핑퐁버퍼를 이용한 DEM 의 효율적인 사진 트리 삼각화

이은석, 이진희, Batamgalan Enkhtsoga, 신병석 *
인하대학교 컴퓨터정보공학과
e-mail : elflee77@nate.com, jhlee07@outlook.com, thebatama@gmail.com, bsshin@inha.ac.kr

An Efficient Quadtree-based Triangulation for DEM using Ping-Pong Buffer

Eun-Seok Lee, Jin-Hee Lee, Batamgalan Enkhtsoga, Byeong-Seok Shin*
Dept. of Computer Science and Information Engineering, Inha University

요 약

최근의 대용량 DEM 데이터는 실시간 렌더링을 하기엔 많은 양의 폴리곤을 필요로 한다. 사진 트리는 이러한 DEM 데이터를 실시간에 렌더링 하기 위해 지형 메쉬를 간략화하는데 널리 사용되는 자료구조이다. 트리구조는 재귀 연산 및 포인터 연산과 같이 GPU 에서 제공하지 않는 기능을 필요로 하기 때문에 일반적으로 CPU 상에서 구현되어 사용된다. GPU 에서 사진 트리 삼각화 기법을 사용하기 위해서 기존의 연구에서는 정점 프리미티브와 스트림 출력 단계를 이용하였다. 하지만 이 방법은 매 프레임 루트 노드부터 리프 노드까지 탐색을 하며 지형 메쉬를 새로 생성 해야하기 때문에 불필요한 연산이 많다. 제안하는 방법은 핑퐁 버퍼를 이용하여 이전 프레임에서 사용한 지형 메쉬를 다음 프레임에서 재활용하여 기존 GPU 기반 사진 트리 삼각화 기법을 가속화한다. 기존 방법이 매 프레임 사각형 패치를 세분화 하면서 지형 메쉬를 생성하는 대신 제안하는 방법은 이전 프레임에서 사용한 메쉬의 각 패치들을 병합하거나 세분화하는 방법을 사용한다. 따라서 본 방법은 GPU 기반 사진 트리 삼각화의 재귀 호출을 제거하여 연산량을 줄이고 매 프레임 CPU-GPU 간의 데이터 전송량도 효율적으로 줄여 기존의 방법을 효율적으로 가속화 한다.

1. 서론

고도필드의 한 종류인 DEM(Digital Elevation Model)은 비디오 게임, 비행 시뮬레이터, 가상 현실 등의 실시간 어플리케이션에서 지형 데이터를 표현하기 위해 널리 사용된다. 최근 DEM 데이터의 용량이 증가함에 따라 고정밀 지형을 렌더링 할 수 있게 되었지만 최신의 그래픽 장치를 사용함에도 불구하고 실시간에 DEM 데이터 그대로 지형을 렌더링 하는 것에는 한계가 있다. 따라서 상세단계를 선별하여 고정밀 지형을 실시간에 렌더링 할 수 있도록 정밀성을 유지한 채 간략화 하는 상세단계(Level-of-detail) 선별 기법들이 제안되었다.

사진 트리(quadtree)[1]는 2 차원 공간을 효율적으로 관리할 수 있는 계층적 자료구조로써 정규 격자를 이용한 지형 메쉬의 재구성에 효과적이다. 사진 트리를 이용하여 간략화 된 지형 메쉬는 적은 폴리곤 갯수로 정밀한 지형을 표현할 수 있다.

과거의 사진 트리 기반 삼각화 기법[2,3,4]은 CPU 에서 주로 수행이 되었으며 이것은 최근 GPU 를 이용한 방법인 기하 클립맵[5,6]이나 청크 메쉬[7,8]를 이용한 방법에 비하여 효율성이 떨어진다. 이전 연구에서는 이러한 단점을 해결하기 위해서 GPU 기반 사진 트리 삼각화 기법인 기하 분할 기법[9,10]을 제안하였다. 이 방법은 쉐이더 파이프라인의 기하 쉐이더의 스트림 출력 단계[11]를 이용하여 기하 데이터들을 재귀적으로 사분할 하면서 지형 메쉬를 생성한다. 이것은 GPU 에서 수행 할 수 없는 재귀연산이나 포인터 연산을 가능하게 하였다. 하지만 매 프레임 메쉬를 생성해 주어야 했기 때문에 반복된 작업이 많은 단점이 있었다.

본 논문에서는 이전 프레임에서 사용된 지형 메쉬를 사각형 패치의 분할 및 병합을 통하여 재사용하는 핑퐁 버퍼를 이용한 렌더링 기법을 소개한다. 핑퐁 버퍼는 2 개의 스트림 출력 버퍼로 구성되며 이전 프레임에서 렌더링에 사용된 버퍼에 저장된 지형 메쉬를 기하 쉐이더 단계에서 세분화 및 병합을 한 후 다른 버퍼에 저장한다. 이러한 작업은 GPU 의 쉐이더 파이프라인에서만 이루어지며 기존 방법의 반복된 연산을 제거하였기 때문에 연산량 감소와 데이터 재전송량을 효율적으로 줄여 렌더링 속도를 가속화시킬 수 있다.

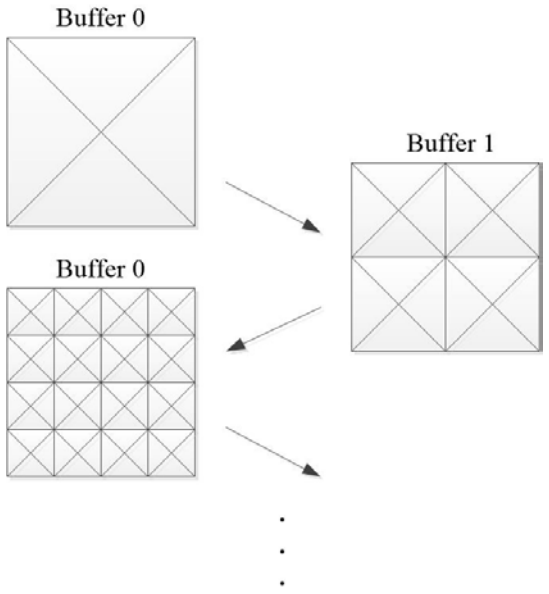
2 절은 세분화 및 병합단계를 수행하는 상세단계 선별 알고리즘과 핑퐁 버퍼에 대해 설명한다. 3 절은

*교신저자

제안하는 방법의 실험 결과를 보여주고 4 절에서 결론을 맺는다.

2. 핑퐁 버퍼를 이용한 상세단계 선별기법

일반적으로 핑퐁 버퍼는 I/O 프로세스를 효율적으로 가속화 하기 위해 두개의 버퍼를 사용하는 더블 버퍼링 기법의 한 종류이다. (그림 1)과 같이 핑퐁버퍼는 상단의 사각형 1 개만 저장된 Buffer 0 의 내용을 읽어 4 개의 사각형으로 세분화 한 후에 Buffer 1 에 저장한 후에 Buffer 1 을 렌더링 한다. 다음 프레임에서는 Buffer 1 을 입력값으로 읽어오고 Buffer 0 에 세분화한 결과를 저장한 후 Buffer 0 를 렌더링한다.

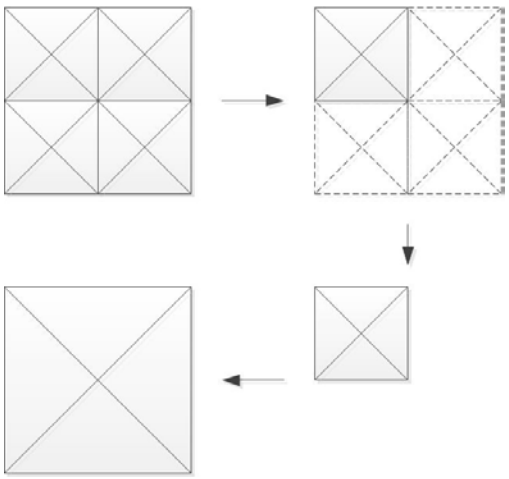


(그림 1) 핑퐁 버퍼를 이용한 지형 세분화

기존 기하분할 기법[9,10]에서 이전 프레임에 사용된 메쉬를 재활용 하지 못하였다. 그 이유는 기하 쉐이더가 세분화에만 특화된 단계이기 때문에 기하 데이터의 병합을 처리하지 못하기 때문이다. 따라서 지형에서 시야가 멀어지거나 평지에 해당하는 지역을 렌더링 할 경우와 같이 간략화가 필요한 경우에 이전 프레임에서 사용하던 메쉬를 재사용 할 수 없었다. 따라서 매 프레임 루트 노드에 해당하는 사각형 패치부터 리프 노드에 해당하는 상세단계까지 탐색하여 간략화를 수행한 후에 결과를 렌더링 해야 했다.

본 논문의 상세단계 기법은 이러한 단점을 해결하기 위해 기하 쉐이더 단계에서 기존에 지원하지 못하던 기하 병합 기법을 제안한다. 기하 쉐이더 단계에서는 프리미티브의 추가/삭제가 가능하다. 따라서 기하 병합을 위해서 (그림 2)와 같은 개념으로 기하 쉐이더 단계의 삭제 기능을 이용하면 기하 병합을 구현할 수 있다.

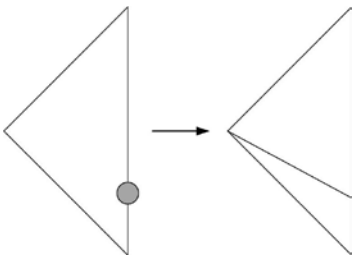
하지만 이 방법은 CPU 기반의 방법과 다르게 각 사각형 패치들이 병합 및 분할 과정을 병렬로 수행해야 한다. 따라서 어느 사각형 패치를 삭제할 것이고 어느 패치를 확대할 것인지를 정해야 하며 이웃 노드들을 탐색하여 크랙 발생 여부의 확인을 각각의 쓰레드가 전부 처리할 수 있어야 한다.



(그림 2) 사각형 패치의 병합과정

제안하는 방법에서는 사각형 패치를 더 세분화 하기 전에 각 사각형 패치를 담은 노드들은 부모 노드를 탐색하여 세분화 여부를 검사한다. 이것은 병합 프로세스가 발생하느냐 안하느냐를 탐색하기 위함이다. 만약 부모 노드에서 세분화 할 필요가 없다면 현재 사각형 패치들은 병합되어야 하는 대상이다. 만약 병합을 하게 된다면 삭제할 패치와 확장해야 하는 패치를 구분해야 한다. 이것은 사진 트리 텍스쳐에 전처리 과정에 미리 입력을 해두면 쉽게 해결 할 수 있다.

크랙 제거를 위한 이웃 노드의 상세단계는 각 사각형 마다 저장하기엔 최악의 경우 이웃한 패치들이 많아져서 무리가 있다. 이 경우엔 각 패치의 변에 이웃 노드의 형태를 저장한 트리구조가 저장되어 있어야 한다. 이것은 출력 할 수 있는 삼각형 개수가 제한되어있는 기하 쉐이더에 적합하지 않다. 이러한 단점을 해결하기 위해서 제안하는 방법은 (그림 3)과 같이 사용하여 크랙을 제거하는 방법을 사용한다. (그림 3)의 삼각형은 (그림 2)의 사각형 패치중에 오른쪽 변을 구성하는 삼각형이다.



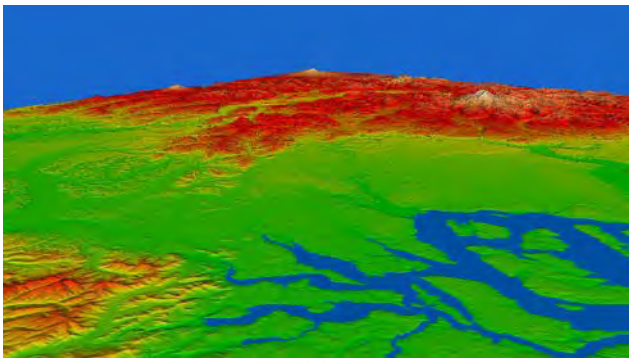
(그림 3) 회색 점들이 크랙이 발생하는 위치이며 삼각형을 세분화하여 크랙을 제거한다

이 방법은 이웃한 패치가 자신보다 하위 상세 단계에 있을 경우 크랙이 발생하기 때문에 발견하는 즉시 한 변을 구성하는 삼각형을 크랙이 발생하는 지점과 맞추어 분할한다. 이 경우 각 패치를 구성하는 정규

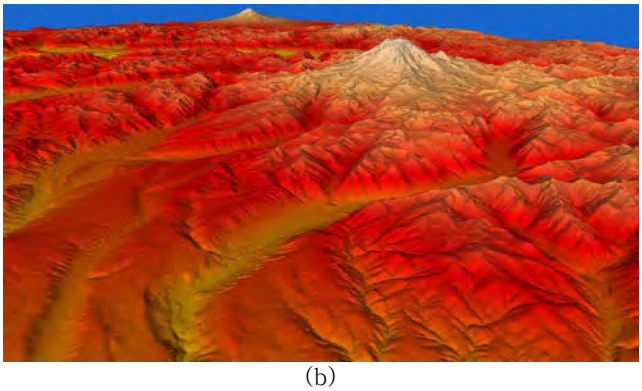
격자의 형태를 유지하지 못하지만 각 사각형 패치의 상세단계를 사각형의 중앙에 해당하는 정점에 저장함으로써 모든 삼각형이 패치의 상세단계 정보와 위치 정보를 공유할 수 있다. 따라서 매 프레임 갱신이 되더라도 본인의 정보를 담은 중앙 정점과 빗변을 분할한 변을 이루는 두 정점에 이웃 노드의 정보를 저장하여 매 프레임 갱신되더라도 크랙 발생을 막을 수 있다.

3. 실험결과

실험은 Intel® Core™ i5-4670 Processor CPU 3.40GHz 에 4GB 의 메인 메모리를 갖는 시스템에서 수행하였다. 그래픽 장치는 1GB 의 비디오메모리를 갖는 NVIDIA™ GTX560Ti 을 사용하였고 그래픽 라이브러리는 DirectX 11.1 을 사용하였다. 데이터셋은 4096^2 의 해상도를 갖는 8 비트 DEM 데이터인 Puget Sound 를 사용하였으며 HD 급 고화질 렌더링을 위해 뷰포트 크기는 1600×900 으로 하였다. 관측 조건은 근거리와 원거리로 하였고 상세단계는 화소 단위의 허용 오차를 1 픽셀 이하로 선별 하였다. (그림 4)는 관측 조건에서 렌더링 한 영상이다.



(a)



(b)

(그림 4) 원거리와 근거리 시점에서 생성된 영상

<표 1> 제안하는 방법과 기존의 방법과 성능비교. Ratio 는 제안하는 방법의 fps 와의 비율이다.

	원거리		근거리	
	fps	ratio(%)	fps	ratio(%)
제안하는 방법	99	100	78	100
기하 분할	52	190.38	27	288.88
이중모드 정점 분할	88	112.5	49	159.18

<표 1>에서는 (그림 4)의 원거리 시각과 근거리 시각에서 기존의 GPU 기반의 사진 트리 삼각화에 대한 연구였던 기하 분할 기법[9] 이중모드 정점 분할 기법[10], 그리고 제안하는 방법과의 성능을 비교하였다. 실험결과 원거리 시점에 비해서 근거리 시점에서 더 큰 속도 향상이 있음을 확인했다. 이것은 근거리에서 더 상세한 지형을 렌더링 해야 하므로 트리 탐색을 위한 기하 데이터의 전송량이 많아지기 때문이다. 기하분할 기법은 사진 트리를 루트부터 리프 노드까지 정점 분할과 삼각형 분할 단계로 각각 1 번씩 총 2 번을 탐색해야 하며 이중모드 정점 분할 기법은 1 번 탐색을 해야한다. 하지만 제안하는 방법은 이전 프레임에서 사용한 데이터를 그대로 사용하기 때문에 현재 노드의 부모 혹은 자식 노드로 1 번만 전파하면 되기 때문에 데이터 전송량이 적고 프로세스가 단순하다.

하지만 정점불할 기법에서 정점을 이용하는 반면 제안하는 방법은 삼각형으로 구성된 사각형 패치를 이용하므로 전반적으로 전송되는 횟수는 적으나 전송해야 하는 기하 데이터의 양이 많기 때문에 같은 삼각형을 이용하여 트리를 탐색하는 기하 분할 기법과는 달리 이중모드 정점 분할에서는 성능 차이가 크게 나지 않는다.

4. 결론

제안하는 방법은 매 프레임 루트 노드부터 리프 노드까지 전부 탐색해야 했던 기존의 방법들의 단점을 해결하고자 핑퐁 버퍼를 이용하여 매 프레임에 이전 프레임에서 사용했던 지형 메쉬를 재사용 하여 트리 탐색 횟수를 줄임으로서 렌더링 속도를 개선하였다. 이 방법을 광선 투사법이나 청크 메쉬등의 사진 트리 기반 방법들에 접목시킴으로 렌더링 성능을 개선할 수 있도록 확장할 것이다.

5. 감사의 글

이 논문은 2015 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2015R1A2A2A01008248).

참고문헌

[1] H. Samet, "The quadtree and related hierarchical data structures", ACM Comput. Surv., vol.16, no.2, pp.187–260, 1984.

[2] P. Lindstrom, D. Koller, W. Ribarsky, L. Hodges, N. Faust, and G. Turner, "Real-time, Continuous Level of Detail Rendering of Height Fields", In: Proceedings of ACM SIGGRAPH'96, Addison Wesley pp.109-118, 1996.

[3] S. Röttger, W. Heidrich, P. Slusallek, and 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, 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", In: Proceedings of Visualization'97, pp.81-88 ,1997.
[5] C.C. Tanner, C.J. Migdal and M.T. Jones, "The clipmap: a virtual mipmap," Proc. ACM SIGGRAPH 1998, pp.151-158, 1998.
[6] F. Losasso and H. Hoppe, "Geometry clipmaps: terrain rendering using nested regular grids," in ACM Transactions on Graphics (TOG), 2004, vol. 23, pp. 769–776.
[7] T. Ulrich, "Continuous LOD terrain meshing using adaptive quadtrees," http://www.gamasutra.com/features/20000228/ulrich_pfv.htm, 2000.
[8] E.S. Lee, J.H. Lee, and B.S. Shin, "Vertex relocation: a feature-preserved terrain rendering method for pervasive computing environments," Multimed Tools Appl, pp. 1–17, Jun. 2015.
[9] 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.
[10] E.S. Lee, J.H. Lee, and B.S. Shin, "Bimodal Vertex Splitting: Acceleration of Quadtree Triangulation for Terrain Rendering," IEICE TRANSACTIONS on Information and Systems, vol. E97-D, no. 6, pp. 1624–1633, Jun. 2014.
[11] D. Blythe, "The direct3d 10 system," ACM Transactions on Graphics (TOG), vol. 25, no. 3, pp. 724–734, 2006.