한국항행학회 논문지 제15권 제4호 2011년 8월

Regular Mesh 기반 지리정보 3D 합성모델

Geographic information 3D Synthetic Model based on Regular Mesh

정지환*, 황선명**, 김성호***

Ji-Hwan Jung*, Sun-Myung Hwang** and Sung-Ho Kim***

요 약

본 연구에서는 지형을 Rendering 기법의 대표적인 방법인 Geometry Clipmaps와 ROAM 2.0을 분석하여 Rendering 연산에 소요되는 연산을 CPU가 아닌 GPU에 중점을 두어 보다 빠르고 넓은 가시화 영역을 보장하는 확장된 Geometry Clipmaps 알고리즘을 제안한다. 확장된 알고리즘은 LOD(Level of Detail)을 통한 각 레벨의 Mesh 구성 방법, 레벨간의 연결망 Mesh 구성 방법, VFC(View Frustum Culling)을 사용하여 Rendering을 최적화 할 수 있는 Mesh Block화 방안 그리고 최대 1m 해상도를 갖는 고해상도 영상 Mapping 방안 등을 포함하고 있다.

Abstract

There are two representative geometry rendering methods. One is Geometry Clipmaps, another is ROAM 2.0. We propose an extended Geometry Clipmaps algorithm which does not focus on CPU operation but the GPU for faster and wider visibility area. The extended algorithm presents mesh configuration method of each level by LOD, how to configurate Mesh network between levels, mesh block method for rendering optimization using VFC, and image mapping method to get high resolution up to 1 m.

Key words : Geometry Clipmaps, Terrain Rendering, VFC, LOD, GPU Programming

I. 서 론

3차원 지도에 관한 이슈는 과거로부터 현재까지 지속적으로 발달해왔다. 하지만 과거에는 그래픽카드의 한계로 인해서 3차원 지도에 관련된 알고리즘들은 그래픽 워크스테이션 이상에서나 실험할 수 있었고 우리가 원하는 실시간으로 변화하는 방법들을 실험하기에는 H/W의 기술이 뒷받침되지 않아 만족할 만한 성과를 거둘 수 없었다. 따라서 과거에 민, 군 주도로 개발되던 다양한 프로젝트들에서는 2D 이미지를 이용한 방법으로 2D 지도를 사용한 사업들만을 수행하였다. 하지만 H/W 기술이 급격하게 변화되면서 그래픽카드에 GPU라는 연산장치가 탑재되었고 조금씩 빠른 속도의 연산을 그래픽카드에서 담당하여 기존의 2D에서도 빠른 연산과 좋은 효과를 보여줄 수 있게 되었고 3D 지도 분야에서도 점점 높은 성

* (주)픽소니어
** 대전대학교
*** 국방과학연구소
· 제1저자 (First Author) : 정지환 · 교신저자 : 황선명
· 투고일자 : 2011년 6월 10일
· 심사(수정)일자 : 2011년 6월 10일 (수정일자 : 2011년 7월 15일)
· 게재일자 : 2011년 8월 30일

능을 보여주며 꾸준히 발전하여 현재에는 대부분의 민,군이 주도하는 사업들 속에 3D 시각화는 이제 필수 항목이라 할 만큼 중요한 비중을 차지하게 되었다.

우리가 다루고자 하는 3D 지형에 관한 알고리즘 또한 그래픽카드의 성능이 떨어지던 시절에는 그릴 수 있는 삼각형의 개수를 최소화하여 그래픽 카드에 최소한의 부하를 주도록 하는 방식의 알고리즘으로 시작하여 현재 엄청난 속도를 보여주는 GPU가 탑재된 그래픽 카드에서는 삼각형의 개수를 줄이기보다는 어떻게 하면 GPU의 성능을 극대화하여 보다 빠른 처리를 할 수 있는 알고리즘으로 변화되었다[1].

LOD(Level of Detail)측면에서 보자면 초창기의 지형 Rendering 방식은 CLOD(Continuous Level of Detail)로 실시간으로 삼각형을 분할하고 병합하는 방식으로 GPU의 계산속도보다 CPU에서 분할/병합 알고리즘에 더 치중하여 Rendering 성능을 높이는 방식을 사용했다[2, 3]. 현재에는 DLOD(Discrete Level of Detail)을 지원하여 CPU에서는 데이터를 로딩하는 연산만을 수행하도록 하고 나머지 연산은 GPU에서 Rendering에 관련된 부분만 집중적으로 담당할 수 있도록 최적화 되어갔다.

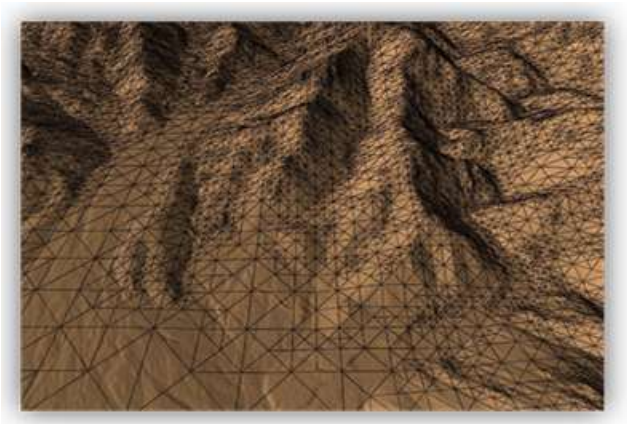

그림 1. CLOD를 사용한 지형렌더링
Fig.1 CLOD using Geometry Rendering

Mesh를 구성하는 측면에서 보면 초창기의 방식은 고도의 차이가 없는 지역은 하나의 Polygon으로 그리고 고도의 차이가 심한 지역만을 많은 Polygon으로 처리하는 Irregular Mesh를 구성하여 GPU에 Polygon의 처리를 최소화하는 방식을 사용했다[4, 5]. 하지만 현재에는 Polygon의 수보다는 Regular Mesh를 구성하여 GPU가 좀더 빠르게 처리할 수 있는 자료구조를 제공하여 연산량을 GPU쪽으로 옮겨 빠른 GPU의 연산을 극대화하는 방식으로 점차 변화하게 되었다.

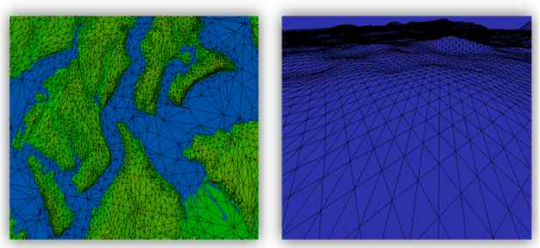

그림 2. Irregular Mesh[6] Regular Mesh의 구성
Fig.2 Composition of Irregular Mesh and Regular Mesh

이러한 지형 Rendering 알고리즘 변화에 맞춰 좀 더 빠르고 좋은 품질을 보장하는 Rendering 알고리즘은 앞으로 민, 군이 주도하는 다양한 3D 관련 사업들에서 반드시 사용하게 된다. 본 연구에서는 현재 GPU의 속도를 극대화 할 수 있는 지형 Rendering 알고리즘을 대표하는 Geometry Clipmaps와 ROAM 2.0을 분석하고 이들보다 더 좋은 품질의 3D 지형을 Rendering 할 수 있는 알고리즘을 제안하도록 하겠다.

Ⅱ. 관련연구

2-1 Geometry Clipmaps

Geometry Clipmaps는 F.Losasso와 H.Hoppe가 제안한 Terrain Rendering Using Nested Regular Grids을 기반으로 하고 있다. Geometry Clipmaps는 기존의 CLOD방식이나 Irregular Mesh를 구성하는 알고리즘에 비해 Polygon의 개수가 기본적으로 많이 들어간다. 하지만 렌더링 알고리즘과 LOD(Level of Detail)의 단순화 그리고 GPU 중심의 Rendering 방식을 사용하기 때문에 CPU 연산의 부하를 데이터 로딩 쪽에 치중하고 실제 렌더링은 GPU 쪽으로 옮겨 렌더링 성능을 극대화하여 기존의 렌더링 알고리즘보다 우수

하고 높은 성능을 보장할 수 있게 하였다[7].

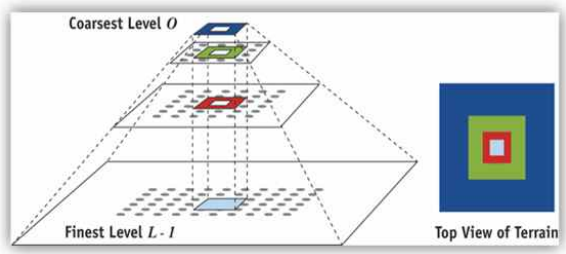

그림 3. Geometry Clipmaps의 기본 개념
Fig.3 Concept of Geometry Clipmaps

Geometry Clipmaps 제일 해상도가 높은 영역인 0 Level 부터 해상도가 가장 낮은 n 레벨을 사용자가 지정하여 LOD(Level of Detail)을 지정할 수 있으며 n 레벨은 n-1 Rendering영역을 제외한 영역만을 포함하도록 한다. 그림4에 Geometry Clipmaps로 Rendering 된 예시 화면을 보면 바깥레벨이 안쪽 레벨을 포함하는 형태로 각각의 레벨이 겹쳐져서 Rendering 되는 것을 확인할 수 있다[8].

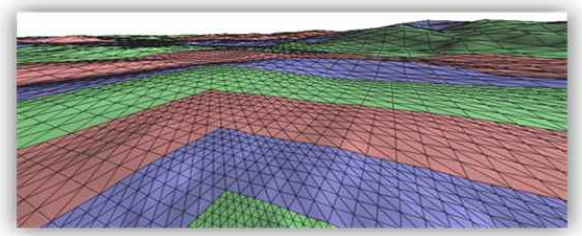

그림 4. Geometry Clipmaps 예시
Fig.4 Example of Geometry Clipmaps

또한 Geometry Clipmaps는 관찰자 중심의 중첩격자(Nested Grids) 형태로 지형을 Rendering하기 때문에 카메라의 이동과 맞춰서 레벨의 Grid도 같이 이동하여 새롭게 갱신하는 방식을 취하기 때문에 DLOD(Discrete Level of Detail)에서 많이 발생하는 Popping 현상을 최소화 할 수 있다는 장점도 존재한다[9].

Geometry Clipmaps 알고리즘은 GPU에 최적화하여 Rendering하기 위해서 1개의 Level을 다음 그림2와 같이 3개의 블록 형태를 사용하여 분할 Rendering 하며 각 레벨간의 연결은 선 형태로 되어있는 연결선을 각 레벨이 만나는 곳을 연결하여 연결망을 구성한다. 3가지 블록들은 GPU에서 한번에 그려질 수 있는 Strip 형태로 구성하여 Rendering 되어지기 때문에 OpenGL 또는 DirectX의 Draw 호출횟수를 최소화할 수 있는 장점이 존재하며 이렇게 분할된 각각의 블록들은 VFC(View Frustum Culling)시 불필요한 블록의 Rendering을 피할 수 있는 장점도 가지고 있다[8, 9].

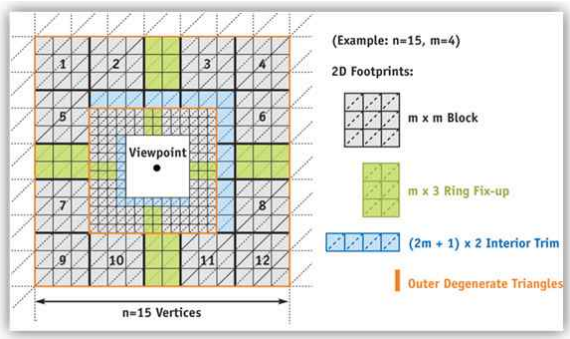

그림 5. Geometry Clipmaps의 블록구조
Fig.5 Block Structure of Geometry Clipmaps

그림5에서 제시하는 블록구조를 사용하여 Rendering을 진행할 때 VFC(View Frustum Culling)이 이루어지게 되면 그림6과 같은 형태로 Rendering이 되어 성능을 좀 더 끌어올릴 수 있다.

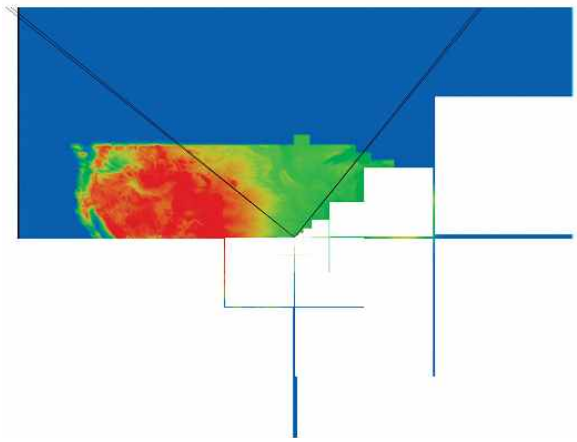

그림 6. VFC를 사용한 Rendering 영역
Fig.6 Rendering Area using VFC

2-2 ROAM(Real-Time Optimally Adapting Mesh) 2.0

ROAM 2.0은 기존의 CLOD(Continuous Level of Detail)를 이용한 지형 Rendering 방법을 수정하여 제

안한 방법이며 CLOD와 DLOD 방법을 동시에 사용한다. 기존의 ROAM은 삼각형을 분할하여 또 다른 삼각형을 생성하는 방식이었던 것에 반하여 ROAM 2.0은 사각형을 분할하여 또 다른 사각형을 생성하는 방식을 취한다. 이렇게 바뀐 중요한 이유는 전 지구를 가시화 할 수 있는 가장 좋은 방식이기 때문이다.

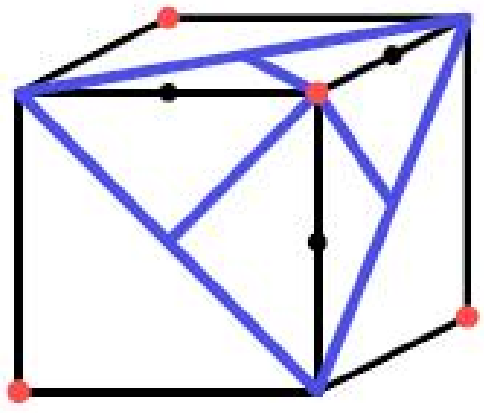

그림 7. ROAM 2.0을 사용한 전 지구 가시화 방식
Fig. 7 Round earth visible Method using ROAM 2.0

위의 그림 7을 보면 빨간 점으로 표시된 Vertex는 지구를 구성하는 구의 실제 좌표를 갖게 된다. 저 상태에서 파란색의 사각형들로 Split이 이루어지고 분할된 Vertex들은 3차원 좌표계 상에서의 Normal Vector로 변환되어 (지구반지름 + 고도)를 곱하여 주면 다시 지구를 구성하는 구의 실제 좌표와 계속하여 Mapping 할 수 있게 된다[10].

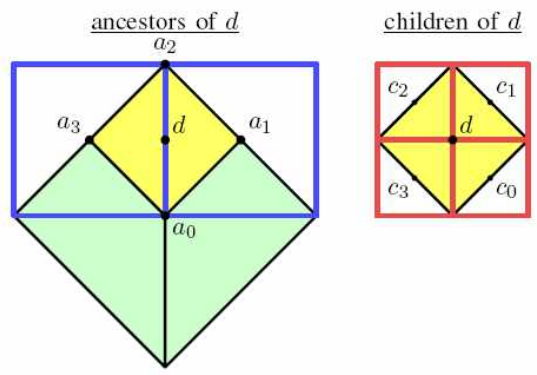

그림 8. ROAM 2.0의 구조
Fig. 8 ROAM 2.0 Structure

ROAM 2.0은 그림 8과 같은 독특한 구조를 갖는데 이는 하나의 사각형은 두 개의 부모를 갖고 4개의 자식을 갖는 형태로 구성이 되어진다. 따라서 트리 구조와 같은 단순한 알고리즘으로 사각형들을 관리할 수 없기 때문에 사각형의 Split/Merge에 두 개의 다른 Queue 사용하여 관리한다. 분할에 사용되는 Queue는 Split Queue, 병합에 사용되는 Queue는 Merge Queue라 정의하여 Split될 사각형과 Merge할 수 있는 사각형들을 따로 관리하게 된다[11].

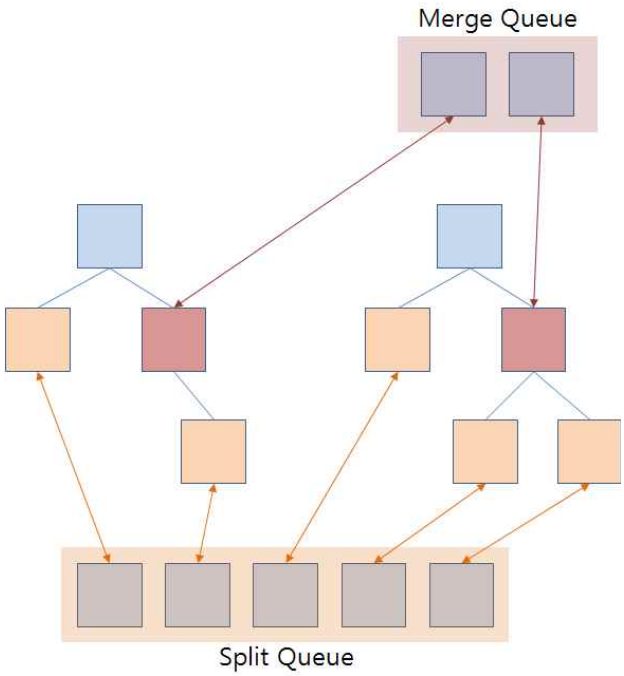

그림 9. ROAM 2.0 Split/Merge Queue
Fig. 9 ROAM 2.0 Split/Merge Queue

ROAM 2.0의 하나의 사각형은 두 개의 삼각형으로 분할하여 Rendering 한다. 이는 Geometry Clipmaps과 유사하게 사각형을 두 개의 삼각형으로 분할하여 하나의 삼각형에 Regular Triangle로 Mesh를 구성하여 GPU 중심의 연산을 수행할 수 있도록 한다.

그림 10은 ROAM 2.0의 Mesh 구성을 보여주며 이러한 Mesh는 부모 사각형과 자식 사각형의 연결지점의 Vertex들이 동일한 Vertex 좌표를 갖는 것을 보이고 있다[12, 13].

하지만 이러한 ROAM 2.0은 전 지구 가시화에도 적합하고 GPU 중심적 연산도 가능하지만 Rendering 시 가장 낮은 LOD(Level of Detail)가 전 지구를 커버

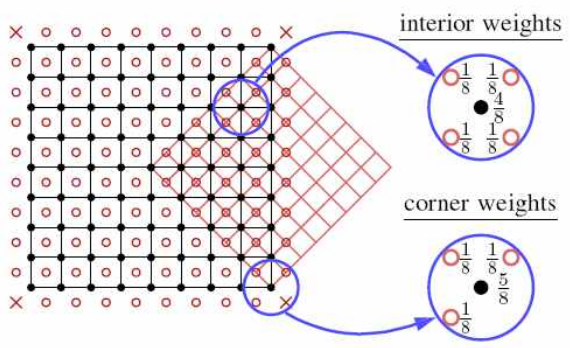

그림 10. ROAM 2.0의 Mesh 구성
Fig.10 Mesh composition of ROAM 2.0

할 수 있을 정도의 낮은 해상도부터 시작하기 때문에 높은 해상도로 들어가면 갈수록 Split /Merge 작업에 굉장히 많은 CPU 연산을 소모할 수 있다. 또한 ROAM 2.0 자체가 DLOD(Discrete Level of Detail) 방식을 취하기 때문에 레벨의 변환이 이루어질 때 타일을 통째로 교체하기 때문에 Popping 현상이 발생한다는 단점을 가지고 있다.

2-3 기존 지형 Rendering 알고리즘의 비교

Geometry Clipmaps와 ROAM 2.0은 모두 광역지형을 Rendering하기에 적합하다 할 수 있는 Rendering 알고리즘들이다. 이 두 알고리즘과 과거에 많이 사용되었던 전형적인 CLOD(Continuous Level of Detail) 알고리즘을 대표하는 QUAD-TREE 알고리즘을 비교하면 다음의 표로 정리할 수 있다

표 1. Rendering 알고리즘의 비교
Table 1. Compare of Rendering algorithms

방법 / 항목	QUAD-TREE	Geometry Clipmaps	ROAM 2.0
렌더링성능	낮음	매우높음	높음
CPU 연산량	높음	매우낮음	다소높음
LOD 방식	CLOD	CLOD/DLOD	CLOD/DLOD
Popping	없음	없음	있음
VFC가능여부	가능	가능	가능
GPU 요구치	낮음	높음	높음
Crack	없음	있음	없음
알고리즘 복잡도	높음	낮음	높음
가시거리	Far설정	Far 설정	짧음

각기 알고리즘들은 각 항목별로 다른 점들을 가지고 있지만 나중에 나온 Geometry Clipmaps나 ROAM 2.0 알고리즘들이 더 좋은 점들을 표를 통하여 확인할 수 있다. 특히 Geometry Clipmaps는 Rendering 성능이 우수하기 때문에 본 연구에서는 Geometry Clipmaps 알고리즘을 개선하여 더 좋은 품질의 알고리즘을 제안한다.

Ⅲ. 확장된 Geometry Clipmaps 알고리즘

3-1 Geometry Clipmaps의 문제점

Geometry Clipmaps 알고리즘은 다른 알고리즘과 비교하였을 때 가장 좋은 성능을 보여주고 있지만 해결하지 못한 몇 가지의 문제점을 안고 있다. 이러한 문제점은 다음과 같다.

첫 번째로 Geometry Clipmap는 광역 지형을 렌더링 하기엔 짧은 가시거리를 갖는다. 가장 낮은 해상도를 갖는 LOD(Level of Detail)의 Grid 개수만큼만 가시거리를 갖기 때문에 넓은 가시거리를 가질 수 없다. 일반적으로 지형 알고리즘에서 사용하는 고도데이터는 Vertex당 간격이 약 30m의 간격을 갖는다. 그럼 Geometry Clipmaps에서 n레벨은 (n-1)레벨의 2배의 간격을 갖게 된다. 공개되어있는 Geometry Clipmaps 알고리즘 논문을 살펴보면 일반적으로 5개의 Level로 구성하며 하나의 Level을 Mesh를 만들기 위해 필요한 Vertex 개수는 256*256 개로 정의하고 있다. 그리고 Geometry Clipmaps 알고리즘은 항상 Rendering 되는 지형의 중심이 3D 카메라의 Eye Point로 설정이 된다. 이러한 조건들을 종합하여 Geometry Clipmaps에서 표현 가능한 가시 거리는 일반적으로

- 480m * 256 / 2 = 61.44Km 이다

위의 식으로 약 61Km 정도의 가시거리를 갖게 된다. 61Km면 맑은 날 항공기에서 볼 수 있는 가시거리를 넘지만 실제 프로젝트 진행 중에는 100Km이상 가시화 하라는 요구사항들이 포함되어있는 경우가 있기 때문에 Geometry Clipmaps로 구현의 한계가 있다.

Geometry Clipmaps은 또한 레벨간의 연결망이 선으로만 존재하여 Crack을 원천적으로 해결할 수 없는 구조로 되어있다.(Crack은 T-Junction이라고도 .하며 서로 다른 레벨의 Mesh가 만났을 때 Vertex 들이 일치하지 않는 것을 의미한다.) 논문들을 참조해보면 대부분 Map의 중심이 항상 카메라의 Eye Point이기 때문에 카메라 움직일 때 항시 Crack이 발생한 부분도 같이 움직이기 때문에 무시할 수 있는 정도라고 말하고 있지만 이는 연구에서는 그렇게 말할 수 있지만 실제 구현하는 부분에서는 반드시 연결망이 존재해서 Crack을 원천적으로 해결할 수 있는 구조의 Mesh 구조를 가지고 있어야 한다.

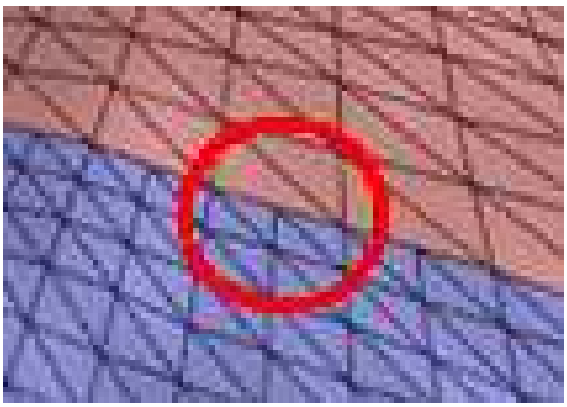

그림 11. Geometry Clipmaps에서 발생하는 Crack
Fig.11 Crack of Geometry Clipmaps

위의 두 가지 문제점은 빠른 속도와 단순한 알고리즘임에도 불구하고 다양한 분야에 적용하기 힘들다는 단점을 가지고 있다. 확장된 Geometry Clipmaps는 넓은 가시거리와 Crack을 원천적으로 해결할 수 있는 장점을 가지고 있다.

마지막으로 Geometry Clipmaps에 영상을 Mapping 할 경우에 고해상도의 영상을 Mcapping 하기 어렵다는 단점을 가지고 있다. 이점에 대해서는 다음 장에서 자세히 설명하도록 하겠다.

3-2 광역지형 3D 합성 알고리즘

확장된 Geometry Clipmaps 알고리즘은 Regular Grid를 사용한 Mesh 구성과 카메라를 기준으로 항시 LOD(Level of Detail)이 구성된다는 점에서 Geometry Clipmaps와 동일한 방식을 채용하고 있다. 하지만 좀 더 넓은 가시화 영역을 확보하기 위해서 카메라를 기준으로 5개의 레벨을 각각 하나씩 갖는 Geometry Clipmaps과 달리 카메라를 기준으로 5개의 레벨을 각각 4개씩 포함하여 갖는다. 이렇게 구성된 방식은 그림 12과 같다.

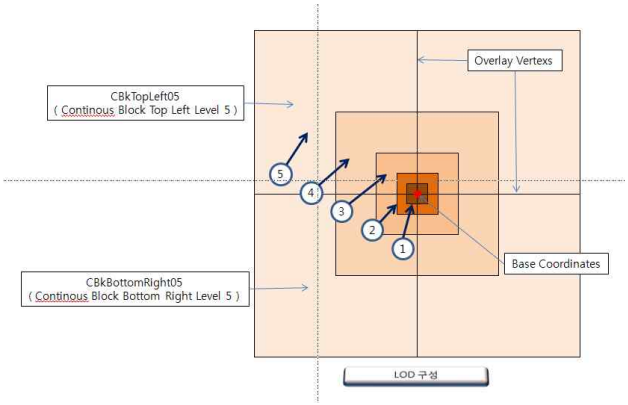

그림 12. 확장된 Geometry Clipmaps
Fig.12 Extended Geometry Clipmaps

Geometry Clipmaps과는 달리 중심으로부터 4방향으로 4개의 영역이 구성되기 때문에 Geometry Clipmaps의 약 2배정도의 가시거리를 확보할 수 있게 되며 DTEDII(약 30m) 해상도의 지형 데이터를 사용할 경우 다음의 표 2과 같은 가시거리를 구성하게 된다.

표 2. LOD 레벨별 가시영역
Table 2. Visible Area of LOD

LOD Level	Grids별 실제거리	Geometry Clipmaps 가시영역	확장된 Clipmaps 가시영역
Level 0	30m	3,840m	7,680m
Level 1	60m	7,680m	15,360m
Level 2	120m	15,360m	30,720m
Level 3	240m	30,720m	61,440m
Level 4	480m	61,440m	122,880m

Geometry Clipmaps의 두 번째 문제였던 각 레벨이 연결되는 부분에 반드시 Crack이 발생하는 부분은 확장된 Geometry Clipmaps에서는 각 레벨간 연결되는 부분의 Mesh 구성을 연결망 구조로 바꾸어 Crack 문제를 원천적으로 해결하였다. 이렇게 연결망을 구성

하면 Mesh 구조가 Regular Grid 형태가 아니기 때문에 약간 복잡하다는 문제를 갖게 되지만 처음 Mesh를 구성하는 Index Buffer 생성시에 이 부분을 고려하여 프로그래밍 한다면 복잡도에 관련된 문제가 발생하지 않는다. Crack을 사용한 연결망 구성은 다음의 그림 13와 같다.

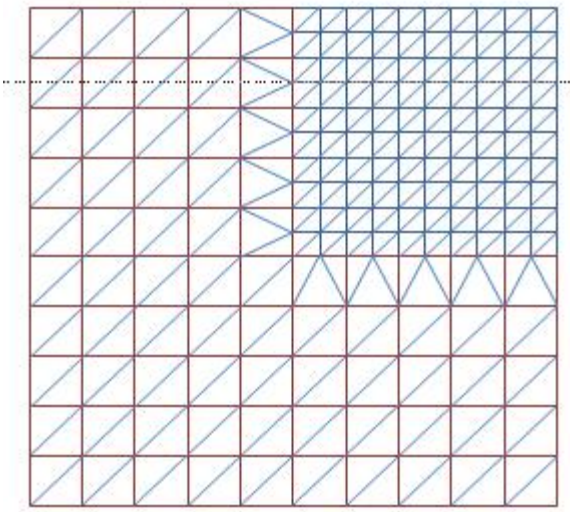

그림 13. Crack을 해결하기 위한 연결망 구성
Fig.13 Network for Crack

Geometry Clipmaps 알고리즘에서는 Rendering 최적화를 위하여 3개의 블록 형태를 사용하여 각각의 레벨을 분할 Rendering 하는 방식을 사용하였다. 이는 각각의 블록들을 VFC(View Frustum Culling)하여 Rendering에서 제외시킬 수 있기 때문이다. 일반적으로 OpenGL/DirectX 에서는 이러한 Culling 기능들이 기본적으로 동작하지만 Geometry Clipmaps에서는 Rendering 시에 GPU에서 많은 연산을 수행하기 때문에 미리 VFC(View Frustum Culling)에서 제외 시키면 그만큼 GPU 연산을 줄일 수 있기 때문에 Rendering 최적화 하는데 도움이 된다.

확장된 Geometry Clipmaps 역시 기존의 방법과 마찬가지로 Rendering을 블록화 하여 수행한다. 하지만 기존의 방법과 달리 정확하게 한 레벨을 16등분으로 조각내어 타일화 한다. 이렇게 타일화하여 Rendering을 수행하면 VFC(View Frustum Culling)에서 걸러진 블록들만 Rendering 되어지는데 이는 다음의 그림 14와 같다.

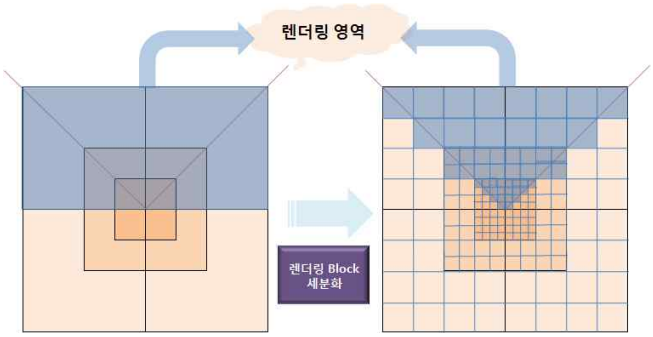

그림 14. VFC를 적용한 레벨별 Block
Fig.14 Block using VFC

위의 그림과 같이 타일화 하면 가장 상세한 Level 0는 16개의 타일로 분해되어지며 Level 1 ~ Level 4는 각각 이전 레벨의 영역을 포함하기 때문에 각각 12개의 타일로 분리되어진다. 이렇게 분리된 타일들은 유사한 형태의 타일들로 분리되어지는데 이는 실제 프로그래밍 시에 Index Buffer 구성을 간결하게 구성할 수 있다는 장점을 갖게 해준다.

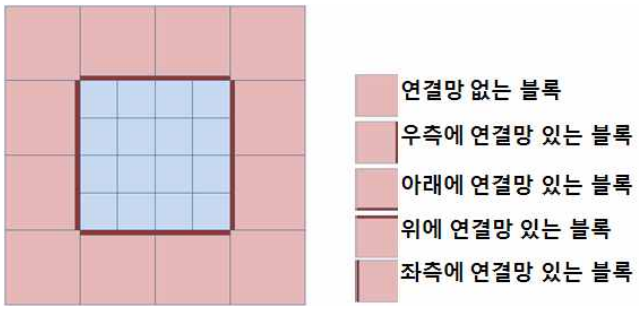

그림 15. Mesh를 구성하는 5가지 블록 Type
Fig. 15 Block Types of Mesh Composition

위의 5가지 타일들은 처음 프로그램이 초기화 될 때 각각의 Type 별로 미리 Index Buffer에 할당이 되어지게 된다. 연결망을 포함하고 있어 다소 복잡해 보일 수 있지만 Index 구성 시에 한번에 그릴 수 있는 구조로 한번만 정의하면 실행 시에는 변경되지 않기 때문에 속도와는 무관하다. 이러한 타일들은 무조건적으로 한번에 그릴 수 있도록 구성이 되어야 하는데 이는 OpenGL/DirectX에서 Draw하는 API를 호출하는 횟수가 Rendering 성능에 상당히 밀접하다는 실험치가 존재한다는 것이다. 실험적으로 테스트 해본 결과 Draw 횟수가 어느 정도 수준까지는 동일한 Polygon을 Rendering 하는데 일정한 속도를 보여주지만 어느

한계치를 넘어가면 기하급수적으로 속도가 감소하는 확인할 수 있었다. 이를 위해 각 타일들은 Triangle Strip 형태로 한번에 한 타일을 그릴 수 있는 구조로 설계 되어야 한다.

3-3 고해상도 영상 매핑 방법

Geometry Clipmaps는 각 레벨에 영상을 Mapping 할 시에 Grid의 개수와 사이즈가 같은 영상을 Mapping 한다. 예를 들어 일반적으로 사용하는 256*256 Grid를 사용하는 Geometry Clipmaps 는 256*256 해상도를 갖는 영상을 Mapping 할 수 밖에 없다. 현재 일반적으로 사용하고 있는 고해상도 영상은 약 Pixel당 1m 정도의 해상도를 갖는 영상을 사용하고 있다. 그러면 DTED II급(약 30m) 해상도의 고도 데이터를 사용시에 3,840*3,840의 해상도의 이미지를 사용하여야 1m급 해상도의 영상을 Texture로 Mapping이 가능하다는 의미를 갖는다. 물론 영상 자체를 Resampling(해상도를 강제로 조정) 할 수도 있지만 이렇게 되면 영상이 뭉개지는 문제가 발생하여 원영상에 훼손을 가져오게 된다. 그리고 OpenGL/DirectX에서 Texture의 사이즈는 2n 크기를 가져야 최적화된 성능을 보장할 수 있다. 따라서 Texture 크기 역시 맞춰야 하는 문제도 가지고 있다.

이러한 문제를 해결하기 위해서 확장된 Geometry Clipmaps 방법은 영상을 Mapping하기 위해 기본적으로 각 LOD별 지형과 동일한 해상도를 갖는 영상을 사용하여 Texture Mapping을 수행한다. 이는 속도와 효율적인 측면에서 유리하도록 설계하기 위한 방법이다. 하지만 DTED II(약 30m)의 지형 데이터를 사용하게 되면 약 30m급 영상(1Pixel = 30m)이 Level0에 Mapping이 되기 때문에 상세한 위성영상을 Mapping 할 수 없다는 단점이 생기게 된다. 특히 Level 0의 영역을 1m 영상으로 Texturing 한다고 가정하면 대략 7680 x 7680 해상도를 갖는 영상을 사용하여 Mapping이 이루어져야 하는데 3D에 최적화된 2n 의 Texture 사이즈도 아닐뿐더러 영상 사이즈가 너무 크기 때문에 최적화에 맞지 않는걸 쉽게 알 수 있다. 이러한 문제점들을 해결하기 위해서 OpenGL/DirectX에서 제공되는 Multi-Texturing 알고리즘을 사용하여 고해상도 영상을 Mapping 하도록 한다. (Multi-Texturing 방식은 하나의 Mesh에 여러개의 Texture를 동시에 Mapping 하는 방식을 말한다.) Level 0 영역의 하나의 블록 사이즈는 약 1920m X 1920m가 된다. 1920 x 1920을 포함하는 최소의 2n Texture 사이즈는 2048 x 2048의 Texture가 가장 최적화된 Texture 사이즈가 된다.

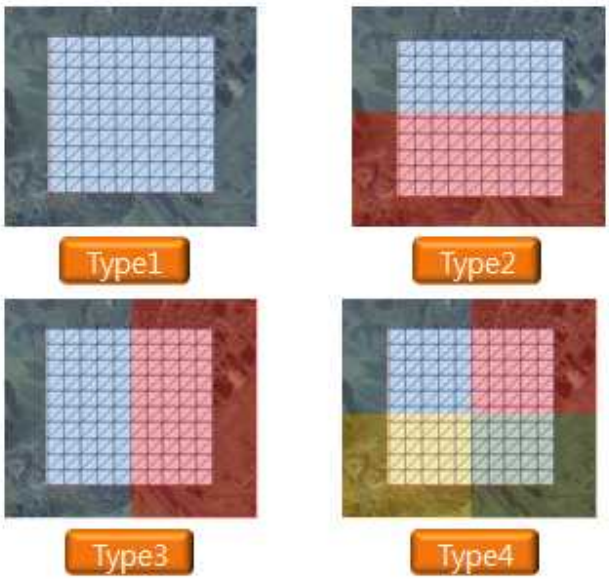

그림 16. Multi-Texturing을 사용한 매핑 방법
Fig. 16 Multi-Texturing mapping method

2048*2048사이즈의 영상은 하나의 블록을 완전히 포함할 수 있는 Texture로 생성이 되며 이를 Multi-Texture를 사용하여 블록에 Mapping 할 경우 그림 15와 같이 4가지 형태의 Mapping 방식이 존재하게 된다.

- Type 1 : 1장의 Texture가 Mesh 블록을 완전히 덮는 경우
- Type 2 : 2장의 Texture가 상, 하 2장으로 Mesh 블록을 완전히 덮는 경우
- Type 3 : 2장의 Texture가 좌, 우 2장으로 Mesh 블록을 완전히 덮는 경우
- Type 4 : 4장의 Texture를 사용하여 Mesh 블록을 완전히 덮는 경우

위의 4가지 Type의 Texture Mapping 방식은 각각의 별도로 작성된 Shader로 동작할 수 있도록 구성한

다. 단점이라고 할 수 있는 것은 현재 Rendering 될 블록이 어떤 형태의 Mapping 방식을 사용해야 할 지에 대해서 Rendering 이전에 결정되어야 한다는 문제가 존재하지만 연산량이 많지 않아 속도에는 지장이 없었다.

3-3 시험 환경

확장된 Geometry Clipmaps를 구현하여 시험한 환경은 다음의 표3과 같다

표 3. 시험환경
Table 3. Test Environment

시험환경	내 용
CPU	Core2Duo 2.4GHz
RAM	2GB
Graphic Card	NVidia Geforce Fx 8600GS (512MB)
운영체제	Window XP
그래픽 라이브러리	OpenGL 2.0/Extension
Shader	GLSL 3.0

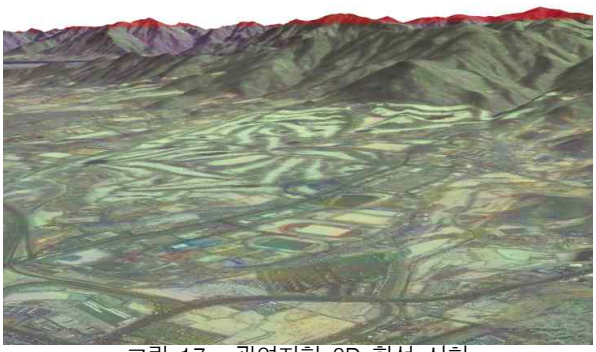

그림 17. 광역지형 3D 합성 시현
Fig.17 Implementation of Large Terrain 3D

Ⅳ. 결론 및 향후 연구과제

현재 존재하는 광역 지형을 Rendering 하기 위한 알고리즘들은 QUAD-TREE, Geometry Clipmaps, ROAM 2.0 P-BDAM등을 비롯하여 무수히 많이 존재한다. 이들은 과거부터 계속 발전된 CLOD(Continuous Level of Detail), DLOD(Discrete Level of Detail)들을 사용하며 다른 방식으로 발전해 왔다. 하지만 이러한 Rendering 알고리즘들은 처음 제안 되었을 당시에 H/W 사양에 따라 최대한의 성능을 이끌어 낼 수 있도록 개발되었기 때문에 그들의 알고리즘 자체에 대해서 어느 것이 더 좋고 그르다 판단하는 것은 쉬운 이야기가 아니다. 특히나 임베디드 장비와 같은 경우는 일반 PC보다 훨씬 낮은 H/W 사양으로 만들어졌기 때문에 각각 주어진 조건에 맞는 알고리즘을 사용하는 것이 좋은 방법이라고 할 수 있겠다. 본 논문에서는 일반 PC 또는 그래픽 워크스테이션 급에서 3차원 지형을 Rendering 하기에 최적화 할 수 있는 알고리즘이 필요했고 최신의 기술을 가장 많이 담고 있는 Geometry Clipmaps 와 ROAM 2.0을 분석해보고 이들 중 더 높은 성능을 낼 수 있는 Geometry Clipmaps의 단점들을 보완하여 확장된 Geometry Clipmaps 알고리즘을 제안했다. 이를 보완한 결과 원래 알고리즘 보다 2배 넓은 가시거리를 확보할 수 있었으며 Crack이 발생하는 문제를 원천적으로 해결하였고 고해상도 영상을 Mapping 할 수 있게 되었다.

실제 구현된 알고리즘은 초당 100프레임 이상의 안정적인 속도를 보장하고 있어 다양한 분야에 적용 가능함이 입증이 되었다. 현재 이 알고리즘을 사용하여 다채널 화면상에 IG(Image Generator), 임베디드 리눅스에 탑재하여 항공기 탑재용 SVS(Synthetic Vision System)에 적용되어 향후 시뮬레이터 분야, 항공기 임베디드 S/W 분야에 지속적으로 사용이 될 수 있을 것으로 예상된다.

하지만 LAN(Local Area Networks)를 통하여 데이터를 전송하는 부분에 대해서도 고려하여 설계, 구현하여 WAN을 통한 데이터 통신을 확장하는 부분과 Google Earth, Virtual Earth 등과 같은 전 지구 가시화 S/W 처럼 전 지구를 자연스럽게 가시화하는 부분에 대해서는 추가적인 연구가 필요하다.

감사의 글

본 연구는 국방과학연구소 민군기술협력지원단의 민군겸용기술사업으로 수행되었다. (09-SN-IC-01)

참 고 문 헌

[1] Samet, H. The Quadtree and Related Hierarchical Data Structures. *ACM Computing Surbeys 16(2),* June 1984, pp. 187~260.

[2] Lindstorm Peter[et al.] RealiTime, Continuous Level of Detail Rendering of Height Fields, *SIGGRAPH.* 1996. -pp. 109~118.(CLOD)

[3] Lario R., Pajarola R., Tirado F. Hyper-block Quad-tree based Triangulated Irregular networks. *In Proceedings of IASTED VIIP*, pages 733~738, 2003

[4] Cignoni, P., Puppo, E., Scopigno, R. 1997. Representation and Visualization of Terrain Surfaces at Variable Resolution. *The Visual computer 13(5)*, 199~217.

[5] Cignoni, P., Ganovelli, F., Gobbetti, E., Marton, F., Ponchio, F., Scopigo, R. 2003a. BDAM - Batched Dynamic Adaptive Meshes for high Performance Terrain Visualization. *Computer Graphics Forum 22(3)*

[6] Cignoni, P., Ganovelli, F., Gobbetti, E., Marton, F., Ponchio, F., Scopigo, R. 2003b. Planet-sized Batched Dynamic Adaptive Meshes(P-BDAM). *IEEE Visualizaiton 2003.*

[7] Tanner, C. C., Migdal, C. J., Jones, M. T. : The Clipmap : a Virtual Mipmap. In SIGRAPH '98 : Proceedings of the 25th annual conference on computer Graphics and Interactive Techniques(New York, NY, USA, 1998), *ACM Press*, pp. 151~158.

[8] Lossa, Frank, and Hugues Hoppe(2004). "Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids.", *ACM Transaction on Graphics(Proceeding of SIGGRAPH 2004) 23(3)*, pp. 769-776.

[9] Arul Asirvatham, Hugues Hoppe. GPU GEMS : Terrain Rendering Using GPU-Based Geometry Clipmaps. *Addison Wesley*, 2004

[10] M. Duchaineau, M. Wolinski, D. Digeti, M. Miller, C. Aldrich, M. Mineev-Weinstein, "ROAMing Terrain : Real-Time Optimally Adaping Meshes", *IEEE Visualizaiton,* Oct. 1997, pp. 81~88

[11] Hakl, H., Zijl, L. V. Diamond algorithm : continuous Level of Detail for Height Fields. *South African computer Journal(2002).*

[12] Lok M. Hwa, Mark A. Duchaineau,, and Kenneth I. Joy(2004). Adaptive 4-8 Texture Hierarchies. *15th IEEE Visualization 2004(VIS'04)* pages 219-226, October 2004.

[13] Lok M. Hwa, Mark A. Duchaineau,, and Kenneth I. Joy(2005). Real-Time Optimal Adaptation for Planetary Geometry and Texture : 4-8 Tile hierarchies. *IEEE Transactions on Visualization and Computer Graphics, March/April 2005(Vol. 11, No. 2).* [10a] page 9, [10b] page 13, [10c] page 12

정 지 환

(주)픽소니어

황 선 명

e-mail : sunhwang@dju.ac.kr
1982년 : 중앙대학교 전자계산학과 (이학사)
1984년 : 중앙대학교 소프트웨어 공학전공 (이학석사)
1987년 : 중앙대학교 소프트웨어 공학전공 (이학박사)

2007년~ 현재 : 한국정보처리학회 S/W공학연구회 운영위원장
2000년~현재 : 한국 S/W프로세스 심사인협회(KASPA) 이사
2000년~2004년 : 한국정보처리학회 논문지 편집위원
1997년~현재 : ISO/IEC JTC7/WG10 한국운영위원
1998년~현재 : 한국정보통신기술협회TTA 특별위원
1989년~현재 : 대전대학교 컴퓨터공학과 교수

김 성 호

국방과학연구소