확장성있는 메타볼 다각형화에 관한 알고리즘

이 종 현, 박 규 호
한국과학기술원 전기 및 전자공학과 컴퓨터공학연구실
Email: johlee@fast.kaist.ac.kr

Scalable Metaball Polygonization Algorithm

Jong Hyun Lee, Kyu Ho Park
Computer Engineering Laboratory, Electrical Engineering Department
Korea Advanced Institute of Science and Technology

요 약

컴퓨터 그래픽의 모델링에 있어서 메타볼은 인체를 모델링하기에 매우 적합한 모델링 요소로서 사용되어왔다. 메타볼을 랜더링하는 방법으로서는 광추적기법(ray-tracing)이나, 메타볼을 다각형화여 다각형랜더링가속기를 이용하는 것이 있는데, 전자는 높은 계산량으로 인해서 실시간으로 랜더링하기 어려운 문제가 있고, 후자인 경우에는 모델링 과정에서 메타볼로 구성된 모델로부터 다각형들을 추출하여 이를 랜더링 시에는 랜더링가속기를 이용해서 실시간으로 랜더링 할 수 있는 장점이 있지만, 다각형을 추출하는 과정이 높은 계산량과 많은 메모리를 필요로하므로 랜더링 시 모델의 특성이 변화하는 경우마다 많은 계산량을 필요로 하는 다각형화를 재 수행해야 하므로 실시간 랜더링이 어려운 문제가 있다. 메타볼로 구성된 모델로부터 다각형을 추출하는 방법으로 많은 연구가 진행되었지만, 이들 대부분이 많은 메모리와 높은 계산량을 요구하고, 다각형화를 가속화하기위한 병렬화가 어려운 문제점이 있다. 이로인해서, 본 논문에서는 다음과 같이 두 가지 특징이 있는 메타볼 다각형화 알고리즘에 대해서 제안하고자 한다. 첫째로 기존의 방법에 비해서 적은 메모리를 사용한다. 둘째로 높은 병렬화로 하여금 계산량의 문제를 해결하고자 한다.

1. 서 론

메타볼은 1982년 미국과 일본에서 Blinn[1], Wyvill[12], Nishimura[9]에 의해서 각각 얼룩모델(blobby model), 소프트오브젝트 ,그리고 메타볼이란 이름으로 제안되었다. 메타볼이란 [식1.1]에 나타나있는 바와 같이 3차원 공간상에서 임의의 점을 중심으로, 중심에서 가장 큰 밀도 값을 가지고, 이로부터 밀도 값이 중심으로 부터의 거리 r 에 반비례하는 밀도 분포함수를 가지는 타원체 또는 구이다.

$$D_i(x,y,z) = b_i e^{-a_i r} \tag{1.1}$$

여기서, r 은 구 i 의 중심으로부터 (x,y,z) 까지의 유클리드거리(euclidean distance)이다. r 이 커지게 되면 밀도값은 매우 작아 지게 되어서 계산상의 의미가 없어지는데, 이 경우의 r 을 메타볼의 유효반경이라 한다. 즉 메타볼의 중심점으로부터 유효반경내에서만 밀도값 (D_i) 이 이 식에 의해서 계산되고 유효반경외의 공간에서의 메타볼에 의한 밀도값은 0이된다.

이러한 분포함수를 가지는 메타볼이 공간상에 여럿 위치한 경우 일정한 밀도값(또는 임계치)을 가진 점들은 [식1.2]과 같이 폐곡면(메타표면)을 구성하게 된다

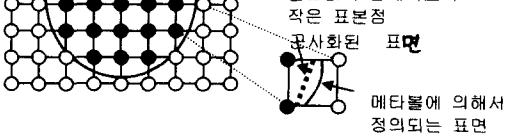
$$Surface = \{(x,y,z) | D_c = \sum_i D_i(x,y,z)\} \tag{1.2}$$

여기서 D_c 는 폐곡면을 정의하는 일정한 밀도값.

메타볼로서 구성된 모델에 대한 랜더링은 이러한 폐곡면을 화면에 표시함으로서 이루어지게 된다. 메타볼로서 구성된 모델에 대한 랜더링방법으로서는 광선추적기법을 이용하는 것이 있는데[1][10], 이는 시점과 화면의 화소를 연결하는 직선과의 폐곡면이 만나는 점을 계산하는 방법인데, [식1.2]에서 알 수 있듯이 비선형방정식의 근을 구하는 것이므로 많은 연산량이 필요하게 되어서 실시간으로 메타볼로 구성된 물체의 랜더링에는 매우 부적합하다. 이러한 실시간 랜더링 문제에 대해서 [식1.2]의 폐곡면을 다각형들로 근사화하는 방법이 제안되었다[4][11][12]. 이 방법은 기존의 광추적기법을 이용한 방법에 비해서 정밀도는 떨어지지만, 모델링 과정에서 추출된 다각형들을 랜더링시 랜더링 가속기를 이용해서 랜더링할 수 있기 때문에 실시간으로 메타볼로 모델링된 물체를 랜더링할 수 있다. 그러나, 기존의 메타볼로 구성된 물체를 다각형화하는 방법은 높은 계산량과 많은 메모리를 필요로 하므로, 물체를 구성하는 메타볼의 위치 같은 속성이 변화하는 시마다, 랜더링과 많은 계산량을 필요로 하는 다각형화를 병행해야하므로 실시간으로 구현하는 것이 매우 어렵다.

2. 메타볼 다각형화에 관한 선행 연구

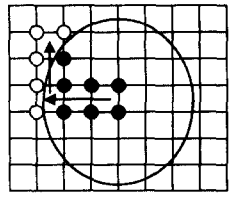
메타볼의 다각형화를 논함에 앞서 이해의 편의를 위해서 2차원상에서 설명하고자 한다.



[그림2.1] 메타볼의 다각형화

메타볼의 다각형화는 [그림2.1]에 나타나 있는 바와 같이, 메타볼을 포함하는 공간에 일정한 간격으로 표본점(sampling point)를 위치시킨 뒤, 표본점에 메타볼들을 정의하는 밀도 분포함수에 의한 밀도값을 저장한 뒤, 이웃하는 표본점들이 메타표면을 정의하는 밀도값보다 한쪽은 크고 한쪽은 작은 경우, 두 표본점 사이에 메타표면이 있다고 가정하고, 위치를 근사화함으로 인해서

전체적으로 근사화된 다면체형태의 폐곡면을 얻는 방법이다. 메타볼의 다각형화는 크게 두 단계로 나누어진다. 첫번째 단계에서는 밀도값을 각 표본점에 저장하고, 두번째는 근사화에 의해서 다각형을 추출하는 과정이다. Wyvill은 메타볼의 다각형화의 첫번째단계인 표본점에서의 밀도값의 계산양을 최소화하는 방향으로 메타볼의 다각형화에 관해서 제안하였다[12].

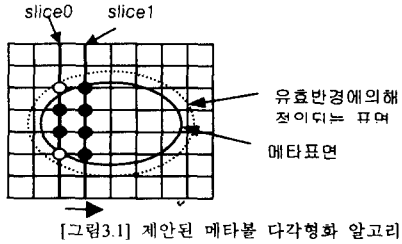


[그림2.2] Wvyill의 메타볼 다각형화의 방법

첫째로 표면에 일정한 간격의 표본점을 할당한뒤, 표본점에서 밀도값을 구하지 않고, 각 표본점에 표본점을 유효반경으로 포함하는 메타볼들의 밀도 분포함수를 linked-list형태로 저장한다. 다음으로 [그림2.2]에 표현되어있는 바와 같이, 각 메타볼들의 중심점으로부터 시작해서 네 개의 표본점들을 취한 후, 각 표본점들에 linked-list형태로 저장된 메타볼들의 밀도 분포함수를 이용하여 표본점들의 밀도값을 계산한다. 만일 네개의 표본점들이 임계치보다 모두 크다면 이웃하는 두개의 표본점을 포함하는 새로운 네개의 표본점을 취하고, 새로 포함된 표본점에서의 밀도값을 계산한다. 만일 네개의 표본점에서 한 개라도 임계치보다 밀도값이 작은 표본점이 있다면, 메타표면이 있다고 가정해서 메타표면을 근사화한다. 3차원인 경우에는 8개의 표본점을 이용해서 메타표면을 근사화하게 된다. Wyvill의 방법은 다음과 같은 단점을 지니고 있다. 첫번째로 메모리의 과다 사용문제다. 초기에 각 메타볼이 포함하는 표본점들에 메타볼을 정의하는 밀도함수를 linked-list형태로 저장하기 때문에 메타볼의 개수가 증가함에 따른 메모리 요구량이 크다. 두번째로 병렬성이 적은 문제가 있다. 즉 표면을 찾기 위해서는 각 메타볼의 중심으로부터 4개의 표본점을 사용하여 표면을 검사하게 되는데, 이러한 방법을 병렬화하여 각 노드당 각기 다른 메타볼로부터 표면을 찾도록 하면, 노드간에 공유할 데이터의 접근빈도가 증가하게 되므로 병렬시스템의 성능이 저하되는 문제가 있다. 이외 다른 방법들은 메타볼을 포함하는 공간에 있는 표본점에 밀도값을 모두 계산하고 난뒤에 다각형을 추출하는 방법에 대해서 다루었으므로 전체 공간의 표본점들에 대해서 밀도값을 저장해야 하므로, 많은 메모리를 사용하는 문제가 있다[4][11]. 이러한 문제에 대해서 본 논문은 이전의 방법에 비해서 적은 양의 메모리를 사용하고, 병렬성이 뛰어난 메타볼 다각형화 방법에 대해서 제안하고자 한다.

3. 제안된 메타볼 다각형화 알고리즘

본 논문에서 제안된 메타볼들로부터 다각형을 추출하는 방법은 Wyvill의 방법과는 다르게 [그림3.1]에 나타나있는 바와 같이 임의의 축과 직교하는 slice단위로 slice내의 표본점들에서의 밀도값을 계산한뒤, 두개의 slice의 표본점들로부터 계산된 밀도값을 이용하여 메타표면을 근사화하는 방법이다.



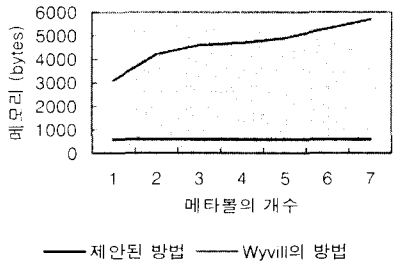
[그림3.1] 제안된 메타볼 다각형화 알고리즘

그러나, slice에서 모든 표본점에서의 밀도값을 계산은 많은 계산량을 필요로 하므로, 이경우에는 메타볼이 정의하는 유효반경내의 표본점에서의 밀도값을 계산하게 되므로, 계산량의 증가는 전체표본점에서의 밀도값을 계산하는 것에 비해서 적다. 제안한 방법을 정리하면 다음과 같다. 첫번째 과정에서는 다각형 생성시 필요한 밀도 정보의 단면(slice)들을 생성하고, 두번째 과정에서는 이러한 단면들로부터 다각형을 추출함과 동시에 다각형의 개수를 줄이는 과정이다. 세번째 과정에서는 두번째 과정에 서 추출된 다각형을 저장한다. 제안된 알고리즘은 첫번째 과정과 두번째 과정을 동시에 병행, 두번째 과정에서 필요로 하는 밀도정보단면들만을 첫번째 과정에서 제공하고, 두번째 과정에서 주어진 밀도정보단면들로부터 정합입방체알고리즘(marching cube)[2][7]을 사용하여 다각형을 추출한다. 이로 인해서 기존의 전영역의 표본점들에 대해서 밀도값을 저장하는 기존의 방법들[4][11]에 비해서 적은 메모리를 사용할 수 있고, 독립적으로 몇 개의 단면들로부터 동시에 다각형을 추출할 수 있기 때문에 병렬성이 뛰어난 점이 있다. 다각형을 추출하기 위해서 사용한 정합입방체 알고리즘은 연속한 4개의 밀도정보단면을 사용하여 다각형을 추출하는 알고리즘인데(다각형을 추출하는 것은 2개의 밀도 정보단면으로도 충분하지만, 4개의 밀도 정보단면을 필요로 하는 것은 법선벡터를 계산하기 위해서 필요하다.), 여기에 필요로 하는 밀도정보단면을 생성하는 알고리즘은, 스캔라인 알고리즘과 매우 유사한 방법을 이용하였다. 이는 단면내 모든 표본점들에 대해서 밀도 값을 구하지 않고, 단면내 메타볼들의 존재영역 즉, 유효반경 내에서만 밀도값을 계산하는 것이다. 메타볼의 존재영역(유효반경)은 타원이므로, 타원의 z축에서의 존재 영역의 최대 최소를 구한다. 그리고 최소값으로 버킷정렬(bucket sorting)한다. 그리고 z를 Z축방향으로 표본간격만큼 증가하면서 각 z값에 해당하는 버킷 버퍼에 있는 메타볼들을 z축에 대한 활성 버퍼로 이동시킨다. 이 활성버퍼에 있는 메타볼들은 현 z값에 대해서 X-Y의 단면 또한 타원들이 된다. 이러한 타원들의 y축에 대한 최대 최소를 구한 다음 최소값으로 버킷 정렬하게 된다. 그리고 y값을 Y축 방향으로 표본간격만큼 증가하면서 각 y값에 해당하는 버킷에 있는 메타볼들을 Y축에 대한 활성버퍼에 이동한다. 그리고 이러한 Y축에 대한 활성버퍼에 있는 메타볼들에 대해서 현 y값에 대한 x구간의 최대 최소를 구한 다음, 이 x구간 사이에 각 메타볼들의 밀도 분포를 사용하여 밀도값들을 추적하게 된다. 이렇게 메타볼의 유효반경을 이용하게 되면, 전공간의 표본점들에 대해서 밀도값을 계산하지 않으므로 기존의 방법들[4][11]에 비해서 계산량을 줄일 수 있다. 본 논문에서 제안한 방법은 밀도 단면들별로 독립적으로 다각형을 추출할 수 있다. 즉 전체 밀도 단면들을 분리한뒤, 병렬컴퓨터의 노드별로 할당하게 되면, 적은 크기의 노드간 데이터를 가지고 메타볼의 다각형화를 수행할 수 있다.

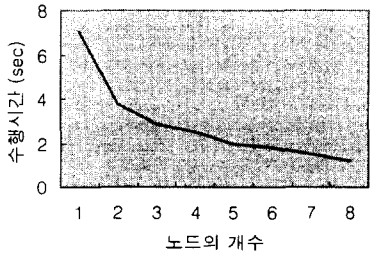
4. 실험결과

[그림4.1]은 메타볼의 개수를 증가함에 따른 메모리의 요구량을 Wyvill의 방법과 제안한 방법인 경우에 따라서 표현한 것이다. 그림에 나타나 있듯이 Wyvill의 방법은 메타볼의 개수를 증가함에 따른 메모리의 요구량이 증가하고 있지만, 제안한 방법은 항상 일정함을 나타내고 있다. 이는 Wyvill인 경우에는 모든 공간의 표본점들에 표본점을 유효반경내에 포함하고 있는 메타볼들에 대한 정보를 포함하고 있으나, 제안한 방법은 모든 공간의 표본점이 아닌 최대 4개의 밀도단면내의 표본점들에 대해서만 밀도값을 저장하고 있으므로, 적은 메모리를 소모하고 있음을 나타내고 있다

[그림4.2]는 제안한 방법을 병렬화시킨 경우, 노드를 증가함에 따른 수행시간을 측정한 것이다. [그림4.2]에 나타나 있듯이 제안된 알고리즘은 노드의 계수를 증가함에 따른 수행시간이 반비례함을 나타내고 있다.



[그림4.1] 메타볼의 개수증가에 따른 메모리요구량의 비교



[그림4.2] 노드의 개수를 증가함에 따른 수행시간의 감소

5. 결론 및 추후 과제

본 논문에서는 메타볼로부터 밀도 정보를 단면단위로 생성할 수 있는 방법을 제안하여 기존의 다각형화 알고리즘을 이용하여 필요한 밀도정보만을 생성, 제공함으로써 메모리를 절감할 수 있게 했고, 단면 단위로 다각형을 추출함으로서 병렬성을 높였다. 실험에서 50개의 밀도 정보단면 (각 단면은 50x50개의 표본점들로 구성)로부터 다각형을 추출하는데 걸리는 시간이 대략 0.9sec(DEC3000기종)가 걸렸다. 소프트웨어에서 이정도의 시간은 하드웨어화 함으로서 20배 이상의 시간을 단축할 수 있을 것이다. 하드웨어화에 관해서 좀 더 생각을 해보면, 메타볼을 다각형화하기 위해서 걸리는 시간의 대부분(80퍼센트)이 다각형 추출시의 법선 벡터와 꼭지점의 위치를 보간하는데 할당되었다. 이 부분을 파이프라인 구조로 전용칩을 사용하면 상당한 시간을 단축할 것이다. 또한 병렬성을 이용하여 확장하게 되면 그 확장하는 모듈 또는 노드의 수만큼의 시간이 줄어 들게 된다.

참고문헌

[1] Blinn J., "A Generalization of Algebraic Surface Drawing", ACM Trans. On Graphics, Vol. 1, No. 3, pp.235-256, July 1982.
[2] Martin J. Durst, Additiona Reference to "Marching Cube", Computer Graphics(SIGGRAPH'88 Proceedings), Vol. 22, pp. 72-73, April 1988.
[3] Gaye L. Graves, "The Art of Metaball", Computer Graphics World, pp. 27-32, May 1993.
[4] Jules Bloomenthal, "Polygonization of Implicte Surfaces", Computer Aided Geometric Design, Vol. 5, No. 4, Nov 1988.
[5] Jules Bloomenthal, Brain Wyvill, "Interactive Technique for Implicite Modelling", Computer Graphics, vol 24, No. 2, pp.109-116, Mya 1990
[6] Nadia Magnenat Thalmann, Prem Kalra, "The Simulation of a Virtual TV Presentor", Proceedings of the 3'rd Pacific Conference on CG&A, Pacific Graphics'95, pp.9-21, Aug. 1995.
[7] William E. Lorensen, Harvey E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm", Computer Graphics, vol. 21, pp. 163-169, July 1987.
[8] Shigeru Muraki, "Volmetric Shape Description of Range Data using Blobby Model", Computer Graphics, Vol. 25, No. 4, July 1991.
[9] Nishimura H, Hirai M. Kawai T. Kawata T, Shirakawa I, Omura K, "Object Modelling by Distribution Function and a Method of Image Generation", Electronics Communication Conference'85, Vol. J68-D, No. 4, 1985.
[10] Wyvill G. and Trotman, "A Ray tracing soft objects", Proceedings of Computer Graphics International, '90, Springer Verlag, 1990.
[11] Jane Wilhelms, Allen Van Gelder, "Octrees for faster isosurface generation extended abstract", Computer Graphics, Vol 24, pp. 57-62, Nov. 1990.
[12] Geoff Wyvill, Craig McPheeters, Brian Wyvill, "Data Structure for Soft Object", Visual Computer Vol.2, No. 4, pp. 227-234, Aug. 1996.