온라인 게임을 위한 실시간 음영 생성 기법

Real-time shadow creation technique for a online game

이승욱, 배재환*
동명대학교, 동명대학교*

Sung-Ug Lee, Jae-Hawan Bae*
Tongmyong Univ, Tongmyong Univ.*

요약

최근 컴퓨터 그래픽 기술의 급속한 발전과 더불어 온라인 게임의 환경도 다양하게 변하고 있다. 사실적인 표현을 위한 원근감, 입체감, 물체의 재질, 그림자, 빛의 세기 등의 다양한 요소가 고려되어야 한다. 그 중 음영의 표현은 객체의 사실적 표현에 중요한 역할을 한다. 음영을 생성하기 위해서는 반복적인 많은 연산을 수행해야 한다. 즉 객체 판단을 위한 수많은 점들에 대한 영역의 분리와 빛의 세기 등을 구별해야 한다. 본 논문은 실시간 그림자를 생성하기 위해 그림자의 영역을 판단하기 위한 데이터를 추출하지 않고, 객체에 대한 LOD를 생성하여 이를 그림자로 매핑 해줌으로써 사실적인 음영을 생성시킬 수 있는 효율적인 방법을 제시한다.

Abstract

Online games' environments have recently changed with developments of computer technique. Various factors such as respective for realistic expression, solidity, the quality of the material, shadow, should be considered. In these factors, shadow expression plays a important role for realistic one. A lot of repeated operation should be executed to provide shadow. In other words, realm division against a lot of dots to estimate objections and the degree of brightness should be divided. This paper provides effective methods to perform realistic shadow through mapping LOD of objections as shadow without drawing data to judge the realm of shadow for shadow creation.

I. 서 론

최근 하드웨어의 급속한 발전으로 인하여 실시간 프로그램에서 음영을 생성하는 효율적인 알고리즘을 지원하는 다양한 기술들이 활용되고 있다. 게임에서 평평한 기준 평면에 음영을 투영하여 처리할 경우 간단한 처리 방법으로 해결할 수 있었다. 그러나 실시간적으로 임의 형태의 객체 음영을 다른 객체나 임의의 지형에 투영하여 처리하는 경우에는 단순한 처리로 가능하지 않다. 게임의 경우 하나의 객체는 수많은 다각형들로 이루어져 있으므로, 객체의 모든 다각형들의 모든 정점들에 대해 충돌 검출을 수행해서 음영을 결정하는 것은 대단히 많은 계산을 필요로 한다. 따라서 실시간적인 음영의 투영을 위해서는 음영 다각형을 세부 분할할 필요를 가진다. 이것은 음영 영역에 속하는 표면의 복잡성에 따라 정확하게 투영하기 위해 많은 비용이 소요된다는 것을 의미한다. 음영 영역에 속하는 표면 정점들의 색상을 어둡게 해서 음영을 표현할 수 도 있지만 표면의 정밀도가 높지 않다면 음영의 정확성을 보장할 수 없다. 또 다른 처리 방법으로 동적인 그림자 맵을 이용하여 처리하는 방법이 있다. 이 처리 방법은 정점의 색상 변경보다 좀 더 정확하고 균일한 결과를 얻을 수 있지만 많은 계산 량을 요구한다[1][2].

본 논문에서는 복잡한 개체에 대한 실시간 음영을 생성하기 위한 처리 방법으로 LOD를 이용하여 조명의 위치에 따라 객체 LOD를 적용한다. 다양한 LOD 객체모델을 실시간적으로 생성하여 적용한다. 이러한 처리 방법은 실시간 그림자를 생성하기 위해 그림자의 영역을 판단하기 위한 데이터를 추출하지 않고, 객체에 대한 LOD를 생성하여 이를 그림자로 매핑 해줌으로써 사실적인 음영을 생성시킬 수 있는 효율적인 처리 방법을 제시한다.

II 기존 연구 및 연구 방법

객체에 대한 정확한 음영을 생성하기 위한 방법으로 사용되고 있는 레이캐스팅이나 래디오 시티기법 등은 계산 비용이 과도한 알고리즘이다. 이러한 알고리즘은 음영을 만들어 내는 객체와 그것을 많든 객체의 형태나 복잡성에 상관없이 음영을 만들어 낸다. 또한 객체와 음영이 적용되는 객체의 형태나 복잡성에 상관없이 항상 제대로 작동하나, 계산 비용이 크다는 단점을 가지고 있다[1][5].

2.1 레이캐스팅 알고리즘

이 알고리즘은 물체를 눈에 보일 수 있도록 객체나 색깔을 모니터에 표현해주는 처리 방법이다. 시야에서 광선이 바라보고 있는 영역의 다른 지점을 향하여 뻗어 나가게 되고 광선은 물체에 부딪히게 되면 부딪힌 지점의 색깔을 계산한다. 즉 픽셀의 색상이 광선에 의하여 계산되어진다. 반복적으로 이러한 계산에 의하여 구해진 객체에 대한 이미지 전체를 계산하게 된다[2][5].

2.2 벡터 카메라를 이용한 음영 위치

벡터 카메라는 세 개의 벡터를 이용하여 자신의 방향을 표시한다. 벡터 U, V, N은 각각 X, Y, Z 원점 벡터들과 평행하다. U 벡터는 오른 쪽, V 벡터는 위쪽, N 벡터는 카메라가 바라보는 방향을 가리킨다. 벡터카메라의 경우 음영의 투영은 카메라의 전역 공간좌표로부터 투영할 정점의 전역 공간 위치를 잇는 벡터를 만드는 과정에서 시작된다. 정점이 카메라의 시야 안에 있다면 벡터카메라의 3D 화면과 그 벡터가 교차하는 위치의 3D 점을 계산한다. 카메라 벡터는 전역 공간좌표 값으로 저장된다. 객체에 대한 좌표 값은 국소-전역변환 행렬을 통하여 생성된다[1].

2.3 명암 생성 알고리즘

명암을 생성하기 위한 알고리즘은 다양하게 제안되었다. 명암을 생성하는 가장 단순한 방법으로 객체의 표면 부분을 모두 같은 색깔로 처리하는 방법이 Faceted 알고리즘이다. 이 처리 방식은 객체의 분할된 면이 한 가지 색깔만을 가진다. 객체의 색깔을 계산할 때 쉐이딩 알고리즘은 객체에 대한 표면의 각도에 따른 조도를 계산한다[3]. 다른 처리 방법으로 Henri Gouraud에 의해 제안된 Ground 쉐이딩 알고리즘은 폴리곤의 표면 노멀을 인위적으로 조절하여 폴리곤간의 경계를 부드럽게 보이도록 만든다. 3차원객체의 버텍스는 두 폴리곤이 만나는 지점에 위치시키고 표면 노멀의 평균치를 적용시킨다. 이러한 처리 방식은 빠른 속도를 얻을 수 있으나 사실적인 표현이 어렵다[4]. 이러한 단점을 개선하기 위해 제안된 알고리즘이 Phong 쉐이딩 알고리즘이다. Bui Tuong Phong에 의해 제안된 알고리즘으로 Ground 쉐이딩 알고리즘이 버텍스의 표면 노멀의 평균대신 표면 노멀의 색깔을 직접 계산하여 정확한 색깔과 빛을 표현하여 폴리곤간의 구분경계를 표현하도록 한다[3][4][5].

2.4 3차원 모델의 표현기법

일반적인 사람의 경우 시각과 같은 감각기관을 통하여 물체를 인식한다. 컴퓨터 그래픽의 경우 3차원 객체에 대한 객관적인 모델의 효과적인 인식 방법은 형태기반의 검색 방법이다. 이러한 검색방법을 적용하기 위해서는 3차원 모델 정보를 인덱싱하는 알고리즘이 필요하다. 현재 많이 사용되고 있는 세 가지 인덱싱 방법으로는 첫째, 기하학적 표현 방법과 둘째, 외관의 특징을 표현하는 방법, 마지막으로 임의의 주석에 의한 표현 방법 등이 있다. 기하학적 표현방법에는 경계, 복셀 표현, CSG(Constructive Solid Geometric tree), point clouds, range image, 음함수 등이 있다. 외곽 특징표현방법에는 색, 질감, BRDFs(Bidirectional Reflectance Distribution Functions) 등이 있다[6][7].

Ⅲ. 본 론

음영의 표현은 3차원 그래픽에서 수학적인 계산에 의해 화면에 표시된 물체에 적절한 색깔 및 밝기를 부여하여 물체의 질감 및 입체감을 나타내는 것으로 많은 시간 계산을 요하는 작업이다. 본 논문은 실시간적으로 임의형태의 객체 음영을 다른 객체나 임의의 지형에 투영하여 처리할 수 있는 효율적인 처리 방법을 제시한다. 이를 위해 필요한 세부 처리 기법에 관하여 본 장에서 논의한다.

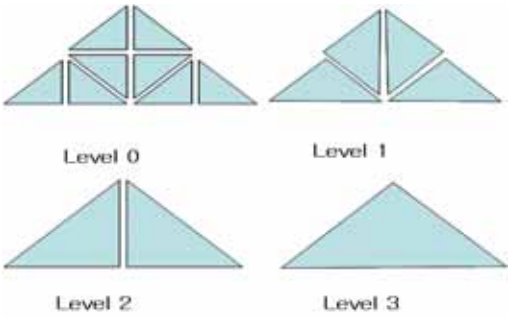
3.1 영역 구분 방법

본 논문에서 제시하는 음영 생성 방법은 면의 방향, 색, 반사함수, 광원, 시점 등에 의하여 결정되며, 면에 대한 광선의 반사의 정도를 계산 할 수 있다. 음영처리에 의해 표현하는 면의 방향이나 재질감등의 생성이 가능하지만 곡면을 다각형의 집합으로 나타낼 경우 각각의 다각형의 형태가 눈에 띄는 현상이 생긴다. 물론 이러한 현상을 감소시키는 스무스 쉐이딩(smooth shading)기법을 이용하면 부드러운 면 처리가 가능하다. 그러나 이러한 처리 방법은 원래 객체가 가지고 있는 영역보다 넓게 인식 되게 된다.

3.2 블록 기반의 간소화를 통한 LOD 적용

3차원 객체는 수 만개의 삼각형들로 이루어진 복잡한 세부 모델로 이루어져 있다. 다각형으로 미세하게 분할될수록 렌더링 처리에 걸리는 시간이 커지며 상대적으로 높은 품질의 렌더링 결과를 얻을 수 있다. 그러나 음영 처리를 위한 LOD는 기존의 LOD와는 개념적으로 차이를 가진다. 즉 음영은 빛의 각도에 따라 객체와 광선이 평면과 충돌에 의하여 발생되는 거리에 비례하여 표현된다. 따라서 게임에서는 조명과 같은 빛은 일정한 위치에서 객체에 조영하게 된다. 그러므로, 이에 대한 객체에 대한 다양한 LOD 사이즈 별로 이력 값을 사실적으로 음영을 생성할 수 있다. 이러한 처리 형태에 기반한 LOD의

생성은 효율적일 것이다. 또한 블록기반의 간소화 작업은 실시간으로 모든 꼭짓점에 대한 간소화 작업을 처리해야 하므로 많은 비용이 소요된다. 이건은 실시간 프레임 처리 속도의 저하를 야기한다. 이에 대한 개선 방법으로 꼭짓점을 포함하는 블록 경계 사각형의 화면 공간 높이를 분석하여 꼭짓점의 그룹간의 간소화 처리를 수행할 수 있다. 이러한 처리는 블록 기반의 간소화를 통하여 객체의 음영을 효율적으로 생성할 수 있다.



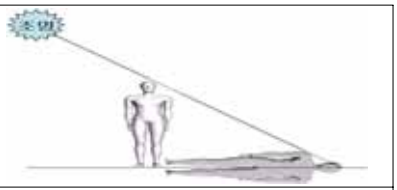
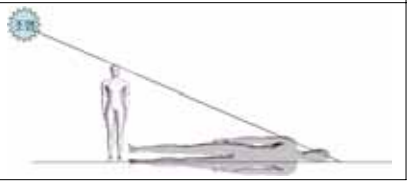
▶▶ 그림 1. 단계별 LOD 적용을 위한 삼각형 간소화

3.3 LOD 모델 적용을 위한 처리방법

객체에 대한 카메라 광선 검출을 통한 객체와 평면과의 충돌검출로 객체와 평면과의 거리를 얻을 수 있다. 이 값을 이용하여 $\cos\Theta$값을 구한다. 또한 최고점과 최저점을 변환/투영하고 그 점들의 화면 좌표들을 적절히 빼면 화면상에 객체의 음영 너비와 높이를 얻을 수 있다. 그런데 이 방법은 객체의 방향에 의존하며, 두 정점들을 처리해야 한다. 확대율은 객체의 위치를 시야 공간으로 변환 후 다음 공식으로 구할 수 있다.

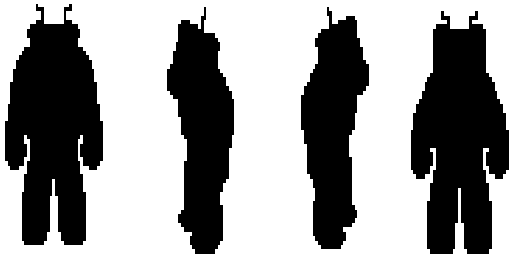
$M = xscale/zview$ 여기서 $xscale$은 투영 방정식에 쓰이는 축척 인수이다.

$$xscreen = (xview \times xscale)/zview + xcenter$$

적용 1. 객체와 동일한 크기의 음영 모델	
적용 2. 객체 보다 적은 크기의 음영 모델	
적용 3. 객체보다 크기가 큰 음영 모델	

▶▶ 그림 2. 다양한 크기가 적용된 음영 모델

시야 공간좌표들은 전역 좌표를 회전/변환한 것이므로 $zview$는 전역 단위로 측정된다. $xscale$은 카메라의 시야에 관련된 것으로 단위 픽셀이다. 따라서 M의 단위는 픽셀당 전역단위이다. M이 크다는 것은 하나의 전역 단위당 픽셀이 많다는 의미이며, M 값의 범위는 객체의 음영 보간을 위한 이력 값으로 사용한다. 이것은 밉맵 텍스쳐에 쓰이는 음영 LOD의 크기이다. 이력문턱 값으로 적용하기 위해서는 모델의 레벨을 결정하고 레벨의 단계에 맞게 이력 문턱 값의 범위에 대한 이력문턱 값을 적용하는 것이다. 단순히 문턱 값을 나타내기 위한 기호 T로 표기한다.



▶▶ 그림 3. 그림자의 앞, 좌, 우, 뒤 모습 샘플

즉 이력문턱 값의 범위는 "(최소 $xscale$ < T<최대 $xscale$)/레벨단계"로 나타낸다. 다각형이나 모델의 개수의 제한은 동일한 객체에 대하여 여러 개의 객체 음영 모델을 가지는 경우 하나 또는 몇 개의 세부 수준 모델만을 사용하여 객체의 음영을 표현한다면 충분한 메모리를 아낄 수 있다. 객체들을 확대율로 정렬하고 크기순으로 N개의 최고 수준, 그 다음 순으로 객체들의 다각형 개수를 기준으로 객체 LOD를 선택할 수 있다. 그림 2는 $\cos\Theta$의 값의 크기에 따라 적용되는 음영 모델의 확대와 축소된 모델의 적용형태를 보여주고 있다. 크기가 다른 LOD 적용 모델의 과정을 볼 수 있다.

Ⅳ. 실험 및 분석

실시간 음영은 객체 모델에 대한 단일한 색상으로 표현된 모습이다. 음영 적용은 카메라 거리와 시야를 통한 포괄적인 값이며, 단순한 카메라 거리만을 의미하는 것은 아니다. 또한 음영의 크기는 조명과 객체의 광선 충돌로 이루어지는 $\cos\Theta$의 값에 의하여 그림자의 크기가 결정된다. 효율적인 처리를 위하여 $\cos\Theta$의 값을 몇 단계로 구분하고 이를 이력값으로 사용함으로써 빠르게 음영 모델을 생성할 수 있다. 그림 4는 동명대학교 학교기업 '라츠인터랙티브'에서 개발한 캐주얼 레이싱 온라인 게임인 '호버런' 실행 화면이다. 레이싱

캐릭터에 그림자를 매치하여 음영이 생성된 처리를 보여주고 있다.

▶▶ 그림 4. 동명대학교 학교기업 '라츠인터랙티브'의 '호버런' 실행시 그림자 생성 데모

그리고 표 1은 객체 모델에 대한 이력 값의 적용 과정을 보여준다. 표 2의 소스는 이력 값에 따른 저수준의 LOD 모델을 생성하는 과정이다. 저수준 LOD를 실시간으로 효율적으로 생성하기 위해서는 다양한 모델을 적용하기는 많은 비용이 필요하다. 게임의 효율적인 처리를 위해서는 많은 수의 LOD 모델을 적용하는 것 보다 사실적인 처리에 기반을 둔 소수의 모델을 표현하는 것이 바람직할 수 있다.

[표 2] LOD 적용을 위한 이력값 처리 단계

```
while (큐에 LOD 적용객체존재)
{
  if (객체모델) then queue.D_U_processed
    if (이력값 cosΘ<객체모델_index)
       for 블록_인덱스 초기화
    else
       for 각 모서리가 아닌 꼭지점처리
       while (이력값 cosΘ>=객체모델_index){
       if (꼭지점이라면) then
          vertex.setEnable =true
          SimplifyBlocks();
       endif
 }
```

[표 3] 실시간 LOD 음영 모델생성 단계

```
void SimplifyBlocks(TopVertex, LeftVerx, RightVertex)
{
if (객체내부노드) then
MidPoint=(LeftVerx + RightVertex)/2
if (MidPoint.IsEnable=true)
'재귀호출로 LOD생성
   SimplifyBlocks MidPoint, TopVertex, LeftVertex
   SimplifyBlocks MidPoint, RightVertex, LeftVertex
endif
}
```

Ⅳ. 결 론

음영 처리를 위해 중요하게 다루어져야할 부분이 카메라이다. 본 논문은 벡터 카메라를 적용하였다. 이것은 행렬 기반의 카메라 적용 기법 보다 3차원 객체의 영역 구분 및 음영 처리에 적합하였다. 특히 벡터 카메라를 적용한 객체 투영 기법은 음영 입체를 생성하기 위한 초점 기반의 객체에 효과적인 것으로 평가된다. 이것은 객체 음영을 표현하기 위한 카메라 적용 모델로 적합하다는 것이다. 또한 실시간 LOD 적용을 통한 몇 개의 저수준 모델들만 사용한다면 객체 표현에 필요한 메모리를 아낄 수 있다. 그리고 확대율로 정렬하고 음영의 크기 순으로 실시간 객체의 음영을 생성하여 처리한다면 사실적인 표현을 위해 만족할 만한 결과물을 효과적으로 생성할 수 있다. 게임 프로세스의 성능 증가와 실시간 게임 모델의 성능개선을 위한 현실적 대안이 될 수 있을 것이다. 본 논문은 게임 그래픽을 위한 실시간 그림자를 생성기법에 관한 처리 방법을 제시하였다. 이 처리 기법은 실시간 음영 생성을 위한 효율적인 처리방법으로 평가된다. 객체의 영역을 통한 데이터 추출에 의한 처리기법과는 다른 각 객체에 대한 LOD를 생성을 통한 그림자를 매핑함으로써 사실적인 음영을 생성시킬 수 있는 처리 방법이다.

▌참 고 문 헌▐

[1] 김병권, 임인성, '이미지와핑을 이용한 실시간 그림자 생성기법' 한국정보과학회논문지, 2002.

[2] A. Woo, P. Poulin, and A. Fournier. A survey of shadow algorithms. IEEE Computer Graphics and Applications. pp:13-32. 1990

[3] Cha Zhang and Tsuhan Chen, "Efficient Feature extraction for 2D/3D object in mesh represention", Proceedings of International Conference on Image Processing, vol.3 pp.935-938,2001

[4] Hugues Hoppe, "Progressive Meshes," Computer Graphics (SIGGRAPH '96 Proceedings) 1996, 99-106.

[5] Hugues Hoppe, "View-Dependent Refinement of Progressive Mesh," Computer Graphics (SIGGRAPH '97 Proceedings) 1997, 189-198.

[6] M. Stamminger and G. Drettakis. Perspective shadow maps. Computer Graphics (Proc. of SIGGRAPH 2002). pp:557-562. 2002

[7] L. Williams. Casting curved shadows on curved surfaces. Computer Graphics (Proc. of SIGGRAPH 78). pp:270-274. 1978