이미지 기반 복층 구조의 지형 자동 생성

최성환, 손영우, 성만규
계명대학교 컴퓨터공학부 게임모바일공학전공
e-mail:shjy1669@naver.com, duddn6238@naver.com, mksung@kmu.ac.kr

Automatic Generation of Duplex Type 3D Terrain by Using a Single Image

SeongHwan Choi, Youngwoo Son, Mankyu Sung
Create Duplex Type Terrain Based in image, Keimyung University

요 약

본 연구는 게임을 비롯한 여러 가지 콘텐츠에서 활용하기 위해서 단일 이미지를 이용한 복층 구조의 지형을 제작하는 방법에 대해서 제안한다. 기존의 하이트맵(Heightmap)을 이용하여 복층구조를 제작했을 때의 문제점을 제시하며 어떻게 단일이미지 하이트맵(Heightmap)을 이용하여 복층 구조의 지형을 제작할수 있는지에 대한 방향을 제시한다. 본 논문에서는 단일 이미지의 RGBA값을 이용한 복층 구조 지형 제작 방식에 대한 실험을 통해 제안한 알고리즘을 검증한다.

1. 서론

요즘 게임과 같은 실시간 콘텐츠에서는 수 많은 지형에 대한 표현이 필요로 하며, 대부분의 엔진이나 프로그램에서는 하이트맵(heightmap)을 이용하여 다양한 형태의 지형(Terrain)을 만들고 있다. 현제 방식으로 지형을 만들면 3D 좌표상 어떤 위치에 대한 하나의 높낮이로만 만들 수 있는 지형을 만들 때 유용하며 동굴 같이 3D 좌표상 동일한 위치상에서 여러 높이를 가지는 지형을 만들기 위해서는 여러 개의 하이트맵을 사용해야 됨으로 많은 이미지를 요구하게 된다. 본 논문은 동굴 같은 동일한 위치에서 여러 높이 값을 가지는 지형을 만들기 위해서 여러 가지의 높이 값에 대한 정보를 가지고 있는 하이트맵이 필요로하는 단점을 해결하기 위한 방법은 제안한다. [1] 본 연구에서는 GPU기반의 OpenGL 쉐이더를 이용하였으며, 렌더투텍스쳐(Rend to Texture)를 이용하여 지형(Terrain)을 만들었으며 하나의 바위 텍스처(Texture)를 이용하여 바위 지형의 느낌을 만들어 냈다. 또한 이미지를 제외한 다른 요소를 지형의 높낮이를 구현하는데 사용하지 않은 장점이 있다.

2. 알고리즘

2.1 하이트맵(Heightmap) 이용한 지형(Terrain) 제작

대부분의 하이트 맵 생성에서는 이미지의 정보를 이용해서 디지털화 된 윤곽에 대한 적절한 높이를 설정한 지형을 제작한다.[2] 이렇게 만들어진 지형은 대부분 단층 구제작이 된다. 이러한 방법으로 복층 구조를 제작하기 위해서는 여러 장의 이미지를 이용해서 만들어야하기 때문에 3D공간상 같은 동일한 위치상에 가지는 높이 값의 개수만큼의 이미지가 필요로 하게 된다. 결국, 복층 구조를 만들기 위해서 여러 장의 이미지가 요구하는 문제점이 있다. 본 연구는 이를 해결하기 위해 컬러 이미지가 갖는 멀티 채널 값을 이용하는 방법을 제안한다.

2.2 RGBA를 이용한 지형(Terrain) 제작

한 이미지에서 RGBA를 이용한 4채널 지형(Terrain)를 생성하는 방식은 기존의 하이트맵(Heightmap)를 이용하여 지형을 생성하는 방식과 높이를 정의 하는 방식에서 크게 달라지는 것은 없다. 다만, 기존의 방식과 다르게 이미지의 각 채널 값을 이용해서 각 각의 지형을 만들어 내는 차이점이 존재 한다. 기존의 방법은 대개 회색톤의 이미지를 이용하며 이 경우, 8비트 회색톤 이미지 좌표(x, y)의 밝기를 $I(x,y)$라고 하고, 하이트의 최대 값을 H라고 한다면 이 위치에서의 하이트는 아래와 같이 계산된다.

$$h = \frac{I(x,y)-255}{255}H$$

본 연구에서는 동굴과 같은 지형을 생성하기 위하여, 윗면, 아래 면에 해당하는 이중구조의 하이트를 아래 위로 연결하여 복층구조의 지형을 생성하였으며 이를 위해 회색톤이 아닌 컬러 이미지를 이용하였다. 컬러이미지는 보통 4개의 채널이 존재하며, 이 채널은 각각 R, G, B, A값을 가지며 이미지 뎁스(Image Depth), 즉 하나의 픽셀에 할당되는 비트수에 따라 각 채널이 가질 수 있는 값의 범위가 결정된다. 예를 들면, 각 채널 달 8비트를 가진 32비트 이미지의 경우, 각 채널은 (0~255)사이의 값을 가진다. 본 알고리짐은 r 채널을 이용하여 이용한 특정 위치 (x,y) 좌표에서의 하이트와 b채널을 이용한 하이트를 계산 후에, 가중치 w를 이용하여 선형 블랜딩하여 사용하여 최종 하이트를 계산한다.

$$h_r = \frac{I(x,y)_r - 255}{255} H$$

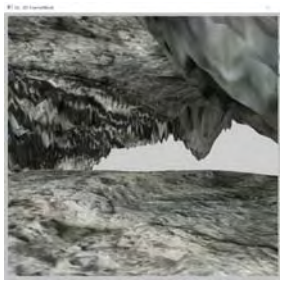
$$h_b = \frac{I(x,y)_b - 255}{255} H$$

$$h = mix(h_r, h_b, w)$$

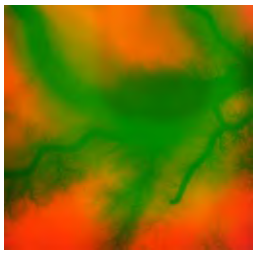
즉, 기존의 하이트맵 방식으로 지형을 만들지만 복층 구조 지형을 위해서 여러 장의 이미지를 사용하는 것을 방지하기 위해서 높이 정보를 각 색에 대입시켜서 동일한 지형을 단일 이미지로 만드는 것이다.

3. 실험

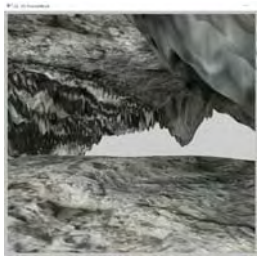
위에서 언급 했듯이 RGBA를 이용하여 단일 이미지로 생성하는 방식과 여러 장의 하이트 맵을 이용해서 만드는 것에 대하여 비교 분석하기 위해 Visual Studio 2013을 이용하여 Windows 10 위에서 구현하였다. 그림1은 단일 하이트맵(Heightmap) 이미지를 이용하여 동굴을 만든 스크린 샷이며 그림2는 해당 동굴을 만들기 위해서 사용한 이미지 파일이다. 그림3은 2가지의 하이트맵(Heightmap) 이미지를 이용하여 만든 스크린샷이며 그림4는 해당 동굴을 구현하기 위해서 사용한 두 개의 하이트맵 이미지이다. 윈도우 생성을 위해 FLTK UI라이브러리를 이용하였으며, 동일한 화이트 맵 이미지를 이용하여 만들었으며 단일 이미지를 만들 때 2이미지를 합치기만 하였다. 또한 천장 지형은 동일 위치상에서 높이 값이 커질수록 내려오게 했으며 아래 지형은 동일 위치상에서 높이 값이 클수록 올라가게 만들었다.



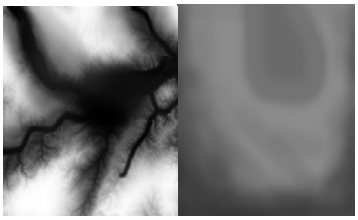
(그림 1) 컬러 이미지 하이트 맵을 이용한 동굴 스크린샷 (256x256)



(그림 2) 단일 이미지 하이트맵(heightmap)

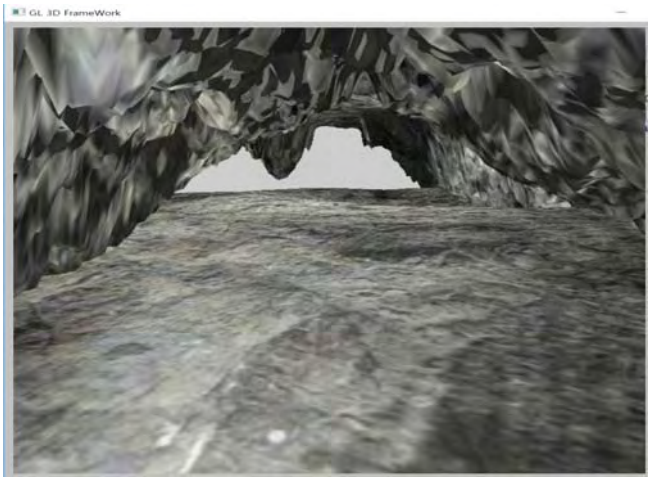


(그림 3) 두 장의 하이트 맵을 이용한 동굴 스크린샷



(그림 4) 두 장의 하이트 맵 (왼쪽:천정부분, 오른쪽 :바닥부분)

그림 1과 그림3을 비교해 보면, 생성된 지형은 시점의 각도 변화와 위치 변환에 따라 변형되지않았음을 확인 하였으며. 생성하기 위한 렌더링 속도 차이에도 변화가 없음을 확인하였다. 그림 6은 동일한 하이트 맵의 해상도를 변화시켜 생성한 동굴의 모습을 나타낸다.



(그림 5)고 해상도 컬러 이미지 하이트 맵을 이용한 동굴 스크린샷 (1024x1024)

4. 결론

본 논문은 3D 공간상 동일 좌표상의 여러 높이를 가지는 지형(terrain)를 제작하기 위한 OpenGL 렌더러를 구현하였다. 현제 여러 가지 높이 값을 저장하기 위해서 RGBA 값을 가질 수 있는 단일 하이트맵(heightmap)를 이용하여 만들었을 때와 여러장의 화이트맵 이미지를 사용하였을 때 동일한 결과를 얻었다. 하지만 산 중간에 동굴을 생성하는 과정에서는 동굴 입구 부분에 폴리곤이 생성되어 입구를 막아서 동굴 밖에서 동굴 안을 볼 수 없는 문제가 있으며 추후 연구를 진행할 예정이다.

Acknowledgment

본 논문은 교육부와 한국연구재단의 대학특성화사업(CK-1)의 지원을 받아 수행된 연구 결과입니다.

참고문헌

[1] David Wolf, OpenGL 4.0 Shading Lanuage Cookbook, PACT publishing, 2011

[2]Ruibin Zhao, Jidong Wang, Bin Yang, A Method for Generating Large-Scale Terrain Based on Image Set, IEEE2010

[4] Johan Andersson, *Terrain Rendering in Frostbite Using Procedural Shader*, WEI, L. 2004. Tile-Based Texture Mapping on Graphics Hardware. In proceedings of ACM SIGGRAPH/Eurographics conference on Graphics Hardware, pp. 55-63. Grenoble, France.