Journal of Korea Multimedia Society Vol. 17, No. 1, January 2014(pp. 69-76)
http://dx.doi.org/10.9717/kmms.2014.17.1.069

GPU 기반의 묶음 LOD 기법을 이용한 지형 렌더링의 가속화 기법

김태권[†], 이은석[††], 신병석[†††]

요 약

대용량 지형 데이터는 최신 그래픽 하드웨어를 사용해도 실시간으로 표현하기가 어렵다. 일반적으로 이런 대용량 지형 데이터를 실시간에 처리하기 위해 연속 상세 단계 기법 같은 메쉬 간략화 기법이 사용된다. 하지만 기하 분할(geometry splitting)과 같이 기존의 GPU기반 사진트리를 사용하는 기법은 트리의 깊이가 깊어질수록 많은 정점을 사용하고 이를 다시 재전송하기 때문에 성능이 저하되며 텍스처를 이용해 트리를 구성하기 때문에 트리의 용량이 커지는 단점이 있다. 이런 단점을 해결하기 위해 본 논문에서는 사진트리 기반의 묶음(chunk)으로 구성된 상세 단계 선별 기법을 GPU에서 처리하는 실시간 지형 렌더링 기법을 제안한다. 제안하는 방법은 기하 분할 기법의 단점을 해결하기 위해 트리 탐색을 제한하고 테셀레이터(tessellator)에서 묶음을 생성한 후 렌더링함으로써 트리 탐색을 줄이고 묶음을 GPU에서 바로 생성할 수 있어 효율적으로 지형을 렌더링할 수 있다.

An Acceleration Technique of Terrain Rendering using GPU-based Chunk LOD

Tae-Gwon Kim[†], Eun-Seok Lee[††], Byeong-Seok Shin[†††]

ABSTRACT

It is hard to represent massive terrain data in real-time even using recent graphics hardware. In order to process massive terrain data, mesh simplification method such as continuous Level-of-Detail is commonly used. However, existing GPU-based methods using quad-tree structure such as geometry splitting, produce lots of vertices to traverse the quad-tree and retransmit those vertices back to the GPU in each tree traversal. Also they have disadvantage of increase of tree size since they construct the tree structure using texture. To solve the problem, we proposed GPU-base chunked LOD technique for real-time terrain rendering. We restrict depth of tree search and generate chunks with tessellator in GPU. By using our method, we can efficiently render the terrain by generating the chunks on GPU and reduce the computing time for tree traversal.

Key words: Real-time Rendering(실시간 렌더링), Terrain Rendering(지형 렌더링), Level of Detail(상세 단계선별)

※ 교신저자(Corresponding Author) : 신병석, 주소 : 인천 남구 용현1,4동 인하대학교 하이테크센터 1403(402-751), 전화 : 032) 860-7452, FAX : 032) 872-7452, E-mail : bsshin@inha.ac.kr
접수일 : 2013년 11월 7일, 수정일 : 2013년 11월 10일
완료일 : 2013년 11월 14일

[†] 인하대학교 컴퓨터정보공학부
(E-mail : program1@inha.edu)
[††] 인하대학교 컴퓨터정보공학부
(E-mail : elflee77@nate.com)
[†††] 인하대학교 컴퓨터정보공학부
※ 이 논문은 2013년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 45256-01)

1. 서 론

최근의 지형 시각화에서는 광범위한 지형을 실시간에 상세하게 표현하기 위해 대용량의 지형 데이터가 사용된다. 대용량 데이터는 연산량이 많아 실시간에 렌더링하기가 어렵기 때문에 GPGPU기술을 이용하여 병렬로 처리해 가속화 하는 기법[1]들이 제안되었다. 하지만 사진트리를 사용하는 기존의 CPU 기반의 연속 상세 단계(CLOD : Continuous Level-of-Detail) 기법들은[2-4] 병렬 처리에 적합하지 않기 때문에 GPU에서 가속화하기 힘들다. 이런 문제점을 해결하기 위해 이전 연구에서는 기하 분할(geometry splitting)을 이용한 사진트리 기반의 삼각화 기법을 제안했다[5]. 이는 기하 쉐이더(geometry shader)의 스트림 출력 단계(stream output stage)를 이용하여 상세 단계를 선별하면서 정점들을 증식시켜 사진트리의 탐색을 GPU에서 할 수 있도록 하는 방법이다. 이 방법은 사진 트리 탐색 과정을 GPU에서 병렬로 처리하므로 CPU기반의 순차적인 사진트리 삼각화 기법에 비해 처리속도가 빠르다. 하지만 정점을 이용하여 트리 탐색을 하기 때문에 트리의 깊이가 깊어질수록 탐색에 사용되는 정점의 개수가 늘어난다. 따라서 GPU를 사용하여 처리를 하더라도 병목현상이 발생할 수 있고 스트림 출력 단계에서 생성된 정점 데이터를 다시 입력값으로 재전송하기 때문에 이로 인한 시간 지연이 발생한다.

이 논문에서는 이런 문제점을 해결하기 위해 사진트리 탐색 프로세스를 최소화 하고 쉐이더 모델(shader model) 5.0부터 지원하는 테셀레이션(tessellation) 기능을 이용해 균일한 격자로 이루어진 묶음(chunk)을 생성한 후 이들을 모아서 지형을 렌더링하는 방법을 제안한다. 이 방법은 사진트리 기반의 삼각화 기법과 달리 묶음을 이용하기 때문에 삼각형 개수가 많아지는 단점이 있으나 트리 탐색에 사용되는 연산으로 인한 병목 현상이 없다. 또한 지형의 세밀한 부분에 대한 묶음을 GPU에서 생성하여 렌더링하기 때문에 트리를 탐색 할 때 사용되는 정점 데이터를 줄일 수 있으므로 전반적인 렌더링 속도가 향상된다.

본 논문의 2절에서는 관련 연구를 소개하고 3절에서는 GPU 기반의 묶음 LOD 기법에 대해 서술하고 4절에서 실험 결과를 보인 후 5절에서 결론을 맺는다.

2. 관련 연구

대용량 지형 데이터의 렌더링을 효율적으로 하기 위해 계층 구조를 이용한 연속 상세 단계 선별 기법들이 연구되었다. Lindstrom과 Rötter는 각각 상향식(bottom-up) 접근법[2]과 하향식(top-down) 접근법[3]을 이용해 사진트리를 탐색하여 지형을 재구성하는 기법을 제안하였다. Duchaineau는 삼각형 이진트리를 사용하는 ROAM(Real-time Optimally Adaptive Meshes)을 제안했고[4] 이를 더 효율적으로 사용하는 방법을 Pomeranz와 White가 제안했다[6, 7]. Evans와 Basu는 right triangle을 사용한 계층 구조를 이용하는 방법을 제안했으며[8,9] Hope는 정점을 사용한 계층 구조를 사용하는 방법을 제안했다[10]. 이 방법들은 CPU를 이용해 연속 상세 단계 선별을 수행하기에는 효율적인 알고리즘이지만 대규모 병렬 처리를 기반으로 하는 GPU에는 적합하지 않다.

위 기법들의 단점을 해결하기 위해 기하 분할(geometry splitting) 기법이 사용된다[5]. 이 방법은 GPU를 이용해 각 정점을 4개로 증식시키는 과정을 반복하여 트리를 탐색한다. 이 과정은 각 정점마다 병렬로 처리되기 때문에 CPU기반의 기법들보다 빠르게 상세 단계 선별이 가능하다.

CPU의 연산을 줄이고 GPU와 CPU간의 데이터 전송을 최소화하기 위해 묶음 LOD와 같은 방법[11,12]은 기하 캐쉬(gepmetry cache)를 이용하여 빠른 속도로 지형을 렌더링을 할 수 있는 장점이 있다. 하지만 이 방법들은 기하 캐쉬로 사용하는 묶음과 같은 패치들은 간략화 할 수 없기 때문에 동일한 지형을 표시하는데 더 많은 기하정보를 필요로 하는 단점이 있다.

Tanner가 제안한 클립맵(clipmap)[13]을 이용하여 Losasso는 대용량 지형 데이터를 실시간으로 시각화하는 기하 클립맵(geometry clipmap)을 제안하였다[14]. 하지만 이 방법은 밉맵(mipmap)을 사용하여 거리기반의 LOD를 수행하므로 밉맵과 같은 상세 단계가 적용된 패치들을 기하 캐쉬를 이용하여 상세 단계 선별을 수행한다. 이 방법은 기하 캐쉬를 이용하여 효과적으로 상세 단계 선별을 할 수 있다는 장점이 있으나 지형과 패치의 거리에 따른 상세 단계 선별을 하므로 시점에 먼 곳의 지형은 사진트리를 사용하는 방법에 비하여 상세하게 표현하지 못하는

문제점이 있다.

3. GPU 기반의 묶음 LOD 기법

이 절에서는 GPU에서 묶음을 사용하여 상세 단계 선별을 하는 기법을 설명한다. 이 방법은 기하 분할 기법의 장점과 CPU기반의 묶음 상세 단계 기법의 장점을 이용하여 GPU에서 사진트리기반의 상세 단계 선별을 하고 묶음을 GPU에서 생성하여 기하 캐쉬 없이 지형을 효율적으로 시각화 하는 방법이다.

3.1 트리 탐색과 묶음 생성

이 논문에서는 기존의 기하 분할 기법을 이용하여 대용량 지형을 렌더링 할 경우 트리 탐색과정에 많은 정점이 필요하던 문제점을 해결하기 위하여 정점을 이용한 탐색이 일정 레벨에 도달하였을 경우 트리 탐색을 중지하고 해당 위치에서 균일한 해상도를 가지는 묶음을 생성하여 지형 메쉬를 렌더링하는 방법을 제안한다.

기존 기하 분할 기법에서 탐색할 트리의 최대 깊이는 지형 데이터의 해상도에 따라 결정된다. DirectX 11에서는 최대 8192×8192텍셀의 해상도를 가지는 텍스처를 GPU의 텍스쳐 버퍼에 적재할 수 있다. 하지만 사진트리를 사용할 경우 DEM데이터는 일반적으로 $(2^n+1)\times(2^n+1)$텍셀 크기를 가지므로 최대 4097×4097텍셀의 해상도를 갖는 데이터까지만 GPU에 적재할 수 있다. 여기서 n은 사진트리의 최대 깊이와 같기 때문에 최대 해상도를 갖는 DEM데이터를 이용하여 사진트리를 생성할 경우 이 사진 트리는 총 12개까지의 레벨을 가질 수 있다.

사진 트리로 인한 메모리 낭비와 탐색에 소요되는 데이터 및 연산을 줄이기 위해 제안하는 방법에서는 묶음을 사용한다. 묶음은 여러 개의 정규 격자들의 묶음으로 기존의 기하 캐쉬를 이용하여 사용하는 방법과는 다르게 GPU에서 헐 쉐이더(hull shader)를 통하여 생성된다. 헐 쉐이더는 하나의 삼각형이나 격자를 테셀레이션을 통하여 세분화 할 수 있도록 테셀레이션 인자(tessellation factor)를 정하는 단계이다. 이 인자값에 따라 다음 단계인 테셀레이터(tessellator)에서 입력된 삼각형 혹은 격자를 테셀레이션 하게 된다. 테셀레이션 인자는 최소 0에서 최대 32의 값을 가질 수 있다. 따라서 트리 탐색으로 인하여 생성된 격자를 묶음으로 변환할 때 각 묶음은 최대 32×32개의 해상도를 갖는 격자들로 구성된다. 묶음의 크기가 클수록 트리 탐색에 소요되는 시간과 트리의 깊이가 줄어들기 때문에 제안하는 방법에서는 묶음을 32×32개의 격자들로 구성한다. 따라서 최대 12의 깊이를 갖는 사진트리는 묶음을 이용할 경우 5단계가 줄어든 7단계로 구성되기 때문에 사진 트리를 저장하는데 소요되는 메모리 공간을 줄일 수 있고 탐색에 소요되는 정점과 연산시간을 절약할 수 있다.

기존의 기하 분할 방법은 각 노드에 해당 상세 단계를 갖는 격자의 기하 오차값을 화면 공간상에서 측정하여 상세 단계를 선별 하였다. 제안하는 방법은 여러 개의 격자들로 이루어진 묶음을 이용하여 렌더링하기 때문에 기존의 방법과는 달리 상세 단계 선별을 위해서 묶음을 이루는 격자들의 기하 오차값들을 측정하여 그 값들의 최대값을 묶음의 오차값 ε로 사진 트리에 저장한다.

그림 1은 ε를 이용하여 화면 공간상의 최대 오차값인 ε'를 구하는 방법을 보여준다. 일반적으로 화면 공간은 투시 투영(perspective projection)을 사용하기 때문에 먼 거리에 있는 물체가 가까이 있는 물체보다 작게 보인다. 따라서 화면 공간상의 최대 오차값 ε'를 구하기 위해서는 그림1과 같이 시점과 가장 가까운 거리에서 ε의 거리를 갖는 두 점을 화면공간상에 투영한 후 그 거리를 측정하여 최대 오차값을 구한다. c가 묶음을 둘러싼 경계구 (bounding sphere)의 중점이라 할 때, 시점과 c를 잇는 직선이

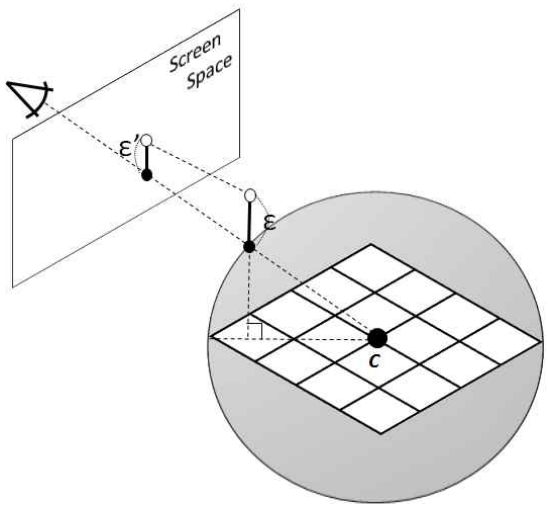

그림 1. 최대 오차값 ε를 이용하여 화면 공간상의 최대 오차값 ε'를 구하는 방법

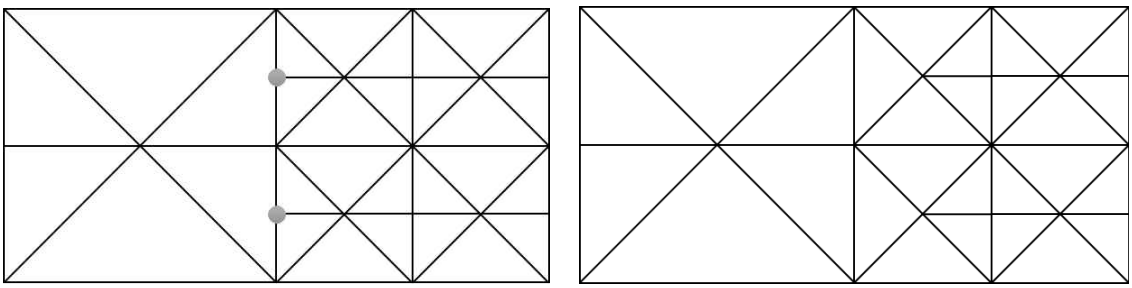

그림 2. 상세 단계의 차이로 인해 크렉이 발생하는 경우(위)와 크랙을 제거한 상태(아래)

경계구와 만나는 점들 중 시점에서 가까운 지점과 그 점에서 묶음과 수직하게 ε 만큼 더 높이 있는 지점을 두 정점의 위치로 한다. 묶음과 수직하게 ε 만큼 높이 있는 점을 사용하는 이유는 지형 데이터의 경우 높이값의 차이를 기하 오차로 갖기 때문에 시점과 가장 가까운 지점에서 최대 에러값 만큼의 높이 차이를 두어 화면 공간상에서 측정하기 위한 것이다.

$$F=\begin{cases}0 \text{ if } \tau \geq D(\varepsilon') \\ 1 \ otherwise\end{cases} \quad (1)$$

이렇게 구한 ε'를 이용하여 식 (1)과 같은 방법으로 트리 탐색을 계속 할지 여부를 결정한다. F는 트리 탐색을 계속 수행할지 확인 하는 플래그(flag)이며 D(ε')는 ε를 화면 공간상에 투영했을 때의 오차값이다. F의 값이 1일 경우 트리를 계속 탐색하고 0일 경우 트리의 탐색을 종료한다. 오차 없는 영상을 출력하기 위해서는 하나의 픽셀에 하나의 정점이 맵핑되어야 하므로 제안하는 방법에서는 τ를 0.5로 설정하였다. 0.5 픽셀보다 에러 값이 작을 경우에는 반올림하였을 경우 0픽셀의 오차를 갖기 때문에 화면상에 오차가 없는 영상을 얻을 수 있다.

3.2 크랙 제거

크랙은 특정 묶음의 상세 단계가 이웃한 묶음들과 다를 경우 발생한다. 이 논문에서는 크랙이 발생한 묶음들의 테셀레이션 인자를 조절하여 크랙을 제거하는 기법을 제안한다. 테셀레이션 인자의 경우 격자의 각 변과 격자의 중앙에 대해 값을 다르게 부여할 수 있다. 그림 2에서 보듯이 상세 단계 차이로 인해 크랙이 발생했을 때 상세 단계가 높은 오른쪽 묶음의 왼쪽 변에 해당하는 테셀레이션 인자를 낮춘다면 이웃한 묶음이 한 단계 덜 세분화 되었다 하더라도 크랙이 발생하지 않는다.

묶음의 최대 해상도는 32×32이므로 이웃 묶음과의 상세 단계가 최대 5레벨 차이 나는 경우까지 크랙 제거가 가능하다. 따라서 4097×4097의 해상도를 갖는 데이터를 사용할 경우 사용되는 사진 트리는 묶음의 해상도를 고려하였을 경우 깊이가 7단계 이므로 최소 2단계는 상세단계 선별과 관계없이 탐색을 해야 한다. 하지만 2단계 이하의 상세 단계를 사용하는 지형은 매우 큰 영역이 평지이거나 지형이 아주 작게 보이는 먼 시점에서 지형을 바라볼 경우이다. 이러한 상황은 일반적인 어플리케이션에서 발생하지 않는다. 따라서 트리의 탐색을 두 번째 단계부터 수행을 하면 효과적으로 크랙을 제거할 수 있으며 이렇게 트리 탐색을 중간부터 수행 할 경우 트리 탐색에 필요한 정점들과 반복연산을 줄일 수 있기 때문에 효과적으로 연산량을 줄일 수 있는 장점이 있다.

이와 같이 테셀레이션 인자를 조정하여 크랙을 제거하기 위해서는 이웃 묶음의 상세 단계를 알아야 한다. 따라서 모든 묶음들은 트리 탐색을 수행할 때 자신의 이웃에 있는 묶음들의 상세 단계에 대한 정보를 저장하고 있어야한다. 트리 탐색은 정점을 이용하여 수행하므로 정점의 RGBA채널에 각각 상하좌우 이웃 묶음의 상세 단계를 저장한다.

그림 3은 하나의 정점에 이웃 묶음의 상세 단계를 RGBA채널에 저장하는 예시이다. 좌측 그림에서 알

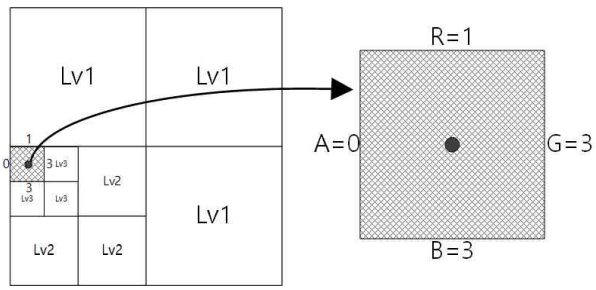

그림 3. 하나의 정점에 주변 묶음의 상세단계를 저장하는 방법의 예시

수 있듯이 A채널에 저장된 0은 지형의 최외곽 변으로 크랙 제거를 수행할 필요가 없는 곳을 나타낸다. 다른 채널들은 이웃한 묶음이 존재하므로 각 묶음들의 상세 단계를 저장한다. 따라서 이 값들을 각 변의 인자로 사용할 경우 효과적으로 크랙을 제거할 수 있다.

이웃한 묶음의 상세 단계는 그림 4와 같이 자신과 같은 레벨에 있는 이웃 묶음들의 트리탐색 여부를 검사하여 수행한다. 그림 4의 회색 묶음의 경우 자기와 동일한 상세 단계를 갖는 가상의 이웃 노드들을 생성한 후 자신의 세분화 여부를 검사함과 동시에 각각의 이웃 묶음들이 더 상세한 단계로 세분화 되어야 하는지 여부를 검사한다.

만약 회색 묶음이 더 세분화 되어야할 경우 상세 단계 선별이 종료된 가상의 이웃 묶음의 레벨을 해당 이웃을 가리키는 색상 채널에 저장한다. 이것은 앞서 설명한 바와 같이 이웃한 묶음의 상세 단계가 회색 묶음보다 상세하지 않다면 회색 묶음에서 해당 이웃과 인접한 면의 위치에 발생할 크랙을 제거해야 하기 때문이다. 이웃한 묶음이 계속 더 상세한 단계로 세분화해야 할 경우 세분화된 회색 묶음과 함께 다시 세분화 여부를 정해야 하므로 상세 단계를 저장하지 않는다.

그림 4의 왼쪽 가상 묶음과 같이 이웃한 묶음이 회색 묶음보다 더 세분화된 단계를 가진다면 회색

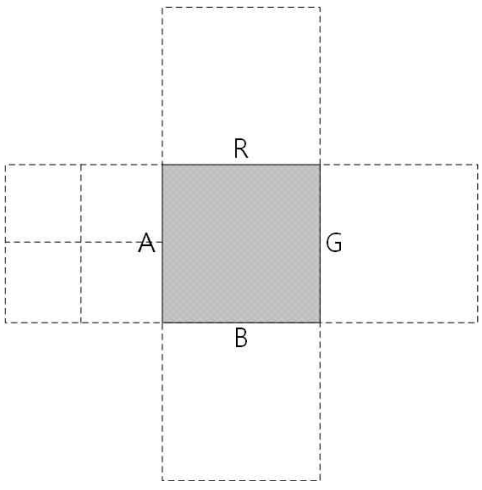

그림 4. 이웃한 묶음의 상세 단계를 검사하는 방법. 가운데 회색 묶음을 기준으로 자신과 같은 상세 단계를 갖는 가상의 이웃 묶음들을 생성하고 그 묶음이 더 상세한 단계로 세분화 되어야하는지 여부를 검사하여 세분화 될 필요가 없다면 해당 위치의 색상 채널에 저장한다.

묶음에서는 크랙 제거를 위해 해당 변의 인자를 변경할 필요가 없다. 따라서 회색 묶음의 상세단계 선별이 종료되었을 경우에는 상세 단계가 저장되지 않은 이웃한 노드에 자기 자신의 상세 단계를 저장하여 자기 자신의 상세 단계를 그대로 테셀레이션 인자로 사용한다.

이렇게 테셀레이션 인자를 이용하여 크랙을 제거할 경우 기존의 묶음 상세 단계 기법에서 사용하는 스커트(skirt)와 같이 추가적인 기하 데이터를 사용하지 않는다. 따라서 렌더링에 사용할 기하 데이터를 줄여 연산속도를 향상시킬 수 있다.

4. 실험 결과

실험은 Intel Core i5 3세대 3570의 3.4GHz로 동작하는 CPU에 8GB의 주 메모리를 갖는 시스템에서 수행하였다. 그래픽 카드는 2GB의 메모리를 갖는 AMD Radeon 7850을 사용하였고 DirectX11의 라이브러리를 사용하였다. 성능향상을 확인하기 위해 제안하는 방법과 기하 분할[5]을 이용한 방법 그리고 적응형 상세 단계 조정 방법[12]을 비교하여 실험하였다.

DEM데이터는 4097×4097의 해상도를 가지는 Puget Sound full(8bit)(A)과 Puget Sound part3 (16bit)(B) 2049×2049의 해상도를 가지는 Grand Canyon(8bit)(C), 1025×1025의 해상도를 가지는 Jeju Island(8bit)(D)를 사용하였다. 그림 5는 제안하는 방법으로 각 데이터 셋을 렌더링한 결과 영상이다.

표 1은 제안하는 방법과 다른 방법들의 데이터 셋에 대한 평균 프레임율을 비교한 표이다. 제안하는 방법과 기하 분할 방법을 비교하였을 때 약 5~35%의 성능 향상이 있었으며 특히 데이터의 해상도가 높을수록 제안하는 방법이 더 효율적임을 알 수 있다. 이는 기하 분할 방법에 비하여 본 방법에서 사용하는 사진 트리의 깊이가 더 적기 때문이다. 또한 적응형 상세 단계 조정 방법에 비해서는 약 83~234%의 성능 향상이 있었는데 이는 적응형 상세 단계 조정 방법이 상세 단계를 선별할 때 지형의 크기와 같은 텍스처에 트리 탐색 결과를 저장하는 것에 비해 제안하는 방법은 트리 탐색 결과를 별도로 저장하는 과정이 없어 병목 현상이 발생하지 않기 때문이다.

표 2는 제안하는 방법에 스커트 기법을 사용하여

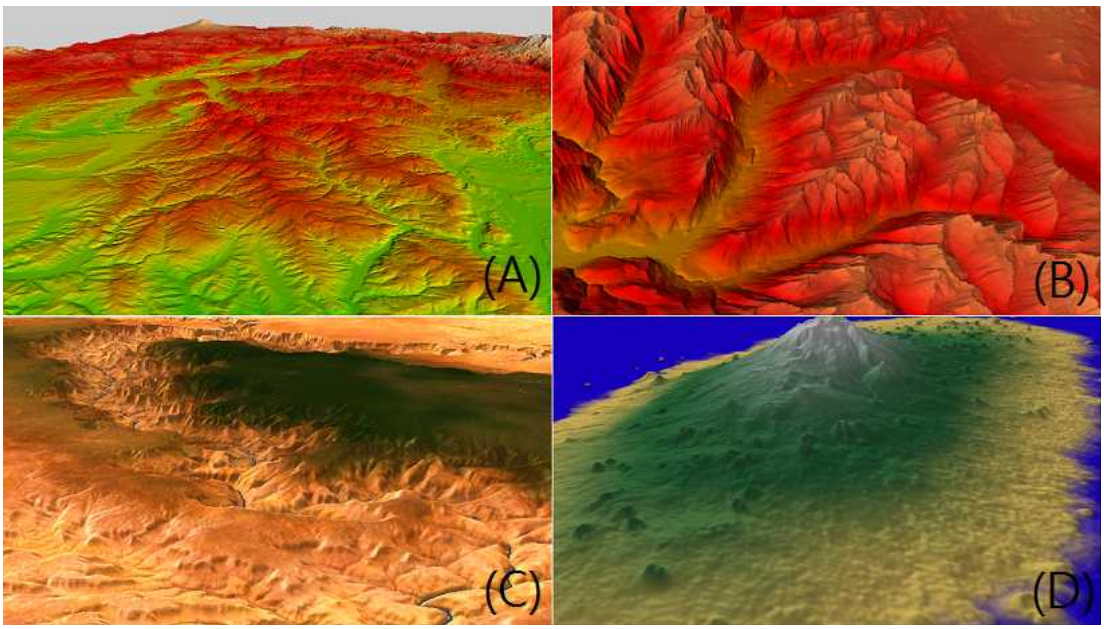

그림 5. 제안하는 방법으로 렌더링한 결과 영상

표 1. 제안하는 방법과 기하 분할, 적응형 상세 단계 조정 기법의 각 데이터 셋에 대한 프레임율 비교 (단위 : fps)

	제안하는 방법 (A)	기하 분할 기법 (B)	적응형 상세 단계 조정 기법 (C)	성능향상	
				A:B	A:C
Puget Sound full (4097×4097) 8bit	29	22	N/A	31%	N/A
Puget Sound part3 (4097×4097) 16bit	27	20	N/A	35%	N/A
Grand Canyon (2049×2049) 8bit	107	84	32	27%	234%
Jeju Island (1025×1025) 8bit	251	237	137	5%	83%

표 2. 제안하는 방법으로 크랙 제거를 수행했을 경우와 스커트(skirt)기법을 사용했을 경우의 프레임율 비교 (단위 : fps)

	제안 하는 방법(A)	스커트 기법 사용(B)	성능향상
			A:B
Puget Sound full (4097×4097) 8bit	29	25	16%
Puget Sound part3 (4097×4097) 16bit	27	24	13%
Grand Canyon (2049×2049) 8bit	107	91	18%
Jeju Island (1025×1025) 8bit	251	221	14%

크랙 제거를 수행하였을 때와 스커트 기법을 사용하지 않고 테셀레이터를 이용하여 크랙 제거를 수행하였을 때의 평균 프레임율을 비교한 것이다. 제안하는 방법을 사용하였을 때 모든 데이터 셋에 대해 약 13~18%의 성능 향상이 있었는데 이는 스커트 기법이 크랙 제거를 수행하기 위해 추가적인 기하정보를 사용하는 반면 제안하는 방법은 추가적인 기하데이터를 사용 하지 않고 테셀레이션 인자를 수정하여 크랙 제거를 수행하기 때문이다.

5. 결 론

이 논문에서는 기존의 기하 분할 기법이 트리의 깊이가 깊을수록 연산량이 많아져서 생기는 병목 현상과 기존 묶음 상세 단계 기법에서 기하 데이터로 인한 과부하 문제를 해결하기 위해 트리 탐색을 제한

하고 테셀레이터를 이용해 묶음을 생성해 렌더링하는 GPU기반의 묶음 상세 단계 선별 기법을 제안하였다. 제안하는 방법은 정규 격자들의 묶음을 사용하여 사진 트리의 규모를 축소하였다. 따라서 트리 탐색에 소요되는 정점수를 줄일 수 있었고 탐색 연산 또한 줄어들어 기존의 방법에 비하여 적은 메모리를 이용하고 빠른 렌더링 속도를 보일 수 있었다. 또한 기존 묶음 기반의 방법들과 같이 기하 캐쉬를 이용하지 않고 GPU에서 테셀레이터를 이용하여 고속으로 묶음을 생성하여 CPU와 GPU간의 전송을 최소화 하였다. 또한 이웃한 묶음과의 상세 단계 차이로 인한 크랙을 추가 기하 없이 테셀레이션 인자값을 조정하여 효과적으로 제거하여 렌더링 속도를 향상시켰다.

참 고 문 헌

[1] 계희원, “GPGPU를 이용한 고속 의료 볼륨 영상의 압축 복원”, 한국멀티미디어학회논문지, 제15권, 제5호, pp. 624-631, 2012.

[2] P. Lindstrom and V. Pascucci, “Terrain Simplification Simplified: A General Framework for View-Dependent Out-of-Core Visualization,” *IEEE Transactions on Visualization and Computer Graphics*, Vol. 8, No. 3, pp. 239-254, 2002.

[3] S. Rötter, W. Heidrich, P. Slusallek, and H. Seidel, “Real-Time Generation of Continuous Levels of Detail for Height Fields,” *Proc. 6th International Conference in Central Europeon Computer Graphics and Visualization*, pp. 315-322, 1998.

[4] M. Duchaineau, M. Wolinsky, D. Sigeti, M. Miller, C. Aldrich, and M. Mineev-Weinstein, “ROAMing Terrain: Real-Time Optimally Adapting Meshes,” *Proc. IEEE Visualization 97*, pp. 81-88, 1997.

[5] E.S. Lee and B.S. Shin, “Geometry Splitting: An Acceleration Technique of Quadtree-Based Terrain Rendering using GPU,” *IEICE Transactions on Information and Systems*, Vol. E94-D, No. 1, pp. 137-145, 2011.

[6] A. Pomeranz, *ROAM using Triangle Clusters (RUSTiC)*, Doctoral Dissertation of University of California, 2000.

[7] M. White, “Real-Time Optimally Adapting meshes: Terrain Visualization in Games,” *International Journal of Computer Games Technology*, Vol. 2008, No. 12, 2008.

[8] W. Evans, D. Kirkpatrick, and G. Townsend, "Right-Triangulated Irregular Networks," *Algorithmica*, Vol. 30, No. 2, pp. 264-286, 2001.

[9] S. Basu and J. Snoeyink, “Terrain Representation using Right-Triangulated Irregular Networks,” *CCCG*, pp. 133-136, 2007.

[10] M. Hope and T. Ertl, “Hardware Accelerated wavelet Transformations,” *Data Visualization 2000*, pp. 93-103, 2000.

[11] T. Ulrich, “Rendering Massive Terrains using Chunked Level of Detail Control,” *Proc. the SIGGRAPH*, 2002.

[12] 최인지, 신병석, “사진트리 기반 지형렌더링을 위한 GPU기반의 적응형 상세단계 조정 방법,” 한국게임학회지, 제8권, 제3호, pp. 61-68, 2008.

[13] C. Tanner, C. Migdal, and M. Jones, “The Clipmap: A Virtual Mipmap,” *Proc. the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 151-158, 1998.

[14] F. Losasso and H. Hoppe, “Geometry Clipmaps: Terrain Rendering using Nested regular Grids,” *ACM Transactions on Graphics* , Vol. 23, No. 3, pp. 769-776, 2004.

김 태 권

2005년 3월～2012년 8월 인하대학교 컴퓨터공학부 학사
2012년 8월～현재 인하대학교 컴퓨터정보공학부 석사과정
관심분야 : 지형렌더링, 상세단계 선택

이 은 석

2002년 3월～2008년 2월 인하대학교 컴퓨터공학부 학사
2008년 2월～2010년 8월 인하대학교 컴퓨터정보공학부 석사
2011년 2월～현재 인하대학교 컴퓨터정보공학부 박사과정
관심분야 : 실시간렌더링, 상세단계 선택, 차세대컴퓨팅

신 병 석

1990년 2월 서울대학교 컴퓨터공학과 학사
1992년 2월 서울대학교 컴퓨터공학과 석사
1997년 2월 서울대학교 컴퓨터공학과 박사

2000년～현재 인하대학교 컴퓨터정보공학부 교수
관심분야 : 불륨그래픽스, 차세대컴퓨팅, 실시간렌더링