점집합을 위한 GPU기반 실시간 전역조명 렌더링*

민혜정[0], 김영준
이화여자대학교 컴퓨터공학과
hjmin@ewhain.net, kimy@ewha.ac.kr

Real-time Global Illumination for Point Sets using GPUs

Heajung Min[0], Young J. Kim
Dept. of Computer Science and Engineering, Ewha Womans University

요약

매끄러운 다양체 표면을 표현하기 적합한 점집합을 가시화하는 과정에서 전역조명 렌더링 기법을 적용하면 빛의 다양한 효과가 더해진 사실감 있는 장면을 렌더링할 수 있다. 실시간 전역조명 렌더링 기법 중 하나인 광선 추적법에 대한 지속적인 요구와 그래픽 하드웨어의 발전을 바탕으로 광선 추적법을 위한 전용 GPU 및 프로그래머블 파이프라인이 근래에 소개되었다. 본 논문에서는 이를 기반으로 하여 점집합으로 구성된 모델에 대해 실시간으로 전역조명 렌더링을 수행한다. 이를 위해 이동 최소 자승법을 적용하여 점집합을 부드러운 음함수 표면으로 근사하고, 광선과 표면의 교차 검사 및 교차 지점에 쉐이딩 효과를 적용하여 전역조명 렌더링을 수행한다. 그 결과 48K개의 점으로 구성된 점집합이 포함된 장면을 2차 반사를 포함하여 수십 fps의 속도로 실시간으로 생성한다.

1. 서론

광선 추적법은 빛의 물리적 성질을 고려하여 사실감 있는 3차원 렌더링을 구현하는 장면 가시화 기법이다. 스크린의 픽셀 색을 계산하기 위해 사용자 시점에서 가상의 3차원 장면을 향해 많은 광선을 발생시키고 장면을 순회하면서 광선의 경로에 따른 쉐이딩 효과, 그림자 효과, 반사 효과 등을 계산하여 최종 스크린 픽셀 색을 결정한다 [4].

근래 들어 광선 추적 계산 가속을 위한 RTX GPU 플랫폼이 소개되었는데 이는 장면을 구성하는 자료 구조의 구축 및 탐색을 가속화하도록 지원하고 새로운 렌더링 파이프 라인을 제공하여 실시간 광선 추적법이 가능한 응용 프로그램을 개발할 수 있게 한다 [1].

* 구두 발표논문, 요약논문 (Extended Abstract).
* 본 연구는 연구재단 중견연구자지원사업(2017R1A2B3012701)의 지원으로 수행되었음.

점집합은 위상정보가 요구되지 않아 개념적 단순성을 이점으로 가지는 기하 표현이다. 무작위적인 점 프리미티브로 구성된 점집합 모델을 GPU로 렌더링하기 위해서는 연속적으로 근사된 표면이 필요하다 [3]. 이 표면과 광선의 교차 검사 및 교차 지점에서의 쉐이딩 효과를 계산함으로써 사실감있는 장면을 렌더링할 수 있다 [2].

본 연구에서는 점의 k-최근접 이웃들을 고려한 바운딩 박스를 정의하고, 바운딩 박스 내 점들에 대해 이동 최소 자승법을 사용하여 음함수 표면을 근사한 후, 이에 광선 추적법을 적용하여 전역조명 효과를 표현한 실시간 렌더링 기법을 제안한다. 광선 추적법의 구현을 위하여 RTX GPU 플랫폼상의 DXR 라이브러리를 사용하여 광선 추적법 전용 렌더링 파이프라인을 이용한다. 그 결과 점집합 모델이 포함된 장면에 대해 효과적으로 광선 추적법이 수행되어 실시간 전역조명 렌더링이 가능하도록 하였다.

2. RTX 기반 점집합의 전역조명 렌더링

2.1. RTX 및 DXR

NVIDIA사가 2018년에 소개한 RTX GPU에 포함된 RT 코어 엔진은 바운딩 볼륨 계층구조 순회 및 광선과 삼각형의 교차 검사를 가속화한다.
DXR은 RTX의 광선 추적법 기능이 통합된 DirectX API로서 렌더링 파이프라인의 쉐이더 객체를 관리하고 광선과 기하와의 효율적인 교차 계산을 위해 기하들을 하위/상위 단계로 나누어 관리한다. 하위 단계에는 삼각형 등의 기하 프리미티브들이 포함되고, 상위 단계에는 변환 행렬과 재질 등이 포함된다 [1].

2.2. 광선과 음함수 표면 교차

점집합을 연속적인 음함수 표면으로 근사하기 위해 먼저 각 점에 대해 k-최근접 이웃들이 포함된 AABB를 정의한다. 각 점의 위치와 법선벡터로 참조평면을 정의하고 이에 이웃점을 투영하여 접공간 지역 좌표계를 구성한다. 이동 최소 자승법을 적용하여 이웃점을 참조평면으로 투

영했을 때 평면까지의 거리와 투영점에서 근사된 표면까지 거리 차를 최소로 만족하도록 하는 표면에 대한 다항함수의 계수를 계산하고 각 점의 자료구조에 오프라인으로 저장해둔다.

온라인 렌더링 과정에서는 광선이 AABB에 충돌하면 광선과 표면의 교차검사를 수행한다. 광선을 접공간으로 변환한 후 광선과 표면을 근사한 다항함수와의 교차점을 계산하고 그 결과 유효한 해가 있다면 교차점으로 정의한다.

교차점의 쉐이딩 효과를 계산하기 위해서 필요한 교차점의 법선벡터를 구하기 위하여 교차점에서 음함수 표면의 기울기 벡터를 계산하여 법선벡터 값으로 이용한다.

2.3. RTX 렌더링 파이프라인 적용

RTX 는 삼각형 프리미티브에 대한 교차 쉐이더를 지원하고 있지만 점 프리미티브에 대해서는 교차 쉐이더를 사용자가 구현하여야 한다. 이를 위해 사용자 정의된 바운딩 박스가 필요하고, 광선과 바운딩 박스 교차 시 광선과 바운딩 박스 내 실제 모델과의 교차 여부를 계산하는 교차 쉐이더를 구현한다.

본 연구에서는 각 점을 중심으로 한 k-최근접 이웃들과의 거리로 크기를 계산한 AABB 를 정의하고, 이는 RTX 의 계층 구조로 구축되어 광선과의 최근접 교차 검색이 가속화된다. 광선과 교차되는 최근접 AABB 가 결정되면 RTX 파이프라인의 교차 쉐이더가 실행되고, 2.2 절에서 언급한 광선과 AABB 내 음함수 표면의 교차 검사를 통해 교차 여부 및 교차점과 법선벡터을 계산한다.
교차점이 정해지면 최근접 쉐이더가 실행되고, 점의 위치와 법선벡터을 바탕으로 그림자와 반사등의 쉐이딩 효과를 계산하여 최종 픽셀의 색을 결정한다.

3. 결과

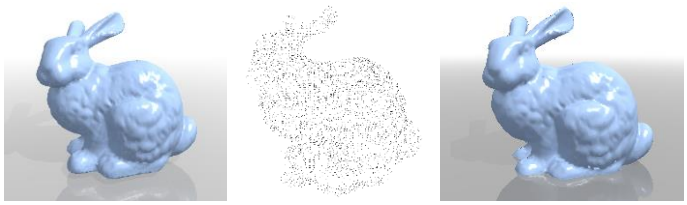

그림 1: (좌) 삼각형 모델을 전역조명 렌더링한 결과, (중간) 36K개의 점집합 모델, (우) 이동 최소 자승법으로 근사된 표면에 전역 조명에 의한 그림자 효과와 반사 효과를 적용하여 점집합 모델을 렌더링 결과이다.

그림1은 NVIDIA사의 RTX 2080 GPU를 이용하여 640 x 640 해상도를 가진 스크린에 전역조명 렌더링을 수행한 결과이다. 36K개의 점으로 구성된 점집합 모델을 전역 조명 렌더링 한 결과 59fps의 렌더링 속도를 보였다.

광선 추적 횟수는 2단계로 하여 그림자와 반사 및 굴절 효과를 처리하였다. 그림자 효과를 위해 광선과 장면 내 기하의 최근접 교차점에서 빛을 향한 방향으로 광선을 추적하여 임의의 기하와 교차가 발생하면 최근접 교차점에 그림자를 표현하였다. 반사 효과를 위해 최근접 교차점에서 반사벡터 방향으로 광선을 추적하여 임의의 기하와 교차 발생 시 교차점 색과 최근접 교차점의 색을 블렌딩하여 렌더링을 수행하였다.

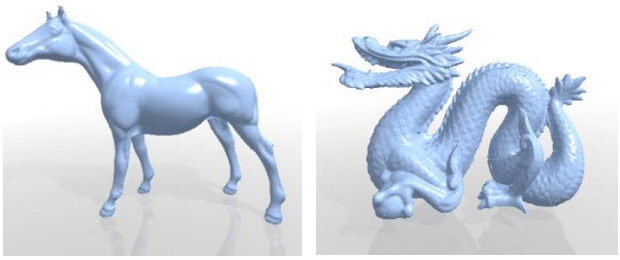

그림 2: (좌) 48K개 점집합, (우) 437K개 점집합

그림 2 는 640 x 640 해상도 스크린에 점집합 모델들에 대해 2 단계 전역조명 렌더링 수행한 결과이다. 렌더링 속도는 48K 개의 점집합의 경우 48fps, 437K 개의 점집합의 경우 4fps 의 결과를 나타냈다.

4. 결론

본 연구는 광선 추적법 가속 엔진이 탑재된 RTX GPU 플랫폼을 기반으로 하여 점집합에 대해 실시간으로 전역조명을 적용해 렌더링하는 기법을 제안하였다.
RTX 에서 지원하지 않는 점집합에 대한 광선 추적법을 적용하기 위해 점집합을 매끄러운 표면으로 근사하고 광선과의 교차 검사를 수행하였다. 또한 다각형 외 점집합 기하가 포함된 장면에 그림자와 반사 효과를 적용하여 사실감 있는 이미지를 생성하였다.

점집합의 개수가 많아질수록 점집합에 포함되는 자료구조의 증가로 인해 그래픽 하드웨어 상 메모리 부하도 높아진다. 이에 추후 대용량 점집합을 위한 전역조명 렌더링을 수행하기 위해서 메모리 관리 및 자료구조 최적화를 적용하여 실용성을 향상시키는 연구를 진행할 것이다.

참고문헌

[1] M. Stich, Introduction to NVIDIA RTX and DirectX Ray Tracing. *https://devblogs.nvidia.com/introduction-nvidia-rtx-directx-ray-tracing/*, 2018.
[2] A. Adamson and M. Alexa, Ray tracing point set surfaces, *Shape Modeling International*, pp.272-279, 2003.
[3] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin and C.T. Silva, Point set surfaces, *IEEE Visualization*, pp.21-28, 2001.
[4] P. Shirley, R. K. Morley, Realistic ray tracing, *AK Peters, Ltd.*, 2008.