**캡스톤 디자인 I**

**종합설계 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *3D Map Viewer* |
| 팀 명 | *Land Mark* |
| 문서 제목 | 결과보고서 - 3D Map VIEWER |

|  |  |
| --- | --- |
| **Version** | 1.5 |
| **Date** | 2014-may-27 |

|  |  |
| --- | --- |
| **팀원** | 배 판근 (조장) |
| 김 현봉 |
| 서 한석 |
| 이 정란 |
| 황 인섭 |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 “3D Map Viewer”를 수행하는 팀 “Land Mark”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “Land Mark”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 결과보고서-3D Map Viewer.doc |
| **원안작성자** | 서한석, 이정란, 황인섭 |
| **수정작업자** | 김현봉, 배판근 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2014-05-20 | 서한석 | 1.0 | 최초 작성 | 초안 작성 |
| 2014-05-22 | 이정란  황인섭 | 1.1 | 내용 추가 | 개발 내용 및 결과물 추가 |
| 2014-05-23 | 김현봉 | 1.2 | 내용 수정 | 향후 추진 계획 수정 |
| 2014-05-24 | 배판근 | 1.3 | 내용 수정 | 참고 문헌 및 개요 수정 |
| 2014-05-25 | 김현봉 | 1.4 | 내용 추가 | 자기 평가 추가 |
| 2014-05-26 | 전체 | 1.5 | 내용 검토 | 전체 내용 검토 |

**목 차**

[1 개요 5](#_Toc388903520)

[1.1 프로젝트 개요 5](#_Toc388903521)

[1.2 추진 배경 및 필요성 6](#_Toc388903522)

[1.2.1 국내 3D Mobile Map Service 현황 6](#_Toc388903523)

[1.2.2 Mobile 기기의 발전 7](#_Toc388903524)

[2 개발 내용 및 결과물 8](#_Toc388903525)

[2.1 목표 8](#_Toc388903526)

[2.2 연구/개발 내용 및 결과물 8](#_Toc388903527)

[2.2.1 연구/개발 내용 8](#_Toc388903528)

[2.2.2 Quad Tree 9](#_Toc388903529)

[2.2.3 평면 방정식 10](#_Toc388903530)

[2.2.4 Frustum Culling 10](#_Toc388903531)

[2.2.5 Crack 12](#_Toc388903532)

[2.2.6 Texture Mapping 13](#_Toc388903533)

[2.2.7 MipMap 14](#_Toc388903534)

[2.3 시스템 기능 및 구조 설계도 14](#_Toc388903535)

[2.3.1 개발 환경 14](#_Toc388903536)

[2.3.2 System Architecture 15](#_Toc388903537)

[2.4 시스템 구현 15](#_Toc388903538)

[2.4.1 프로그램 실행 과정 15](#_Toc388903539)

[2.4.2 Quad Tree 생성 16](#_Toc388903540)

[2.4.3 Image Load 17](#_Toc388903541)

[2.4.4 Frustum Culling 적용 18](#_Toc388903542)

[2.4.5 Level Of Detail 적용 19](#_Toc388903543)

[2.4.6 Rendering 20](#_Toc388903544)

[2.4.7 Camera Event 20](#_Toc388903545)

[2.4.8 Class Diagram 23](#_Toc388903546)

[2.5 활용/개발된 기술 24](#_Toc388903547)

[2.5.1 단일 Level에 대한 MipMap 24](#_Toc388903548)

[2.5.2 Multi Level에 대한 MipMap 27](#_Toc388903549)

[2.6 현실적 제한 요소 및 그 해결 방안 28](#_Toc388903550)

[2.6.1 Quad Tree간 Crack 현상 28](#_Toc388903551)

[2.6.2 경/위도 정보의 부재 28](#_Toc388903552)

[2.7 기대효과 및 활용방안 29](#_Toc388903553)

[3 자기평가 30](#_Toc388903554)

[3.1 기대효과 30](#_Toc388903555)

[3.2 사용가능성 30](#_Toc388903556)

[4 참고 문헌 31](#_Toc388903557)

[4.1.1 기타 참고 자료 31](#_Toc388903558)

[5 부록 32](#_Toc388903559)

[5.1 사용자 매뉴얼 32](#_Toc388903560)

[5.2 운영자 매뉴얼 32](#_Toc388903561)

[5.3 배포 가이드 32](#_Toc388903562)

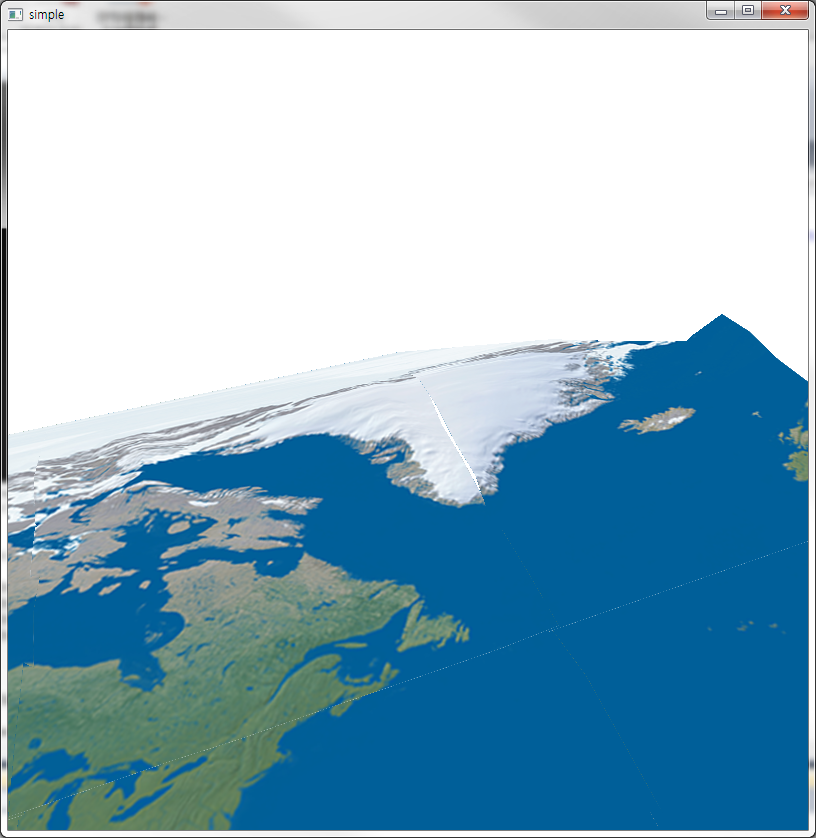
[5.4 XXX 매뉴얼 32](#_Toc388903563)

[5.5 XXX에 대한 기술 문서 32](#_Toc388903564)

# 개요

## 프로젝트 개요

현재 국내에서 3D Mobile Map Service를 제공하는 곳은 국토교통부에서 제작된 ‘VWorld’ 가 유일하다. ‘VWorld’는 오픈플랫폼이 제공하는 클라이언트 프로그램을 활용하여 새로운 서비스를 구축하기 용이하다는 장점이 있지만, 오픈플랫폼 클라이언트의 기능적 한계를 벗어나는 서비스를 구축할 수 없다는 단점이 존재하며, 서비스 지역이 한국이기 때문에, 한국을 벗어난 다른 국가의 경우에는 3차원 지형 서비스를 지원하지 않는다.



(그림 1 - 1) 프로그램 실행 화면

따라서 본 프로젝트에서는 위성사진과 고도데이터 사진을 이용하여 3차원의 지형정보를 지원하는 Viewer를 제작하는 것이 목표이다. 사용자들은 어플리케이션을 이용하여 서버로부터 PNG 형태의 이미지(위성사진, 고도데이터)를 다운로드 하여, 지형을 3차원으로 가시화 할 수 있다.

[표 1 - 1] 3D Map Viewer 기능

|  |  |
| --- | --- |
| 분 류 | 기 능 |
| 카메라 이동 | 카메라 위치에 따른 지형 전환 |
| Zoom-In/Out | 카메라의 고도에 따른 지형 레벨 전환 |

현재 개발된 3D Map Viewer의 기능으로는 크게 ‘Camera 이동’ 기능과 ‘Zoom-In/Out’ 기능이 있다. ‘Camera 이동’ 기능은 카메라가 이동한 위치에 따라서, 불필요한 지형을 삭제하고, 새로운 지형을 서버에서 다운로드 받아 화면에 출력하는 기능이며, ‘Zoom-In/Out’ 기능은 카메라와 지형간의 거리에 따라 해상도를 높임으로써, 가까운 거리에서 지형을 볼 경우 높은 해상도의 지형을 볼 수 있다.

## 추진 배경 및 필요성

### 국내 3D Mobile Map Service 현황

정보통신기술의 발전으로 점차 GIS(Geographic Information System)가 발전하고 모바일 GIS 또한 발전하고 있다. 모바일 GIS는 민간, 공공 모든 부분에서 ITS, LBS, 텔레매틱스 기술과 함께 이들을 지원하기 위한 기반 기술로 활용분야도 점차 다양화 되고 있다.

[표 1 - 2] 회사 별 지원가능한 Map Viewer

|  |  |
| --- | --- |
| 회 사 | 프로그램 |
| 구글 | Google Map / Google Earth |
| 네이버 | Naver Map |
| 다음 | Daum Map |
| 애플 | Flyover |
| 국토교통부 | VWorld |

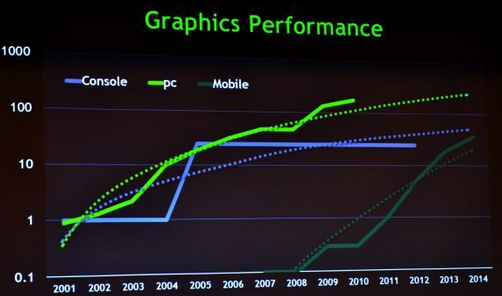
이러한 GIS는 사용자가 어려움 없이 편리하게 위치정보를 활용할 수 있도록 하는 웹2.0 기반의 GIS 패러다임이다. 또한 이것은 일반 대중 및 개발자들이 개방형의 지도 데이터 및 기능들을 이용해 다양한 위치기반 서비스들을 만들 수 있도록 하는 개방형 공간정보 플랫폼이다. 현재는 Google, Microsoft, Yahoo 등 대표적인 포털 업체들이 기술을 주도해 가고 있으며 다양한 서비스를 제공되고 있다. 이는 사용자들이 자신의 요구에 부합되는 지도서비스를 개발하고 공유할 수 있도록 사용자 참여 지도 제작 서비스를 보편화 시키는데 큰 역할을 하고 있다.

[표 1 - 3] Map Viewer의 3D 지원 여부

|  |  |  |  |
| --- | --- | --- | --- |
| 프로그램 | PC | Mobile | 기 타 |
| Google Earth | O | O | 국내 지원 불가 |
| Naver Map | X | X |  |
| Daum Map | O | X |  |
| Flyover | O | O | 주요 도시만 가능 |
| VWorld | O | O | 국내 지역에 한정 |

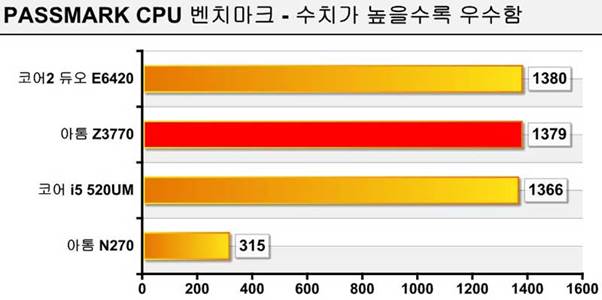
하지만 이러한 끊임없는 노력에도 불구하고, 국내의 3D 모바일 지도 서비스 현황은 매우 열악하다. 특히 국내에서 Google Earth를 모바일로 사용할 수 없게 되면서, 모바일에서 사용가능 한 3차원 지도 서비스가 필요하게 되었고 이 서비스는 서비스국토교통부에서 제작 ‘VWorld’ 하나에 불과하다. ‘VWorld’는 오픈플랫폼이 제공하는 클라이언트 프로그램을 활용하여 새로운 서비스를 구축하기 용이하다는 장점이 있지만, 오픈플랫폼 클라이언트의 기능적 한계를 벗어나는 서비스를 구축할 수 없다는 단점이 존재한다.

### Mobile 기기의 발전



(그림 1 – 2) 기기별 그래픽 성능 그래프 (2012, NVIDIA)

현재 모바일 기기의 발전은 기존 기기(콘솔, PC)에 비해 두드러지게 나타나며, 기존 기기의 성능을 따라잡고 있는 실정이다. NVIDIA의 경우 2012년 기자간담회에서 ‘모바일 기기의 성능이 현재의 콘솔 그래픽을 넘어설 것’이라고 밝힌 바 있다.

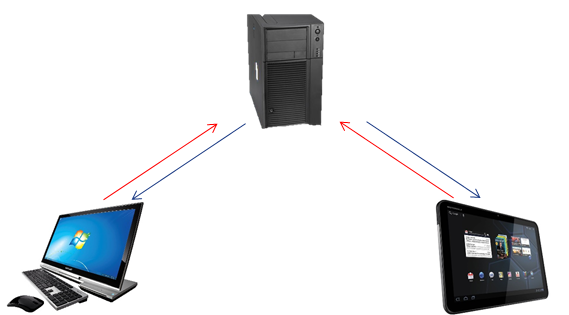


(그림 1 – 3) 모바일 CPU와 구형 PC CPU의 성능 비교(2013, Intel)

또한, 2013년 후반기에 출시된 베이트레일 기반 아톰인 ‘Z3770’의 경우, 구형 데스크탑용 듀얼코어 ‘E6420’과 유사하며, 2010년에 출시된 노트북용 저전력 프로세서인 ‘I5 520UM’을 상회하는 성능을 발휘하는 것을 볼 수 있다. 이처럼, 모바일 기기의 하드웨어 발전은 보다 다양한 환경에 쓰일 수 있을 것으로 예상된다.

# 개발 내용 및 결과물

## 목표



(그림 2 – 1) 3D Map Viewer의 목적

따라서, 본 프로젝트에서는 기존 3차원 모바일 지형 서비스의 한계에서 벗어남과 동시에 세계의 고도 데이터와 위성 영상을 이용하여, PC, Android 기기에서 실행 가능한 3D Map Viewer를 개발하는 것을 목표로 한다. 또한, 다른 지도 서비스와 차별화 하여 직접 고도 데이터를 플랫폼에서 계산하여 표현함으로써, 서버의 부담을 최소화 할 수 있으며, 파일의 경로만을 수정하여 손쉽게 지형 정보를 변경할 수 있도록 한다.

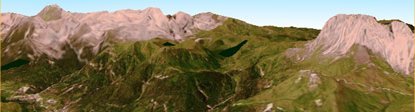
## 연구/개발 내용 및 결과물

### 연구/개발 내용

[](http://www.opensg.org/projects/opensg/attachment/wiki/Tutorial/OpenSG1/NodeCores/lod.png)

(그림 2 - 2) LOD(Level Of Detail) 방식

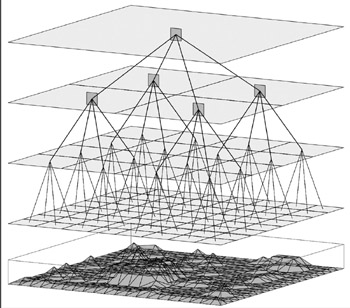
LOD 방식은 Level Of Detail의 약자로서, 메시 모델링 데이터의 정밀도를 단계별로 조정하는 기술이다. 카메라와 오브젝트 간의 거리나 화면에 투영된 면적에 따라 모델 디테일을 다르게 렌더링하는 것이며, 멀리 떨어져있는 오브젝트 같은 경우에 하이퀄리티 맵핑을 계속 쓰게 되면 무겁고, 비효율적일 수 있다. 따라서, 카메라가 가까이 보여줄 땐 하이퀄리티 맵핑을 보여주게 되지만 카메라가 멀 경우 무게를 줄이기 위해 적은 수의 폴리곤을 갖는 모델링 데이터를 사용하는 로우맵을 보여주는 기술이다.

[](http://api.viglink.com/api/click?format=go&jsonp=vglnk_jsonp_13958366481868&key=9d57a91dbe2c04751e4fa138fb7bd152&loc=http://www.rez3d.com/&v=1&libId=e69bc15b-7697-489e-967c-3d5287be6440&out=http://www.livedolomiti.com/3d/Launcher.htm&ref=http://www.webring.org/l/rd?ring=3d;id=1;url=http://www.rez3d.com/&title=RezIndex&txt=%20)

(그림 2 - 3) Flyover 지도에 적용된 거리기반 LOD

3D 지형 서비스에서 LOD방식을 많이 사용하고, 주로 거리기반의 LOD 방식을 이용한다. 또한, 고도데이터의 값을 서버에서 계속해서 갱신하는 방식을 사용하기 때문에, 동적 LOD 방식이 효율적이다.

### Quad Tree



(그림 2 - 4) Quad Tree를 이용한 3D 지형의 표현

Quard Tree란 자료구조의 트리를 기반으로 자식 노드가 4개인 트리를 말한다. 3D데이터를 표현하기 위한 자료구조를 Scene Graph라고 부르는데 이것 역시 그에 포함된다. 이 트리는 2차원 공간을 4개의 자식 노드로 재귀적으로 분할하는 방법이다. Quard Tree를 사용하면 초기에 하나의 넓은 월드의 지형을 1/4로 나누고 그렇게 나온 4개의 노드를 부모로서 다시 1/4로 나뉘기 때문에 매우 거대한 지형을 빠르게 검색할 수 있으며, 필요 없는 데이터를 제거함으로써, 3D 엔진이 처리해야 할 데이터의 양을 빠르게 줄일 수 있다.

### 평면 방정식

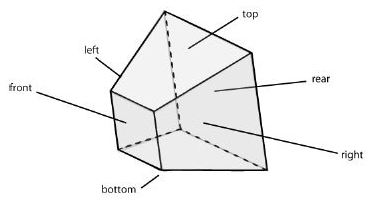
평면 방정식이란 무한 평면을 정의하는 수학적 도구로서, 간단한 수식으로 평면을 표현한 것이다. 평면 방정식의 정의는 다음과 같다.

ax+by+cz+d = 0

여기서, a, b, c는 이 평면의 방향을 나타내는 법선 벡터이며, d는 평면과 원점간의 거리를 나타내는 값이다. 또한, (x, y, z) 좌표값은 모두 이 평면 위에 있는 점들이다. 우리는 평면 방정식의 연산 결과를 통해 어떠한 한 점이 평면의 앞에 있는지, 뒤에 있는지를 판단할 수 있으며, 이러한 성질을 이용하여 절두체 내부에 점이 포함되는지 판단할 수 있으며, 얼마나 떨어져 있는지도 알 수 있다.

### Frustum Culling

Frustum이란 투영 행렬을 계산할 때, 화면에 보이는 범위를 정의하는데, 이 모양이 머리를 자른 것 같다고 해서 절두체(Frustum)라고 부른다.



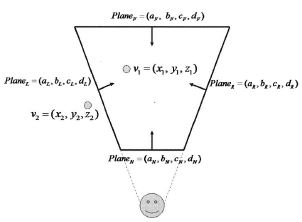
(그림 2 - 5) 절두체의 정의

Frustum culling이란 3차원 영역에서 실제로 카메라의 시야 범위에 포함되는 것들만 렌더링하고, 나머지 것들은 렌더링하지 않는 기법을 말한다. 이러한 Frustum culling은 3D 엔진을 개발하는 데 있어서 중요한 속도 증가 기법 중의 하나이다.

[표 2 - 1] 절두체를 이루는 6개의 평면

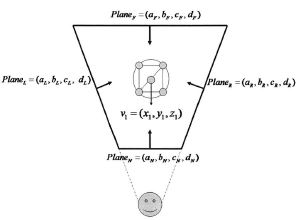
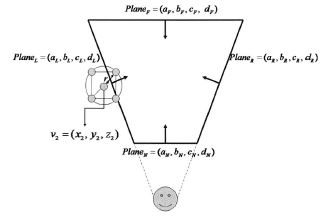
|  |  |
| --- | --- |
| 이 름 | 위 치 |
| 근평면 (Near plane) | 카메라와 수직하며 제일 가까운 곳의 시야 범위를 나타내는 평면 |
| 원평면 (Far plane) | 카메라와 수직하며 제일 먼 곳의 시야 범위를 나타내는 평면 |
| 좌평면 (Left plane) | 카메라의 좌측 시야 범위를 나타내는 평면 |
| 우평면 (Right plane) | 카메라의 우측 시야 범위를 나타내는 평면 |
| 상평면 (Top plane) | 카메라의 상단 시야 범위를 나타내는 평면 |
| 하평면 (Bottom plane) | 카메라의 하단 시야 범위를 나타내는 평면 |

이러한 절두체는 총 6개의 평면으로 이루어져 있으며, 이렇게 6개의 평면에 포함되지 않는 영역은 오히려 렌더링 속도를 저해하는 요소로 작용한다. 따라서 6개의 평면 내부와 외부를 판단하기 위해서 평면 방정식을 도입하여 계산하며, 6개의 평면 방정식에 점의 좌표를 대입해서 모든 평면 방정식의 결과값이 양수(+)이면, 이 점은 절두체의 내부에 포함되어 있다고 판단한다.



(그림 3 - 3) 절두체와 점의 포함관계

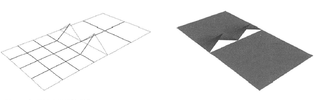
우리가 사용할 Culling 기법은 경계구 컬링(Bounding sphere culling) 기법으로, 메시 전체를 감싸는 경계구를 지정하고 이 구가 절두체에 포함되는지를 판단하는 것이다. 따라서, 평면 방정식 ax+by+cz+d=0에 경계구의 중심점(x, y, z)의 값을 대입해서 나온 결과값이 반지름 r보다 작다면 평면과 중심점의 거리가 반지름보다 작은 것이므로 경계구에 포함되는 것으로 판단한다. 이러한 경계구 기법을 사용하는 이유는, 메시의 일부분이 절두체에 포함되는 경우가 상당히 많기 때문에, 이를 정상적으로 판정하기 위해서 사용한다.



(그림 2 - 6) 경계구 컬링 기법의 포함 유무 확인

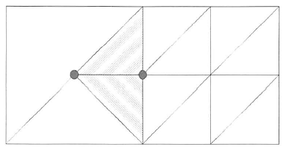
### Crack

Quad Tree 에서의 균열(Crack)은 LOD를 적용할 때, LOD 단계가 서로 다른 메시들의 경계 부분에서 디테일이 낮아지게 되어 인덱스가 참조하는 정점이 줄어들게 되면서, 지형에 구멍이 뚫리는 현상을 말한다.



(그림 2 - 7) 디테일 차이에 의한 크랙현상

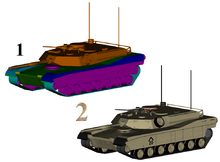
따라서, 이러한 크랙문제를 해결하기 위해서 LOD 단계가 서로 다른 메시의 경계부분에 삼각형을 추가하여, 균열이 발생하는 것을 막을 수 있으며, 상하좌우 네 가지 방향으로 만들어 모든 메시에서 발생할 수 있는 크랙을 해결할 수 있다.



(그림 2 - 8) 삼각형을 이용한 크랙현상 해결

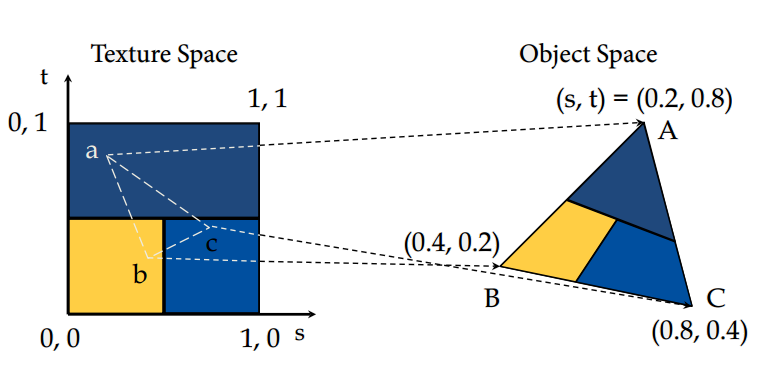
### Texture Mapping

텍스처 매핑(Texture mapping)은 컴퓨터 그래픽스 분야에서 가상의 삼차원 물체의 표면에 세부적인 질감의 묘사를 하거나 색을 칠하는 기법이다. 일반적으로 수식이나 2차원의 그림을 3차원 물체의 표면에 여러 가지 방법을 통하여 적용하고, 컴퓨터 그래픽 화면을 만들어 나갈 때 마치 실제의 물체처럼 표현한다.



(그림 2 - 9) 텍스처 매핑(Texture mapping)

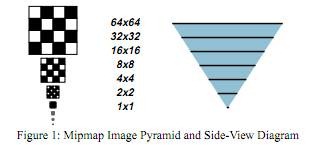
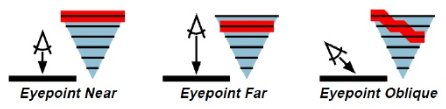
텍스처 공간(Texture space)의 경우, 그림의 크기의 상관없이 (0,0) ~ (1,1)의 범위로 표현되기 때문에, 오브젝트 공간에서 이를 상용할 경우 텍스처 공간에 맞게 별도로 계산하는 과정이 필요하게 된다.



(그림 2 – 10) 텍스처 공간과 오브젝트 공간

하지만, 이러한 매핑 방식의 경우 확대(Magnification)와 축소(Minification)의 방식에 따라 Aliasing 현상이 발생할 수 있다. 이러한 경우 Filtering이나 Mipmap을 이용하여 Aliasing 현상을 최소화 하는 것이 가능하다.

### MipMap

(그림 2 - 11) Mipmap Image Pyramid

GIS에서 사용하는 데이터가 매우 크기 때문에, 데이터를 하드웨어가 처리할 수 있는 작은 타일들로 분할하여 관리한다. 이때, Mipmap 이라 부르는 개선된 자료 구조를 사용한다. Mipmap은 다양한 크기의 텍스처를 준비하고 Mapping되고자 하는 폴리곤에 가장 근사한 크기를 갖는 텍스처를 선택하는 방식으로 데이터 구조들의 단일화, 부드러운 영상 변화 및 안정적인 렌더링 속도를 보장한다는 장점이 있다.

## 시스템 기능 및 구조 설계도

### 개발 환경

* 운영체제

Windows 7 64bit

* GDAL

사용 언어 : C++

개발 환경 : MS Visual Studio 2010/2012

* FreeGlut

사용 언어 : C++

개발 환경 : MS Visual Studio 2010/2012

* OpenGL

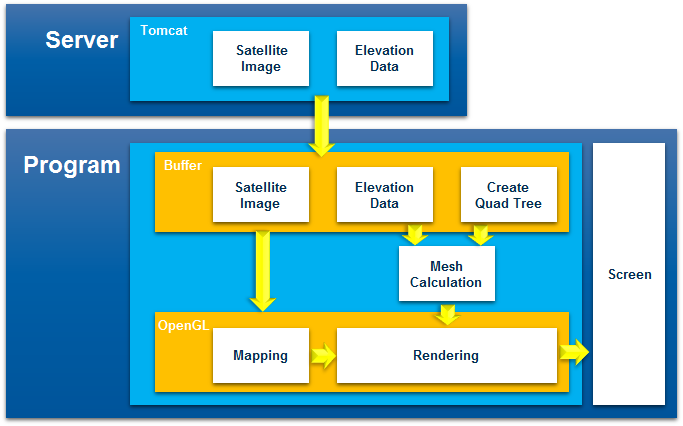
사용 언어 : JAVA

개발 환경 : Eclipse Android

* Tomcat

개발 환경 : Eclipse

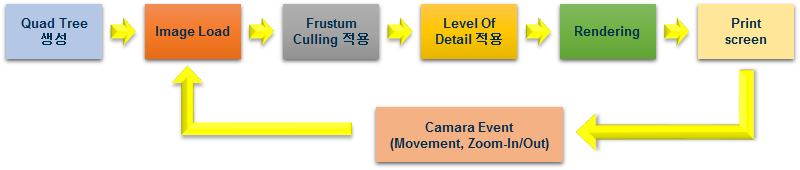
### System Architecture



(그림 2 – 12) System Architecture

## 시스템 구현

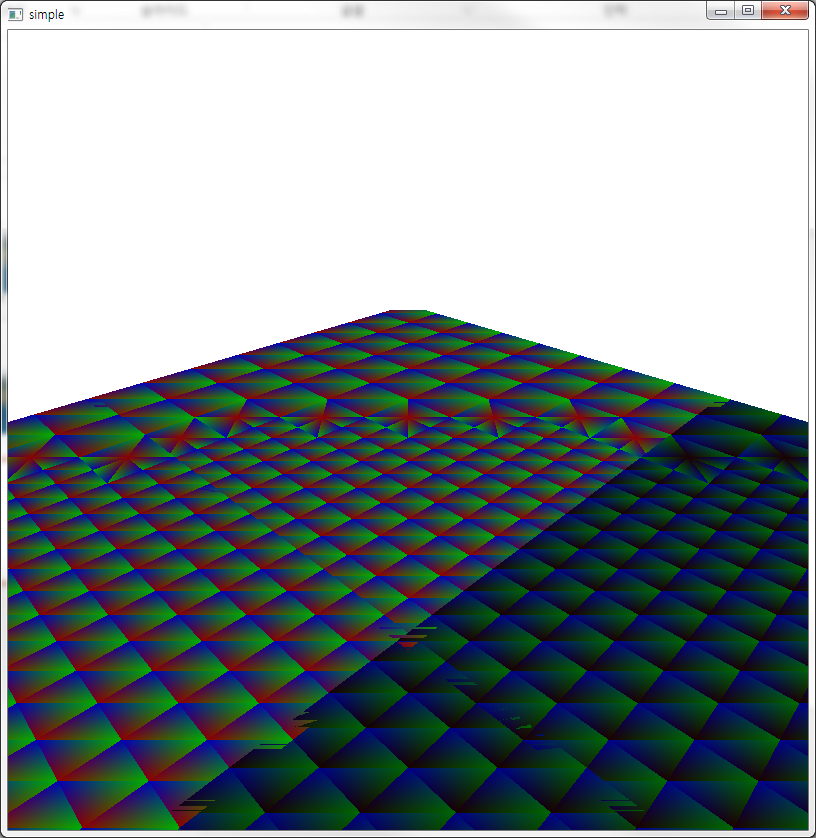
### 프로그램 실행 과정



(그림 2 - 13) 프로그램 실행 과정

프로그램 실행 시, 실행과정으로는 크게 “Quad Tree 생성”, “Image Load”, “Frustum Culling 적용”, “Level Of Detail 적용”, “Rendering”으로 나뉜다. “Quad Tree 생성” 단계에서는 9방향에 대한 각각의 Quad Tree를 생성하며, “Image Load” 단계에서 최초에 실행될 지형에 대한 고도데이터와 위성사진을 서버에서 다운로드 받아 메모리에 적재한다. 또한, “Frustum Culling 적용” 단계에서는 현재 현재의 카메라의 위치에서 보이는 시야에서 불필요한 부분을 제거한다. “Level Of Detail 적용” 단계에서는 Frustum Culling 후에 남은 메시 중에서, 카메라의 위치에서 거리가 먼 메시의 Detail을 줄이며, 마지막으로 “Rendering” 단계에서, 메시에 고도 값을 추가하여 3차원 지형으로 가시화한다.

### Quad Tree 생성



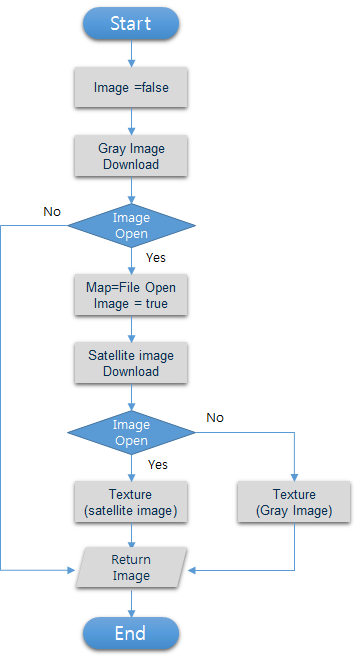
(그림 2 - 14) Quad Tree 생성

먼저 “Quad Tree 생성” 단계에서는 ‘3D Map Viewer’에서 사용될 9방향의 Quad Tree를 생성한다. 각각의 Quad Tree의 경우 좌측 하단을 (0, 0)으로 하는 512 \* 512 크기를 가지며, 각 방향값이 지니는 좌표의 값과 Quad Tree 내부의 좌표의 값을 더함으로써, 해당 Pixel의 위치를 표현한다.

[표 2 – 2] 각 지역별 방향과 방향값 및 좌표값

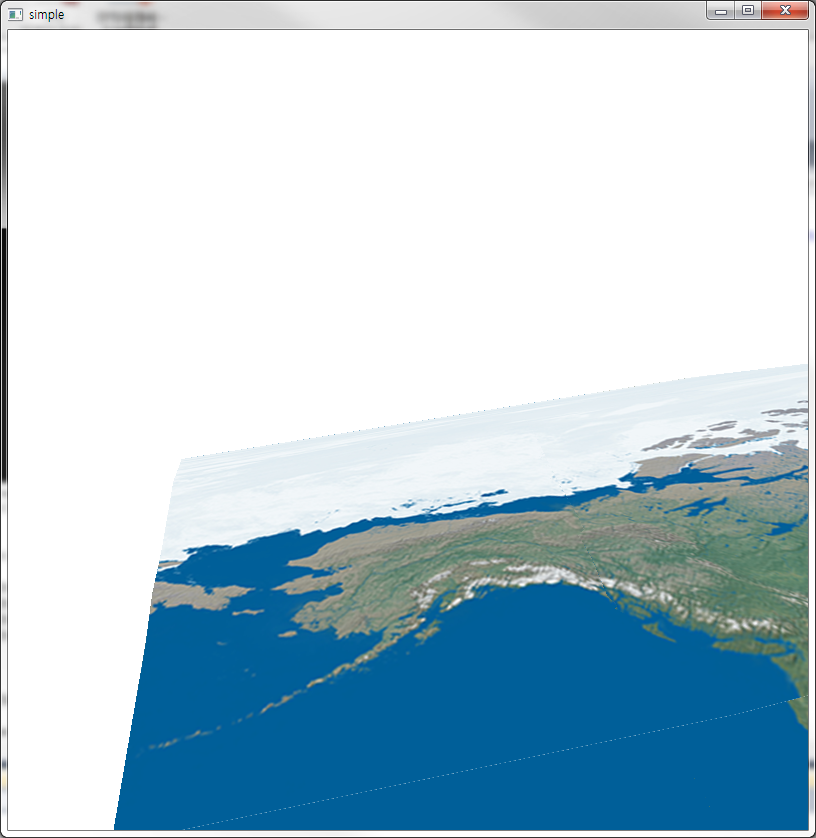
|  |  |  |  |
| --- | --- | --- | --- |
| 종 류 | 방향(값) | 좌표값 (x, y) | 비 고 |
| CUR | 현재 지역(0) | (-256, -256) |  |
| WEST | 서쪽 지역(1) | (-768, -256) |  |
| EAST | 동쪽 지역(2) | (256, -256) |  |
| SOUTH | 남쪽 지역(3) | (-256, -768) |  |
| SOUTH\_WEST | 남서쪽 지역(4) | (-768, -768) | SOUTH + WEST |
| SOUTH\_EAST | 남동쪽 지역(5) | (256, -768) | SOUTH + EAST |
| NORTH | 북쪽 지역 (6) | (-256, 256) |  |
| NORTH\_WEST | 북서쪽 지역 (7) | (-768, 256) | NORTH + WEST |
| NORTH\_EAST | 북동쪽 지역 (8) | (256, 256) | NORTH + EAST |

### Image Load



(그림 2 - 15) Image Load 과정

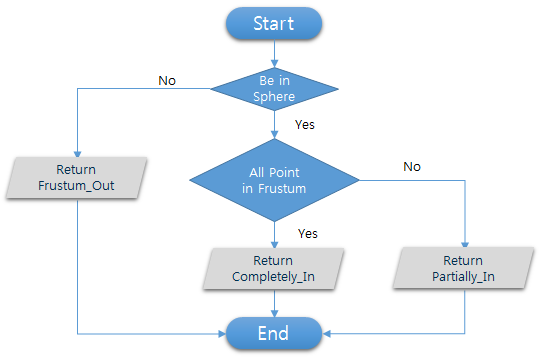
“Image Load” 단계에서는 9방향의 고도데이터와 위성사진을 서버에서 다운로드하여 메모리에 적재하는 단계이다. 함수가 호출되면, 해당 위치에 해당하는 고도데이터(Gray Image)를 서버에서 다운로드하며, 파일을 열 수 없을 경우 파일이 존재하지 않는다고 판단한다. 만약 고도데이터가 존재할 경우, 파일이 존재한다고 표시하며(Image = true), 이후 위성사진을 다운로드한다. 위성사진이 존재할 경우 위성사진을 이용하여 Texture를 바인딩하며, 존재하지 않으면 위성사진 대신 고도데이터를 이용하여 바인딩 작업을 수행한다.



(그림 2 - 16) 존재하지 않는 영역에 대한 표현

본 프로젝트에서는 고도데이터와 위성사진이 존재한다는 가정하에 진행되며, ‘고도데이터가 존재하지 않을 경우 그 지역은 존재하지 않는다’고 판단한다. 따라서, 고도데이터가 없는 지역의 경우 (그림 2 - 16)과 같이 표현되지 않는다.

### Frustum Culling 적용



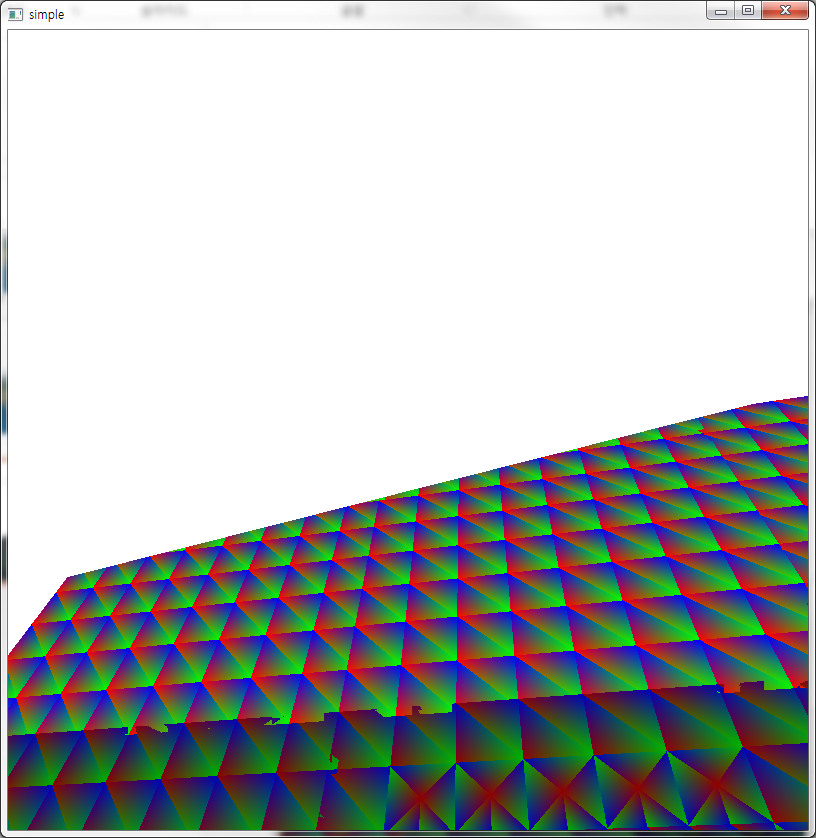
(그림 2 - 17) Frustum Culling 판단

“Frustum Culling 적용” 단계에서는 현재의 카메라의 위치에서 표현될 수 있는 노드를 계산한다. 먼저 ‘Be in Sphere’를 통해, ‘노드의 값이 경계구 안에 포함되는가?’를 판단하며, 경계구 안에 포함되지 않을 경우 ‘Frustum\_Out’을 리턴한다. 경계구에 포함되는 노드의 경우, 노드의 각 정점(TopLeft, TopRight, BottomLeft, BottomRight)이 경계구에 포함되는가에 대해서 판별하며, 모든 정점이 포함될 경우 ‘Completetly\_In’을, 그렇지 않으면 ‘Partially\_In’을 리턴한다.

[표 2 - 4] Frustum Culling 결과

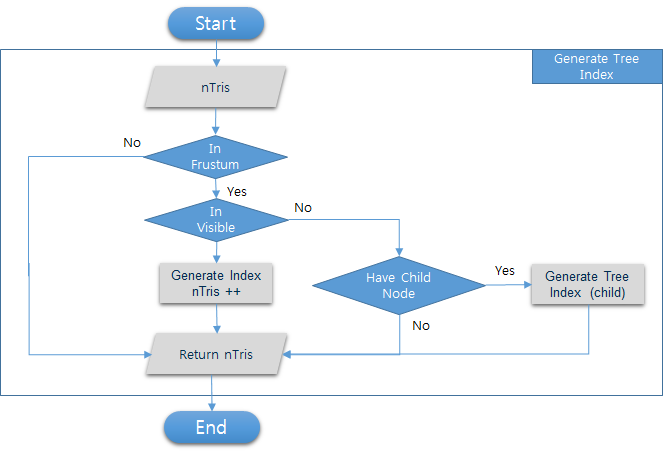
|  |  |  |
| --- | --- | --- |
| 분 류 | 설 명 | 자식 노드 계산 |
| Frustum\_Out | 경계구에서 벗어남 | X |
| Completetly\_In | 경계구에 완전히 포함 | O |
| Partially\_In | 경계구에 부분적으로 포함 | O |

Frustum Culling 함수의 판단 결과에 따라, 노드의 자식 노드가 경계구 안에 포함되는가를 판단할 수 있다. ‘Frustum\_out’의 경우 4개의 정점 모두 경계구 안에 포함될 가능성이 없지만, ‘Completetly\_In’과 ‘Partially\_In’의 경우 자식 노드는 경계구 안에 포함될 수 있기 떄문에, 자식 노드에 대해서도 Frustum Culling을 계산한다.



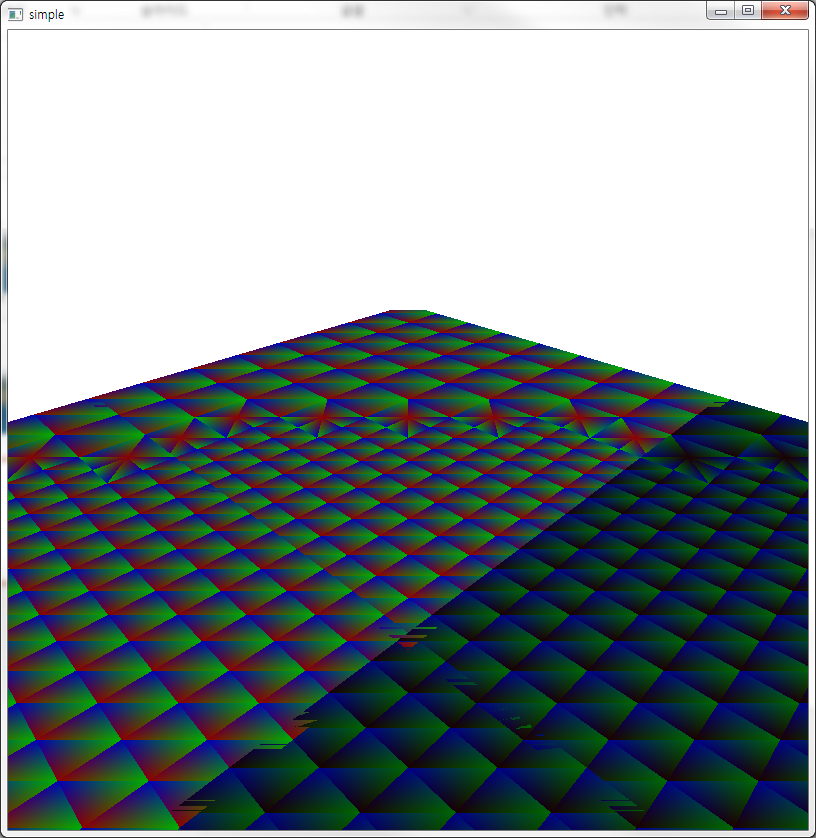
(그림 2 - 18) Frustum Culling의 적용

### Level Of Detail 적용



(그림 2 - 19) Level Of Detail 단계

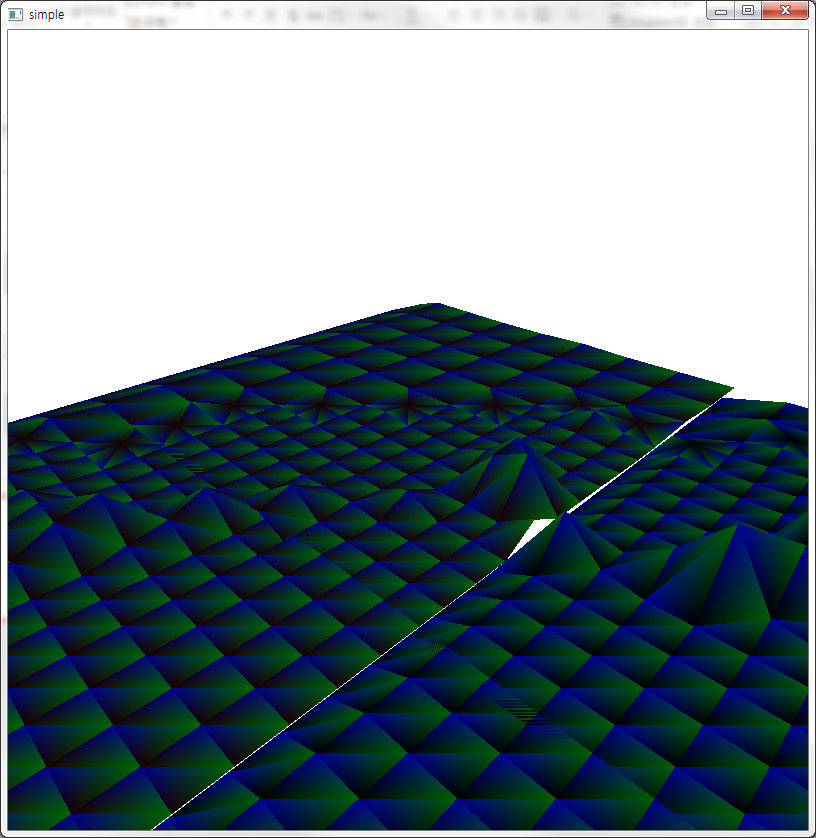
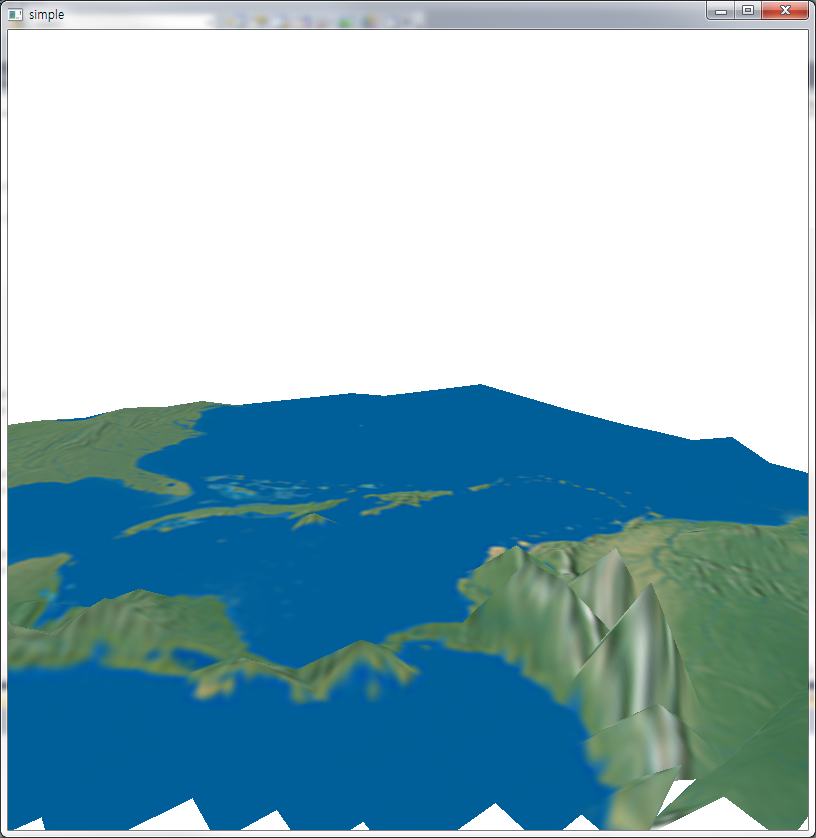
“Level Of Detail 적용” 단계에서는 9개의 Quad Tree 중, 고도데이터가 존재하는 방향에 대해서 실제 출력될 메시의 정보를 계산한다. ‘In Frustum’에서는 해당 노드의 값이 경계구 안에 포함되는지를 판단하며, 경계구에 포함될 경우 ‘In Visible’에서 노드의 Width 값이 LOD Leve보다 작은지 비교한다. 만약 ‘In Visible’의 값이 참이라면, 현재 카메라의 위치에서 얻을 수 있는 가장 작은 Detail을 의미하며, 해당 노드의 메시정보를 추가한다. 반대로, ‘In Visible’의 거짓이라면 Detail을 높여야 하므로, 자식 노드에 대해서 검색하여 Detail을 높인다.



(그림 2 - 20) Level Of Detail 적용

LOD의 경우, 카메라와 오브젝트 간의 거리를 통해 계산되기 때문에, 서로 다른 Quad Tree에 대해서도 매끄럽게 LOD가 적용되는 것을 확인할 수 있다.

### Rendering

(그림 2 - 21) 고도 데이터의 3차원 표현과 Mapping 결과

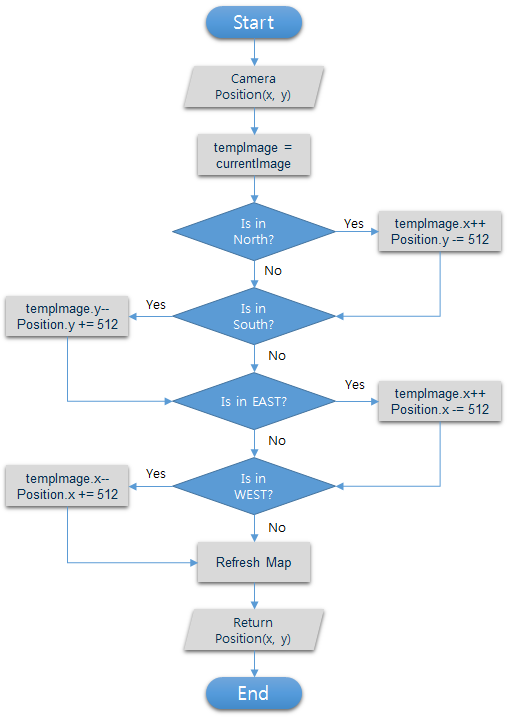
“Rendering” 단계에서는 메시의 값을 이용해서 3차원으로 가시화하는 단계이다. 메시의 Pixel 값을 이용하여 고도데이터의 고도값을 얻어낼 수 있으며, Mapping을 통해 이미지를 표현한다.

### Camera Event

[표 2 - 3] 3D Map Viewer의 Camera Event

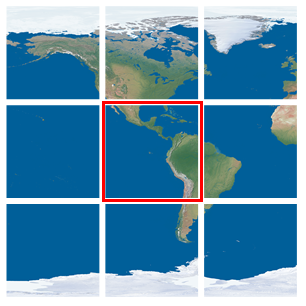
|  |  |  |
| --- | --- | --- |
| 분 류 | 기 능 | 비 고 |
| 카메라 이동 | 카메라 위치에 따른 지형 전환 | 카메라의 평면 좌표(x, y)값 |
| Zoom-In/Out | 카메라의 고도에 따른 지형 레벨 전환 | 카메라의 고도(z)값 |

3D Map Viewer의 Camera Event는 “카메라 이동”과 “Zoom-In/Out”이 존재한다. “카메라 이동”은 카메라의 평면 좌표(x, y)의 값에 의해 발생하며, “Zoom-In/Out”은 카메라의 고도값의 변화(z의 변화)에 의해 발생한다.



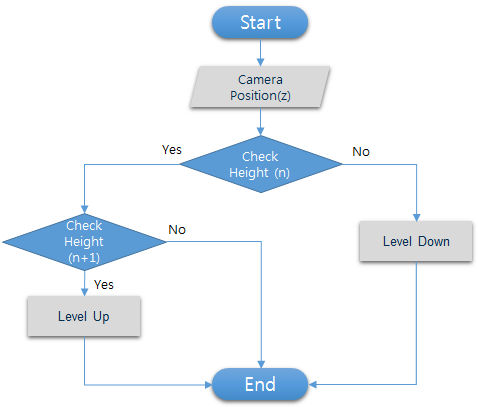
(그림 2 - 22) 카메라 이동 이벤트

“카메라 이동” 이벤트는 카메라의 x, y 값에 의해서 발생하는 이벤트로 카메라가 Quad Tree의 어느 부분에 존재하는지에 대해서 판별한다. 카메라가 위치한 x, y값에 따라 일정한 수를 더함으로써, 카메라의 최종 방향을 판단할 수 있으며, 카메라가 중앙의 Quad Tree에서 벗어나지 못하도록 좌표값을 새로 계산한다.



(그림 2 - 23) 카메라가 존재할 수 있는 영역

이는 카메라가 이동할 수 있는 범위를 가운데 지형으로 고정시킴으로써 마치 카메라가 움직이면서 새로운 지형을 불러오는 것과 같은 효과를 내기 위한 것이다. 또한, 다음레벨과 이전 레벨의 지형에 대한 계산을 좀 더 쉽게 할 수 있다.

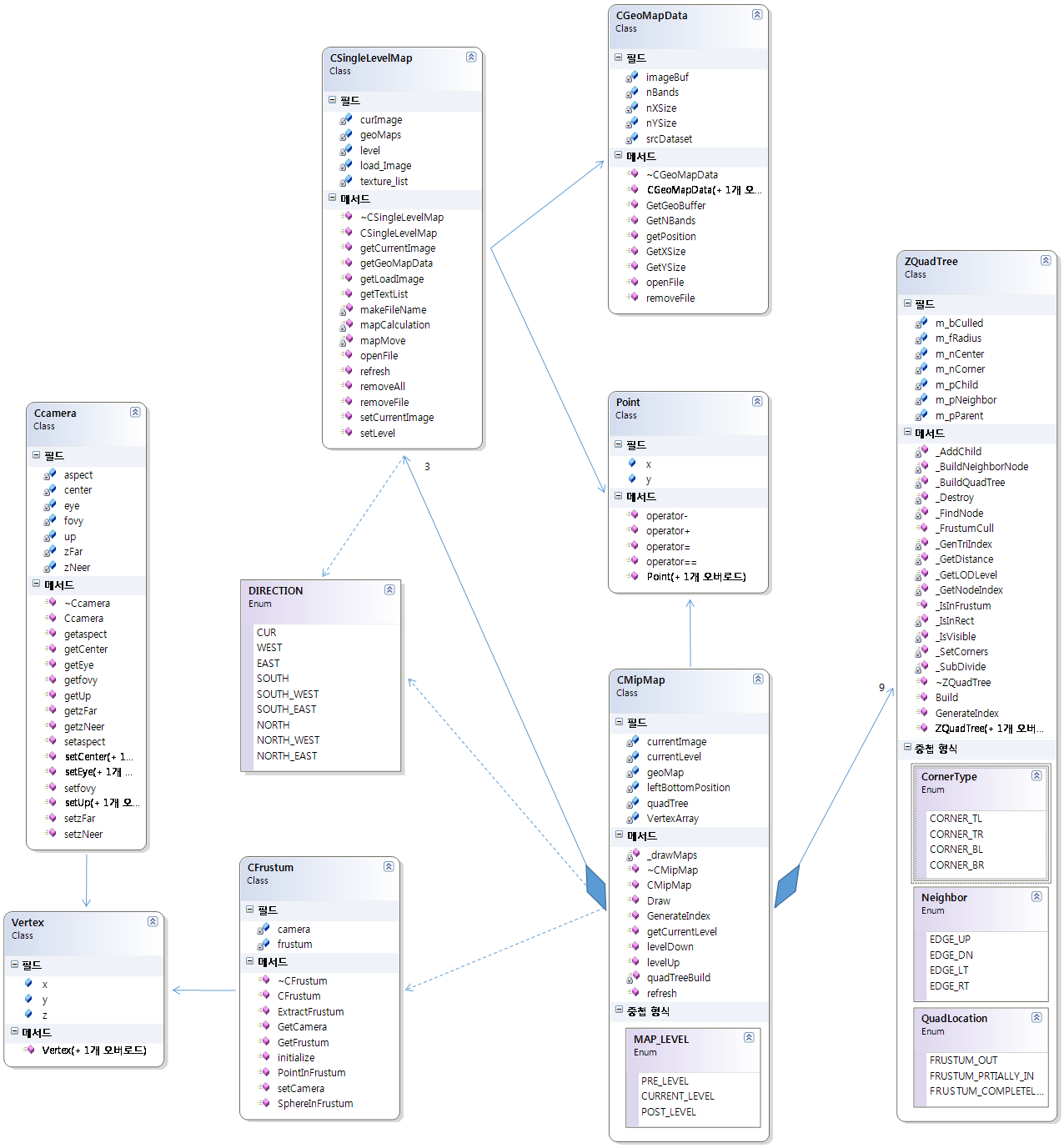


(그림 2 - 24) 카메라 Zoom-In/Out 이벤트

“Zoom-In/Out” 이벤트는 카메라의 z축에 의해 발생하는 이벤트이며, 카메라의 고도값이 지형으로부터 얼마나 떨어져 있는가에 따라 정해진다. 카메라의 고도값이 현재 레벨에서의 기준 고도값보다 높다면, ‘Level Down’함수를 호출해서 지형의 레벨을 낮추며, 반대로 다음 레벨 고도값에 비해 낮으면, Detail이 상승했다는 의미이기 때문에 ‘Level Up’ 함수를 호출한다.

‘Level Up’, ‘Level Down’ 함수의 경우 현재의 지형정보를 Swap하여, 각각 다음레벨, 이전레벨로의 지형을 전환한다는 의미이다. 이러한 Swap함수를 통해 지형정보를 전환한 후, 불필요해진 지형에 대해서, 새로운 지형을 다운로드하여, 지형정보를 최신화한다.

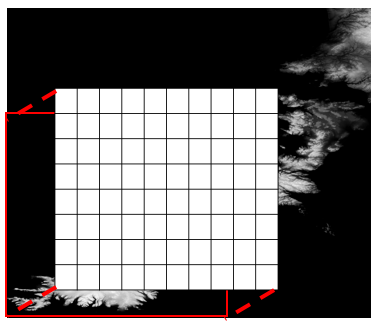
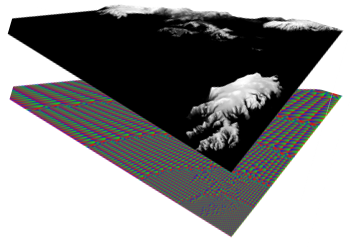
### Class Diagram



(그림 2 - 25) Class Diagram

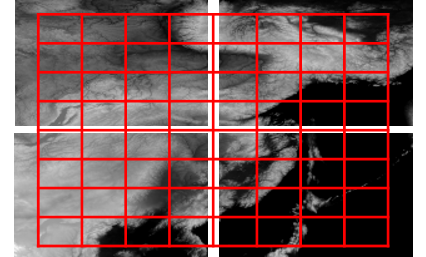
## 활용/개발된 기술

### 단일 Level에 대한 MipMap

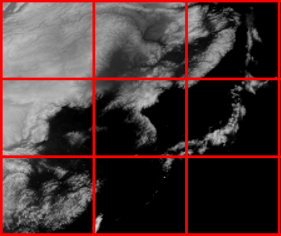
(그림 2 - 26) 단일 Quad Tree를 이용한 3D Map View

3D Map Viewer를 렌더링 할 때, 하나의 Quad Tree를 이용하여 지형정보를 3차원 가시화 하는 방안으로 프로젝트를 수정하였다. 이는, Quad Tree의 경우 이미지의 가로, 세로의 픽셀의 개수를 이용하여 Quad Tree를 구성하는데, 지형정보의 특성상, 지도의 크기는 균일하다고 가정하면, 모든 Quad Tree의 초기 Parameter(가로, 세로의 픽셀의 수)는 동일하기 때문이다.



(그림 2 - 27) 1개의 Quad Tree와 4개의 고도데이터

하지만, 기존의 하나의 고도데이터를 이용하여 3차원 가시화를 수행한 경우 정상적으로 작동할 수 있지만, 여러 개의 고도데이터를 사용하기 위해서는 효율적인 메모리 관리가 필요하다. 특히, Quad Tree에 의해 사용되지 않는 고도데이터의 경우 메모리에서 제거함으로써, 메모리의 효율을 늘리는 것이 가능하다.



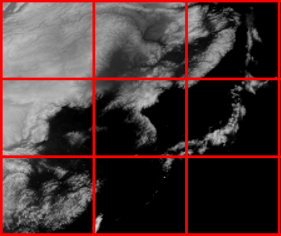
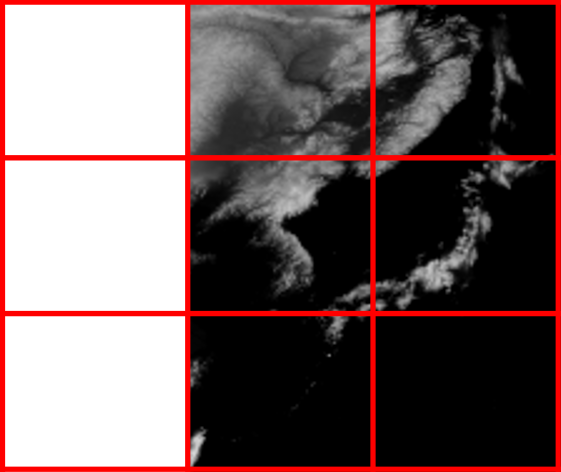
(그림 2 - 28) 단일 Level Clipmap

단일 Level에 대한 Mipmap의 경우, 최초에 설정된 지점의 고도데이터를 포함한 8방향의 고도데이터를 메모리에 저장한다. 각각의 고도데이터는 방향값을 가지고 있으며, 이 방향값을 통해 고도데이터가 현재 좌표에서 어느 방향에 위치하는 지 확인할 수 있다.

[표 2 – 5] 각 지역별 방향과 방향값

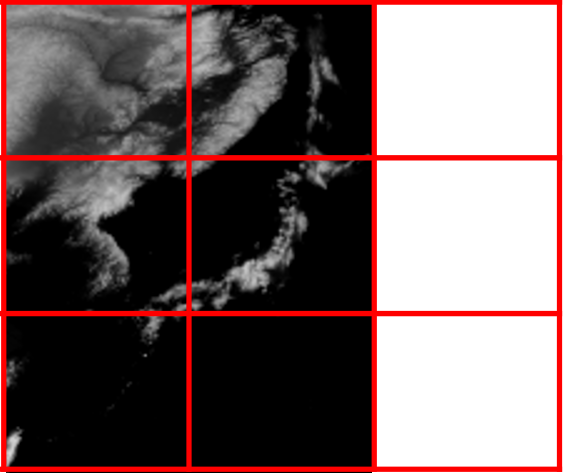
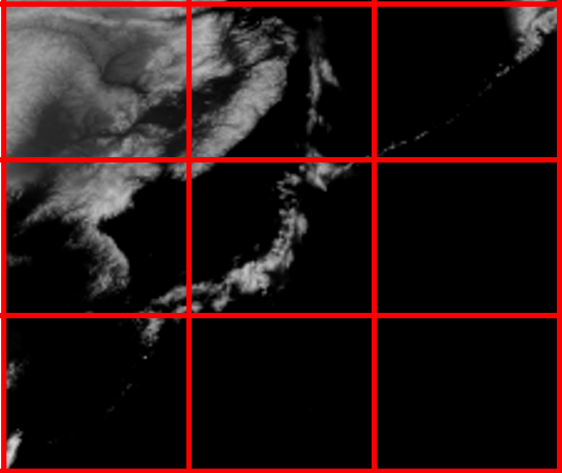
|  |  |  |
| --- | --- | --- |
| 종 류 | 방향(값) | 비 고 |
| CUR | 현재 지역(0) |  |
| WEST | 서쪽 지역(1) |  |
| EAST | 동쪽 지역(2) |  |
| SOUTH | 남쪽 지역(3) |  |
| SOUTH\_WEST | 남서쪽 지역(4) | SOUTH + WEST |
| SOUTH\_EAST | 남동쪽 지역(5) | SOUTH + EAST |
| NORTH | 북쪽 지역 (6) |  |
| NORTH\_WEST | 북서쪽 지역 (7) | NORTH + WEST |
| NORTH\_EAST | 북동쪽 지역 (8) | NORTH + EAST |

그 후, Quad Tree의 위치가 변경될 때마다 현재의 위치를 갱신하며, 변경된 위치가 현재 지역을 벗어난 경우 고도데이터를 갱신하게 된다. 만약, 현재의 지점에서 우측으로 계속 이동하여, 현재 지역의 고도데이터를 벗어난다고 가정할 경우에 기존에 저장한 서쪽 지역의 데이터는 불필요하다. 따라서, 불필요한 서쪽 지역의 데이터를 모두 삭제한다.

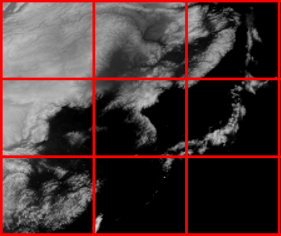
(그림 2 - 29) 좌표의 갱신 및 불필요한 고도데이터 삭제

마지막으로, 변경된 좌표의 동쪽 지역의 고도데이터를 메모리에 로드하기 위해서 기존의 고도 데이터를 모두 한 칸 왼쪽으로 옮긴다. 그 후, 새로운 지역의 고도데이터를 로드함으로써, Mipmap을 유지하게 된다.

(그림 2 – 30) 고도데이터 로드

이러한 방식으로 Mipmap을 구현할 경우 새로운 지역을 계산하고 디스크에서 불러오는 시간동안 해당 지역을 렌더링하는 것을 막기 위해, 카메라의 최대 시야 범위가 고도데이터 1개의 x값과 y값보다 작아야 한다. 따라서, 이 부분의 경우 카메라의 시야 거리를 조절하거나 Frustum Culling을 이용하여, 가시거리를 제한해야 한다.



(그림 2 - 31) 카메라의 최대 가시거리(점선)

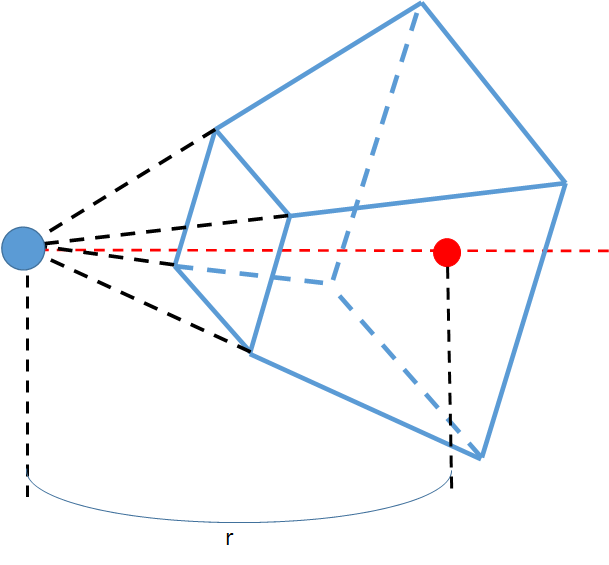
### Multi Level에 대한 MipMap

Multi Level에 대한 Mipmap의 경우, 단일 Level의 Mipmap 방식에서, 최초에 설정된 Map Level의 이전/다음 Level의 지형까지 미리 Load하는 방식이다. 현재 구현된 Mipmap의 경우 총 3개의 Level에 대한 지형 정보를 저장하고 있으며, 각각의 Level은 9개의 방향에 대한 고도데이터와 위성정보를 저장한다.

[표 2 – 6] 각 레벨과 레벨값

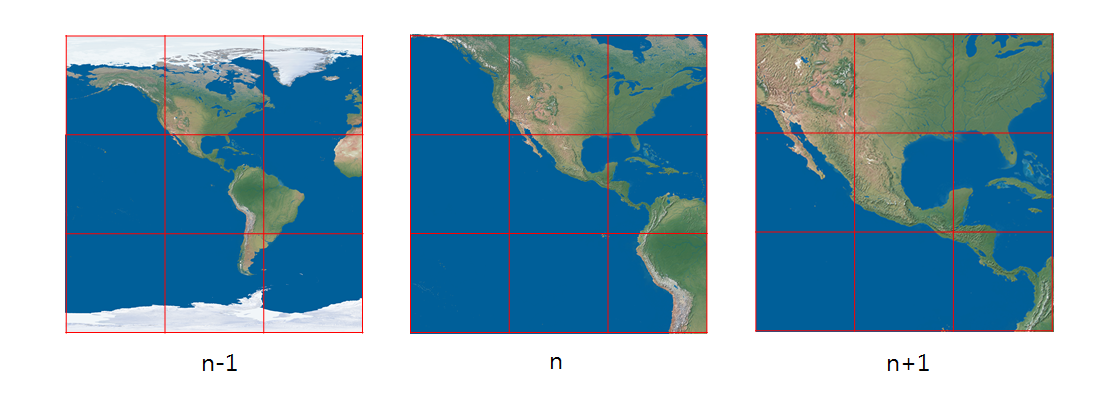
|  |  |  |
| --- | --- | --- |
| 종 류 | 레벨(값) | 비 고 |
| PREVIOUS\_LEVEL | 이전 레벨(0) | Resoultion Down |
| CURRENT\_LEVEL | 현재 레벨(1) | - |
| NEXT\_LEVEL | 다음 레벨(2) | Resoultion Up |

이러한 형태의 Mipmap의 경우 사용자가 지도를 Zoom-in/out을 할 경우, 지도가 사용자로부터 떨어진 거리에 따라 자동으로 Map Level을 전환한다. 카메라를 사람의 눈이라고 가정했을 때, 카메라로부터 물체까지의 떨어진 거리가 r이면 상의 크기는 1/r^2에 비례한다.



(그림 2 - 32) 카메라와 물체 사이의 거리에 따른 시야

따라서, 물체가 정사각형의 형태를 지니고 있을 경우, 카메라가 물체로부터 2배정도 떨어지게 되면, 물체의 크기는 1/4가 되며, 한변의 길이는 1/2가 된다. 따라서, Level 0일 때의 카메라와 지도의 거리를 기준으로, (1/2)^n의 거리가 될 때, 현재 지형 정보의 Level을 n으로 변화시키는 방식을 이용한다.

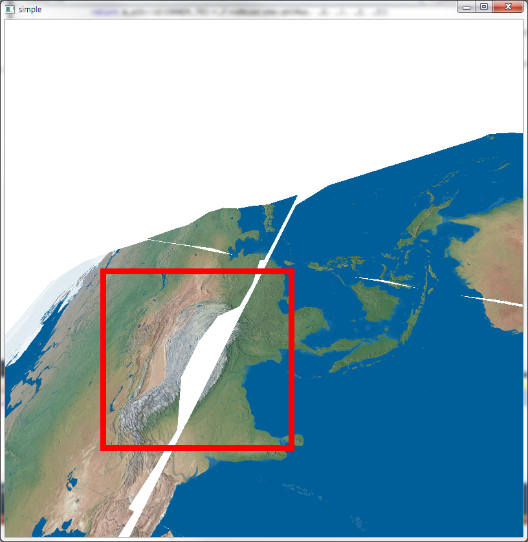


(그림 2 - 33) n레벨에서의 지형이미지 저장

n-Level 에서의 화면에서 카메라를 Zoom-in을 하여 n-level에서의 카메라와 지도 사이의 거리보다 가까워질 경우, 내부적으로 지형 정보를 swap하여, n+1의 지도로 교체된다. 또한, 사용되지 않는 n-1 level의 경우 기존의 지형정보를 삭제하고 n+2 level의 지도를 새로 다운로드 할 수 있도록 개선하였다.

## 현실적 제한 요소 및 그 해결 방안

### Quad Tree간 Crack 현상



(그림 2 - 34) Quad Tree 경계에서의 Crack 현상

현재 구현된 프로젝트의 경우, 각 Quad Tree 간의 Crack현상을 해결하는 부분이 빠져 있는 상황이다. 따라서, Quad Tree 간의 경계 부분에서 Crack 현상이 발생할 수 있으며, 이는 후에 주변의 Tree도 Neighbor Node를 검색함으로써, 해결할 수 있다.

### 경/위도 정보의 부재

현재 구현된 프로젝트는 PNG 파일을 사용해서 3차원 지형정보를 제공하기 때문에, 해당 좌표의 경, 위도 좌표에 대한 정보가 존재하지 않는다. 이는 서버에서 PNG파일이 아닌, TIFF 파일을 이용하여 해결할 수 있으며, TIFF 형태의 파일 내부에 경, 위도의 값을 삽입하여(GeoTIFF), 클라이언트가 TIFF 파일 내부에 있는 경, 위도 값을 통해 사용자의 카메라 위치에 대한 좌표를 알 수 있다.

## 기대효과 및 활용방안

# 자기평가

## 기대효과

## 사용가능성

# 참고 문헌

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 번호 | 종류 | 제목 | 출처 | 발행년도 | 저자 | 기타 |
| 14-1 | 서적 | 3D Engine Design for  Virtual Globes | Taylor & Francis Group | 2011 | Patrick Cozzi & Kevin Ring |  |
| 14-2 | 서적 | 3D 게임 프로그래밍 |  | 2005 | 김용준 |  |
| 14-3 | 논문 | WebGL Earth | Masaryk University Faculty of Informatics | 2011 | Petr Sloup |  |

### 기타 참고 자료

|  |  |  |  |
| --- | --- | --- | --- |
| 번호 | 제목 | 출처 | 기타 |
| 14-1 | Terrain Rendering Using GPU Based Geometry Clipmaps | http://research.microsoft.com/en-us/um/people/hoppe/gpugcm.pdf | PDF |
| 14-2 | The OpenGL ES Shading Language. | http://www.khronos.org/registry/gles/specs/2.0/GLSL\_ES\_Specification\_1.0.17.pdf | PDF |
| 14-3 | Fast Terrain Rendering Using  Geometrical MipMapping | http://www.flipcode.com/articles/article\_geomipmaps.pdf | PDF |
| 14-4 | GDAL - Geospatial Data  Abstraction Library | http://www.gdal.org/ | Homepage |

# 부록

## 사용자 매뉴얼

**설치 가이드, 따라하기 등 포함**

## 운영자 매뉴얼

**설치 가이드, 설정 및 운영 가이드 등**

## 배포 가이드

## XXX 매뉴얼

## XXX에 대한 기술 문서