2002년 한국정보처리학회 춘계학술발표논문집 제9권 제1호

Occlusion Culling 처리를 위한 1-패스 렌더링 시스템 구조 설계

이은지, 최문희, 박우찬, 김신덕
연세대학교 컴퓨터과학 · 산업시스템공학과
e-mail:ejlee@kurene.yonsei.ac.kr

The Design of 1-pass Rendering Pipeline for Occlusion Culling

Eun-Ji Lee, Moon-Hee Choi, Woo-Chan Park, Shin-Dug Kim
Dept of Computer Science, Yonsei University

Abstract

최근 컴퓨터 그래픽스 분야에서 보다 현실감 나는 영상을 제공하기 위해, 많은 기하로 구성될 뿐만 아니라 깊이 복잡도가 매우 높은 데이터가 요구되어지고 있다. 또한 기하학적으로 늘어나는 데이터를 실시간에 처리해 줄 수 있는 고성능의 렌더링 시스템에 대한 요구도 높아지고 있다. 이에 본 논문에서는 OpenGL 기반에서 occlusion culling을 1-패스에 처리하여 고성능을 보여주는 렌더링 구조를 제안한다. 이는 2-패스의 기존 구조에서 반복적으로 발생되는 불필요한 연산을 효과적으로 제거하여 성능을 높여주고 파워 소모도 최소로 하고 있다. 실험을 통해 제안 구조가 기존 구조에 비해 1.2 ~ 1.5배 성능 향상을 보임을 알 수 있었다.

1. Introduction

최근 3차원 그래픽스에서는 사실적인 영상을 생성하기 위해 보다 정교한 모델링이 요구되고 있다. 이로 인해 게임 및 비행 시뮬레이션과 같은 응용 분야에서는 처리해야 할 3차원 데이터의 양이 급격히 증가하고 있고, 이 데이터를 실시간에 처리할 수 있는 시스템도 필요하게 되었다 [1][2][3]. 현재 3차원 그래픽 처리 기술의 발전 속도는 상당히 성장하고 있으나. 현실감있는 영상의 모델링에 요구되는 데이터의 증가 속도에는 미치지 못하고 있다. 이에 그래픽스 분야에서는 이를 극복하기 위해 visibility culling, level-of-detail rendering과 같은 처리 데이터를 대폭적으로 감소시키는 기술이 연구되어왔다 [2][3]. 이 기법들은 실제 그려지는 영상의 질에 영향을 미치지 않으면서, 처리되는 데이터의 상당량을 제거해주고 있다.

은면 제거로 알려진 visibility culling기법은 컴퓨터 그래픽에 있어서 상당히 중요한 작업이다. 이는 시점과 시점 방향 그리고 보여질 물체가 주어졌을 때, 눈에 보이는 부분만을 효과적으로 판단하여 그려질 영상을 만드는 사용된다 [1]. 이전의 렌더링 시스템에서는 주로 z-버퍼링을 이용하여 보이지 않는 부분을 제거해 주었다. 모든 프레그먼트(fragment)들의 최종 결과값의 생성후에 z-버퍼링을 수행하는 이 방법은 시야(view frustum)에 속하는 모든 데이터, 즉 보여지는 데이터 뿐만 아니라 보이지 않는 데이터까지 모두 처리해 준다. 따라서 상당한 연산이 요구되고, 그만큼의 손실을 가져온다. 그러나 visibility culling 기법을 렌더링 과정에 적용하게 되면, 불필요한 데이터의 대부분을 렌더링 시스템의 상단, 즉 z-버퍼링 이전단계에서 미리 제거하게 되므로 연산 처리양을 줄일 수 있다. 최근에 출시되는 대부분의 고성능 그래픽 그래픽 가속기에 visibility culling 기법을 적용할 정도로 그 효과는 매우 크다 [4][5]. Visibility culling 기법을 대표적으로 연구하는 [6]와 [7]에서는 기존의 OpenGL 기반의 시스템에 visibility culling 중 occlusion culling 알고리즘을 적용하는 구조를 제안하였다. 이 구조는 occlusion mode를 정의하고, 별도의 하드웨어에서 occlusion culling을 수행하도록 한다. 이는

occlusion 모드와 렌더(render) 모드간의 전환을 통해 렌더링을 수행하는 2-패스 구조로서, 연산의 중복 수행으로 인하여 전체적인 성능의 저하 및 파워 손실을 가져오는 문제점이 있다.

본 논문에서는 이러한 문제를 해결하기 위해 occlusion culling 모드와 렌더링 모드간의 전환이 없이 1-패스에 occlusion culling을 수행할 수 있는 구조를 제안한다. 즉, occlusion culling 담당 부분과 렌더링을 담당하는 부분을 분리하여 두 연산이 동시에 수행하도록 한다. 제안 구조는 기존의 구조에 비해 상당히 높은 프레임 처리율을 가지고, 불필요한 연산의 반복을 없애 주었고, 연산에 필요한 파워도 최소화하였다. 실험을 통해, 제안한 구조는 기존 구조와 비슷한 제거율, 즉 전체 기하 중 영상 생성에 필요하지 않는 기하의 약 96.32%를 평균적으로 제거한다. 그리고 제안 구조는 기존 구조에 비해 1.2 ~ 1.5배의 프레임 처리 향상을 보여준다.

본 논문의 구성은 다음과 같다. 2장에서는 occlusion culling과 관련된 연구들에 대해 서술한다. 3장에서는 제안 구조 및 그의 특징에 대해 서술하며 4장에서는 제안 구조의 실험결과를 보여준다. 마지막으로 5장에서는 결론을 맺는다.

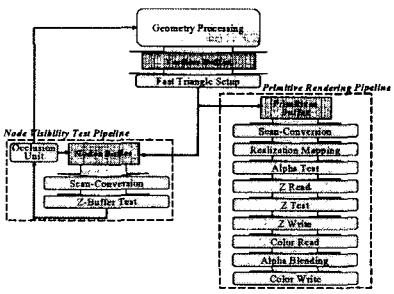
2. Related Work

Occlusion Culling은 1976년 Clark [8]에 의해 처음 사용된 이래로, 컴퓨터 그래픽스 분야에서 상당한 연구가 진행되어 왔다. 초기의 접근 방법들은 장면의 계층적 분할 블록들을 시야(view frustum)에 대해 culling하는 데 기반을 두고 있다 [9]. 이 방법은 간단하고, 근접(close-up)한 장면에 효과적이지만, 시야(view frustum) 내에 있으면서 depth complexity가 높은 장면들에 대해서는 그다지 적합하지 않다.

1993년에 Greene et al.은 hierarchical z-buffer(HZB) algorithm을 제안하였다 [2][10]. 이는 z-pyramid를 통해 객체 공간의 일관성(coherence)을 이용하여 객체의 계층적 구조에 대한 occlusion test를 하였다. 장면을 octree로 분할한 후, 각각의 octant를 시야(view frustum)에 대해 culling한다. 그 후, 남은 octant들이 visible한 지를 체크하기 위해 octant들의 윤곽을 주사 변환하여 프레임 버퍼(frame buffer)로 보낸다. 만약 이들 octant들이 visible하면, octant 내에 있는 모든 객체들 또한 visible한 것으로 간주한다. visible한 객체들에 의해 프레임 버퍼의 변화된 후에, 재 계산에 의해 수정된 z-pyramid를 체크함으로써 visibility query를 수행한다. 그러나 z-pyramid를 이용한 query는 일반적인 그래픽 하드웨어에 의해 지원되지 않는다. 또한 z-pyramid를 유지하는 것은 상당한 비용을 수반하는 연산이다.

Zhang et al.은 hierarchical occlusion map (HOM)을 이용하여 occlusion culling을 하는 방법을 제시하였다 [11]. 이 방법은 장면 데이터베이스로부터 가리개(occluder) database를 생성한다. 이 가리개(occluder)들을 이용하여 장면 database의 잠재적인 피가리개(occludee)들의 bounding box들과 투영된 가리개(occluder)들의 이미지 계층(image hierarchy)과의 overlap을 테스트한다.

HP는 occlusion culling을 위한 OpenGL extension을 제안하였다 [6]. 이는 hierarchical z-buffer 방법과 비슷한 방법으로, 훨씬 더 복잡한 기하로 표현되는 그래픽 프리미티브들을 가시성 결정을 위해 occlusion test mode에서 렌더링한다. 이 결과에 따라 해당 기하는 렌더링되거나 무시된다. Bartz et al.는 OpenGL 시스템에서 occlusion query를 향상시키기 위한 방법을 제시하였다 [7]. 이 방법은 OpenGL 렌더링 시스템의 depth test 단계 뒤에 occlusion culling unit을 삽입하여 occlusion query를 빠르게 처리할 수 있도록 하였다. 그러나 [6]과 [7]은 occlusion culling을 처리하기 위해 occlusion mode와 render mode를 수행하는 2-패스 구조로, 불필요한 OpenGL 시스템 연산 수행으로 인한 전체적인 성능 저하와 processing power의 소모를 가져오는 단점이 있다.



[그림 1] Proposed Rendering System.

3. Proposed Architecture

3.1 Proposed Occlusion Culling 시스템

[그림 1]은 제안된 occlusion culling 시스템이다. 이는 기존의 z-buffer를 기반으로 하는 occlusion

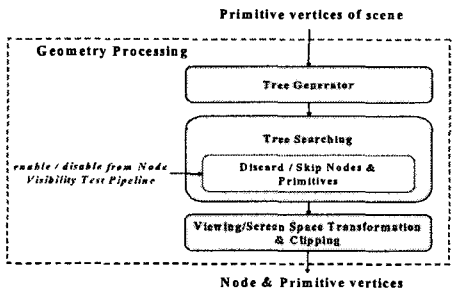
culling 기법들에 쉽게 적용할 수 있는 기법이다. 이는 프레임을 생성하는 과정을 2-패스에 처리하는 기존의 렌더링 파이프라인을 약간의 하드웨어를 추가하여 1-패스에 처리할 수 있도록 하였다.

기하학 처리단계에서는 공간상 근접한 장면의 객체들을 bounding box로 묶어 계층적인 트리를 생성하고, 보여질 가능성이 있는 노드를 탐색하여 다음 처리단계로 보내주는 역할을 한다. Node Visibility Test 시스템에서는 screen-space oriented visibility 정보를 이용하여 occlusion 판단을 위한 정보를 생성한다.

프리미티브 렌더링 시스템에서는 생성된 occlusion 판단 정보를 이용하여 눈에 보이지 않는 영역에 속하는 프리미티브를 판별한다. 보이지 않는 것으로 판별된 프리미티브는 무시하고, 보이는 것으로 판별된 프리미티브는 주사변환을 통하여 프리미티브 내의 픽셀의 값을 생성한다. 또한 Node Visibility Test 시스템과 프리미티브 렌더링 시스템은 병렬적으로 수행된다.

따라서 제안 구조는 2-패스에 occlusion 판단 정보를 생성하는 기존 구조의 문제점인 파이프라인 지연과 불필요한 연산처리로 인한 processing power의 소모를 효과적으로 제거할 수 있다. 또한 fast triangle setup은 주사변환을 위한 빠른 변위값의 생성 및 하드웨어 비용을 줄이기 위해 사용된다.

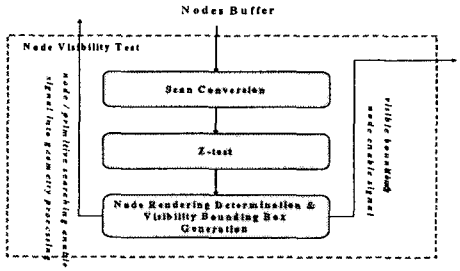
3.2 기하학 처리 단계



[그림 2] 기하학 처리 Flow.

기하학 처리에서는 occlusion culling의 전처리 단계로 [그림 2]와 같이 수행된다. 이 단계는 트리 생성, 트리 탐색, 변환 및 Clipping을 수행한다. 이중 트리 탐색은 가장 중요한 과정으로, Node Visibility Test 시스템으로부터 생성된 node enable 정보를 통해 트리의 보여질 가능성이 있는 노드를 결정하여 다음 단계인 occlusion culling 처리단계로 보내주는 역할을 수행한다.

3.3 Node Visibility Test 단계

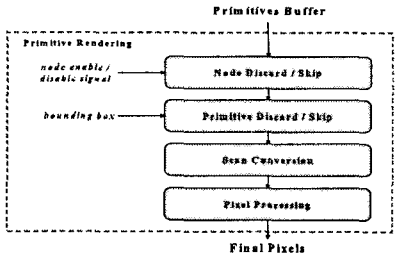


[그림 3] Node Visibility Test Flow.

[그림 3]은 Node Visibility Test의 흐름을 보여준다. 이 단계에서는 스크린-투영된 노드에 대한 주사변환, z-test, 노드 및 프리미티브 렌더링을 위한 정보를 생성하는 역할을 수행한다. 또한 기하학 처리의 노드 결정을 위한 node/프리미티브 searching enable signal을 생성한다.

3.4 프리미티브 Rendering 단계

프리미티브 Rendering 단계는 visible한 노드의 프리미티브를 렌더링하는 역할을 수행하는데, 이의 수행 흐름도는 [그림 4]와 같다.



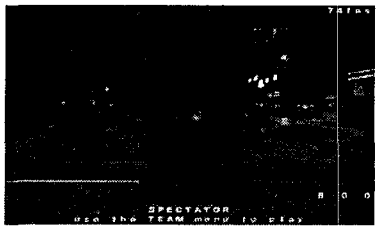
[그림 4] 프리미티브 Rendering Flow.

Node visibility test 단계에서 생성된 node enable/disable signal에 의해 렌더링될 노드를 결정하고, 그 노드의 bounding box를 이용하여 노드내의 프리미티브의 가시성(visibility)을 결정한다. 이 과정을 통과한 프리미티브만을 렌더링하여 최종의 픽셀을 생성한다.

4. Experimental Results

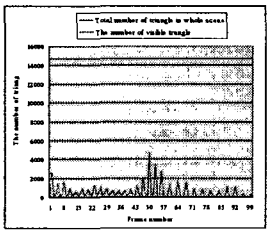
본 논문에서 제안된 알고리즘의 효율성을 검증을 위해 OpenGL graphics API의 Open Source implementation인 Mesa 3.4를 이용하였고, 제안된 구조를 위해 occlusion culling simulator를 제작하였

다.

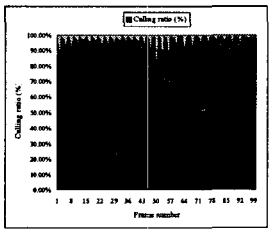


[그림 5] One scene of Quake III.

본 실험에서는 실시간 렌더링 시스템의 시뮬레이션에 많이 사용되는 Quake III Arena를 사용하여 실험하였다. 실제 사용자의 인터렉티브한 조정(control)을 통하여 [그림 5]와 같은 Quake III의 장면들의 생성함으로서 제안 구조의 전체적인 성능을 측정하였다.



[그림 6] 프레임 당 cull된 triangle의 수.



[그림 7] 프레임당 cull된 triangle의 비율.

[그림 6]은 100 프레임에 걸쳐 제거된 프리미티브의 수를 보여준다. 전체적으로 일정한 비율의 culling 비율을 보여주나, 부분적으로 많은 수의 프리미티브를 처리해주는 프레임이 존재하는데, 이는 열린 공간의 경우에 해당한다. 열린 공간은 가려지는 데이터의 비율이 상대적으로 낮음을 알 수 있다. 이를 전체 장면에서의 culling 비율로 표현하면 [그림 7]과 같으며, 전체 프레임에 걸쳐 평균적으로 96.32%의 culling 비율을 갖는다.

5. Conclusion

본 논문에서는 OpenGL 시스템 상에서 occlusion culling을 처리할 수 있는 고성능의 실시간 렌더링 시스템 구조를 제안하였다. 이는 1-패스에 occlusion culling 처리할 수 있는 렌더링 시스템 구조로, 기존 구조의 문제점인 불필요한 연산의 반복 및 processing power의 소모를 효과적으로 줄였으며, 또한 렌더링 처리에서의 지연을 제거하여 성능을 향상시켰다. 결과적으로 본 제안 구조는 전체 모델을 구성하는 기하 중 실제 영상 생성에 영향을 미치지 않는 기하를 평균적으로 96.32% 제거하였으며, 기존의 OpenGL 시스템 -based 렌더링 구조에 비해 1.2 ~ 1.5배 성능이 향상되었다.

References

[1] T. Moller, E. Haines, Real-Time Render ing, A K Peters, 1999.

[2] N. Greene. Hierarchical Rendering of Complex Environments. PHD thesis, University of California at Santa Cruz, June 1995.

[3] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. Computer Graphics: Principles and Practice. Addison-Wesley, second edition, 1996.

[4] T. Pabst. High-Tech And Vertex Juggling NVIDIA's New GeForce 3 GPU. Toms Hardware Guide, February 2001.

[5] S. Morein. Ati radeon hyper-s technology. In presentation at Hot3D Proceedings, part of Graphics Hardware Workshop, 2000. http://www.merl.com/hwws00/presentations/ATIHot3D.pdf

[6] R.Cunniff. Visualize fx graphics scalable architecture. In presentation at Hot3D Proceedings, part of Graphics hardware Workshop, 2000.

[7] D. Bartz, M. Meiner and T. Huttner, Extending Graphics Hardware for Occlusion Queries in OpenGL, 1998 EUROGRPHICS/ SIGGRPH workshop on graphics hardware, ACM

[8] J. H. Clark, "Hierarchical Geometric Models for Visible Surface Algorithms," Comm. ACM. vol. 19, pp. 547-554, Oct. 1976.

[9] Garlick B, Baum D, Winget J. Interactive viewing of large geometric databases using multiprocessor graphics workstations. SIGGRAPH'90 course notes: Parallel Algorithms and Architectures for 3D Image Generation, 1990

[10] N. Greene, M. and G. Miller, Hierarchical Z-buffer visibility, 20th annual conference on Computer Graphics, ACM, 1993

[11] H. Zhang, D. Manocha, T. Hudson and K. E. Hoff, Visibility culling using hierarchical occlusion maps. In SIGGRAPH '97 Conference Proceedings, pages 77-88, Los Angeles, California, Aug. 1997. ACM Computer Graphics. vol. 31, no. 4.