Journal of Korea Game Society 2013 Oct; 13(5): 55-62
http://dx.doi.org/10.7583/JKGS.2013.13.5.55

게임 프로그래밍

웨이블릿 압축 계수의 RGBA채널 인덱싱을 이용한 대용량 지형 렌더링 기법

김태권, 이은석, 신병석
인하대학교 컴퓨터정보공학과
program1@inha.edu, elflee77@nate.com, bsshin@inha.ac.kr

Massive Terrain Rendering Method Using RGBA Channel Indexing of Wavelet Coefficients

Tae-Gwon Kim, Eun-Seok Lee, Byeong-Seok Shin
Dept. of Computer Science and Information Engineering, Inha University

요 약

대용량 지형 데이터는 전체를 CPU나 GPU메모리에 적재할 수 없기 때문에 하드디스크와 같은 보조기억장치에서 필요한 부분을 읽어와 렌더링하는 out-of-core기반의 방법이 사용된다. 하지만 out-of-core 기반의 방법은 하드디스크로부터 GPU메모리까지 데이터를 읽어올 때 대역폭 한계로 인해 데이터의 전송시간이 길어진다. 이 논문에서는 Direct Compute를 이용하여 대용량 지형 데이터를 GPU에서 웨이블릿 기법으로 압축한 후 계수들을 이미지의 RGBA채널에 대응시켜 저장하고 렌더링 단계에서 이를 압축 해제하여 사용하는 방법을 제안한다. 이 방법은 GPU를 이용하여 압축된 지형 데이터를 빠르게 압축 해제해 사용함으로써 데이터의 전송량을 줄이고 웨이블릿 계산을 병렬적으로 수행하므로 전체 렌더링 시간을 단축할 수 있다.

ABSTRACT

Since large terrain data can not be loaded on the GPU or CPU memory at once, out-of-core methods which read necessary part from the secondary storage such as a hard disk are commonly used. However, long delay may occur due to limited bandwidth while loading the data from the hard disk to memory. We propose efficient rendering method of large terrain data, which compresses the data with wavelet technique and save its coefficients in RGBA channel of an image us, then decompresses that in rendering stage. Entire process is performed in GPU using Direct Compute. By reducing the amount of data transfer, performing wavelet computations in parallel and doing decompression quickly on the GPU, our method can reduce rendering time effectively.

Keywords : Terrain rendering(지형렌더링), Wavelet(웨이블릿), Level of Detail(상세단계)

Received: Sep. 18, 2013 Accepted: Oct. 11, 2013
Corresponding Author: Byeong-Seok Shin(Inha University)
E-mail: bsshin@inha.ac.kr

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

3D게임, 비행시뮬레이션, 지리정보시스템 (GIS : Geographic Information System)등 다양한 어플리케이션에서 넓은 지형을 상세하게 표현할 것을 요구함에 따라 최근의 지형 렌더링에서는 대용량 데이터의 처리가 매우 중요하다. 하지만 메모리 공간이 제한되어 있어서 최신의 하드웨어를 사용하더라도 대용량 지형 데이터를 모두 적재할 수 없으며 이 때문에 실시간으로 시각화하기 어렵다.

out-of-core 기법[1]은 지형 데이터를 하드디스크와 같은 보조기억장치에 저장하고 렌더링 할 때 필요한 부분만을 메모리에 적재하여 사용하는 기법이다. 지형 렌더링 분야에서는 보조기억장치에서 메모리로 데이터를 전송할 때 소요되는 시간을 단축시키기 위해 out-of-core기반의 상세단계(LOD: Level-of-Detail) 기법[2,3]들을 사용하여 데이터 전송량을 줄인다. 하지만 이 방법들은 시점변경으로 상세단계가 바뀔 때마다 보조기억장치에서 데이터를 읽어온 뒤 렌더링을 하여야 하므로 처리속도가 느리다.

이 논문에서는 웨이블릿[4] 압축을 이용해 DEM 데이터를 압축하고 웨이블릿 계수들을 이미지의 RGBA채널에 적용하여 보조기억장치에 저장한 후 GPU를 이용해 데이터를 빠르게 압축 해제 하는 방법과 테셀레이션(tessellation)[5]을 이용하여 상세단계 선택을 효과적으로 수행하는 방법을 제안한다. 이 방법을 사용하면 압축된 데이터를 RGBA 채널에 나누어 저장하므로 데이터 전송량이 줄어들고 A채널에는 지형을 렌더링 하는데 필요한 데이터를 저장하고 RGB채널에는 압축 해제에 필요한 계수들을 저장하므로 웨이블릿을 계산하는데 필요한 4개의 계수가 하나의 텍셀에 저장되어 인덱싱(indexing)하기가 쉬워진다. 또한 테셀레이션을 이용해 상세단계 선택을 수행하므로 모든 연산이 GPU상에서 처리되어 빠르고 크랙 제거를 위해 별도의 연산을 수행할 필요가 없다. 제안하는 방법은 시점의 변경으로 인해 고해상도의 데이터가 요구될 때 보조기억장치에서 데이터를 다시 읽어오지 않고 이미 적재되어 있는 데이터를 다른 비율로 압축 해제하여 사용하므로 보조기억장치에 접근하는 횟수를 줄일 수 있다.

이 논문의 2절에서는 관련 연구들을 소개하고 3절에서는 이미지의 RGBA채널에 맞춰 인덱싱한 웨이블릿 압축 방법과 테셀레이션을 이용해 상세단계 선택을 수행하는 방법에 대해 서술하고 4절에서 실험 결과를 보인 후 5절에서 결론을 맺는다.

2. 관련 연구

Hope는 OpenGL extensions를 이용해 고정 기능(fixed function) 파이프라인을 사용하는 2차원 이산 웨이블릿 변환을 제안하였다[6]. Wong는 이산 웨이블릿 변환을 GPU를 이용해 가속화하는 방법[7]을 OpenGL과 Cg로 구현해 JPEG2000의 유명한 코덱 중 하나인 JasPer[8]와 비교하였다. 하지만 이 방법은 완벽한 병렬화가 되지 않는 단점이 있다. Laan은 CUDA를 이용하여 GPU의 병렬처리에 맞는 코드를 설계해 압축 속도를 대폭 향상시켰다[9]. Treib는 여러 압축 기법들을 이용해 대용량의 지형을 편집하는 시스템을 제안하였다[10].

계층구조를 이용한 상세단계 기법들은 지형 렌더링 분야에서 대용량 데이터를 효과적으로 렌더링하기 위해 사용된다. Lindstrom[11]은 상향식(bottom-up) 접근법을 사용해 가장 상세한 단계에서부터 사진트리의 분할과 병합을 결정해 지형 메쉬를 간략화 하였다. Rötter[12]는 하향식(top-down) 접근법을 이용하였는데 전처리 시간에 미리 오차값을 계산해 정점을 제거하므로 렌더링시 빠르게 오차를 제거할 수 있다. Duchaineau는 삼각형 이진트리를 이용하여 시점에 따라 동적으로 지형을 생성하는 ROAM(real-time optimally adaptive meshes)[13]을 제안하였다. 하지만 이런 계층 구조에 대한 연산은 CPU에서만 가능하였다.

Out-of-core 기법을 사용한 대표적인 방법으로는 Losasso가 제안한 기하 클립맵(geometry clipmap)[2]이 있다. 이 방법은 Tanner가 제안한 클립맵[3]의 밉맵 피라미드 구조를 이용해 대용량 DEM 데이터를 효과적으로 읽어와 실시간으로 지형을 시각화 한다. 하지만 밉맵 피라미드를 사용하여 거리 기반의 상세단계를 선택하기 때문에 클립 영역이 작을 경우 기하 오차가 발생하고 시점 이동시 보조기억장치에서 지속적으로 데이터를 불러와 처리 속도가 느리다.

3. 웨이블릿 압축을 이용한 대용량 지형 렌더링

기존의 계층구조를 사용하는 기법들[11,12,13]은 CPU 친화적인 순차적 알고리즘을 사용하기 때문에 멀티코어 기반의 병렬 처리가 특징인 GPU에 적용하기에는 적합하지 않다. 웨이블릿 변환은 병렬 처리가 가능한 형태로 변형이 가능하고 그래픽 하드웨어는 텍스처를 빠르게 처리하는데 특화되어 있으므로 이미지의 RGBA채널에 웨이블릿 계수들을 대응시키면 GPU를 이용하여 효율적인 연산이 가능하다. 따라서 이번 절에서는 GPU를 이용해 대용량 DEM 데이터를 웨이블릿 변환을 통해 압축한 후 이미지의 RGBA채널에 적용하여 렌더링 하는 방법과 테셀레이션을 이용해 상세단계 선택을 수행하는 방법을 소개한다.

제안하는 방법은 먼저 전처리 단계에서 웨이블릿 변환을 수행해 DEM데이터를 압축한 결과로 나오는 계수들을 이미지의 RGBA채널에 매핑하여 보조기억장치에 저장하는 인덱싱 작업을 수행한다. 2차원 데이터는 압축을 한번 할 때마다 해상도가 1/2씩 줄어들고 압축을 한번 해제할 때마다 해상도가 2배씩 늘어난다. 따라서 밉맵처럼 모든 해상도의 데이터를 미리 만들어 둘 필요가 없고 압축 횟수에 따라 여러 해상도로 복원 가능한 다해상도(multi-resolution) 데이터를 구성할 수 있다. 렌더링 단계에서는 지형을 표현하기 위해 필요한 데이터를 보조기억장치에서부터 GPU메모리까지 불러와 현재 시점에 적절한 상세단계까지 압축을 해제하고 상세단계 선택을 수행한 후 렌더링한다.

3.1 웨이블릿을 이용한 압축과 인덱싱

DEM데이터는 정수로 이루어져 있으므로 웨이블릿 변환 시 웨이블릿 계수의 양자화를 수행하지 않는 리프팅 스킴(lifting scheme)[14]을 이용하는 정수 웨이블릿 변환(IWT: Integer Wavelet Transform)을 적용한다. 이것은 적은 연산으로 무손실 압축이 가능하다. 이 논문에서는 정수 웨이블릿 변환인 CDF 5/3 웨이블릿[15]을 GPU로 구현해 사용하였다.

$$\begin{cases} d_{2n} = X_{2n} - \frac{1}{2}(X_{2n-1} + X_{2n+1}) \\ c_{2n+1} = X_{2n+1} + \frac{1}{4}(d_{2n+1} + d_{2n+2} + 2) \end{cases} \quad \text{(eq. 1)}$$

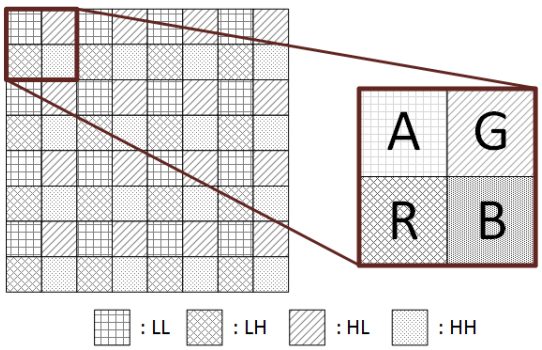
$$\begin{cases} X_{2n+1} = d_{2n+1} - \frac{1}{4}(c_{2n} + c_{2n+2} + 2) \\ X_{2n} = c_{2n} + \frac{1}{2}(X_{2n-1} + X_{2n+1}) \end{cases} \quad \text{(eq. 2)}$$

(eq. 1)은 데이터를 압축할 때, (eq. 2)는 압축해제할 때 사용되는 식이다. X는 원본 데이터, d는 웨이블릿 변환을 통해서 나온 상세계수, c는 웨이블릿 변환을 통해서 나온 근사계수를 나타내며 아래첨자들은 해당 데이터의 위치를 나타낸다.

2차원 데이터에 대한 웨이블릿 변환은 수평과 수직성분이 분리 가능(separable)하다. 따라서 행(가로방향)에 대해서 식을 적용하는 수평 통과 단계와 열(세로방향)에 대해서 식을 적용하는 수직 통과 단계로 분리해 각 단계별로 1차원 웨이블릿 변환을 수행한다. 이 과정은 각 통과 단계에 대해 결과값이 독립적이므로 효율적으로 병렬 처리가 가능하며 이 논문에서는 계산 쉐이더(compute shader)[16]를 이용해 수행하였다. 상세단계를 구

성하기 위해 웨이블릿 압축 횟수를 지정할 수 있으며 압축 횟수가 한번 증가할 때 마다 결과 해상도는 1/2씩 줄어든다.

[Fig. 1]는 2차원 데이터에 대한 웨이블릿 변환 결과를 RGBA채널에 맞춰 인덱싱 하는 모습을 보여준다. 2차원 DEM데이터에 대해 웨이블릿 변환을 수행하면 왼쪽과 같이 각 계수들이 격자 형태로 나열되어있는 결과가 나온다. 이 결과 값들은 하나의 텍셀에 하나의 계수가 저장되므로 그대로 저장할 경우 용량이 크고 압축을 여러 번 수행하면 해제 할 때 각 텍셀마다 필요한 계수들의 위치를 각각 계산해야하므로 비효율적이다. 하지만 그림의 오른쪽처럼 각 웨이블릿 계수들을 RGBA채널에 맞춰서 인덱싱 하면 각 통과 단계를 수행한 4개의 계수들이 모두 한 텍셀에 저장되기 때문에 데이터의 용량이 줄어들고 압축 해제 시 해당 텍셀의 계수들과 이웃한 한 텍셀의 계수들을 이용하기 때문에 인덱싱하는데 많은 비용이 들지 않는다. 또한 렌더링에 필요한 DEM데이터는 A채널에만 저장되기 때문에 인덱싱이 용이하다.



[Fig. 1] Storing wavelet coefficients after compression of two dimensional data without indexing (left) and mapping those coefficients to RGBA channels(right)

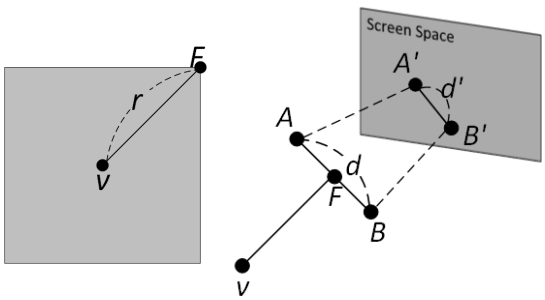
이 논문에서는 $2^n \times 2^n$의 해상도를 가지는 DEM 데이터를 사용한다. 계산 쉐이더를 사용하기 위해서는 스레드 그룹과 그 그룹 안에 있는 스레드의 개수를 먼저 설정한다. 쉐이더 모델 5.0에서 계산 쉐이더의 스레드 그룹과 스레드는 3차원 배열 (x, y, z)형태로 구성되며 최대 개수는 각각 제한이 있으므로 스레드를 분배할 때는 이 값을 넘지 않아야 한다[17]. 먼저 2차원 데이터의 웨이블릿 변환 시 행과 열에 대해 각각 1차원 웨이블릿 변환으로 수행된다는 점을 이용하고 스레드 그룹과 스레드 그룹의 스레드가 각각 3차원 형식으로 구성되기 때문에 스레드 구성을 2차원 배열 형태로 구성해 식을 적용하기 쉽게 설계하여 스레드를 배분한다. 2차원 DEM데이터의 한 축의 길이를 E라 하고 렌더링 시 사용되는 최소 크기를 가지는 DEM데이터의 한 축의 길이를 P라 할 때 스레드 그룹은(E, E/P, 1)로 설정하고 각 스레드 그룹이 가지는 스레드의 개수는 크기를 (1, P, 1)로 설정한다. 이렇게 설정된 스레드들은 (eq. 1)을 이용하여 연산을 한 후 [Fig. 1]과 같은 형태로 RGBA채널에 저장된다. 이 방법은 $P \times P$ 해상도 보다 큰 $2^n \times 2^n$ 해상도를 가지는 모든 데이터에 적용 가능하다. 복원은 압축 해제할 DEM데이터의 한 축의 길이가 e이고 제안하는 방법으로 압축했을 때의 한 축의 길이를 p이라 했을 때 계산 쉐이더를 이용한 스레드 그룹의 구성은 (e, e/p, 1)로 설정하고 각 스레드 그룹이 가지는 스레드의 개수는 (1, p, 1)로 설정해 복원할 지점의 해당 텍셀의 계수들과 이웃한 한 텍셀의 계수에 대해 (eq. 2)를 적용하여 계산한다. 저장된 데이터는 렌더링할 때 필요한 부분만을 GPU 메모리로 불러온다. 만약 불러온 데이터의 해상도가 필요한 해상도 보다 낮으면 계산 쉐이더를 이용해 원하는 해상도까지 압축을 해제한 후 렌더링 한다.

3.2 상세단계 선택

대용량 지형을 효과적으로 렌더링 하기 위해 압축한 데이터는 클립맵 형식으로 GPU에서 사용되고 상세단계 선택은 쉐이더 모델 5.0부터 지원하는 테셀레이션[5] 기능을 이용해 구현하였다.

여기서는 한 변의 길이가 클립맵에서 가장 거친 단계의 샘플링 간격에 해당하는 정규 격자를 생성

하고 시점과의 거리를 기반으로 격자를 테셀레이션 하는 상세단계 방법을 사용하였다. 격자의 분할을 위해서는 각 격자를 얼마나 분할할지를 결정하는 테셀레이션 인자가 필요하다. 이 인자는 헐 쉐이더 (hull shader) 단계에서 설정되며 쉐이더 모델 5.0 에서는 최소 0에서 최대 32까지 값을 설정할 수 있다. 격자의 분할은 격자의 중앙과 각 변에 대해 수행되므로 네 변에 대한 인자와 중앙에 대한 인자를 설정해야한다. 오차 없는 지형을 렌더링하기 위해서는 지형을 표현하는 격자를 구성하는 정점 사이의 거리가 화면 공간에 투영되었을 때 2픽셀을 초과해서는 안 된다. 이 논문에서는 클립맵을 이용하므로 오차 없는 영상을 출력하기 위해서는 화면 공간상에서 해당 클립 영역의 샘플링 간격보다 정규 격자를 구성하는 삼각형의 한 변의 길이가 더 작아야 한다. 따라서 테셀레이션 인자는 화면 공간상에서 해당 격자가 참조하는 클립 영역의 샘플링 간격보다 한 변의 길이가 작거나 같도록 설정한다. 격자 중앙의 인자는 네 변에 대한 인자의 평균값으로 설정하였다.



[Fig. 2] An example of representing the farthest position F of the clip region from the viewpoint v (left), and a projected image of AB with the sampling interval d (right)

테셀레이션 인자를 구하기 위해서는 오차가 발생하지 않는 격자의 최대 길이 τ를 알아야 한다. 화면 공간상에서 τ는 시점과의 거리에 비례하기 때문에 최소 τ를 구하기 위해서는 [Fig. 2] 왼쪽 그림과 같이 색칠된 부분이 클립 영역이라 할 때 시점 v에서 동일 클립 영역내의 가장 먼 지점인 F와의 거리 r을 이용한다. [Fig. 2] 오른쪽 그림의 $\overline{AB}$는 $\overline{VF}$와 수직하며 해당 영역을 표현하는 클립맵의 샘플링 간격 d를 길이로 갖는 선분이다. 이 선분을 화면 공간상에 투영하여 해당 지점에서 화면 공간상의 오차가 발생하지 않는 격자의 최대 길이 d'을 구할 수 있다. 따라서 d'은 시점과 수직한 $\overline{AB}$로부터 측정하였기 때문에 시점과의 거리 r을 이용한 비례식을 이용하여 r과 다른 거리 R에서 요구하는 τ를 (eq. 3)과 같이 구할 수 있다.

$$d' : r = \tau : R \qquad \text{(eq. 3)}$$

따라서 제안하는 방법은 테셀레이션 인자를 시점과 격자를 구성하는 한 변의 거리 R을 기준으로 조절하여 상세단계를 선택한다.

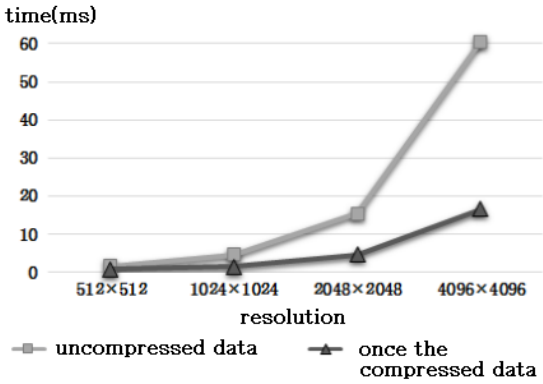
3.3 시각 절두체 선별

이 논문에서 시각 절두체 선별은 테셀레이션 기능을 이용하여 구현하였다. 렌더링 단계에서 화면에 표현되지 않는 부분을 렌더링하면 시간이 오래 걸릴 수 있으므로 시각 절두체의 외부에 있어 표현하지 않아도 되는 부분의 테셀레이션 인자를 0으로 설정하였다. 이렇게 하면 해당 삼각형은 삭제되기 때문에 속도가 빨라진다.

4. 실험 결과

본 실험은 Intel Core i5 3세대 3570 3.4GHz CPU에 8GB의 주 메모리를 갖는 시스템에서 수행했다. 그래픽 카드는 2GB의 메모리를 갖는 AMD Radeon 7850을 사용하였고 DirectX11 라이브러리를 사용하였다. DEM데이터는 16384×16384의 해상도를 가지는 Puget Sound를 이용하였고 DirectX11의 쉐이더 파이프라인에서 지원하는 텍스처의 최대 크기는 8192×8192이지만 그래픽 메모리의 여유를 두기 위해 DEM 데이터를 4096×4096

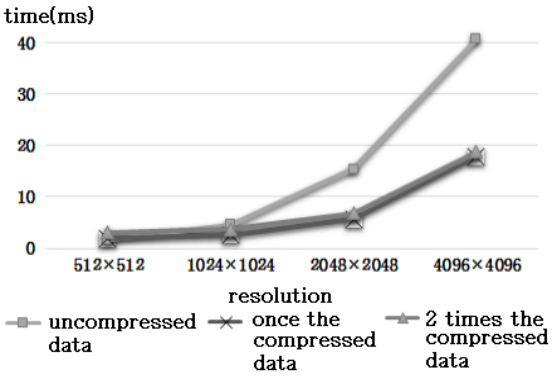
으로 8분할하여 제안하는 방법으로 압축하였으며 렌더링 할 때 사용되는 클립 영역의 최소 해상도는 512×512로 설정하였다. 실험은 제안하는 방법으로 압축되어진 데이터와 기하 클립맵 방식을 이용한 데이터가 GPU메모리에 적재되기까지의 시간을 비교하였다.



resolution	upload time(ms)	
	uncompressed data	once the compressed data
512^2	1.67	0.81
1024^2	4.59	1.44
2048^2	15.39	4.62
4096^2	60.63	16.59

[Fig. 3] Comparison of time to upload original data and compressed data applying the proposed method only once for the same area (unit : ms)

[Fig. 3]는 동일한 영역을 표현하기 위해 비압축 데이터를 적재하는 시간과 제안하는 방법을 적용한 지형 데이터에서 추출된 512^2 크기의 데이터를 GPU메모리로 적재할 때까지 소요되는 시간을 비교한 것이다. 그림에서 보듯이 해상도가 낮은 데이터에서 512^2 해상도의 지형을 임의로 추출해 GPU메모리에 적재하는 경우는 두 방법이 큰 차이를 보이지 않았으나 해상도가 높은 지형의 데이터일수록 제안하는 방법이 더 빠르다.



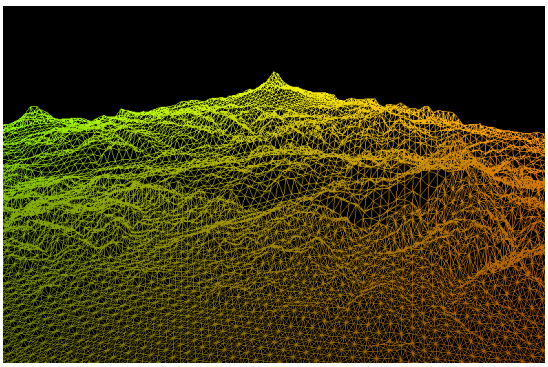
resolu-tion	total turnaround time(ms)		
	uncompressed data	the proposed method	
		one time compression	2 times compression
512^2	1.67	1.90	3.01
1024^2	4.59	2.52	3.64
2048^2	15.39	5.71	6.82
4096^2	60.63	16.28	17.40

[Fig. 4] Comparison of rendering time (including upload and decompression) of original data and compressed data with the proposed method (unit : ms)

제안하는 방법은 기하 클립맵 방법을 바탕으로 렌더링을 하지만 필요한 상세 단계의 데이터를 구성하기 위해 원본 기하 클립맵에서와 달리 압축을 해제하는 연산이 추가로 필요하다. 512^2 해상도의 데이터를 압축 해제해 하는데 걸리는 시간은 1회 압축(1단계 복원)된 경우 1.08ms이고 2회 압축(2단계 복원)된 경우 2.20ms이다. [Fig. 4]는 특정 해상도를 가지는 데이터에서 512^2 해상도만큼의 데이터를 추출해 데이터를 GPU메모리에 완전히 적재하기 까지 걸리는 소요 시간을 나타낸다. 원본 기하 클립맵의 경우 GPU메모리에 데이터 적재가 완료되는 시점까지의 소요 시간이고 제안하는 방법의 경우 GPU메모리에 데이터를 적재한 후 데이터를 압축 해제해 복원하는 시간까지 포함된 것이다. [Fig. 3]에서는 압축 해제 시간을 고려하지 않아서 해상도와 관계없이 제안 방법이 빠르지만 압축해제

시간까지 포함된 [Fig. 4]를 보면 낮은 해상도를 가지는 지형에서는 원본 기하 클립맵의 적재시간이 미세하게 더 빠른 것을 알 수 있다. 그러나 그 차이는 매우 미미한 편이며 해상도가 증가하면 제안하는 방법의 성능이 급격히 좋아짐을 확인할 수 있다. 이는 제안하는 압축방법이 대용량 지형 데이터 처리에 더 적합함을 의미한다.

[Fig. 6]은 제안하는 방법으로 렌더링한 결과 영상을 와이어 프레임으로 표현한 것이다. 시점과의 거리에 따른 테셀레이션을 이용해 상세단계 선택을 수행하였으므로 시점과 가까운 부분에는 많은 삼각형이 만들어지고 먼 곳은 적은 삼각형이 만들어지는 것을 확인할 수 있다.



[Fig. 5] A resulting image of wireframe rendering with the proposed method

5. 결 론

이 논문에서는 DEM데이터를 웨이블릿으로 압축한 후 이미지의 RGBA모델에 맞춰 보조기억장치에 저장함으로써 전송되는 데이터의 양을 줄이고 테셀레이션을 이용해 상세단계 선택을 효율적으로 수행하는 방법을 제안하였다. 제안하는 방법을 기존의 클립맵과 비교하였을 때 512^2의 해상도를 가진 데이터에 대한 수행 속도는 제안하는 방법이 느렸으나 그 이상의 해상도를 가진 데이터에 대해서는 더 빠르게 수행되었다. 이는 지형 데이터의 크기가 점점 증가하는 최근 추세에 제안하는 방법이 적합함을 의미한다. 또한 테셀레이션을 이용해 상세단계 선택을 수행하므로 효율적으로 오차 없는 영상을 출력할 수 있었다.

ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 45256-01).

REFERENCES

[1] P. Lindstrom, V. Pascucci, “Terrain simplification simplified: A general framework for view-dependent out-of-core visualization.”, IEEE Transactions on Visualization and Computer Graphics, Vol. 8, pp. 239-254, 2002.

[2] F. Losasso, H. Hoppe, “Geometry clipmaps: terrain rendering using nested regular grids.” ACM Transactions on Graphics (TOG), Vol. 23, pp. 769-776, 2004.

[3] C. Tanner, C. Migdal, M. Jones. “The clipmap: a virtual mipmap.” Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp. 151-158, 1998.

[4] A. Graps, “An introduction to wavelets.” Computational Science & Engineering, Vol. 2, pp. 50-61, 1995.

[5] Microsoft Corp, “Tessellation Overview”, http://msdn.microsoft.com/en-us/library/windows/desktop/ff476340(v=vs.85).aspx

[6] M. Hope, T. Ertl, “Hardware accelerated wavelet transformations.”, Data Visualization 2000, pp. 93-103, 2000.

[7] T. Wong, C. Leung, P. Heng, J. Wang, “Discrete wavelet transform on consumer-level graphics hardware.” IEEE Transactions on Multimedia, Vol. 9, No. 3, pp. 668-673, 2007.

[8] M. Adams, F. Kossentini, “JasPer: A software-based JPEG-2000 codec implementation.”, Proceedings of 2000 International Conference on Image Processing, Vol.2, pp. 53-56, 2000.
[9] W.J. van der Laan, A.C. Jalba, Jos B.T.M. Roerdink, “Accelerating wavelet lifting on graphics hardware using CUDA.”, IEEE Transactions on Parallel and Distributed Systems, Vol. 22, No. 1, pp. 132-146, 2011.
[10] M. Treib, F. Reichl, S. Auer, R. Westermann, “Interactive editing of gigasample terrain fields.”, Computer Graphics Forum, Vol. 31, pp. 383-392, 2012.
[11] P. Lindstrom, D. Koller, W. Ribarsky, L. Hodges, N. Faust, G. Turner, “Real-time, continuous level of detail rendering of height fields”, Proceedings of ACM SIGGRAPH 96, pp. 109-118, 1996.
[12] S. Rötter, W. Heidrich, P. Slusallek, H. Seidel, “Real-time generation of continuous levels of detail for height fields.” Proceedings of 6th International Conference in Central Europeon Computer Graphics and Visualization, p. 315-322, 1998.
[13] M. Duchaineau, M. Wolinsky, D. Sigeti, M. Miller, C. Aldrich, M. Mineev-Weinstein, “ROAMing terrain: real-time optimally adapting meshes”, Proceedings of IEEE Visualization 97, pp. 81-88, 1997.
[14] W. Sweldens, “The lifting scheme: A construction of second generation wavelets.”, SIAM Journal on Mathematical Analysis, Vol. 29, pp.511-546, 1998.
[15] A. Cohen I. Daubechies, J. Feauveau, “Biorthogonal bases of compactly supported wavelets.”, Communications on pure and applied mathematics, Vol. 45, pp485-560, 1992.
[16] Microsoft Corp, “Compute Shader Overview”, http://msdn.microsoft.com/en-us/library/windows/desktop/ff476331(v=vs.85).aspx
[17] Microsoft Corp, “ID3D11DeviceContext:: Dispatch method”, http://msdn.microsoft.com/en-us/library/windows/desktop/ff476405(v=vs.85).aspx

김 태 권 (Kim, Tae-Gwon)

2012년 8월 인하대학교 컴퓨터공학부(학사)
2012년-현재 인하대학교 컴퓨터정보공학부(석사)

관심분야 : 지형렌더링, 상세단계 선택

이 은 석 (Lee, Eun-Seok)

2008년 2월 인하대학교 컴퓨터공학부(학사)
2010년 8월 인하대학교 컴퓨터정보공학부(석사)
2011년 2월-현재 인하대학교 컴퓨터정보공학부(박사)

관심분야 : 실시간렌더링, 상세단계 선택, 차세대컴퓨팅

신 병 석 (Shin, Byeong-Seok)

1990년 2월 서울대학교 컴퓨터공학과(학사)
1992년 2월 서울대학교 컴퓨터공학과(석사)
1997년 2월 서울대학교 컴퓨터공학과(박사)
2000년-현재 인하대학교 컴퓨터정보공학부 교수

관심분야 : 볼륨그래픽스, 차세대컴퓨팅, 실시간렌더링