메쉬 기반 마칭큐브

김현준[0], 김도훈, 김민호
서울시립대학교 컴퓨터과학과
(kamu1324, kdh0810, minhokim)@uos.ac.kr

Mesh-Based Marching Cubes

Hyunjun Kim[0], Dohoon Kim, Minho Kim
Dept. of Computer Science, University of Seoul

요약

본 논문에서는 삼각형 집합(triangle soup)형식의 등가면(isosurface)을 추출하는 기존의 마칭큐브 기법을 개선하여 연결된 메쉬(connected mesh)형식으로 추출하는 실시간 기법을 제안한다. 이를 통해 기존에는 불가능했던 다양한 렌더링 기법을 사용하여 동적으로 변하는 등가면을 렌더링할 수 있고, 등가면의 연결방식(topology) 등의 정보를 추출할 수 있다. 본 기법은 기존 마칭큐브에 삼각형들을 연결하는 후처리 과정을 추가한 기법에 비해 최대 30%의 향상된 성능을 보인다.

1. 서론

마칭큐브 기법[1]은 볼륨데이터로부터 등가면을 추출하는 가장 고전적이고 보편적인 기법으로, 이후 GPU를 통한 병렬연산 등 이를 개선한 여러 기법이 제안되었다. 하지만 마칭큐브의 결과물은 서로 연결되지 않은 삼각형집합으로 이루어져 있기 때문에 중복된 정점들의 개수가 매우 많고 삼각형의 연결정보가 필요한 고급 렌더링기법을 사용할 수 없다는 등의 단점이 있다. 이를 개선하기 위해 같은 좌표의 정점들을 묶어 삼각형들을 연결하는(welding) 후처리 과정을 추가할 수 있으나[2][3] 이는 정점들의 개수에 따라 성능이 크게 좌우된다는 단점이 있다. (그림3)
본 논문에서는 이러한 마칭큐브 기법을 개선하여 연결된 메쉬형식의 등가면을 실시간으로 추출하는 기법을 제안한다. 우선 볼륨데이터의 엣지(edge)별로 정점들을 추출한 후 이들을 참조한 삼각형들을 생성하는 방식을 통해 정점버퍼(vertex buffer)와 인덱스버퍼(index buffer)를 생성하고 이를 오픈GL을 사용해 렌더링하게 된다.

2. 메쉬 기반 마칭큐브 기법

---

* 구두 발표논문

본 마칭큐브 기법은 다음과 같은 네 단계를 거쳐 등가면을 생성한다. 그림1은 그림2의 마칭스퀘어(marching squares) 예제에 본 기법을 적용했을 경우의 전체 흐름도를 보여준다.

2.1 교차점 검사(intersection test)

오픈CL에서는 할당된 메모리의 크기를 동적으로 늘이거나 줄이는것이 불가능하므로, 생성되는 정점의 개수와 삼각형의 개수를 미리 계산하여야 한다. 이를 위해 우선 셀(cell) 단위로 볼륨데이터를 스캔하면서 해당 셀의 세 방향 엣지에서 각각 교차점이 존재하는지, 존재한다면 몇 개의 삼각형으로 이루어졌는지 검사하여 결과를 각각 엣지테이블(edge table)과 셀테이블(cell table)에 저장한다.

2.2 직렬화 및 압축(serialization and compactification)

이 단계에서는 두 가지 작업을 수행한다. 우선 앞 단계에서 검출한 교차점들을 직렬화하여 정점의 개수를 파악하고 이를 압축하여 정점버퍼(vertex buffer)를 만든다. 그리고 셀테이블을 직렬화하여 전체 볼륨에서 생성되는 모든 삼각형의 개수를 계산하는데, 이 정보는 다음 단계에서 인덱스버퍼를 만들때 사용하게 된다. 이러한 직렬화 과정은 병렬 프레픽스 합계(parallel prefix sum) 기법을 사용하여 빠르게 수행할 수 있다[4]. 본 기법에서는 교차점들을 압축하는 과정에서 법선 벡터도 함께 계산한다.

2.3 마칭큐브

본 단계에서는 기존의 마칭큐브를 이용하되, 정점의 좌표가 아닌 인덱스를 사용하여 삼각형들을 생성하여 인덱스버퍼를 만든다.

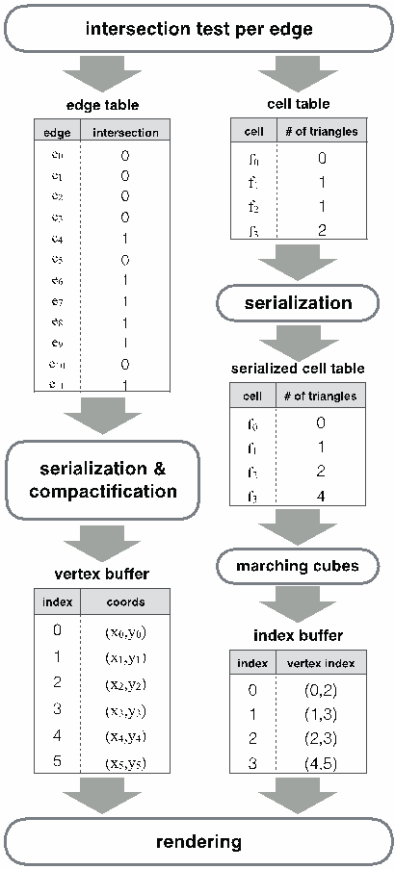

그림1. 전체 흐름도

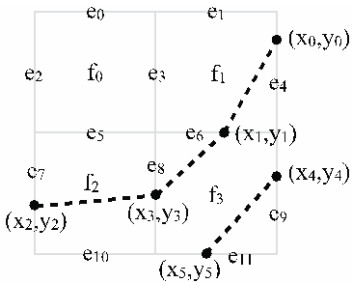

그림2. 마칭스퀘어 예제

2.4 실시간 렌더링

오픈CL과 오픈GL은 그래픽스 메모리를 공유할 수 있다. 앞 단계에서 생성한 정점버퍼와 인덱스버퍼를 오픈GL 파이프라인에 전송하여 실시간 렌더링이 가능하다. 이는 인덱스 렌더링(indexed rendering) 방식이기 때문에 지오메트리 쉐이더(geometry shader)를 활용하는 각종 고급 렌더링 기법을 사용할 수 있다.

3. 퇴화삼각형(degenerate triangles)의 제거

레벨값(isolevel)과 복셀의 값이 같을 경우, 마칭큐브 과정에서 퇴화삼각형이 생성되는 경우가 있다. 이는 해당 복셀의 값에 약간의 노이즈를 추가하여 제거할 수 있다. 이 과정에서 추가적인 연산이 필요로 하지만 그 연산량이 적기 때문에 전체적인 성능에는 큰 영향을 미치지 않는다.

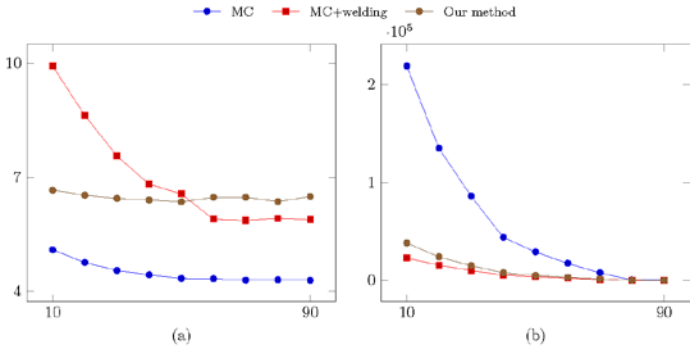

그림3. (a) 레벨값에 따른 연산시간(milliseconds) (b) 레벨값에 따른 정점의 개수

4. 성능 측정 및 결과 비교

본 장에서는 기존 기법과 본 논문에서 제안하는 기법의 성능을 비교한다. 기존 기법에 적용한 후처리 과정(welding)은 Thrust로 구현하였다[2]. 사용한 모델은 128^3 크기의 Hydrogen이다. 그림 3은 (a) 레벨값에 따른 각 기법의 성능과 (b) 정점의 개수를 나타낸다. 기존의 마칭큐브는 연산속도가 가장 빠르지만 중복된 정점이 많이 생기는 것을 알 수 있다. 후처리를 통해 정점을 연결할 경우, 레벨값에 따라 성능의 편차가 심한데 이는 후처리 과정이 정점의 개수에 영향을 많이 받기 때문이다. 이에 비해 본 논문이 제안하는 방법은 레벨값에 상관없이 안정적인 성능을 보이고 중복된 정점도 제거되는 것을 알 수 있다.

5. 결론 및 후속연구

본 논문에서는 기존의 마칭큐브 기법을 개선하여 연결된 메쉬형태의 등가면을 추출하여 기존 기법의 단점을 개선하는 기법을 제안한다. 이후 최적화를 통해 좀 더 성능을 높이는 방법을 연구할 계획이다.

참고문헌

[1] Lorensen, William E., and Harvey E. Cline, *Marching cubes: A high resolution 3D surface construction algorithm,* ACM siggraph computer graphics. Vol. 21. No. 4. ACM, 1987.
[2] N. Bell. High-productivity CUDA development with the thrust template library, 2010.
Conf. 2010, San Jose CA, Sept. 20-23, 2010.
[3] Griffin, Wesley, et al. "GPU curvature estimation on deformable meshes." Symposium on Interactive 3D Graphics and Games. ACM, 2011.
[4] Harris, Mark, Shubhabrata Sengupta, and John D.

Owens. "Parallel prefix sum (scan) with CUDA." *GPU gems* 3.39 (2007): 851-876.