CEMTool 환경에서 3D FEM Visualization

한세경, 박정훈, 권욱현
서울대학교 전기컴퓨터공학부

3D FEM Visualization on CEMTool

Sekyung Han, Jung Hun Park, and Wook Hyun Kwon
School of Electrical Engineering and Computer Sciences, Seoul National University

Abstract - 본 논문에서는 범용 공학 소프트웨어인 셈툴과 연계한 FEM 패키지의 3D Visualization 및 Operation에 대해 기술 한다. FEM 해석을 위한 Graphical Interface로서 각종 오브젝트의 생성, 로드와 더불어 해당 오브젝트를 3D Visualization하여 각종 Manipulation을 하는 기법에 대해 서술하도록 한다. 특히, 해석에 필요한 경계 조건이나 하중을 부여하기 위해 노드를 Picking 하는 기법과, 해석 후 객체의 내부 하중 상태 등에 대한 컨투어링을 위해 객체를 분할하는 Cut Section 등에 대해서도 다루도록 한다.

1. 서 론

FEM해석, 특히 3D FEM 해석에서는 필연적으로 Boundary Condition이나 하중 부여, 그리고 Deforming 등을 시각적으로 보여주기 위해 최신의 3D 기술이 접목되게 된다. 이 때 일반적으로 캐드에서 설계 당시부터 노드의 포지션뿐만 아니라 이 노드를 이용한 기하학적인 물체형상 정보가 도입되게 된다. 본 논문에서는 그러한 기하학 정보가 없는 일반 3D Geometry 파일을 로드하여 동적으로 노드 및 Face picking을 하는 기법과 이를 Tetrahedral 메쉬로 변환하여 FEM 해석을 한 후, 임의의 단면을 보기 위해 객체를 절단하고 절단면을 보간하여 컨투어링을 할 수 있는 알고리즘에 대해 다루도록 한다.

2. 본 론

2.1.1 Basic Node Picking Algorithm

일반적으로 3D 좌표계를 가지는 노드들은 각 오브젝트의 중심을 기준으로 디자인이 되어 있다. 이렇듯 오브젝트마다 서로 다른 Origin으로 디자인된 노드들을 하나의 공통 Origin으로 수정하는 작업을 보통 World Transform이라고 한다. 그리고 이렇게 공통 Origin 을 가지게 한 후 물체를 관측하고자 하는, 즉 카메라의 위치에 맞도록 객체를 카메라 위치의 역으로 변환하는 것을 View Transform이라고 한다. 마지막으로 이렇게 변환된 3차원 노드들을 2차원의 스크린 상에 맵핑하는 과정을 Projection Transform이라고 한다. 결국 하나의 노드는 World, View, Project의 세 가지 변환을 하여 화면상에 랜더링이 되는 것이다. 이 때 이러한 노드 중 하나를 마우스로 유저가 선택할 수 있게 하는 작업을 바로 Picking, 특히 노드를 선택하는 것을 노드 Picking이라고 한다.

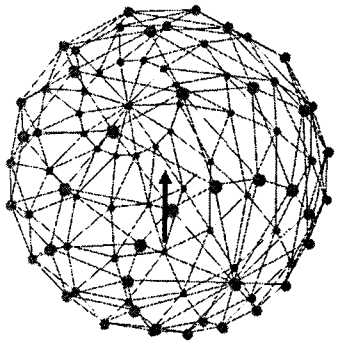
노드 Picking에 있어서 첫 번째 고려해야 할 사항이 2차원의 마우스 좌표를 3차원화 하는 것이다. 특히 이를 위한 전 처리 작업으로 우선 스크린 상의 마우스 X,Y 좌표를 3D World를 비추는 카메라의 2차원 좌표로 Scaling을 해준다. 그런데 이 때 카메라상의 좌표는 Projection이 된 상태의 좌표기 때문에 깊이 정보가 없게 된다. 따라서 해당 좌표에서부터 무한하게 카메라가 바라보는 쪽을 향하여 빛을 쏘고 이 빛과 Intersect하는 노드를 찾는 것이 기본적인 Picking에서의 개념이다.

이 때 빛의 진행 방향에 대한 방향 백터는 앞서 스케일링 된 좌표를 랜더링 과정의 반대로 Inverse View Transform을 하고 다시 Inverse World Transform을 하여 구할 수 있게 된다. 또한 이 빛의 Origin은 통상적으로 방금 구한 역변환 행렬의 Transform Element들로 이루어지게 된다.

다음으로는 이렇게 얻은 빛의 원점과 진행방향 벡터를 이용하여 이 빛을 각 노드와 Hit Test하여야 하는데, 이 때 실제 노드는 부피를 가지고 있지 않으므로 Hit Test를 할 수 없게 된다. 따라서 각 노드를 중심으로 일정 크기의 반지름을 지니는 구를 상정하여 이 구와 빛 간의 기하학적인 Hit Test를 통해 어떤 노드가 선택되었는지를 판단하는 것이다. <그림 1>은 이러한 방법을 도식한 것으로 구 형태의 오브젝트에 각 노드를 일정 볼륨을 가지도록 구의 형태로 확대하였다.

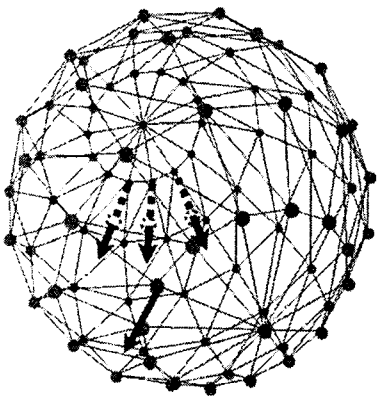
2.1.2 Excluding of Invisible Node

2.1.1에서 설명한 것과 같이 오브젝트의 모든 노드에 대해 가상의 빛과 Hit Test를 하여 선택된 노드를 Picking하면 되는데 이 때 한가지 문제가 발생하게 된다. 바로 Invisible한 영역에 있는 노드를 어떻게 배제할 까 하는 것이다. 그림 1에서와 같은 3차원 오브젝트에서는 정면에 보이는 부분 외에도 뒷면에 실제로는 보이지 않는 곳에도 노드가 존재하며 무한 진행하는 빛과 Hit Test를 하게 되면 이러한 Invisible한 노드들까지 모두 Picking 되게 된다. 가령 단일 노드를 선택하여야 하는 경우 눈에 보이는 곳에 있는 노드들이 다수 선택될 때에는 빛의 Origin에서 가장 가까운 곳에 있는 노드를 선택하면 되지만 Inivisible한 영역에 있는 노드들만이 선택될 경우 이를 구별해 낼 수 있는 알고리즘이 필요하게 된다. 이를 위해 본 논문에서는 노드 Picking 후 후처리를 통해 선택된 노드들을 선별하게 된다. 이 후처리 과정에서는 우선 Hit Test에 성공하여 선택된 노드 모두를 대상으로 이번에는 해당 노드를 원점으로 하고 진행방향을 카메라 쪽으로 한 무한 진행의 가상 빛을 상정하여 이 빛과 본 오브젝트와의 Hit Test를 실시한다. 그리하여 여기서 Hit Test가 성공하게 된 노드들은 카메라에 가까운 쪽에 오브젝트의 일정 부분이 존재하게 되므로 Invisible한 노드로 판단하여 Picking 대상에서 제외 시킨다. 이렇게 하여 Invisible한 노드를 모두 배제시킨 후 남은 노드들 가운데, Hit Test용 원점과 가장 가까이 있는 노드를 최종적으로 선택하여 Picking 과정을 마치게 된다.



<그림 1> 무한 진행하는 빛과 확대된 노드간의 Hit Test

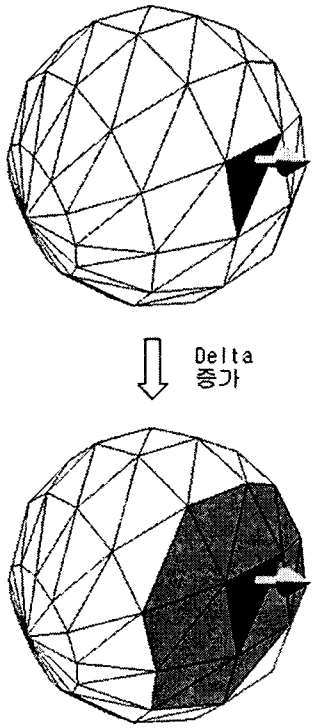
그런데 Invisible 노드를 선별하기 위한 후처리에서 역방향의 가상 빛과 본 오브젝트의 Hit Test는 3차원 상에서 구와 직선의 Intersection과 같이 단순한 형태로 할 수가 없다. 왜냐하면 오브젝트의 형상이 어떻게 이루어져 있는지 모르므로 하나의 기하학적 모델로 표현할 수가 없기 때문이다. 따라서 이를 위해 또 다른 아이디어가 필요하게 되며 이 때 필요한 것이 바로 삼각형과의 Hit Test이다. 형상과 상관없이 3차원 오브젝트를 이루는 기본 단위는 항상 삼각형이므로 구를 이루는 모든 삼각형 Face와 역방향 빛의 Hit Test를 하여 이 중 하나라도 Intersection되는 Face가 존재한다면 Invisible Node로 간주할 수 있게 된다. <그림 2>는 이러한 과정을 도식화한 것이다.



<그림 2> Invisible 노드를 선별하는 후처리 과정

2.2 Dynamic Face Picking

본 절에서는 앞서 언급한 것과 같이 자체적으로 기하학적 Face정보를 지니지 않은 3D 오브젝트에서 Dynamic하게 Face Picking하는 알고리즘에 대해 다루도록 한다. 우선 앞서의 노드 Picking에서와 같이 마우스의 좌표점에서 시작하여 카메라 방향으로 무한 진행하는 가상의 빛을 상정하도록 한다. 그리고 이 빛과 본 오브젝트를 구성하는 모든 최소 삼각형들 간의 Hit Test를 실시한다. 이리하여 스크린으로부터 가장 가까이 있는 삼각형 Face를 선택하고 이 삼각형 Face의 Normal Vector를 구한다. 그리고 이 삼각형 Face에 대해 내부적으로 마킹을 해두고 이 삼각형 Face와 인접한 삼각형들에 대해 Recursive하게 탐색을 시작한다. 그리고 매 탐색된 삼각형들의 Normal Vector를 구하여 최초의 Reference 삼각형에서 구한 Normal Vector와의 내각을 구한다. 그리하여 이 내각이 임의로 정한 작은 값, delta 안에 존재하게 되면 이 삼각형은 우리고 찾고자 하는 기하학적 평면의 일부에 속하는 것으로 간주한다. 그리고 한번 탐색이 된 삼각형 Face는 마킹을 해 두어 재탐색이 되는 것을 방지한다. 이렇게 Recursive한 깊이 우선 탐색으로 통해 모든 삼각형 Face에 대한 탐색이 끝나면 앞서 마킹된 삼각형들로 이루어진 기하학적 평면을 찾아내게 된다. 이 때 delta를 0으로 설정하면 완벽한 기하학적 평면을 얻어내게 되며, delta를 높혀갈수록 차량 본넷과 같이 다소 곡률이 있는 면도 선택할 수 있게 된다. 이는 사용자에게 기하학 면 선택의 자유도를 높혀 주는 효과가 있다. <그림 3>은 이러한 과정을 나타낸 그림이다.



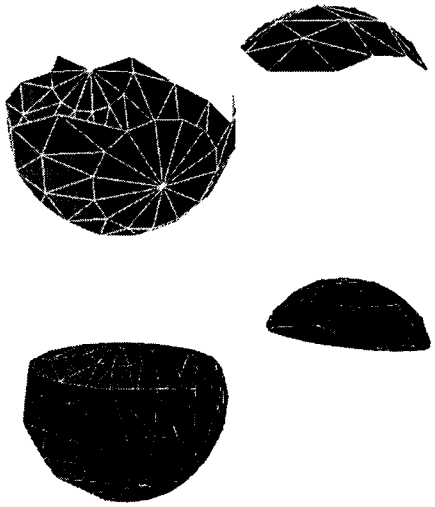
<그림 3> Delta를 조정함으로서 곡률을 지닌 기하학 평면을 선택

3 Dynamic Cut Section

FEM 해석을 한 후에 일반적으로 외관의 스트레스보다 내부의 Stress에 관심을 가지는 경우가 많다. 이를 위해 본 논문에서는 내부 스트레스를 Graphical하게 볼 수 있도록 임의의 단면으로 오브젝트를 절단할 수 있는 기법에 대해 설명하도록 한다. 특히 여기서는 Tetrahedral 형태로 Mesh화된 형태에 대해 다루도록 한다.

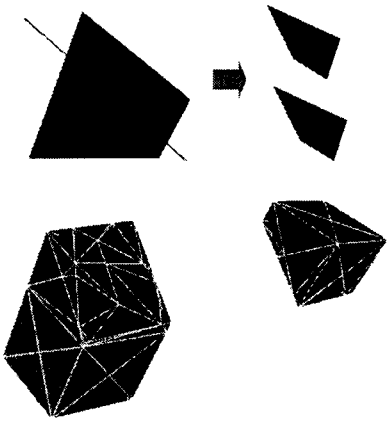
앞서 언급한 바와 같이 기본적으로 3차원 랜더링을 하기 위해서는 3D 오브젝트는 삼각형 형태의 Face가 기본 단위로 구성되어 있어야 한다. 그런데 임의의 단면으로 오브젝트를 절단하게 되면 이 삼각형 구조가 파괴되게 되므로 이를 보정해주기 위해 동적으로 삼각형의 Vertex를 생성하고 이를 연결하여 새로운 삼각형 Face를 생성하여야 한다. 이를 위해 다음 <그림4>와 같이 삼각형 구조가 파괴된 부분에 다시 Vertex를 생성하고 삼각형 형태로 나누어주게 된다. <그림5>는 이런 처리를 하지 않을 경우와 하였을 경우에 보이는 단면 모양을 나타내고 있다.

<그림 4> 잘려진 삼각형 Face의 복원 과정



<그림 5> 삼각형 Face의 복원 과정이 없을 경우 단면

또한 오브젝트 표면상의 삼각형 Face뿐 아니라 내부의 Tetrahedral이 잘려졌을 경우에도 역시 삼각형 구조가 파괴되는 경우가 있으므로 이 역시 동적으로 보간해주어야 한다. <그림6>은 이렇게 삼각형 구조가 파괴되는 절단면과 이를 보간한 후 나타낸 결과이다.



<그림 6> Tetrahedral 절단과 보간후 오브젝트 절단면

3. 결 론

본 논문에서는 3D 오브젝트 자체에 기하학적 면 정보가 포함되지 않은 경우에도 동적인 계산으로 일정부분 곡면을 포함하는 면까지도 Picking해 내었고 또한 임의 단면 생성을 가능하게 하였다. 또한 Invisible 영역에 있는 노드를 선택 범주에서 배제시키는 알고리즘에 대하여서도 논하였다. 이러한 형태의 기법을 사용할 경우 기본 FEM 해석 툴에 비해 훨씬 User Friendly한 UI를 제공할 수 있으며 실제로 이를 구현하여 교육용 FEM 해석 툴을 CEMTool과 연계하여 개발함으로서 초보자도 친숙히 다가설 수 있도록 배려하는 계기가 되었다.

[참 고 문 헌]

[1] Mason McCuskey, "Special Effects Game Programming with DirectX",Premier Press.Inc,2001

[2] Peter j. Kovach, "Inside Direct3D", Microsoft Press, 2000