에 의해 생성된 셰이더의 바이트 코드(byte code)를 효율적으로 수행할 수 있는 가상 기계(virtual machine)가 필요하다.

셰이더 언어보다 렌더러 하부 구조에 더 밀접한 수준에서 기능을 확장하는 방법도 제공되고 있는데, PRMan이나 mental ray는 C/C++ 방식의 API가 제공되고 있다. PRMan의 경우 셰이더 언어에서 기본 제공하고 있지 않은 기능을 DLL 형태로 제작할 수 있도록 하고, 이를 셰이더 언어에서 호출할 수 있도록 하였다. PRMan에 광선 추적 기법이 구현된 것은 2003년인데, 그 전에는 PRMan에서 광선 추적이 필요했던 경우는 API 방식으로 광선 추적기능을 구현하여 셰이더 언어에서 호출하여 사용하였다. PRMan에서 사용하는 이러한 방식의 셰이더 기능 확장을 DSO라고 한다. Mental ray의 경우도 API가 활발히 사용되고 있다. 영화 "Matrix Reloaded"의 제작에서, 측정기반 BRDF 셰이더가 이러한 방식으로 구현되어 사용된 것은 유명한 예이다[11].

Ⅲ. 고속 프리뷰 기술

3차원 컴퓨터 그래픽의 렌더링 기술이 도입된 이후로 지난 수십 년간 많은 기술 발전이 있어 왔으며, 이와 더불어 렌더링 속도 또한 많은 향상이 있어 왔다. 그러나, 이러한 속도향상에도 불구하고 시대가 변함에 따라 요구되는 영상의 품질이 이전에 비하여 월등히 증가되어 전체 렌더링 시간은 크게 변화되지 않았다. 이러한 장시간에 걸친 렌더링 프로세스 요구는 작업자들에게 적지 않은 부담으로 존재하기 때문에 작업의 효율성을 저해하는 커다란 요인으로 자리잡고 있다. 특히, 렌더링 프로세스에서 조명의 변화나 셰이더의 변화는 가장 빈번히 발생하는 작업의 형태이며, 이러한 작업에 대하여 그 결과를 점검하

기까지 다시 렌더링을 수행하고 기다리는 행위는 가히 치명적인 효율성 저하를 유발하게 된다. 그런 이유로, 최근 들어 실시간 재조명에 관련된 연구와 실시간 셰이딩에 대한 연구 결과가 발표되고 있다. 이번 장에서는 이중 실시간 재조명 시스템과 셰이딩 인터페이스 시스템에 대하여 소개하고자 한다.

1. 고품질 실시간 재조명 시스템

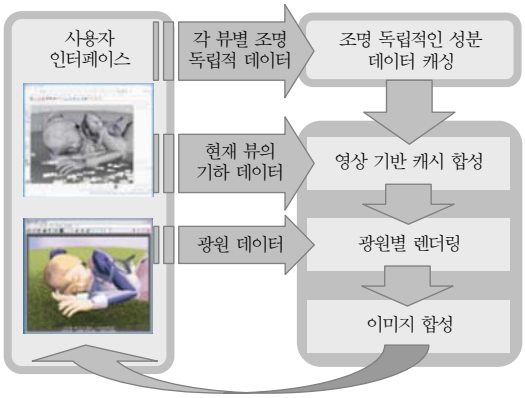
디지털 영상 제작에서 조명작업이란 장면 내에 광원을 적절한 위치에 배치하고 각 광원에 대한 빛의 방향, 세기, 퍼짐 각 등을 조정하여 렌더링 될 장면이 기획한대로 연출되도록 하는 작업을 말한다. 조명을 설정하는 과정은 일반적 예상과 달리 많은 시간과 노력을 요구한다. 가장 큰 이유는 조명을 수정한 결과를 확인할 수 있는 효과적인 피드백 기능이 현재 사용중인 대부분의 제작 도구에서 지원되지 않기 때문이다. 실시간 프리뷰를 수행하는 경우는 렌더링 품질이 좋지 못하여 피드백의 정확성이 많이 떨어지고, 반면에 최종 렌더링(final rendering)에서는 긴 렌더링 시간을 기다려야 하기 때문에 피드백 횟수의 제한이 있다. 이러한 문제는 결과적으로 시각적으로 훌륭한 영상을 얻기 위해서 조명 작업에 많은 인력과 시간을 투입해야 하는 문제를 낳는다.

전문적인 3D 디자이너 한 명이 애니메이션의 한 장면에 대한 조명 작업을 하는 데는 수 일에서 수 주일의 시간이 걸린다. 전체 제작 과정에서 본다면 조명 작업에 소요되는 시간은 표면 재질의 셰이딩 작업을 포함하여 전체 제작 기간의 3분의 1에서 절반 정도를 차지한다. 이러한 시간의 대부분은 중간 결과 확인을 위한 프리뷰 렌더링 시간이기 때문에 이 과정을 가속화 한다면 전체 렌더링 시간을 상당히 단축할 수 있게 되며, 이는 제작 생산성 향상에 크게 기여하게 된다.

일반적인 3차원 장면의 렌더링 가속에 관해서는 그 동안 컴퓨터 그래픽스 분야에서 많은 연구가 이루어져 왔지만, 아직까지 복잡한 장면을 실시간으로 최종 렌더링하는 것은 고성능의 하드웨어를 사용하더라도 매우 어려운 일이다. 따라서, 일반적인 렌더링의 가속에 초점을 맞추는 대신 조명 작업에 특화된 렌더링의 가속으로 범위를 제한하여 실시간 고품질 렌더링을 구현하려는 노력이 나타나고 있으며, 2005년에 SIGGRAPH에서 발표되었던 픽사의 Lpics 시스템[12]이 조명 변화에 대한 실시간 프리뷰가 가능함을 보였다.

조명작업은 대상이 되는 장면 내의 광원을 결정하는 작업으로, 광원 개수, 광원의 세기, 방향, 퍼짐각, 감쇄율, 색상 등이 조명 작업에서 결정되는 속성이다. 조명 작업은 보통 콘텐츠 제작 프로세스의 후반부에 위치하기 때문에 3차원 객체의 모델링 데이터, 표면 재질, 카메라 설정 등은 고정시켜 두고 작업하게 되며, 조명작업 중의 프리뷰 렌더링에서는 많은 부분에서 똑같은 계산이 반복적으로 이루어진다. 이러한 점에 착안하여 실시간 재조명 시스템에서는 고정된 요소들에 의한 계산은 전처리 단계에 미리 계산하여 일종의 캐시 이미지에 저장해 두고, 광원 정보 수정 시에는 미리 계산된 데이터를 그대로 사용하여 프리뷰 렌더링의 성능 향상을 꾀하고 있다.

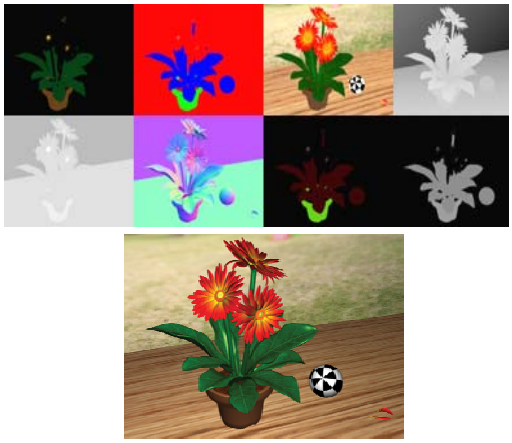
전체적인 시스템은 (그림 9)와 같은 구조로 구현된다. 여기서 재조명 렌더링 시스템은 세 부분으로 구성되어 있는데, 장면 데이터에서 조명 독립적 성분을 미리 계산하여 캐시화하는 모듈(조명 독립적 성분 캐시화), 각 광원별로 렌더링하는 모듈(광원별 렌더링), 최종 이미지를 합성하는 모듈(이미지 합성)들이다. 먼저 사용자가 조명 작업을 시작하면 초기



(그림 9) 실시간 재조명 시스템의 구성도

화 과정으로 조명 독립적 성분이 픽셀단위로 계산되어 여러 장의 캐시 이미지에 저장된다. 다음 과정은 캐시 이미지를 파라미터로 사용하여 각 광원마다 렌더링을 수행하고 중간 이미지들을 얻게 되며, 이 이미지들은 최종적으로 하나의 이미지로 합성되어 프리뷰 윈도를 통해 화면에 보여진다. (그림 10)은 각 캐시의 이미지와 합성 이미지를 보여주고 있다.

(그림 10)의 캐시 이미지를 얻기 위하여 먼저 각 픽셀의 셰이딩 계산에 필요한 파라미터를 미리 추출하여 이미지 형태로 저장한다. 재조명 렌더링 과정에서는 이 캐시 이미지에 저장된 파라미터 값을 이용하여 광원별로 각 픽셀의 셰이딩 계산을 수행한다. 광원별로 계산된 픽셀의 컬러를 모두 더하면 최종 이미지가 얻어진다. 이 방식의 장점은 재조명 렌더링에 걸리는 시간이 장면 복잡도가 아닌 이미지 해상도에 따라 결정된다는 점이다. 즉 아무리 복잡한 장면에 대한 재조명 렌더링도 지정된 이미지 크기에 따라 계산시간이 결정되게 되어 항상 일정한 프레임률을 보장해 준다.



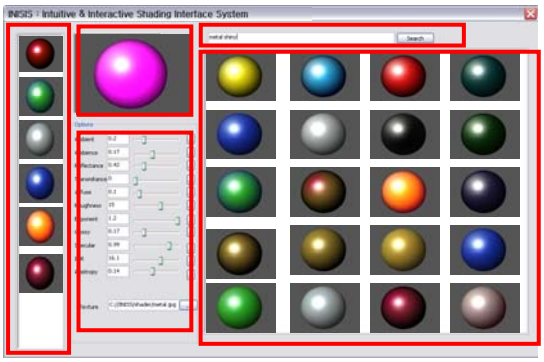
(그림 10) 광원 독립적 캐시 이미지 생성과 합성

2. 직관적 셰이더 인터페이스 시스템

렌더링을 통해 디자인 객체를 보다 실제에 가깝게 보여주기 위해서는 셰이더의 표현이 무엇보다 중요한 작업이 된다. 셰이더란 사물의 질감이나 색상 등 형태를 제외한 사물을 표현하는 요소로써, 실제와 가까운 셰이더를 표현하기 위해 수십 개 이상의 많은 속성(attribute)이 존재한다. 그래픽 디자이너는 이 셰이더 속성들을 조절함으로써 셰이더를 표현하고 렌더링을 통하여 그 결과를 확인할 수 있다.

마야를 비롯한 기존의 시스템들은 사실감 있는 영상을 만들기 위해서 셰이더의 속성들을 일일이 조절해야 하고, 사용자가 셰이더의 각 속성을 조금이라도 조절하면 반복적으로 렌더링을 통하여 그 결과를 확인해야 하며, 이 과정은 많은 처리 시간을 요구한다. 대체적으로 셰이더의 표현이 익숙하지 않은 사용자들은 속성 조절시 시행착오를 많이 겪기 때문에 오랜 시간을 낭비하게 되며, 또한 전문가라고 할지라도 셰이더 속성을 조절할 때마다 렌더링을 수행하여 결과를 확인해야 하므로 작업 시간의 낭비를 피할 수는 없다.

이를 해결하기 위하여 기존의 렌더링 시스템들은 몇 가지 샘플 셰이더들을 제공하고 있으며, 사용자는 제공된 샘플들 중 자신이 원하는 결과와 유사한 셰이더를 고른 후 속성들을 수정함으로써 자신이 원하는 결과를 렌더링하고 있다. 그러나 이 샘플들의 수는 매우 한정적이기 때문에 사용자가 원하는 셰이더가 존재할 가능성이 매우 낮고, 결국 사용자는 자신이 원하는 최종 디자인 객체를 얻기 위하여 수많은 렌더링을 통한 시행착오를 겪어야 하는 문제가 여전히 존재한다. (그림 11)은 직관적 셰이더 인터페이스 시스템(INISIS)의 사용자 인터페이스를 보여준다. 오른쪽에 많은 수의 유사 셰이더를 동시에



(그림 11) 직관적 셰이더 인터페이스 시스템

제공하고 있으며, 선택된 셰이더의 파라미터 변화에 대한 변형 셰이더 또한 리스트로 보여주도록 구성되어 있다. 또한 키워드 검색이나, 셰이더 카테고리 검색, 사용된 셰이더 저장 기능 등도 제공하고 있어 렌더링 작업시 가장 많은 시간이 할당되는 셰이더 작업의 효율성을 증가시킬 수 있다.

직관적 셰이더 인터페이스 시스템은 기존 렌더링 시스템들의 문제점을 해결하기 위하여 사용자가 원하는 셰이더와 유사한 셰이더를 효과적으로 찾아내는 기능을 제공하며, 이는 기본적인 샘플 셰이더들을 카테고리별로 나누어 두고, 사용자가 셰이더 카테고리를 선택한 후 그 카테고리 안에서 원하는 샘플 셰이더를 선택하게 하며, 사용자가 선택한 샘플 셰이더의 속성들을 분석하여 유사한 셰이더의 집합을 사용자에게 보여주는 기능을 수행하게 된다. 이 때, 보여지는 셰이더 집합 중에서 사용자가 원하는 셰이더를 다시 선택하면 그와 좀 더 유사한 샘플 셰이더들을 사용자에게 보여주게 되고, 이 과정을 반복함으로써 사용자가 원하는 셰이더와 거의 흡사한 셰이더를 선택할 수 있도록 유도한다. 이러한 과정은 데이터베이스의 샘플 셰이더들을 사용자에게 보여주는 과정으로서 렌더링이 아니며, 사용자가 선택한 샘플 셰이더를 바탕으로 약간의 속성을 수정함으로써 최종적으로 렌더링을 수행할 수 있게 한다.

이 시스템은 유사 셰이더 검색을 통하여 사용자가 원하는 결과와 흡사한 셰이더를 제공할 수 있으므로 렌더링 횟수를 크게 줄일 수 있고, 사용자들이 만든 셰이더 결과를 데이터베이스에 저장함으로써 샘플 셰이더 데이터 수를 지속적으로 확장 가능하고, 사용자가 원하는 결과를 쉽게 찾을 수 있도록 하는 데 그 목적이 있으며, 전체 렌더링 시간을 크게 감소시킬 수 있는 방법을 제시하고 있다.

Ⅳ. 결론

렌더링 과정은 컴퓨터 그래픽스 파이프라인의 최말단에서 최종 영상의 품질을 결정하게 되며, 가장 많은 처리시간과 비용을 필요로 하는 핵심 요소이다. PRMan과 mental ray 등의 상용 렌더러들은 고속성과 고품질의 두 가지 목적을 한꺼번에 얻기 위한 노력을 계속하고 있으며 하드웨어 성능의 발전과 함께 렌더링 속도도 급속히 빨라지고 있으나, 실제와 구별할 수 없을 수준의 사실적 영상을 생성하기 위해서는 아직도 상당한 렌더링 시간을 필요로 하고 있다.

외국 유명 콘텐츠 제작사들은 고유한 장면 표현을 위해 상용 렌더러와 함께 자체적으로 개발한 인-하우스(in-house) 렌더러들을 함께 사용하고 있어, 상용 렌더러에서 부족한 기능을 보완하고 있으나 국내 콘텐츠 제작사들은 자체 렌더러를 전혀 보유하고 있지 않아, 외국 제품에 의존할 수 밖에 없는 실정이나 그나마 활용 능력이 미흡한 상태로, 새로운 기능 개발을 통한 창의적인 표현에 제약을 받고 있으며, 결과적으로 차별화된 콘텐츠 제작에 어려움이 있는 상태이다.

렌더링이 컴퓨터 그래픽스 파이프라인에서 상당히 중요한 위치를 차지함에도 불구하고, 국내에서 연구/개발/활용을 담당할 전문 인력이 매우 부족한 상태로 렌더링 관련 연구 수행 내용이 해외와 비교할 때 비교적 역사가 깊지 못하고 내용도 다양하지 못한 편이다. 국내에서 요소기술 개발에 대한 시도는 산발적으로 진행되어 왔으나 요소기술을 개발 통합한 렌더러가 개발된 경우는 아직까지 없는 상태이다. 국내 보유 렌더링 관련 기초 기술의 취약성은 국내 제작 영상물의 품질이 세계 최고 수준에 도달하는 데 주요 장애 요소로 작용하고 있으므로 현재의 기술 격차가 더욱 심화되기 전에 렌더링 관련 기술을 외국과 대등한 수준으로 높임과 함께 콘텐츠 제작 업체와 기술 개발 주체간 상호 협력 체제를 갖추어 나가야 할 것이다.

약 어 정 리

BRDF	Bidirectional Reflectance Distribution Function
DART	Designer's Augmented Reality Toolkit

DSO Dynamic Shader Object
INISIS Intuitive and Interactive Shading Interface System
REYES Render Everything You Ever Saw
RSL RenderMan Shader Language
SAH Surface Area Heuristics

참 고 문 헌

[1] C. Wylie, G.W. Romney, D.C. Evans, and A.C. Erdahl, "Halftone Perspective Drawings by Computer," FJCC 67, Thompson Books, Washington, DC, 1967, pp.49-58.

[2] Robert L. Cook, Loren Carpenter, and Edwin Catmull, "The Reyes Image Rendering Architecture," *Computer Graphics(SIGGRAPH '87 Proceedings)*, pp.95-102.

[3] Robert L. Cook, "Stochastic Sampling in Computer Graphics," *ACM Transactions on Graphics*, Vol.5, No.1, 1986, pp.51-72.

[4] T. Whitted, "An Improved Illumination Model for Shaded Display," *Communications of the ACM*, Vol.23, No.6, 1980, pp.343-349.

[5] R. Cook, T. Porter, and L. Carpenter, "Distributed Ray Tracing," *Computer Graphics(SIGGRAPH Proceedings)*, 1984, pp.137-145.

[6] A. Reshetov, A. Soupikov, and J. Hurley, "Multi-level Ray Tracing Algorithm," *ACM Transactions on Graphics*, Vol.24, Issue 3, 2005, pp.1176-1185.

[7] H.W. Jensen, "A Practical Guide to Ray Tracing and Photon Mapping," *SIGGRAPH 2004*, Course 4, Aug. 2004.

[8] P.H. Christensen, "Fast Photon Map Global Illumination," *Journal of Graphics Tools*, Vol.4, No.3, 1999, pp.1-10.

[9] V. Havran, R. Herzog, and H.P. Seidel, "Fast Final Gathering via Reverse Photon Mapping," *Computer Graphics Forum*, Vol.24, No.3, 2005, pp.323-332.

[10] Rober L. Cook, "Shade Trees," *Computer Graphics (SIGGRAPH '84 Proceedings)*, pp.223-231.

[11] G. Borshukov, Measured BRDF in film production: realistic cloth appearance for "The Matrix Reloaded," *in SIGGRAPH Sketches & Applications*, 2003, p.1.

[12] F. Pel lacini, K. Vidimce, A. Lefohn, A. Mohr, M. Leone, and J. Warren, "Lpics: a Hybrid Hardware Accelerated Relighting Engine for Computer Cinematography," *in Proc. of ACM SIGGRAPH 2005*, 2005, pp.464-470.