3D 게임엔진을 위한 객체 간략화에 적용 가능한 EQEM 방법

민경필* · 한태화** · 전준철***

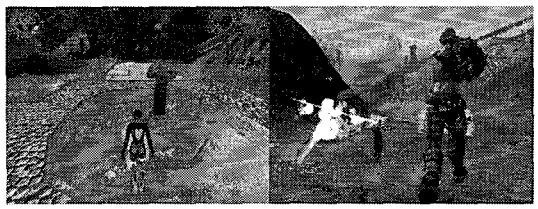
1. 서 론

군사적인 목적으로 처음 이용된 3D 기법은 1970년대에 단순한 와이어 프레임으로 탱크 모양을 표현하거나 벡터 드로잉 스타일에 의한 항공기 조종 시뮬레이터 등에 사용되기 시작했다. 3D 게임은 이런 군사적 목적의 항공기 조종 시뮬레이터로부터 개발되었으나, 당시 하드웨어와 소프트웨어의 한계로 인해 많이 보급되지는 못했다. 90년대 들어서 컴퓨터 하드웨어의 성능이 급속히 발전하면서 3D 게임의 개발이 가능해졌으며, 또한 3D 게임엔진의 개발로 PC상에서도 다양한 게임의 효과를 줄 수 있게 됐고, 반복과정을 좀 더 원활하고 편리하게 작업하는 것이 가능해졌다.

3D 게임엔진에 대한 정의는 아직까지 명확하게 규정되지 않고 있지만 일반적으로 다음과 같은 정의를 포함한다[1].

- 그래픽 · 음악 · 효과음을 출력하고 입력장치에 의한 입력을 받는다.
- 캐릭터를 움직이기 위한 알고리즘을 제공한다.
- 다양한 지형을 처리한다.
- 인공지능 역할을 수행한다.
- 네트워크를 지원하며 네트워크에서 일어나는 여러 가지 사건을 모니터한다.

일반적으로 위의 역할을 수행하면 3D 게임엔진이라 한다. 즉 기본적으로 제공되는 DirectX나 OpenGL 같은 API를 활용하여 보다 쉽고 효율적으로 원하는 게임을 만들 수 있도록 가공 처리한 집합이라고 할 수 있다. 대표적인 3D 게임엔진으로는 그림 1 에서와 같이 퀘이크 엔진, 언리얼 엔진, 리스텍 엔진, 하프라이프 엔진 등이 있다.



(a) 퀘이크 엔진 적용 (b) 언리얼 엔진 적용

(c) 리스텍 엔진 적용 (d) 하프라이프 엔진 적용

그림 1. 각종 3D 게임엔진을 이용한 게임

이러한 3D 게임엔진은 역할에 따라 렌더링 엔진, 애니메이션 엔진, 물리 엔진, 인공지능 엔진, 네트워크 엔진, 3D 사운드 엔진, 맵 에디터 등으로

* 경기대학교 전자계산학과
** 경기대학교 전자계산학과
*** 경기대학교 정보과학부 부교수

구성된다. 렌더링 엔진은 폴리곤 및 폴리곤 메쉬를 생성하는 기능과 3D 객체의 아키텍처를 정의하는 기능, 공간상에서 3D 객체의 위치를 파악하는 기능, 카메라 제공 기능 등이 있다. 애니메이션 엔진은 게임에서 캐릭터가 움직일 때 캐릭터의 행동에 대한 움직임을 시간에 따라 렌더링 해주거나 일정 패턴을 지정하여 움직일 수 있도록 하며, 물리 엔진은 3D 객체의 움직임에 현실감을 부여하기 위해 현실세계에서 적용되는 물리 법칙을 그대로 3D 객체에 적용하는 역할을 한다. 인공지능 엔진은 사용자들이 컴퓨터와 게임을 할 때 마치 사람과 상대하는 것 같은 착각을 불러일으킬 수 있도록 컴퓨터의 행동을 지정하는 역할을 하며, 네트워크 엔진은 네트워크 게임을 위한 모든 기능을 모아서 한꺼번에 수행해주는 역할을 한다. 마지막으로 사운드 엔진은 사운드를 게임에 자유롭게 첨가하고 사용할 수 있도록 제반 환경을 지원하는 기능을 수행한다.

본고에서는 3D 게임엔진 중 중추적 역할을 하는 렌더링 엔진에 대해 살펴본다. 특히, 일반 PC 게임뿐만 아니라 최근 이용자가 급속히 늘고 있는 온라인 게임에서도 3D 그래픽을 적용하기 위해서는 데이터의 실시간 처리 능력을 요구한다. 렌더링 엔진의 기능 중 실시간 객체 출력을 위해 적용되는 알고리즘을 조사해보며, 빠른 렌더링 처리를 위해 3D 게임에 적용 가능한 메쉬 간략화 알고리즘인 EQEM(extended quadric error metric) 방법을 소개하도록 한다.

2. 렌더링 엔진

렌더링 엔진이란 게임 상에 등장하는 각종 그래픽 데이터를 게임 진행 상황에 맞게 화면에 실시간으로 출력해주는 프로그램 모듈이다. 다시 말하면 사용자에게 게임의 진행 상황이나 사용자의 입력에 대한 게임 캐릭터의 반응, 그리고 게임 진행 중에 발생하는 각종 이벤트 정보를 화면에 출력하는 모듈이라고 할 수 있다. 렌더링 엔진의 구성은 그림 2와 같다[2].

2.1 렌더링 엔진의 기능

3D 게임의 제작을 위해서는 렌더링 엔진은 다음과 같은 기능을 제공하여야 한다[3].

2.1.1 3D 객체화면 출력 기능

그래픽 디자이너에 의해 생성된 객체의 형상, 표면 속성, 그리고 객체에 입혀진 각종 텍스처가

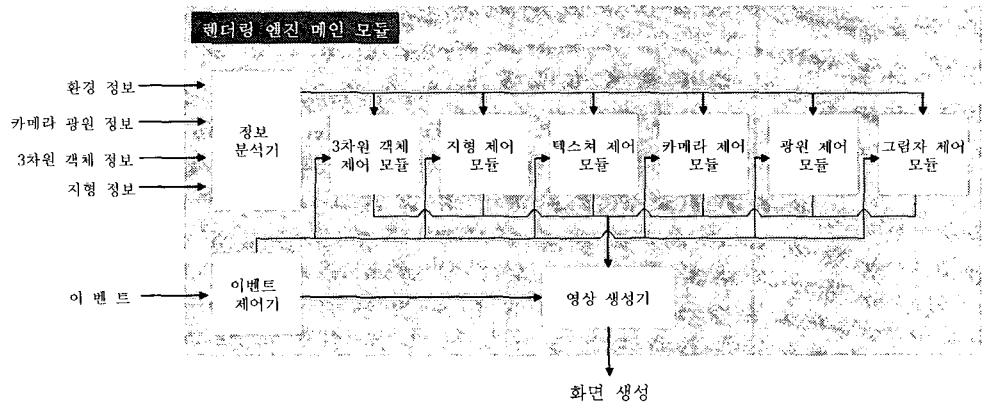


그림 2. 렌더링 엔진의 구성

렌더링 엔진에서 올바르게 처리되어 화면에 적절하게 출력되도록 하며, DirectX나 OpenGL, 최신 그래픽 하드웨어 가속장치의 기능을 최대한 살려내도록 한다.

2.1.2 실시간 객체 출력을 위한 각종 게임 객체 제어 기능

임의의 객체의 다각형의 수가 많아지면 영상 생성에 소요되는 시간이 많아지게 되어 렌더링 속도가 줄어들게 된다. 이러한 현상을 방지하기 위하여 렌더링 엔진은 전체 게임 환경 중 카메라 시야에 나타나지 않는 게임 객체를 제거하거나, 화면에 나타나더라도 아주 적은 수의 화소만 차지할 경우 보다 작은 수의 다각형으로 표현할 수 있도록 한다.

2.1.3 객체 애니메이션 지원 기능

렌더링 엔진은 객체의 움직임을 표현하기 위해 애니메이션 엔진과 함께 통합되어 게임 컨텐츠로 구성되며, 경우에 따라서는 렌더링 엔진 내부에 객체 동작 제어 기능을 갖고 있다.

2.1.4 게임 환경 제어 및 처리 기능

3D 객체의 전체 지형 표현 방식과 지형과 객체 간 충돌 여부를 파악하는 알고리즘들에 해당되는 실내 지형을 위한 방식과 실외 지형을 위한 방식을 처리할 수 있는 기능을 포함한다.

2.1.5 카메라 이동, 회전, 확대, 축소 제어 기능

카메라의 시점을 자유롭게 변형시킬 수 있도록 카메라의 이동 및 회전 그리고 초점거리를 변화시켜 자유롭게 시야를 확대 및 축소하는 기능을 제공한다.

2.1.6 광원 및 그림자 제어 기능

그래픽 라이브러리에서 제공하는 기능 이외에도 메쉬의 정점 컬러를 이용한 광원 그리고 텍스처를 이용한 광원 맵 기능을 제공하여야 하며, 자연스러운 객체의 출력을 위해 그림자 맵 기능을 비롯한 다양한 그림자 생성 및 출력 기능을 제공한다.

2.1.7 각종 쉐이더 활용 기능

그래픽 하드웨어 가속장치 내의 내부 렌더링 연산을 개발자가 직접 작성한 쉐이더를 이용할 경우 이를 렌더링 엔진에 포함시켜 렌더링을 수행하도록 해 주는 기능을 제공한다.

3. 실시간 객체 출력을 위한 LOD 기법

3D 게임에서 임의의 객체를 화면상으로 출력하는 렌더링 속도는 그래픽 하드웨어 가속장치가 처리해야하는 다각형의 개수에 따라 달라진다. 그러므로 빠른 렌더링 처리를 위해서는 객체의 상대적 중요성을 조사하여 다각형 개수를 조절할 필요가 있다. 객체 LOD(level of detatil) 기법은 3D 게임엔진을 제작할 때 데이터의 최적화를 위한 방법으로, 3D 모델의 형상을 상세한 단계에서 단순한 단계까지 물체와 시점 사이의 거리에 따라 여러 단계로 LOD 모델 구조의 데이터로 최적화시킨 후, 카메라와 물체와의 거리에 따라 출력 형태를 조절하는 방법이다.

여러 LOD 기법 중에서 메쉬 간략화 과정은 원래 모델의 형상과 특징을 유지하며 폴리곤의 수를 감소시키는 방법으로 Hoppe[4], Garland와 Heckbert[5], Cohen[6], Luebke와 Erickson[7], Lindstrom과 Truk[8] 등의 기술이 주로 사용되며, 특히 Garland와 Heckbert이 가장 간편하게 사용 될 수 있는 알고리즘으로 많이 이용되고 있다.

3.1 메쉬 간략화를 위한 기하 연산

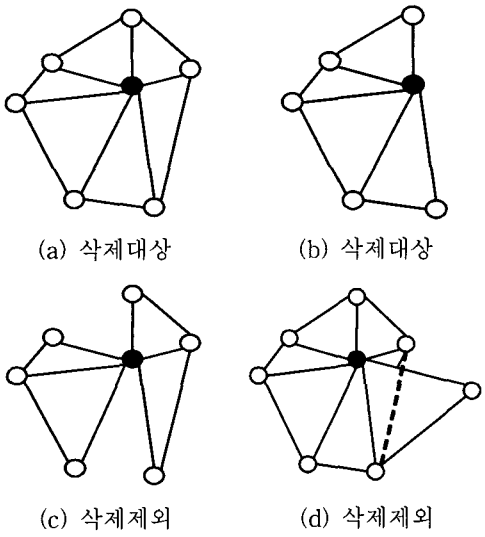
다양한 메쉬 간략화 알고리즘들은 메쉬를 구성

하는 기하학적 데이터를 줄이는 방법에 따라 크게 세 가지 기하 연산의 범주로 나눌 수 있다.

3.1.1 정점 제거

Schroeder의 간략화 알고리즘[9]이 대표적인 정점 제거 방법(vertex decimation)으로 반복적인 간략화 방법을 적용한다. 주어진 대상 메쉬에서 정점을 제거하고 생긴 구멍에 대해 다시 메쉬를 구성하는 방법이다.

그림 3과 같이 정점 제거 방법은 먼저 주어진 기준에 의하여 정점들을 분류하여 다양체와 비다양체로 나누어 다양체에 속하는 제거할 정점을 선택하고, 선택된 정점을 제거하여 그 정점을 공유하고 있는 모든 삼각형들을 삭제한다. 그리고 정점의 삭제로 생긴 홀에 삼각화 알고리즘을 적용하고 이와 같은 과정을 반복적으로 수행한다. 정점 제거 방법은 결과 구멍에 재삼각화 알고리즘을 적용해야 하는 비용이 추가적으로 들뿐 아니라 다해상도 렌더링 시스템에서는 제약점을 가지게 된다. 또한 모델의 전역적 위상을 고려하지 않고 지역적인 위상만을 유지하면서 메쉬를 간략화 하기 때문에 간략화된 결과 메쉬의 질이 떨어지는 단점이 있다.



(a) 삭제대상 (b) 삭제대상

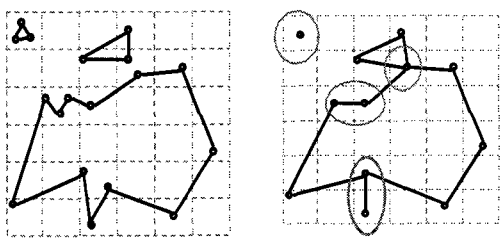
(c) 삭제제외 (d) 삭제제외

그림 3. 제거 대상 정점의 분류

3.1.2 정점 군집화

정점 군집화 방법은 정점의 집합을 군집들의 집합으로 공간적으로 분할한 후 같은 군집내의 모든 정점들을 통합한다. 정점 군집화 방법은 Luebke[7]의 HDS(hierarchical dynamic simplification) 알고리즘이 대표적인 응용 예이다.

그림 4와 같이 정점 군집화 방법은 초기 메쉬를 감싸는 바운딩 박스를 만들고 일정한 간격으로 바운딩 박스를 분할한다. 그리고 정점트리를 생성하는데, 이때 정점트리의 각 노드는 그 노드안의 정점들 중에서 가장 곡률이 큰 정점을 대표 정점으로 정한다. 정점 트리는 간략화 과정에서 생성되는 간략화된 메쉬의 기하학 정보를 대표 정점을 이용하여 얻어낼 수 있다. 따라서 이 방법은 비교적 빠른 간략화 속도를 지원한다. 그러나 간략화 수행 과정에서 메쉬의 형태에 영향을 덜 미치는 정점이 더 크게 영향을 주는 정점보다 나중에 간략화 되는 경우가 발생하여 결과 메쉬의 질을 저하시키게 된다.



(a) 정점 군집화 전 (b) 정점 군집화 후

그림 4. 정점 군집화를 이용한 간략화

3.1.3 에지 축약

에지 축약은 에지를 구성하는 두 개의 정점을 하나의 정점으로 만들어 내는 단순한 작업으로 메쉬의 기하학적 데이터 수를 줄여 나간다. 따라

서 앞의 정점 삭제 방법에서와 같이 재삼각화 알고리즘을 적용하지 않아도 된다. 이 방법은 Hoppe[4], Garland[5] 등의 간략화 기법에서 사용되며, 에지 붕괴라고 불리기도 한다. 에지 축약을 사용한 알고리즘들은 삭제할 대상 에지를 선택하는 방법에서 차이가 난다. 에지 축약 방법을 사용하면, 대상 에지를 삭제한 후 생성된 면의 방향이 바뀌는 경우가 발생할 수 있다. 일반적으로 이런 경우는 제거 대상에서 제외시킨다. 또한 에지 축약 후 생성된 면의 종횡비가 일그러져 부분적으로 나빠질 수 있는데 이 경우도 축약 고려대상에서 제외시킨다.

특히, 이러한 에지 축약 방법을 이용하는 Hoppe의 Progressive 메쉬 방식은 DirectX 8.0 이후 버전부터는 그래픽 라이브러리 자체에 주어진 메쉬에 대해 Progressive 메쉬를 생성하는 기법을 제공하고 있기 때문에 쉽게 이 기능을 활용할 수 있다.

3.2 기하 정보를 이용한 메쉬 간략화

메쉬 모델의 간략화 방법은 주로 메쉬의 기하 정보를 이용하여 수행된다. 일반적으로 다각형 메쉬 모델 M은 정점 V와 면 F의 집합으로 구성되었다. 각 정점 $v \in V$는 그 기하 정보 $g_v \in R^3$와 특정 스칼라 값인 속성 정보 m, $s_v \in R^m$의 집합으로 구성되어있다. 기하 정보 요소와 속성 정보 요소는 각각 정점 $v_v = (g_v, s_v)^T \in R^{3+m}$의 열 벡터로 g_v와 s_v를 구성하고, 점 $v = (g, s)^T \in R^{3+m}$으로 표현할 수 있다.

메쉬 모델의 기하 정보는 모델을 구성하는 일련의 정점 v 들의 위치 정보와 연결 정보인 면 정보 f로 구성된다. 본고에서 기하 정보의 오차 척도로 이용하고자 하는 QEM(quadric error metric)은 Garland와 Heckbert가 제안한 알고리즘을 적용한다[5]. 기본 QEM은 한 점에서 모든 평면까지의 거리의 제곱의 합으로 그림 5과 같이 나타낸다.

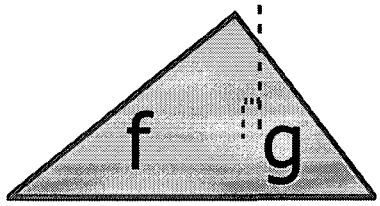


그림 5. 점-면의 제곱 거리:QEM

기하 정보만을 고려한 QEM에서는 위의 속성 정보 m에 대해 $m=0$인 경우를 해결한다. 원래 메쉬의 각 면 f는 한 점 $v=(g) \in R^3$에서 그 면을 포함한 평면까지의 제곱 거리와 같은 이차식 $Q^f(v)$로 정의된다. 원래 메쉬 상의 각 정점 v에 이웃하는 면에 대한 이차식에 면의 넓이 (S)만큼의 가중치가 합해진 이차식의 합으로 다음 수식 1과 같이 할당된다.

$$Q^g(v) = \sum_{v \in f} S(f) \cdot Q^f(v) \qquad \text{수식 1}$$

간략화 과정의 에지 축약, $(v_1, v_2) \rightarrow v$ 된 후 새로운 정점 v는 이차식 $Q^v(v) = Q^{v_1}(v) + Q^{v_2}(v)$가 최소가 되는 위치의 점으로 할당된다. 그리고 다음으로 축약될 에지는 가장 작은 비용을 가진 것을 선택하여 이를 반복적으로 수행하여 메쉬 간략화를 하게 된다.

평면 $f=(v_1, v_2, v_3)$에 대해서 이차식 $Q^f(v)$는 다음 과정을 통해 구할 수 있다. 평면 f를 포함하는 임의의 점 v에서 거리는 점과 법선 벡터의 정의에 의해 $n^T v + d = 0$로 정의된다. 이때 속성 정보를 고려하지 않는 $m=0$인 정점을 $v=(g)$라 하면, g에서 면 f를 포함하고 있는 면 P의 방정식은 $n^T g + d$로 유도할 수 있다. 이때 면의 법선

$n=(g_2-g_1)\times(g_3-g_1)/\|(g_2-g_1)\times(g_3-g_1)\|$ 이고, 스칼라 $d=-n^Tg_1$ 이다. 이 식은 $\|n\|=1$이라는 제약 조건 상에서, 다음 선형 시스템 수식 2의 해를 구하면서 파라미터를 얻는다.

$$\begin{pmatrix} P_1^T & 1 \\ P_2^T & 1 \\ P_3^T & 1 \end{pmatrix}\begin{pmatrix} n \\ d \end{pmatrix}=\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{수식 2}$$

따라서, 한 점 g와 평면 P사이의 제곱 거리는 $Q^f(v=(g))=(n^Tv+d)^2=v^T(nn^T)v+2dn^T+d^2$로 나타낼 수 있고, 이때 이차 함수는 $v^T(A)v+2b^Tv+c$라 표현할 수 있다. 여기서 A는 대칭 3×3 행렬, b는 크기 3의 열 벡터, c는 상수이다. 그러므로 이차식 Q^f는 수식 3과 같이 표현된다.

$$Q^f=(A,b,c)=((nn^T),(dn),d^2) \quad \text{수식 3}$$

이때, 각 계수는 6, 3, 1로서 총 10개의 계수가 이용된다. 이차식은 4×4의 대칭 행렬로 동차 행렬로 표현할 수 있기 때문에 수식 1의 이차식 Q^f이 10개의 계수 벡터들의 간단한 선형 조합을 이용하면 얻을 수 있다는 점에서 이 표현은 매우 효율적인 표현 방식이다. 에지 축약 후, 수식 1을 최소화하는 $v_{\min}$의 위치는 1차 미분 $\triangle Q_v(v)=2Av+2b$가 0이 되는 조건을 만족하면 되므로, 다음 선형 시스템 수식 4의 해가 바로 그 위치가 된다.

$$Av_{\min}=-b \quad \text{수식 4}$$

그리고 에지 축약 후 새로운 위치 v에 대한 이차식은 축약된 v_1과 v_2에서의 면적이 가중된 수식 5와 같은 이차식의 합으로 할당된다.

$$Q^V(v)=\sum_{v_1\in f} S(f)\cdot Q^f+\sum_{v_2\in f} S(f)\cdot Q^f \quad \text{수식 5}$$

3.3 모델의 추가된 특성 정보들을 위한 이차 오차 척도 확장

앞서 설명된 기하 정보만을 이용한 QEM 방법을 메쉬 표면의 특성 정보까지 포함한 확장 모델을 유도하는 과정을 설명한다. 기하 정보에 대한 모델의 간략화가 속성 차수 $m=0$인 경우를 고려하여 간략화를 수행한 반면, 정점의 속성 정보를 고려하면, 이 이차식은 속성 정보를 가진 ($m>0$) 이차식으로 확장 할 수 있다[10]. QEM의 점과 평면 (R^3)과의 거리 방정식을 확장시켜서 정점의 속성 정보가 포함된 차원으로 확장하면, 점과 초평면(R^{3+m})사이의 거리로 간략화를 수행할 수 있다. $v=(p,s)\in R^{3+m}$ 인 정점 v에 대하여 $Q^f(v)$는 점 v로부터 세 정점 (v_1,v_2,v_3)의 확장된 아핀 공간($P'\subset R^{3+m}$)상의 거리로 정의할 수 있다. 그림 6과 같이 축약으로 생긴 v의 새로운 점 v'에 대해서 이차 오차 $Q^f(v)=\|v-v'\|^2$은 기하 거리 오차인 $\|p-p'\|^2$과 속성 오차인 $\|s-s'\|^2$인 두 요소로 구성된다고 할 수 있다.

확장된 공간을 R^n이라 하면 점 v는 R^n공간상의 위치 정보와 속성 정보를 가진 벡터 성분으로 표현 할 수 있다. 기하 성분에 색상 속성 성분을 추가하여 확장하면, 기존의 위치정보 $[x,y,z]^T$ 성분은 $[x,y,z,r,g,b]^T$의 $n=6$인 6차원 공간의 벡터 성분을 가진 정점이 된다. 기존의 기하 정보만을 고려한 이차 오차 척도를 확장하여 다음과 같이 유도할 수 있다.

확장된 공간상의 삼각형 T를 이루는 세 점 $p,q,r(T=(p,q,r))$은 색상을 가진 표면상의 한 점 p는 $p=[p_x,\ p_y,p_z,p_r,p_g,p_b]^T$와 같이 기하 정보와

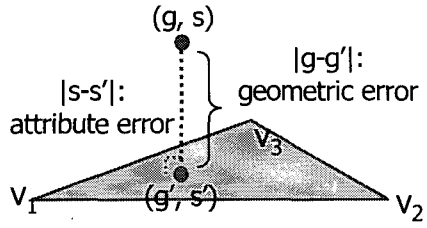


그림 6. 기하 정보와 오차 정보의 차이

속성 정보를 함께 표현할 수 있다. 세 개의 선형 독립된 점은 항상 2차원 평면을 정의하고, 이때 세 점은 R^n 공간의 이차 평면을 정의하면, 주어진 평면에서, R^n 공간상의 한 점과 평면과의 제곱 거리를 측정할 수 있는 이차식을 구할 수 있다. 2차원 평면이 삼각형 T를 포함한다면, 한 점과 두 개의 직교 벡터를 정의할 수 있다. 임의의 한 점 p에서, $h=q-p$와 $k=r-p$로 각각 h와 k를 정의하면, Gram-Schmidt의 직교 정규화 과정에 의해 그림 7과 같이 두 개의 직교 법선 벡터 e_1과 e_2를 계산 할 수 있다.

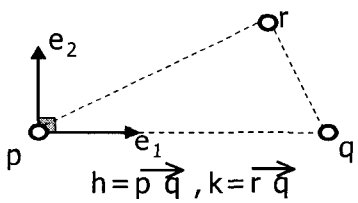


그림 7. e_1, e_2의 계산

$$e_1 = h \;/\; \|h\|$$
$$e_2 = (k-(e_{1\cdot k})e_1) \;/\; \|(k-(e_{1\cdot k})e_1\|$$

이것은 점 p를 원점으로 하고 e_1과 e_2를 축으로 하는 공간을 구성할 수 있으며, 이론적으로 $e_1, \ldots, e_n$을 계산하면 축을 확대하여 공간을 확장 구성할 수 있다. 기존의 QEM을 일반화시켜보면 이때의 e_1과 e_2를 구할 수 있다. R^n 공간상의 점 v에 대해서 T의 평면으로의 제곱 거리 Q^v를 고려하여 이를 구할 수 있다. 벡터 $u=p-v$의 제곱 길이는 수식 6으로 풀어서 표현할 수 있으며, 이는 직각 이론에 따라 e_3항 이후와 축이 되는 e_1, e_2의 관계로부터 수식 7로 재정리할 수 있다.

$$\|u\|^2 = u^T u = (u^T e_1)^2 + \ldots + (u^T e_n)^2 \qquad \text{수식 6}$$

$$(u^T e_3)^2 + \ldots + (u^T e_n)^2 = \|u\|^2 - (u^T e_1)^2 - (u^T e_2)^2 \qquad \text{수식 7}$$

수식 7의 좌항은 T의 평면에 수직인 모든 축을 따라 난 u의 제곱 길이로서, 점 v로부터 T의 2차 평면까지의 직교 제곱 거리임을 의미한다. 수식 7의 우항을 정리하면 수식 8로 정리할 수 있다.

$$\begin{aligned}&\|u\|^2-(u^T e_1)^2-(u^T e_2)^2\\ &= u^T u-(u^T e_1)(e_1^T u)-(u^T e_2)(e_2^T u)\\ &= v^T v-2p^T v+pp\\ &-v^T(e_1 e_1^T)v+2(pe_1)e_1^T v-(pe_1)^2\\ &-v^T(e_2 e_2^T)v+2(pe_2)e_2^T v-(pe_2)^2\end{aligned} \qquad \text{수식 8}$$

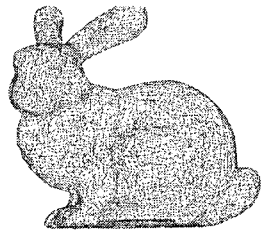
이 식은 $v^T Av+2b^T v+c$의 이차 함수 형태로 정리하면 다음과 같다.

$$\begin{aligned}A &= I-e_1 e_1^T-e_2 e_2^T\\ b &= (pe_1)e_1^T+(pe_2)e_2^T-p\\ c &= pp-(pe_1)^2-(pe_2)^2\end{aligned} \qquad \text{수식 9}$$

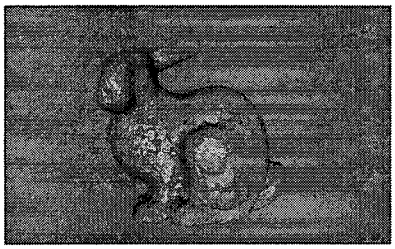
따라서 표면 정보의 속성이 이차식에 추가가 되면, A는 $n\times n$ 차의 행렬이 되고, b는 $n-$벡터로 이차식의 확장이 가능하다. 색상 정보뿐만 아니라 질감이나 법선의 속성을 더 추가할 경우에는 추가적인 차원 증가가 필요하다. 메쉬 표면의 정점의 위치 정보이외에 속성 정보를 더 추가할 경우 속성의 요소가 증가되고 계산에 수반되는 수행시간 역시 증가한다.

그림 8는 bunny 모델의 실험 결과이다. 그림 8(b)는 원 모델과 기하학적 위치 정보만을 이용한 QEM의 간략화 결과이고, 그림 8(d)는 기하 정보와 rgb 색상에 대한 속성 정보를 이용했을 때, 여기에 법선 정보를 추가하여 간략화 했을 때의 결과를 상호 비교하였다. 원래 bunny 모델은 69451개의 면(f)으로 구성되어있다. 이를 기하 정보만을 이용하여 1500개의 면으로 간략화를 수행한 경우 그림 8(b)와 같은 메쉬 모델로 표현된다. 메쉬만으로 구성된 모델은 현실감이 부족하므로 속성 정보를 추가한 방법에 의해 같은 면의 개수로 구성된 모델을 간략화 하였을 경우 그림 8(c)와 같은 결과

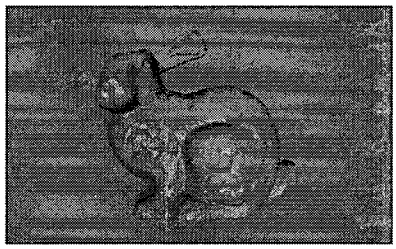
(a) bunny 원 모델 f :69451



(b) *geometry* f : 15000



(c) 속성-*color* 이용 f : 15000



(d) *color+normal* f: 15000

그림 8. bunny 모델의 실험 결과 : *color* +*normal* 속성 추가

를 얻을 수 있다. 그림 8(d)는 기하 정보에 색상 정보뿐만 아니라 법선 정보까지 함께 고려하여 간략화한 결과로 좀 더 원 모델에 시각적으로 충실한 간략화 결과를 볼 수 있다.

4. 결 론

3D 게임 분야는 하드웨어보다는 소프트웨어적인 요소가 강한 분야이다. 최근 국내에도 많은 국내 3D 게임엔진이 제작되고 있지만, 게임 소프트웨어의 기술 수준은 아직 선진국에 비해 미약하므로 대부분의 국내 게임들은 외국의 3D 게임엔진을 들여와 사용하고 있다. 그러므로 게임엔진의 각 핵심 부분을 개선할 수 있는 알고리즘들의 개발이 필요하다.

게임엔진은 게임의 핵심 기능적 요소를 구성하는 라이브러리와 이를 뒷받침해주는 지원 부분의 집합체로 정의할 수 있으며, 엔진의 핵심 부분은 렌더링 엔진, 애니메이션 엔진, 물리 엔진, 네트워크 엔진, 인공지능 엔진 등으로 구분될 수 있다. 본고에서는 3D 게임을 제작하는데 필요한 3D 게임엔진에서 렌더링 엔진 기능에 필요한 렌더링 속도 개선 방법에 대해 알아보았다. 게임 화면에 나타나는 각종 요소에 대한 처리를 담당하는 렌더링 엔진은 실시간 처리를 위한 빠른 렌더링 작업을 위한 속도개선 기능을 반드시 포함하고 있어야 한다. 렌더링 속도를 증가시키기 위한 방법으로는 클리핑(clipping), 컬링(culling), LOD, 버퍼 지원 기능 등이 포함된다.

LOD는 카메라와의 거리에 따라 객체의 정밀도를 조정하는 방법으로서 정밀도의 조정에 필요한 메쉬 간략화를 위해 Hoppe의 방법과 Garland의 방법이 주로 이용되고 있다. Garland의 간략화 방법은 QEM을 이용한 알고리즘을 적용하였으며, 본 논문에서는 기존 QEM에 색상, 법선, 그리고 질감 정보와 같은 속성 정보를 포함한 EQEM을 적용한 결과를 통해 기존의 방법에 비해 원 모델에 가까운 간략화 모델을 생성함을 알 수 있었으며, 이러한 알고리즘은 3D 게임에도 적용 가능할 것이다.

참 고 문 헌

[1] 한국산업개발원, "게임엔진 품질 평가 기술", 도서출판 정일, 2003.

[2] 김현빈 외 11인, "온라인 3D 게임엔진 개발", 한국전자통신연구원, pp. 27-60, 2003.

[3] 정보통신부 한국정보통신교육원 한국컴퓨터게임학회, "GAME 엔진 개발론", 홍릉과학출판사, pp. 39-53, 2003.

[4] H. Hoppe. "Progressive Meshes". In SIGGRAPH 96, pp. 99-108, 1996.

[5] M. Garland and Paul S. Heckbert. "Mesh simplification with quadric error metrics". In SIGGRAPH 97, pp. 209-216, 1997.

[6] J. Cohen, M. Olano and D. Manocha. "Appearance-preserving simplification", In SIGGRAPH'98, pp. 115-122, 1998.

[7] D. Luebke and C Erikson. "View-dependent simplification of arbitrary polygonal environment". In SIGGRAPH'97, pp. 199-208, 1997.

[8] P. Lindstrom and G. Turk. "Fast and memory efficient polygonal simplification", IEEE Visualization98, pp. 279-286, 1998.

[9] W.Schroeder, J.Zarge, and W.Lorensen, "Decimation of Triangle Meshes", in the Proceeding of SIGGRAPH'92, pp. 65-70, 1992.

[10] T. H. Han and J. C. Chun, "Simplification using Property Clustering on Polygonal Surface", in the Proceeding of CIE2004, pp.527-532, 2004.

민 경 필

- 1996년 경기대학교 수학과(학사)
- 1998년 경기대학교 전자계산학과(석사)
- 2000년~현재 경기대학교 전자계산학과 박사과정
- 관심분야 : face detection and recognition, 3D face modeling, Animation

한 태 화

- 1996년 경기대학교 수학과(학사)
- 1998년 경기대학교 전자계산학과(석사)
- 2005년 경기대학교 전자계산학과 (박사)
- 관심분야 : face detection and recognition, 3D face modeling, Animation

전 준 철

- 1984년 중앙대학교 전자계산학과(학사)
- 1986년 중앙대학교 전자계산학과(석사)
- 1987년~1988년 삼성반도체 통신연구소 연구원
- 1992년 The Univ. of Connecticut, 컴퓨터 공학과(석사)
- 1995년 The Univ. of Connecticut, 컴퓨터 공학과(박사)
- 2001년~2002년 미시건 주립대학 패턴인식 및 영상처리 연구실 객원교수
- 1995년~현재 경기대학교 정보과학부 부교수
- 관심분야 : 컴퓨터 그래픽스(3D face modeling & feature detection), 영상처리(2D, 3D face detection), 의료영상처리(MRI, CT 영상처리)