# CLOD를 위한 옥트리 분할 기법

°
이승욱 박경환
동아대학교 컴퓨터공학과

# A Partitioning Method of Octree for CLOD

°
Sookng-ug Lee Kyung-hwan Park

## 요 약

본 논문은 기존의 3D게임엔진에 실시간으로 상호작용이 가능하고 3D MMORPG(Massive Multi-play Online Role Playing Game) 게임에 적합한 가상공간을 표현하기위한 필요한 기술을 분석하고 이를 활용하려 한다. 기존의 머드 게임에 3차원 기술적용하고, 3차원 물체를 모델링 하는데 있어서 메쉬나 버텍스, 혹은 폴리곤으로 사실적인 지형처리와 렌드링 속도향상을 위하여 3차원 개체의 폴리곤을 동적으로 생성시키기 위한 방법으로 Height field처리 기법과 거리에 따라 다르게 모델링 된 데이터를 선택적으로 사용하는 LOD(Level of Detail)처리기법과 가시성 판단이나 충돌 검출을 위한 입체 컬링 방법으로 옥트리를 이용하여 가상공간을 분해 하기 위한 자료 구조로 사용한다. 본 논문은 기존의 3차원 공간을 표현하기 위하여 사용되고 있는 옥트리 구조를 이용하여 공간을 분할하고, 이를 세부수준으로 나누어 처리하기위한 LOD 개념을 이용하여 외부지형을 폴리곤으로 표현하는 방법에 대한 처리 기법을 제시하려고 한다

## 1. 서론

3D 게임엔진을 이용하여 실시간으로 그때그때 상황에 따라 배경 및 캐릭터의 모습이 달라 보이게 처리한다는 것은 PC의 연산 처리 속도와 메모리의 한계로 많은 어려움이 따른다. 그래픽가속기의 등장과 최적화된 알고리즘과 어셈블리로 구성된 라이브러리를 사용하므로 실시간 렌드링이 어느 정도 가능하게 되었다. 3D 게임엔진과 같은 실시간 사용자와 상호 연동되어 작동하면서 초당 30프레임 이상을 보여줄 수 있도록 처리해야 한다. 실시간 렌드링을 하기 위해서는 그래픽 라이브러리를 사용하여 프로그래밍을 통하여 구현한다. 본 논문은 그래픽 라이브러리로 Microsoft의 DirectX와 C++을 이용하여 3D 게임엔지에서 필요한 3차원 외부지형 및 개체를 폴리곤으로 표현하는 방법에 대한 처리 기법을 제시하려고 한다.

## 2. 3차원 공간 지형 및 개체의 표현 방식

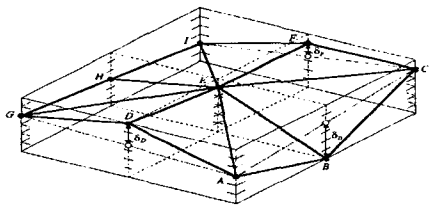
3차원 공간 상에 사용되는 거대한 지형 데이터와 개체를 실시간으로 처리하기위해서는 많은 어려움이 따른다. 고려해야 할 사항을 세가지로 요약한다면 다음과 같은 것들이 있다. 우선은 3차원 게임에 사용되는 모델의 폴리곤 수를 최대한 줄여야 한다. 실사와 같은 영상물을 얻기 위해서는 폴리곤의 수가 많으면 많을수록 좋겠지만 게임과 같은 환경에서는 무엇보다도 속도가 문제가 되기 때문에 될 수 있으

면 폴리곤의 수를 줄인다. 두 번째로 미리 계산할 수 있는 것들은 미리 계산하여 처리한다. 실제로 광원의 처리 같은 경우 이를 계산하는 것은 많은 시간을 필요로 하기 때문에 실시간 3차원 그래픽에 사용하기에는 적당하지 않다. 세 번째로 공간을 쪼개서 렌더링 파이프라인에 들어가는 물체의 수를 줄이고 카메라의 뷰볼륨 안에 있는 물체만 렌더링 하도록 한다. 이러한 부분들은 개임에 있어 그래픽처리성능을 향상시키는 아주 중요한 요소이다. [3][7][10]

## 3. 3차원 가상 공간의 지형데이터를 처리하는 방법

### 3.1 Height field

3차원의 가상 공간의 지형데이타를 처리하는 방법으로 거대한 지형데이타를 실시간으로 처리할 경우 지형을 고정적인 정점좌표들을 주어서 표현할 경우 지형이 광범위해서 당연히 속도 저하가 생긴다. 광범위한 지형의 각 정점의 좌표를 거리에 따라 효율적으로 적절한 폴리곤 수를 조절할 수 있는 데이터 저장방법으로 사용한다. 지형데이터를 표현할 때 지형의 높이 정보만을 저장하고 폴리곤은 내부적인 알고리즘으로 생성해 내는 방법이다. Height field란 일정한 간격이 (x,y)좌표에 대한 z축성분인 높이값을 말한다.



[그림 1] height field

그림 1은 height field에 의한 지형자료를 표현한 그림이다. height field에 의한 지형자료는 x, y축의 성분이 일정한 간격으로 존재하기 때문에 z축의 데이터만을 저장하면 방대한 지형에 대한 저장공간을 크게 줄일 수 있다.[2][10]

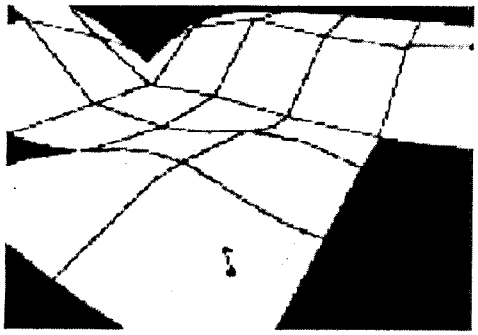
지형데이터를 저장하기위해서는 지형데이터를 일정한 크기로 잘라서 보이는 부분만을 계산해서 폴리곤을 생성하는 페이지형식으로 저장해서 각각 n × n 크기의 Height field를 가지는 페이지 형식으로 지형을 표현해서 카메라의 위치에 따라 보일 수 있는 페이지만 렌더링 함으로 속도향상을 얻을 수 있다.

### 3.2 LOD

컴퓨터에 표현되는 개체들은 기하학적 모델로 표현되며, 거리에 따라 서로 다르게 모델링 된 데이터를 선택적으로 사용하는 LOD(Level Of Detail) 기법으로 표현할 수 있다. 이것은 카메라와 렌더링 될 대상 물체와의 거리에 따라 폴리곤의 개수가 다른 모델링 데이터를 선택적으로 렌더링 하는 방법이다. 렌더링 될 때 사용되는 폴리곤의 수를 줄여서 렌더링 속도를 향상시키는 기법이다. LOD 렌더링을 구하려면 하나의 개체에 대해 다양한 모델을 만들어야 한다. 개체의 수준이 바뀔 때 세부수준을 선택하는 방법으로 카메라와 개체사이의 거리에 따라 문턱값을(threshold)을 적용함으로 빠르게 반복되는 수주 전환의 문제도 해결할 수 있다.

### 3.3 CLOD(Continuous LOD)

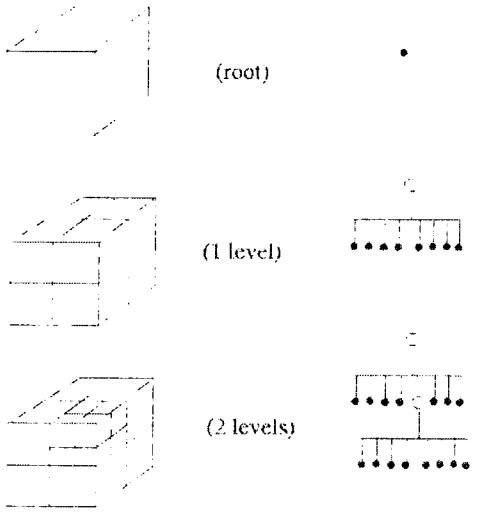
넓은 지형의 Height field 데이터와 같이 고정된 정점 데이터를 저장하는 방식이 아니고 실행시간에 폴리곤을 동적으로 생성해 내기 때문에 LOD(Level Of Detail)를 사용할 경우 동적으로 바뀌는 카메라의 위치에 따른 적절한 LOD 폴리곤들을 얻을 수 있다. 이것은 기존의 LOD와는 다르게 CLOD(Continuous LOD)라고 부른다. LOD와 CLOD의 차이점은 LOD의 경우 렌더링될 폴리곤들의 집합을 미리 거리에 따라 만들어 놓고 적절한 LOD 데이터를 선택적으로 렌더링하는 반면 CLOD는 거리의 변화에 따라 지형을 표현하는 vertex들을 병합 또는 삭제함으로써 그 표현의 정도를 동적으로 달리 할 수가 있다. 물론 실행시간에 폴리곤을 생성하는 오버헤드는 있지만 정점으로 표현될 경우 방대한 양의 LOD 폴리곤 데이터를 여러 개 갖게 되는 메모리의 낭비를 막을 수 있고 카메라의 위치와 방향에 따라 적절한 폴리곤을 생성해 낼 수 있는 장점이 있다.



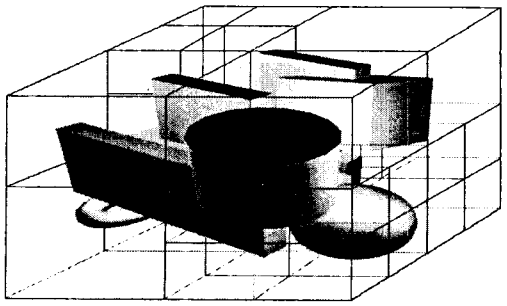
[그림 2 ]
카메라의 위치와 방향에 따른 폴리곤 생성의 변화

3.4 옥트리

임의의 형태의 입체를 공간적으로 분할하는 일종의 화면 분할법 중의 하나로 화면을 8조각으로 계속 나누어 관리하는 방법을 말합니다. 옥트리는 정적인 지형에 적합하나 하나의 장면에서 동적으로 움직이는 개체들에 대한 부착 목록을 저장하는 데에도 사용된다. 각노드당 최대 8개의 자식들을 가질수 있는 트리구조로 노드의 자식들은 부모입방체를 8등분하는 동일한 크기의 입방체들이 된다. 공간 분할은 자식노드를 통한 공간 분할은 입방체가 어떤 특정한 크기가 될 때까지, 또는 각 노드에 담긴 다각형들이 일정한 개수가 될 때까지 반복된다




[그림 3 ] 옥트리의 데이터 구조



[그림 4] 주위의 입방체로부터 세부수준으로 분할된 모습

옥트리의 각노드는 자신의 공간안에 있는 모든 다각형에 대한 포인터들을 담는다. 가시성 판단의 경우 트리의 루트 노드의 경계 입방체가 완전히 보인다고 판정되는 경우라면 그 입방체가 감싼 개체 전체를 랜더링하면 된다. 일부만 보이는 것으로 판정되었다면, 트리를 탐색해 가면서 자식노드를 탐색한다. 어떤 노드의 경계 입방체가 시야 각뿔대로부터 완전히 벗어났다면 그 시점에서 탐색은 끝나며, 그 노드의 모든 자식을 렌드링으로부터 제외시킨다.[1][2][4][6][8]

## 4. 구현 방법

본 논문은 3차원의 가상 공간의 지형데이타를 처리하는 방법으로 거대한 지형데이타를 실시간으로 처리할 경우 지형을 고정적인 정점 좌표들을 주어서 표현할 경우 지형이 광범위해서 당연히 속도 저하가 생긴다. 광범위한 지형의 각 정점의 좌표를 거리에 따라 효율적으로 적절한 폴리곤 수를 조절할 수 있는 데이터 저장방법이 필요하다. 지형데이터를 표현할 때 지형의 높이 정보만을 저장하고 폴리곤은 내부적인 알고리즘으로 생성해 내는 방법이다. Height field란 일정한 간격이 (x,y)좌표에 대한 z축성분인 높이 값을 말한다. Height field의 경우에는 저장 공간을 줄이기 위한 방법으로 z축의 데이터만을 저장한다. 하지만 매번 z-buffer를 읽어야 하기 때문에 속도 저하기 야기되고, 매번 역행렬을 계산해야 하므로 속도가 저하된다. 이를 개선하기위한 방안으로 Height field에 의해 저정된 점 좌표를 8개의 화면으로 균등 분할한다.

데이터를 처리하기위한 방법으로 옥트리를 이용하여 화면을 단순히 분할하는게 아니라 폴리곤의 개수를 기준으로 균등 분활을 시도함으로써, 메모리의 오용이 적고 클리핑 및 충돌 처리시 폴리곤을 정확하고 빠르게 검출해 낼 수 있다. 옥트리의 렌더링의 기본 개념은 일단 가장 최상단의 옥트리부터 시작하여 8개 점의 각도를 구해서 화면 시야 여부를 결정한다. 8개의 점이 다 시야 안이라면 그 안의 옥트리를 그냥 더 이상의 클리핑도 필요 없이 렌더링에 필요한 노드를 검출한다. 8개의 점이 모두 밖에 있다면 그 안의 옥트리는 화면에 보이지 않는게 확실하므로 노드가 검출되지 않고 2가지의 경우가 아니라면 다시 하위 옥트리로 내려가 다시 앞의 2가지를 비교하게 된다. 옥트리에 의해 화면 분할이 끝나면 카메라의 viewing furstum과 옥트리는 모든 노드와의 포함연산을 통해 보여지는 노드를 검출한다. 카메라의 거리가 아닌 화면상에 나타난 크기를 기준으로 이력 문턱 값이 적용된 LOD가 적용하게 되는데, LOD 적용시 확대율의 계산의 경우 발생할 수 있는 카메라의 거리가 아닌 화면상에 나타난 크기를 기준으로 모델의 세부 수준을 결정해야 한다는 문제와 빠르게 반복되는 수주 전환의 문제를 이력 문턱 값을 적용함으로 하나의 값이 아니라 하나의 범위에 대한 문턱 값 적용으로 LOD의 빈번한 변화를 방지할 수 있다.

[적용 단계]

1 단계 3차원의 가상 공간의 지형데이타를 Height field를 이용하여 3차원 점좌표로 변환한다.

2 단계 옥트리를 이용하여 Height field점좌표를 8개의 화면으로 균등 분할한다.

3 단계 카메라의 viewing furstum과 옥트리를 모든 노드와의 포함연산을 통해 보여지는 노드를 검출한다.

4 단계 LOD를 적용하여 실시간으로 폴리곤을 생성한다.

1 단계 구현

- 3차원의 가상 공간의 지형데이타를 Height field를 이용하여 3차원 점 좌표로 변환 처리부분

```
void HEIGHTFIELD_Transform(HEIGHTFIELD* pWorld,
                           CAMERA* pCamera)
 {
  int nStartX, nEndX;
  int nStartZ, nEndZ;
  int x, z;
  pWorld->fXTrans=pCamera->POV.x/pWorld->fTileWidth;
  pWorld->fZTrans=pCamera->POV.z/pWorld->fTileDepth;
 // 일정 범위로 제한하기 위해, 처리할 영역을 계산
  nStartX = (int)pWorld->fXTrans;
  nStartZ = (int)pWorld->fZTrans;
  nEndX=min((nStartX+SEIGHT_WIDTH),pWorld-
  nWidth);
  nStartX=max(0,(nStartX-SEIGHT_WIDTH));
  nEndZ=min((nStartZ+SEIGHT_DEPTH),pWorld->nDepth);
  nStartZ=max(0,(nStartZ-SEIGHT_DEPTH));
 // 해당 정점을 변형
  for ( x = nStartX; x < nEndX; x++ ) {
       for ( z = nStartZ; z < nEndZ; z++ )
                    MATRIX_TransformWithHomogenous
(
  pCamera->W2P,
  pWorld->pVertex[z * pWorld->nWidth + x] );
       }
 }
 pCamera      : 카메라 개체
 pCamera->POV  : 시점자의 위치(PointOfView)
```

2 단계

- 옥트리를 이용하여 Height field점 좌표를 8개의 화면으로 균등 분할 방법 구현

화면을 8등분하고 화면의 중앙에서 8개 박스를 만들 수 있다. 그 바운드 박스 안에 들어온 폴리곤을 골라 임시 변수에 등록한다.(일정한 한계를 결정하는데, 2가지 방법이 대표적으로 쓰인다. 하나는 이 옥트리 내의 폴리곤 개수가 어느 정도 이하면 더 이상 쪼개지 않는 것과, 옥트리를 나누는 한계를 결정하는 것. 보통 유동적이고 효율적인 관리를 위해

서는 전자의 방법이 일반적임) 임시 저장된 폴리곤이 어느 정도 이상이면 옥트리안에 등록시키지 않고 다시 이 옥트리를 분할한다. 계속 분할하다 폴리곤의 개수가 어느 정도 이하로 떨어지면 옥트리구조체 내에 그 데이터들을 등록시키고, 분할을 멈춘다.

```
BuildOctree()
{
                                if(NumPolys>POLY_THRESHOLS)
        for(int i =0;i < 8; i++)
        {
      BuildNode(n->Child[i],lin);
            BuildOctree(n->Child[i];
          }
}
```

BuildOctree()의 기능을 살펴보면 다음과 같다.

- 노드에 대한 경계 입방체를 만든다.
- 그 입방체 안에 어떤 다각형들이 들어가는지 판단한다.

3 단계

- 카메라의 viewing furstum과 옥트리를 모든 노드와의 포함연산을 통해 보여지는 노드를 검출한다.

```
TriInCube(Tri T,Cube C)
{
Vector Trans = C.Center;
Vector Scale = 1.0 / C.Size;
For (int i=0;i<3;i++)
    T.Vert[i] = (T.Vert[i] Trans) / Scale;
If (TriInVoxe[T])
   Return true;
Return false;
}
```

4 단계

- LOD를 적용하여 실시간으로 폴리곤을 생성

옥트리에 의해 공간 분할된 3차원지형을 세부 수준으로 적용하기위한 방안으로 고려할 수 있는 부분을 을 살펴보자. 렌더링 할 세부 수준을 선택하는 간단한 방법은 카메라와 개체사이의 거리에 어떠한 문턱값을 적용하는가에 있다. 이방법에는 다음과 같은 문제점을 내포하고 있다. 첫째로 카메라와 시야 사이를 고려하지 않은 부분으로 개체가 카메라로부터 얼마나 떨어져 있다고 해도 카메라와 시야 각이 매우 작으면 화면상에 나타난 개체의 크기가 생각보다 크기 때문에 저 수준 모델을 사용해서는 안 된다.

즉 카메라의 거리가 아니라 화면상에 나타난 크기를 기준으로 모델의 세부수준을 결정한다는 것이다. 두 번째 문제는 개체가 문턱값 거리부근에서 계속 움직이는 경우 세부 수준의 전환이 대단히 여러 번 일어나게 된다. 세부수준을 결정을 위해서는 이러한 문제를 해결하기위해서 확대율과 이력 문턱값의 적용을 통하여 해결할 수 있다.

1. 이력 문턱 값 적용 방법

이력 문턱값의 적용은 하나의 값이 아니라 하나의 범위에 대한 문턱값을 적용하는 것이다.

단순문턱값과 이력문턱값의 차이를 살펴보면 다음과 같이 정의 할 수 있다.

```
단순문턱값 T
output =      {input >= T 이면 1
              {input <  T 이면 0
이력 문턱값 (t)
              {input >= T_high 이면 1
output(t) =   {input <  T_low 이면 0
              {그 외의 값이면 output(t-1)
```

입력이 증가하는 경우, 입력이 $T_{high}$를 넘어가기 전까지는 출력이 그대로 0이다. 입력이 감소하는 경우 입력이 $T_{low}$에 이르기 전까지는 출력이 그대로 1이다 입력이 증가하든 감소하든, $T_{low}$와 $T_{high}$ 사이에 있으면 출력은 변하지않는다.이러한 접근 방식을 LOD 선택에 적용하면 어떤 하나의 위치에서 개체가 왔다갔다해도 LOD가 계속 변하지 않는다. [2]

문턱 값이 적용된 LOD 선택 알고리즘의 의사 코드 구현 부분 (세가지 수준으로 고려된 경우)

```
int computelod:
worldpos     개체의 전역 위치
lodprev              이전프레임에서 선택된 LOD
```

```
{
        viewpos
        M=xscale/viewpos.z
        if ( M < T_mlower)
                Lod = low
        else if ( M < T_mmupper)
                Lod=lodprev
중간.저수준에 대한 이력 범위
        else if ( M < T_hlower )
                Lod=medium
        else if ( M < T_hupper )
                Lod = lodprev
고/중간 수준에 대한 이력 범위
        else
                Lod = high (m>=T_hupper)
return Lod
}
```

MOI $T_{low}$와 $T_{high}$ 사이에 있으면 이 함수는 MOI 저수준이라 하더라도 항상 이전의 LOD로 돌려준다.

## 5. 결론

본 논문은 3차원 게임엔진설계에서 발생되는 속도 문제를 어느 정도 해결하기위한 방안을 모색하려고 Height field와 옥트리가 가지고 있는 화면분활 방법에 세부 수준의 서로 다른 모델을 적용시킴으로써 렌더링의 성능과 시각적 품질을 높이기 위한 방법으로 모색하였다. Height field는 광범위한 지형을 각 정점의 좌표를 거리에 따라 효율적으로 적절한 폴리곤 수를 조절할 수 있는 데이터 저장방법과 옥트리를 이용하여 Height field 점 좌표를 8개의 화면으로 균등 분할함으로써, 메모리의 오용이 적고 클리핑 및 충돌 처리 시 폴리곤을 정확하고 빠르게 검출해 낼 수 있다. 옥트리는 정적인 지형에 적합하며, 동적으로 움직이는 개체들에 대한 부착 목록을 저장하는 데에도 사용된다. 옥트리는 가시성 판단이나 충돌 검출,개체관리에 효율적이다. 옥트리에 의해 화면 분할이 끝나면 카메라의 거리가 아닌 화면상에 나타난 크기를 기준으로 이력 문턱값이 적용된 LOD가 적용되게 된다. 본 논문에서 제시된 방법은 현재 3차원 가상공간에서의 외부지형과 개체표현에 적용한다면 효율적인 처리능력을 발휘할 수 있다.

참고문헌


[1] Jaap Suter, Introduction To Octrees 1999.4 URL:http://www.flipcode.com/tutorials/tut_octrees.shtml

[2]Mark Deloura, Game Programming Gems , CHARLES River MEDIA,INC.2000.8

[3] Astheimer, P. and P che, M.-L. Level-of-detail generation and its applications in virtual reality. 1994. In Virtual Reality Software and Technology (Proceedings of VRST'1994,8 23-26

[4]Mike Krus,Patrick Bourdot,Fran oise Guisnel, and Guillaume Thibault Levels of Detail & Polygonal Simplification Singapore), pages 299-312

[5]Foley, van Dam, Feiner, and Hughes, Computer Graphics: Principles and Practice 2nd Edition, 1987, p 550555.

[6]Hoff, Kenny, Fast ABBB/View-Frustum Overlap Test

[7]Moller and Haines, Real-Time Rendering, 1999, p. 206211, 310312.

[8]Suter, Jaap, Introduction to Octree , 1999.4 URL:http://www.flipcode.com/tutorials/tut_octrees.shtml

[9]Bryan Turner ,Real-Time Dynamic Level of Detail Terrain Rendering with ROAM,Gamasutra 2000.4 URL:http://www.gamasutra.com/features/20000403/turner_01.htm

[10]박현우,최혁종, Real Time 3D Graphics Technique.1999.5 URL: http://www.3dartisan.com/