정점증식을 이용한 사진트리 기반 지형 시각화 기법

이은석[O] 신병석

인하대학교 컴퓨터·정보공학과

elflee77@inha.edu bsshin@inha.ac.kr

Quadtree-based Terrain Visualization Using Vertex Multiplication

Eun-Seok Lee[O] Byeong-Seok Shin

Dept. of Computer Science and Information Engineering, Inha University

요 약

사진트리는 지형 시각화를 위한 점진적인 메쉬 생성에 널리 사용되는 자료구조이다. 사진트리는 빠른 상세단계 선택과 효과적인 시각절두체 선별이 가능하다. 하지만 계층적인 자료구조이므로 CPU에서만 사용할 수 있고 그래픽 하드웨어의 렌더링 파이프라인에서는 활용할 수 없다. 따라서 기존의 사진트리 기반 지형 시각화 기법들은 GPU를 이용한 다른 시각화 방법들에 비하여 CPU 의존도가 크고 처리시간이 오래 걸린다. 본 논문에서는 정점증식을 이용하여 GPU만으로 지형을 렌더링 하는 방법을 소개한다. 이 방법은 기존의 CPU를 이용한 사진트리기반 방법들에 비하여 화질의 저하 없이 빠른 속도로 렌더링 할 수 있다.

Abstract

In terrain visualization, the quadtree is the most frequently used data structure for progressive mesh generation. The quadtree provides an efficient level-of-detail selection and view frustum culling. However, most applications using quadtrees are performed by the CPU, since the hierarchical data structure cannot be manipulated in a programmable rendering pipeline. For this reason, quadtree-based methods show lower performance and higher dependancy of CPU in comparison to GPU-based methods. We present a quadtree-based terrain-rendering method for GPU execution that uses vertex multiplication. It offers higher performance than previous CPU-based quadtree methods, without loss of image quality.

키워드: 지형 렌더링, GPU기반 렌더링, 실시간 렌더링, 상세단계

Keywords: Terrain rendering, GPU-based rendering, Real-time rendering, Level of detail

1. 서론

지형 시각화는 3D게임이나 비행 시뮬레이션 등에서 사실적인 장면을 표현하는데 널리 사용된다. 일반적으로 거대한 입력 데이터를 다루기 때문에 이에 대응하는 메쉬를 생성하면 대용량의 비디오 메모리를 탑재한 최신 그래픽 하드웨어조차도 처리하기 어렵다. 따라서 결과 영상의 화질을 적절히 유지하는 메쉬 간략화 기법들이 필요하다.

사진트리(quadtree)[1]는 2차원 공간을 효율적으로 표현하는 계층적 자료구조로써 지형 렌더링에서 많이 사용되는 메쉬 간략화 기법인 연속상세단계 선택과 시각절두체 선별 기법에 효과적이다. 이는 고정밀 지형 영상을 실시간으로 생성할 수 있게 한다. 하지만 그래픽 하드웨어의 렌더링 파이프라인에서는 정점 배열이나 텍스쳐 등 순차적인 데이터만 처리가 가능하므로 사진트리와 같이 지형 렌더링에서 사용되는 계층적 자료구조는 사용할 수 없다. 따라서 사진트리를 이용하는 방법들은 CPU에서만 수행되고 이로 인해 화질과 속도가 저하되는 문제가 생긴다.

본 논문에서는 지형 렌더링에 필요한 사진트리를 정점증식 기법을 이용하여 그래픽 하드웨어의 렌더링 파이프라인에서 사용 가능하도록 함으로써 GPU만을 이용하여 사진트리 기반의 지형 렌더링 기법을 제안한다.

2장에서는 지형데이터를 시각화 하는데 사용되는 여러 가지 알고리즘들을 소개하고, 3장에서는 정점증식기법을 이용하여 GPU만으로 지형을 시각화 하는 방법을 기술한다. 4장에서는 기존의 방법과 비교한 결과를 보이고 결론을 맺는다.

2. 관련 연구

연속상세단계 기법 (CLOD : continuous level-of-detail)은 대용량 지형 데이터를 효과적으로 렌더링 하기위한 방법이다. 계층적 자료구조를 이용한 대표적인 CLOD방법에는 지형 데이터를 재귀적으로 4개의 균일한 영역으로 나누는 사진트리기반의 방법[2,3]과 삼각형을 재귀적으로 2분할하여 관리하는 이진트리기반의 방법[4] 그리고 삼각형의 가장 긴 변을 기준으로 분할과 병합을 반복하여 지형 메쉬를 관리하는 방법[5]등이 있다. 하지만 기존의 CPU기반 CLOD 기법들에는 두 가지 문제점이 있다. 우선, 공간 분할을 통해 시점 기반의 시각절두체 선별기법 적용에 유리하지만, 관측조건의 변화에 따라 렌더링 속도가 달라지는 문제점과 기존 CLOD 기법들은 매 프레임마다 CPU에서 기하정보를 생성하여 비디오 메모리에 업로드해야 하기 때문에 실시간 처리가 어렵다는 문제점을 갖는다. 첫 번째 문제점은 Shin 및 Choi가 실시간으로 삼각형 개수를 조절하여 불규칙한 렌더링 속도를 안정화시키는 방법[6]을 제안함으로써 해결되었고, 두 번째 문제점은 Levenberg가 제안한 기하 캐쉬방법[7]을 통해 해결되었다. 하지만 해당 기법은 제한된 캐쉬의 용량 때문에 대용량의 데이터의 실시간 렌더링이 어렵다. 이런 문제점을 해결하기 위해 Dick등의 연구자는 기하정보의 압축을 통하여 비디오 메모리로 전송될 데이터의 용량을 줄이고 그래픽 하드웨어의 렌더링 파이프라인 상에서 압축을 해제하는 방법[8]을 제안하였다. 하지만 이 방법 역시 전처리 과정에 기하정보들을 미리 생성해야하는 문제점이 있기 때문에 효과적인 CLOD 기법이라고 하기 어렵다.

최근 대용량의 지형을 효과적으로 렌더링하기 위한 방법으로 다양한 해상도의 밉맵(mip-map)을 이용하는 Clipmap 기법[9],[10]이 있다. 하지만 이것은 상세단계가 제한된 몇 개의 사각 영역의 보편 공간상에서 선택되기 때문에 CLOD 기법을 이용하는 방법들에 비해 정확한 상세단계 선택이 어려울 뿐만 아니라 밉맵을 사용하기 때문에 멀리 있는 지형은 선명하게 표현할 수 없다.

3. GPU를 이용한 사진트리기반 지형의 시각화 방법

기존의 지형 시각화 방법에서 계층적 자료구조는 CPU에서만 사용할 수 있었다. 본 논문에서는 GPU를 이용하여 사진트리기반의 지형을 시각화 기법을 소개한다. 그림1은 제안한 방법이 GPU에서 처리되는 과정을 보여준다.

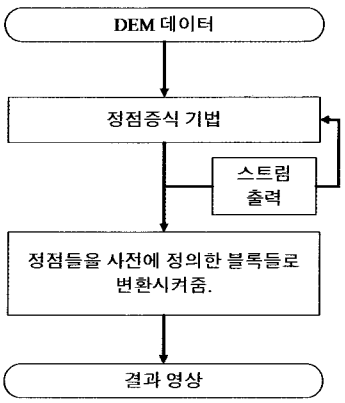

그림1 : 본 논문에서 제안하는 방법의 처리절차

입력으로는 $n \times n$ ($n = 2^{d-1} + 1$, d 는 사진트리의 깊이) 크

기를 갖는 DEM(digital evaluation map) 데이터를 사용한다. 이 데이터는 지형의 높이값이 저장된 고도필드(height field)이다. 입력받은 데이터를 렌더링 하기 위해서 쉐이더 모델 4.0[11]에서 지원하는 기하 쉐이더(geometry shader)와 스트림 출력(stream output)를 이용하여 상세단계 선택을 하면서 정점들을 재귀적으로 증식시킨다. 이 정점들은 블록 메쉬들로 변환 된다. 이 블록 메쉬들을 모아서 지형 메쉬를 렌더링한다.

3.1 거리기반의 상세단계 선택

DEM 데이터는 3차원 위치 정보를 가지고 있기 때문에 정점집합으로 쉽게 변환할 수 있다. 하지만 대용량 지형 데이터는 최신형 그래픽 하드웨어를 사용해도 렌더링 하기 힘들기 때문에 상세단계 선택을 통해 지형을 간략화 하여 렌더링하는 방법을 사용한다. 본 절에서는 사진트리를 이용한 거리기반의 상세단계 선택 방법을 설명한다.

사진트리의 루트노드에는 전체 지형의 크기를 갖는 사각형 블록이 저장된다. 자식노드들은 이 블록을 사분할한 4개의 블록들을 각각 갖는다. 이 과정을 사진트리의 깊이 d만큼 재귀적으로 수행하여 트리를 완성한다.

생성된 사진트리를 이용한 거리기반의 간략화 기법은 (식1)을 이용하여 트리를 하향 탐색 한다.

$$f = \begin{cases} 0 & \text{if } |v,p| < \dfrac{d\tau}{l} \\ 1 & \text{otherwize} \end{cases} \qquad \text{(식1)}$$

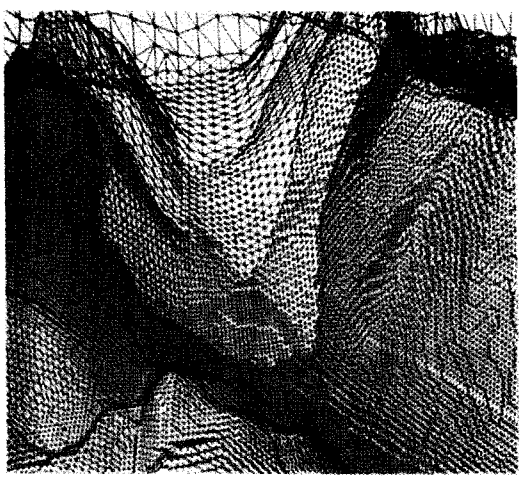

그림2 : 거리기반의 상세단계 선택기법을 적용한 결과 (Puget Sound)

(식1)은 각 노드가 갖는 블록을 둘러싼 경계구 p와 시점 v와의 거리가 트리의 깊이인 d와 사용자가 지정한 문턱값 τ의 곱을 블록의 레벨 l로 나눈 값보다 작다면 트리를 더 탐색하고 그렇지 않으면 탐색을 중단하는 것을 의미한다. f는 탐색여부를 결정하는 플래그 값이다. 이 방법을 통하여 그림2와 같이 시점에 가까운 지형은 상세하게 렌더링하고 멀리 있는 지형은 간략히 표현할 수 있다.

3.2 정점증식기법

정점증식기법은 효과적인 지형의 렌더링을 위해 사진트리 탐색을 GPU에서 할 수 있도록 한다. 정점증식은 기하 쉐이더에서 수행된다(그림3 참조).

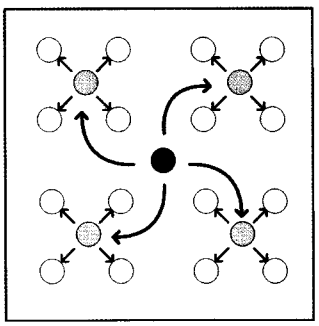

그림3 : 정점증식기법. 검정색 정점이 루트노드이다. 회색은 검정색의 자식노드이고, 흰색은 회색의 자식노드이다.

정점증식기법은 하나의 정점에서 시작한다. 이 정점은 그림3의 검정색 정점으로 사진트리의 루트노드에 해당한다. 이 정점은 루트노드에 대응되는 블록의 중앙에 위치한다. 루트노드부터 하향식 방식으로 트리를 탐색하기 위해 루트 정점은 상세단계 선택 과정을 거쳐 탐색을 해야 할 경우 4개의 회색 정점으로 증식한다. 이 회색 정점들은 검정색 정점을 기준으로 나뉘어진 4개의 블록의 중앙점에 위치한다. 이렇게 생성된 정점들은 스트림 출력을 통해 메모리로 저장되고 다시 렌더링 파이프라인으로 입력이 된다. 재입력된 회색 정점들은 다시 흰색 정점들로 증식한다. 이 과정이 트리의 깊이 d-1만큼 수행되고 사진트리를 이용하여 상세단계 선택을 마친 정점집합이 결과물로 나온다.

이 정점집합들은 시야 바깥의 블록들도 포함하고 있으므로 시각절두체 선별을 통해 시야 안에 있는 정점만을 남긴다. 이 과정은 기하 쉐이더에서 이루어진다. 최종적으로 생성된 정점집합들은 다음절에서 설명할 사전에 정의한 블록 메쉬들로 변환되어 지형 메쉬를 완성한다.

3.3 패치를 이용한 메쉬 생성

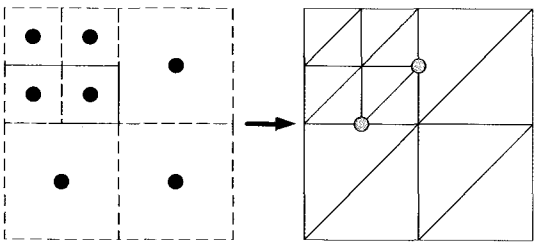

그림4 : 정점을 블록 메쉬로 변환하는 과정. 회색 정점들이 T-정점이다.

서로 다른 레벨을 가진 노드들이 인접해 있는 경우에 그림4와 같이 T-정점들이 생긴다. T-정점들은 크랙 발생의 원인이 된다. T-정점을 제거하기 위해서는 T-정점이 나타나는 변에 정점이 추가된 블록을 생성해 주어야 한다.

본 논문에서 제안하는 렌더링 방법은 거리기반의 상세단계 선택을 하기 때문에 이웃한 블록들 간의 두 레벨 이상 차이나지 않는다. 이것은 블록의 각 변에 하나 이하의 T-정점이 존재하며 하나의 블록에 최대 4개의 T-정점이 생긴다는 것을 의미한다. 따라서 총 16(2^4)개의 경우가 발생하게 되므로 본 논문에서는 그림5와 같이 16가지의 블록들을 미리 정의해 둔 후 필요한 블록 모양을 선택하여 기하 쉐이더에서 변환하는 방법을 택하였다.

GPU상에서 블록의 모양을 선택하기 위해서는 이웃정점의 정보를 알아야 한다. 하지만 기하 쉐이더에서는 이웃정점의 정보를 알 수 없기 때문에 그림6과 같이 가상의 정점을 통해서 이웃정점의 레벨을 계산한다.

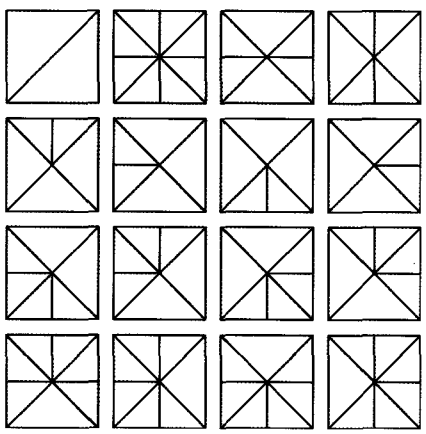

그림5 : 사전에 정의된 16가지 블록들

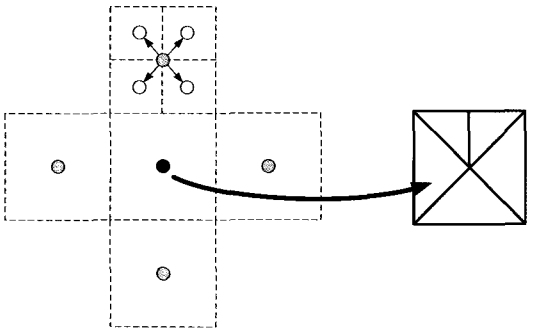

그림6 : GPU에서 이웃노드를 탐색하는 방법. 검은 정점에 대응하는 T-정점 위치는 자신과 동일한 레벨을 가진 4개의 이웃에 대한 상세단계 선택과정을 통해 알 수 있다. 검은 정점 상단의 회색 정점과 같이 더 상위레벨로 탐색이 가능할 경우 해당 위치에 T정점이 생기므로 이것을 제거한 블록을 선택한다.

상의 정점들은 각 노드와 같은 레벨을 가지며 노드의 상하좌우에 위치한다. 이 가상의 정점들은 (식1)을 이용하여 상세단계 선택과정을 거쳐 상위단계로 탐색이 가능한지 검사한다. 가상의 정점이 상위 단계로 탐색이 가능할 경우 해당하는 가상정점의 위치에 T-정점이 생기게 된다. 이렇게 하여 GPU에서 T-정점의 위치를 판별하고 해당 위치에 사전에 정의한 T-정점을 제거할 수 있는 블록을 선택하여 정점을 변환시킨다. 이 블록들을 모으면 지형 메쉬가 되고 최종적으로 픽셀 쉐이더를 거쳐 텍스쳐 맵핑이 된 후 화면에 보여진다.

4. 실험결과

본 실험은 Intel(R) Core™2Duo CPU E8400 3.00GHz에 4GB의 주 메모리를 갖는 시스템에서 수행되었다. 그래픽 카드는 1GB의 메모리를 갖는 NVIDIA GeForce™ 9800GTX(+)를 사용하였고 DirectX 10 라이브러리를 사용하였다. 사용된 데이터는 8비트의 Grand Cannyon (2048×2048)과 Puget Sound(4096×4096)이고, 뷰포트의 크기는 1024×768로 하였다. 문턱값 τ는 투영 행렬의 근평면(near plane)에서 원평면(far plane)까지 거리를 1.0으로 가정하여 표1과 같이 3가지 기준으로 측정하였다. 실제 근평면에서 원평면까지 거리는 지형의 대각선 길이와 같다.

표1 : 기존 CPU기반 사진트리를 이용한 방법[2]에서 문턱값 따른 렌더링 속도(*fps*) 변화 (고정된 시점)

Dataset	τ	*fps*	# of triangles
Puget Sound	0.25	4	190k
	0.5	n/a	640k
	1.0	n/a	1600k
Grand Canyon	0.25	5	170k
	0.5	n/a	490k
	1.0	n/a	1300k

표2 : 본 논문에서 제안하는 방법의 문턱값에 따른 렌더링 속도(fps) 변화 (고정된 시점)

Dataset	τ	*fps*	# of triangles
Puget Sound	0.25	147	190k
	0.5	45	640k
	1.0	9	1600k
Grand Canyon	0.25	153	170k
	0.5	61	490k
	1.0	14	1300k

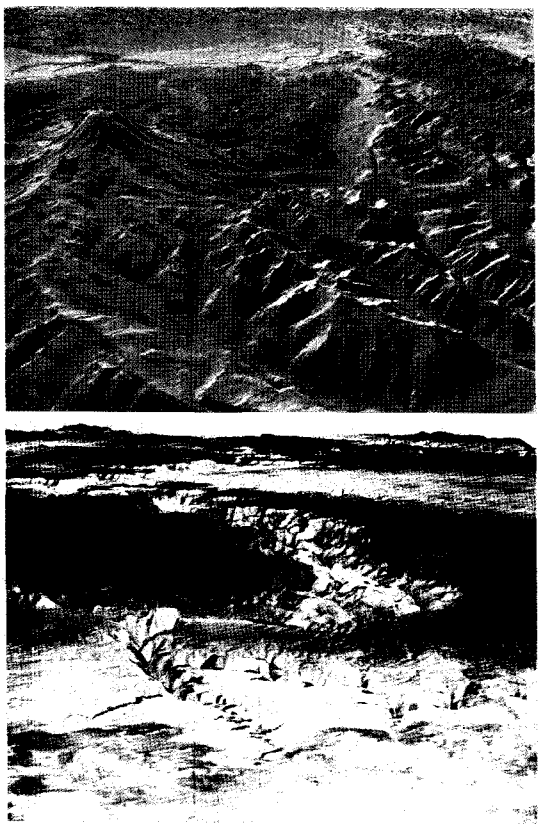

그림7 : 본 논문이 제안하는 방법을 이용한 결과영상. 위쪽은 Puget Sound, 아래쪽은 Grand Canyon의 결과

표1과 표2는 기존의 CPU를 이용한 사진트리 기반의 렌더링 기법과 본 논문에서 제안한 GPU기반 렌더링 기법의 결과를 보여준다. 각 표는 고정된 시점에서 문턱값 τ에 따른 프레임율과 삼각형 개수를 측정한 결과를 보여준다. τ=1.0인 경우에는 상세단계 선택과정은 생략되며 시각절두체 선별만 적용된다. τ값이 0.5, 0.25일 경우에는 상세단계 선택과정을 거치면 렌더링 속도가 향상되고 그려지는 삼각형 개수가 감소하였다. 표1은 기존의 사진트리를 사용한 방법[2]인데, 이 어플리케이션은 사진트리로 메쉬를 생성한 후에 GPU를 이용하여 렌더링 하였다. 라이브러리는 DirectX 9를 사용하였다. 각 방법을 적용한 결과영상은 그림7과 같다.

표3 : dataset에 따른 T-정점에서의 최대 pixel error

Dataset	τ	최대 pixel error	평균 pixel error
Puget Sound	0.25	7	3.68
	0.5	3	0.73
	1.0	0	0
Grand Canyon	0.25	5	2.73
	0.5	2	0.96
	1.0	0	0

τ가 1.0인 경우에는 상세단계 선택과정을 거치지 않으며 변화가 일어나지 않기 때문에 기하 팝핑(geometry popping)이 일어나지 않는다. 하지만 τ가 0.5이거나 0.25인 경우에는 거리기반의 상세단계 선택으로 인하여 지형을 간략화하기 때문에 기하 팝핑이 일어난다. 표3은 T-정점이 생기는 위치와 실제 지형 데이터 값의 화면공간상의 오차를 비교한 값들 중 최대값과 평균값을 측정한 것이다. τ가 0.5인 경우 평균 0.7~0.9픽셀 오차가 발생하며 최대 2~3픽셀 정도의 오차가 발생한다. τ가 0.25일 경우 평균 2.7~3.7픽셀 정도의 오차가 발생하며 최대 5~7픽셀의 오차가 발생한다.

표4 : 쉐이더 스테이지별 이용 빈도

Shader Stage	이용빈도
Vertex Shader	7% ~ 10%
Geometry Shader	51% ~ 64%
Pixel Shader	3% ~ 4%

본 논문에서 제안한 방법과 기존의 사진트리를 이용한 방법을 비교한 결과 기존의 방법에서는 실시간 렌더링이 불가능했지만 본 논문이 제안한 방법에서는 안정적인 속도로 렌더링 할 수 있었다. 이는 지형 메쉬를 생성할 때 기존의 CPU를 사용한 방법에서는 블록의 레벨을 정하기 위해 트리탐색을 순차적으로 하였지만, 본 논문이 제안

한 방법에서는 GPU에서 트리탐색을 병렬로 처리하기 때문이다. 또한 본 논문이 제안한 방법은 GPU만을 이용하므로 기존의 방법에 비해 CPU의 사용을 효과적으로 줄여 CPU에서 추가적인 작업이 가능하게 했다.

표4는 본 논문에서 제안한 방법의 쉐이더에서 스테이지별 사용빈도를 보여준다. 정점 쉐이더는 이웃노드 탐색과 완성된 지형 메쉬에 높이값을 적용하는 작업을 수행하고 기하 쉐이더는 상세단계 선택과 시각절두체 선별 그리고 블록 생성 등의 작업을 수행한다. 그리고 픽셀 쉐이더는 텍스쳐 맵핑 작업을 수행한다. 대부분의 작업들은 총 3개의 패스로 구성된 기하 쉐이더에서 수행된다.

5. 결론

본 논문에서는 사진트리기반 지형을 그래픽 하드웨어만을 이용하여 렌더링 파이프라인 상에서 시각화하는 방법을 소개하였다. 쉐이더 모델 4.0의 기하 쉐이더와 스트림 출력은 각 프리미티브(primitive) 별로 병렬처리가 된다. 따라서 이를 이용한 정점증식기법은 트리 탐색과 지형 메쉬의 생성을 병렬로 처리하기 때문에 기존의 CPU기반의 사진트리에 비해 렌더링 성능이 획기적으로 향상됨을 확인할 수 있었다.

제안한 방법은 하향식 사진트리를 이용하여 in-core 방식으로 지형 렌더링을 수행하였다. 추후 GPU에서 프레임 일관성(Frame Coherency)을 고려한 상향식 사진트리의 사용과 대용량 지형을 위한 out-of-core 방식의 렌더링기법에 대해 연구할 것이다.

감사의 글

이 논문은 2009년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2009-0067404).

참고 문헌

[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 Proc. of ACM SIGGRAPH'96*, 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," *In Proc. 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 Proc. of Visualization''97*, pp.81-88, 1997.

[5] P. Lindstrom and V. Pascucci, "Terrain simplification simplified: A general framework for view-dependent out-of-core visualization," *IEEE Trans. Vis. Comput. Graph.*, Vol.8, No.3, pp.239-254, 2002.

[6] B. Shin and E. Choi, "An Efficient CLOD Method for Large-Scale Terrain Visualization," *In Proc. of The Entertainment Computing - ICEC Lecture Notes in Computer Science*, Vol.3166, 2004.

[7] J. Levenberg, "Fast view-dependent levels-of-detail rendering using cached geometry," *In Proc. of Visualization''02*, pp.259-266, 2002.

[8] C. Dick, J. Schneider, and R. Westermann, "Efficient Geometry Compression for GPU-Based Decoding in Realtime Terrain Rendering," *Computer Graphics Forum*, Vol.28, No.1, pp.67-83, 2009.

[9] C. Tanner, C. Migdal, and M. Jones, "The clipmap: a virtual mipmap," *In Proc. of ACM SIGGRAPH'98*, pp.151-158, 1998.

[10] F. Losasso and H. Hoppe, "Geometry clipmaps: terrain rendering using nested regular grids," *ACM Trans. Graph*, Vol.23, No.3, pp.769-776, 2004.

[11] D. Blythe, "The Direct3D 10 System," *ACM Trans. Graph*, 2006.

〈저자소개〉

이은석

- 2008년 2월 인하대학교 컴퓨터공학부(학사)
- 2008년 ~ 현재 인하대학교 정보공학부 (석사)
- 〈관심분야〉 지형 렌더링, level-of-detail

신병석

- 1990년 2월 서울대학교 컴퓨터공학과(학사)
- 1992년 2월 서울대학교 컴퓨터공학과(석사)
- 1997년 2월 서울대학교 컴퓨터공학과(박사)
- 2000년 ~ 현재 인하대학교 컴퓨터공학부 부교수
- 〈관심분야〉 실시간 렌더링, 볼륨 그래픽스, 의료 영상