(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2008년11월03일
(11) 등록번호 10-0866573
(24) 등록일자 2008년10월28일

(51) Int. Cl.
G06T 15/00 (2006.01)
(21) 출원번호 10-2008-0016233
(22) 출원일자 2008년02월22일
심사청구일자 2008년02월22일
(56) 선행기술조사문헌
서모영, 정우남, 한탁돈, '깊이 기반 3차원 영상 렌더링에서 투명한 스플랫을 사용한 가시성 기법', 한국정보과학회: 학술대회지 , 한국정보과학회 05 추계 학술발표논문집(1) , pp.706-708 , 2005
KR1020010085426
전체 청구항 수 : 총 6 항

(73) 특허권자
인하대학교 산학협력단
인천 남구 용현동 253 인하대학교
(72) 발명자
신병석
경기 안양시 동안구 갈산동 샘마을한양아파트 114동 402
이진희
인천 강화군 강화읍 관청리 947-8 태영빌라 B동 101호
(74) 대리인
특허법인태동

심사관 : 조우연

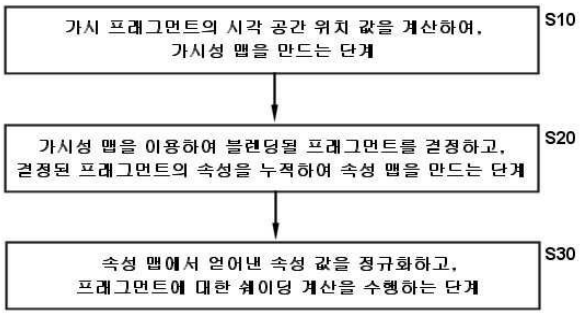
(54) 가시성 맵을 이용한 점-기반 렌더링 방법

(57) 요 약

본 발명은 점-기반 렌더링 방법에 관한 것으로서, (1) 가시 프래그먼트(visible fragment)의 시각-공간 위치 값을 계산하여 가시성 맵(visibility map)을 만드는 단계, (2) 상기 단계 (1)에서 만들어진 상기 가시성 맵을 이용하여 블렌딩될 프래그먼트를 결정하고, 결정된 상기 프래그먼트의 속성을 누적하여 속성 맵(attribute map)을 만드는 단계, 및 (3) 상기 단계 (2)에서 만들어진 상기 속성 맵에서 얻어낸 속성 값을 정규화하고 프래그먼트에 대한 쉐이딩(shading) 계산을 수행하는 단계를 포함하는 것을 그 구성상의 특징으로 한다.

본 발명의 점-기반 렌더링 방법에 따르면, 깊이 값 기반의 가시성 계산 방법 대신에 거리 값 기반의 가시성 계산 방법을 사용하고, 평균 위치 맵과 적응형 프래그먼트 처리 기법을 적용함으로써 화질 개선과 렌더링 성능 향상을 동시에 가능하게 한다.

대 표 도 - 도1



특허청구의 범위

청구항 1

점-기반 렌더링(point-based rendering) 방법으로서,

(1) 가시 프래그먼트(visible fragment)의 시각-공간 위치 값을 계산하여 가시성 맵(visibility map)을 만드는 단계;

(2) 상기 단계 (1)에서 만들어진 상기 가시성 맵을 이용하여 블렌딩될 프래그먼트를 결정하고, 결정된 상기 프래그먼트의 속성을 누적하여 속성 맵(attribute map)을 만드는 단계; 및

(3) 상기 단계 (2)에서 만들어진 상기 속성 맵에서 얻어낸 속성 값을 정규화하고 프래그먼트에 대한 쉐이딩(shading) 계산을 수행하는 단계

를 포함하는 방법.

청구항 2

제1항에 있어서,

상기 단계 (2)에서, 프래그먼트의 거리 값을 이용하여 블렌딩될 프래그먼트를 결정하는 것을 특징으로 하는 방법.

청구항 3

제2항에 있어서,

상기 가시성 맵으로부터 샘플링한 위치와 시각-공간(view-space)에서 계산된 프래그먼트 위치 사이의 거리가 미리 지정된 임계값보다 클 경우에는 클리핑(clipping) 하는 것을 특징으로 하는 방법.

청구항 4

제1항에 있어서,

상기 단계 (2)에서, 상기 프래그먼트의 속성으로서 상기 프래그먼트의 평균 위치 값(averaged position value)을 이용하는 것을 특징으로 하는 방법.

청구항 5

제4항에 있어서,

상기 프래그먼트의 상기 평균 위치 값은, 하나의 픽셀로 블렌딩되는 여러 프래그먼트들의 시각-공간 위치에 대한 가중 평균 값으로 정의되는 것을 특징으로 하는 방법.

청구항 6

제1항에 있어서,

상기 단계 (1)에서, 상기 프래그먼트를 형성하는 스플랫(splat)의 투영된 크기가 픽셀 크기보다 작은 경우, 해당 프래그먼트의 시각-공간 위치 값을 계산하는 대신에 상기 스플랫의 중심점을 해당 프래그먼트의 시간 공간 위치 값으로 사용하는 것을 특징으로 하는 방법.

청구항 7

삭제

명 세 서

발명의 상세한 설명

기 술 분 야

<1> 본 발명은 점-기반 렌더링 방법에 관한 것으로서, 가시 프래그먼트(visible fragment)의 시각-공간 위치 값을 계산하여 가시성 맵(visibility map)을 만들고, 만들어진 가시성 맵을 이용하여 블렌딩될 프래그먼트를 결정하고, 결정된 프래그먼트의 속성을 누적하여 속성 맵(attribute map)을 만든 후, 만들어진 속성 맵에서 얻어낸 속성 값을 정규화하고 프래그먼트에 대한 쉐이딩(shading) 계산을 수행함으로써, 화질을 개선하면서 동시에 렌더링 성능을 향상시킬 수 있는 점-기반 렌더링 방법에 관한 것이다. 다시 말해, 본 발명은 깊이 값 기반의 가시성 계산 방법 대신에 거리 값 기반의 가시성 계산 방법을 사용하고, 평균 위치 맵(averaged position map)과 적응형 프래그먼트 처리 기법을 적용함으로써 화질 개선과 렌더링 성능 향상을 동시에 가능하게 하는 점-기반 렌더링 방법에 관한 것이다.

배 경 기 술

<2> 3차원 디지털화 기술이 발전하면서 디지털 미켈란젤로 프로젝트의 "다비드 상(the Statue of David)"와 같이 실세계에서의 모형들이 시각화 (visualization)의 중요한 대상으로 떠오르고 있다. 그런데 이러한 모델들은 보통 표면을 매우 정밀하게 표현하기 위해서 엄청난 수의 샘플 점을 가지고 있기 때문에 전통적인 다각형기반의 렌더링 방법으로는 시각화하는 것이 거의 불가능하다. 이에 따라, Levoy와 Whitted가 제안한 점-기반 렌더링 방법이 이러한 모델들을 시각화 할 수 있는 적절한 대안으로 제시되고 있다. 점-기반 렌더링에서는 다각형을 이용하지 않고, 또한 점들 간의 연결성 정보(topology)도 사용하지 않는다. 그럼에도, 많은 연구를 통해 점-기반 렌더링 방법을 이용하면 고속의 렌더링을 하면서 다각형-기반 렌더링 수준의 화질을 얻을 수 있음이 입증되었다.

<3> 대화식 시각화를 위해서 여러 가지 상세 단계(Level Of Detail; LOD) 기술들이 개발되었다. 또한 타원형 가중평균(Elliptical Weighted Average; EWA) 필터와 같은 정교한 필터링 기법이 고안되어서 앨리어싱(aliasing)이 발생하지 않는 고화질 영상을 만들어낼 수 있게 되었다. 이와 더불어, 프로그래밍 가능한 파이프라인 기술의 발전으로 그래픽스 하드웨어를 활용한 여러 가지 가속화 기법들이 소개되었다.

<4> Botsch등은 본래 다각형-기반 렌더링을 위하여 고안된 지연 쉐이딩(deferred shading) 기법을 점-렌더링에 적용하였다. 지연 쉐이딩의 기본 아이디어는 비용이 많이 소요되는 쉐이딩 계산을 가능한 렌더링 프로세스의 마지막 부분으로 이동시킴으로써 최종 영상의 각 픽셀 당 한 번의 쉐이딩 계산만 하도록 하는 것이다. 지연 쉐이딩의 또 다른 장점은, 래스터화(rasterization)와 쉐이딩 단계를 확실하게 구분해줄 수 있다는 것이다. 비록 이 방법을 사용하면 상당한 화질 개선 효과가 생기지만 좀 더 개선할 여지가 많다. 주로 사용되는 깊이 값 조정(depth offsetting) 방법을 이용하는 깊이 값 기반 가시성 계산 기법은 가끔 프래그먼트들을 오버-블렌딩(over-blending)하는 문제점을 야기한다. 더욱이 프래그먼트들의 깊이 값을 이용하기 때문에 쉐이딩 계산이 잘못될 가능성도 있다.

발명의 내용

해결 하고자하는 과제

<5> 본 발명은 기존에 제안된 방법들의 상기와 같은 문제점들을 해결하기 위해 제안된 것으로서, 깊이 값 기반의 가시성 계산 방법 대신에 거리 값 기반의 가시성 계산 방법을 사용하고, 평균 위치 맵(averaged position map)과 적응형 프래그먼트 처리 기법을 적용함으로써 화질 개선과 렌더링 성능 향상을 동시에 가능하게 하는 점-기반 렌더링 방법을 제공하는 것을 그 목적으로 한다.

과제 해결수단

<6> 상기한 목적을 달성하기 위한 본 발명의 특징에 따른 점-기반 렌더링(point-based rendering) 방법은,

<7> (1) 가시 프래그먼트(visible fragment)의 시각-공간 위치 값을 계산하여 가시성 맵(visibility map)을 만드는 단계;

<8> (2) 상기 단계 (1)에서 만들어진 상기 가시성 맵을 이용하여 블렌딩될 프래그먼트를 결정하고, 결정된 상기 프래그먼트의 속성을 누적하여 속성 맵(attribute map)을 만드는 단계; 및

<9> (3) 상기 단계 (2)에서 만들어진 상기 속성 맵에서 얻어낸 속성 값을 정규화하고 프래그먼트에 대한 쉐이딩(shading) 계산을 수행하는 단계를 포함하는 것을 그 구성상의 특징으로 한다.

<10> 바람직하게는, 상기 단계 (2)에서, 프래그먼트의 거리 값을 이용하여 블렌딩될 프래그먼트를 결정할 수 있다.

<11> 더욱 바람직하게는, 상기 가시성 맵으로부터 샘플링한 위치와 시각-공간(view-space)에서 계산된 프래그먼트 위치 사이의 거리가 미리 지정된 임계값보다 클 경우에는 클리핑(clipping)할 수 있다.

<12> 바람직하게는, 상기 단계 (2)에서, 상기 프래그먼트의 속성으로서 상기 프래그먼트의 평균 위치 값(averaged position value)을 이용할 수 있다.

<13> 더욱 바람직하게는, 상기 프래그먼트의 상기 평균 위치 값은, 하나의 픽셀로 블렌딩되는 여러 프래그먼트들의 시각-공간 위치에 대한 가중 평균 값으로 정의될 수 있다.

<14> 바람직하게는, 상기 단계 (1)에서, 상기 프래그먼트를 형성하는 스플랫(splat)의 투영된 크기가 픽셀 크기보다 작은 경우, 해당 프래그먼트의 시각-공간 위치 값을 계산하는 대신에 상기 스플랫의 중심점을 해당 프래그먼트의 시간 공간 위치 값으로 사용할 수 있다.

<15> 바람직하게는, 상기 단계 (2)에서, 적응형 프래그먼트 처리(adaptive fragment processing) 기법을 적용할 수 있다.

효 과

<16> 본 발명의 점-기반 렌더링 방법에 따르면, 가시 프래그먼트의 시각-공간 위치 값을 계산하여 가시성 맵을 만들고, 만들어진 가시성 맵을 이용하여 블렌딩될 프래그먼트를 결정하고, 결정된 프래그먼트의 속성을 누적하여 속성 맵을 만든 후, 만들어진 속성 맵에서 얻어낸 속성 값을 정규화하고 프래그먼트에 대한 쉐이딩 계산을 수행함으로써, 화질을 개선하면서 동시에 렌더링 성능을 향상시킬 수 있다. 특히, 본 발명의 점-기반 렌더링 방법에 따르면, 깊이 값 기반의 가시성 계산 방법 대신에 거리 값 기반의 가시성 계산 방법을 사용하고, 평균 위치 맵과 적응형 프래그먼트 처리 기법을 적용함으로써 화질 개선과 렌더링 성능 향상을 동시에 가능하게 한다.

발명의 실시를 위한 구체적인 내용

<17> 이하에서는 첨부된 도면들을 참조하여, 본 발명에 따른 실시예에 대하여 상세하게 설명하기로 한다.

<18> 도 1은 본 발명의 일 실시예에 따른, 점-기반 렌더링 방법의 구성을 나타내는 도면이다. 도 1에 도시된 바와 같이, 본 발명의 일 실시예에 따른 점-기반 렌더링 방법은, (1) 가시 프래그먼트의 시각-공간 위치 값을 계산하여 가시성 맵을 만드는 단계(S10), (2) 단계 S10에서 만들어진 가시성 맵을 이용하여 블렌딩될 프래그먼트를 결정하고, 결정된 프래그먼트의 속성을 누적하여 속성 맵을 만드는 단계(S20), 및 (3) 상기 단계 S20에서 만들어진 속성 맵에서 얻어낸 속성 값을 정규화하고 프래그먼트에 대한 쉐이딩 계산을 수행하는 단계(S30)를 포함한다.

<19> 단계 S10은, 가시 프래그먼트의 시각-공간 위치 값을 계산하여 가시성 맵을 만드는 역할을 한다. 여기서, 가시 프래그먼트는 시점을 기준으로 가장 앞쪽에 있는 프래그먼트를 의미한다. 상기 단계 S10에서, 프래그먼트를 형성하는 스플랫(splat)의 투영된 크기가 픽셀 크기보다 작은 경우, 해당 프래그먼트의 시각-공간 위치 값을 계산하는 대신에 스플랫의 중심점을 해당 프래그먼트의 시간 공간 위치 값으로 사용할 수 있다.

<20> 단계 S20은, 단계 S10에서 만들어진 가시성 맵을 이용하여 블렌딩될 프래그먼트를 결정하고, 결정된 프래그먼트의 속성을 누적하여 속성 맵을 만드는 역할을 한다. 상기 단계 S20에서, 프래그먼트의 (기존의 깊이 값 대신에)거리 값을 이용하여 블렌딩될 프래그먼트를 결정할 수 있으며, 가시성 맵으로부터 샘플링한 위치와 시각-공간(view-space)에서 계산된 프래그먼트 위치 사이의 거리가 미리 지정된 임계값보다 클 경우에는 클리핑(clipping) 할 수 있다. 또한, 상기 단계 S20에서, 프래그먼트의 속성으로서 프래그먼트의 평균 위치 값(averaged position value)을 이용할 수 있으며, 여기서 프래그먼트의 평균 위치 값은, 하나의 픽셀로 블렌딩되는 여러 프래그먼트들의 시각-공간 위치에 대한 가중 평균 값으로 정의될 수 있다. 또한, 상기 단계 S20에서, 적응형 프래그먼트 처리(adaptive fragment processing) 기법을 적용할 수 있다.

<21> 단계 S30은, 단계 S20에서 만들어진 속성 맵에서 얻어낸 속성 값을 정규화하고 프래그먼트에 대한 쉐이딩 계산을 수행하는 역할을 한다.

<22> 이하에서는, 본 발명의 일 실시예로서, 상기 단계 S10 내지 S30과 관련하여, 래스터화, 렌더링, 최적화, 실험 결과로 나누어 보다 구체적으로 살펴보기로 한다.

<23> 1. 래스터화(rasterization)

<24> 먼저 래스터화와 관련하여, 본 발명의 특징에 따른 가시성 맵과 평균 위치 맵에 대하여 설명한 후, 나아가 스플랫에 대한 타원형 래스터화 연산과 거리 검사(distance test)에 대해서 설명하기로 한다.

<25> (1) 스플랫 래스터화(splat rasterization)

<26> 기존의 연구에서 적용한 것과 유사한 래스터화 연산을 적용한다. 각각의 스플랫은 화상 평면에 투영되어 정사각형을 이루는 일련의 프래그먼트들을 생성한다(도 2 참조). 도 2에서, 스플랫은 근접 평면(near plane)으로 투영되며, 각각의 프래그먼트에 대한 시각-공간 위치 p_k가 근접 평면 상의 프래그먼트의 위치 p'_k를 이용하여 계산된다. p_k와 스플랫의 중심 위치 사이의 거리를 테스트함으로써, 타원형 스플랫이 만들어진다. 각각의 프래그먼트에 대하여 시각-공간 위치 p_k를 계산하기 위해서 p'_k를 지나는 관측 광선이 스플랫의 서포팅 평면(supporting plane)과 교차하는 점을 구한다. 만약 프래그먼트가 스플랫의 외접 사각형 밖에 위치하면 그 프래그먼트는 무시된다. 프래그먼트의 시각-공간 위치 계산도 기존의 다른 연구에서 한 것과 유사한 방식으로 한다. 타원을 구성한 후 특정 픽셀에 대한 각 프래그먼트의 기여도는 p_k와 스플랫 중심점간의 거리로 추정할 수 있다. 추정된 기여도는 쉐이딩 단계에서 가중 평균 속성을 계산할 때 사용된다.

<27> 가시성 맵 생성 단계(S10)에서는, 가시성 맵에 가시 프래그먼트의 거리 값 대신에 시각-공간에서의 위치 값을 저장한다. 이러한 위치 값들은 두 번째 단계(S20)에서 샘플링되어 블렌딩할 프래그먼트를 결정하는데 사용된다. 가시성 맵은 스칼라 깊이 값 대신에 (x, y, z) 좌표 값을 저장하기 때문에 깊이 버퍼 크기의 3배에 해당하는 기억공간을 필요로 한다. 예를 들어, 화상의 해상도가 1024 * 1024인 32bit 깊이 버퍼를 사용할 경우 4MB의 텍스처 메모리가 필요한 반면, 가시성 맵은 12MB의 기억공간을 필요로 한다. 그러나 최근에 나오는 일반적인 그래픽 가속기들은 충분한 텍스처 메모리를 가지고 있으므로 문제될 것이 없다. 이러한 기억공간의 부담에도 불구하고 가시성 맵은 깊이 버퍼에 비해서 여러 가지 탁월한 장점들을 가지고 있다.

<28> (2) 거리-기반 블렌딩(distance-based blending)

<29> 속성 계산 단계(S20)에서는, 블렌딩될 프래그먼트들을 결정하고 이것을 속성 맵에 누적해서 마지막 쉐이딩 단계(S30)에서 사용될 수 있도록 해야 한다. 기존의 방법들에서는 이러한 분류(블렌딩될 프래그먼트들의 결정)를 위해서 오프셋 깊이 검사 방법을 이용하였다. 그러나 기존의 오프셋 깊이 검사 방법을 이용할 경우, 관측 광선의 각도에 따라서 같은 깊이 구간이 시각-공간상에서 서로 다른 거리 값을 가질 수 있다는 문제점이 있다. 이것은, 깊이 값 검사를 매우 복잡하게 만들고 이에 따라 여러 가지 문제를 야기할 수 있다. 더구나 이러한 깊이 오프세팅을 잘못하면 최종 영상에서 잘못 가려지는 부분이 만들어질 수도 있다. 이와 같이 기존의 오프셋 깊이 방법을 이용하여 블렌딩될 프래그먼트들을 결정하는 경우의 문제점은 도 3을 참조하면 분명하게 확인할 수 있다. 즉, 도 3에 따르면, 프래그먼트들 간의 거리가 2개의 광선 r_1과 r_2에 따라 달라지지만(d_1과 d_2로 표시함), 깊이 간격(z로 표시함)은 두 쌍에 대하여 동일함을 확인할 수 있다.

<30> 상기와 같은 문제점에 대한 인식으로부터, 본 발명에서는 깊이 검사 대신에 단계 S10에서 만들어진 가시성 맵을 사용하여 거리 검사를 한다. 입력되는 프래그먼트는 거리 값을 이용하여 블렌딩될 프래그먼트와 제외될 프래그먼트로 분류된다. 만약 가시성 맵으로부터 샘플링한 위치와 시각-공간에서 계산된 프래그먼트 위치 사이의 거리가 미리 정의된 임계값보다 클 때는 클리핑 할 수 있다.

<31> (3) 평균 위치 맵(averaged position map)

<32> 기존의 방법에서는 쉐이딩 단계(S30)에서 필요한 시각-공간 위치를 최전방 프래그먼트(예컨대, 도 4의 f_1)의 깊이 값으로부터 얻어왔다. 그러나 여러 개의 프래그먼트들이 하나의 픽셀로 블렌딩될 때 최전방 프래그먼트와 평균 프래그먼트 사이의 차이가 커질 수도 있는데, 이와 같은 경우 쉐이딩 단계에서의 쉐이딩 계산이 잘못될 수 있다.

<33> 깊이 값으로부터 직접 얻어지는 시각-공간상의 위치를 이용하는 경우의 또 다른 문제점은 여러 프래그먼트들의 깊이 값이 아주 조금 차이가 날 경우 소위 말하는 z-경합(z-fighting) 문제가 생길 수도 있다는 것이다. 예를 들어, 도 4에서 f1의 깊이 값이 f2의 깊이 값과 거의 유사할 경우 시각-공간 위치는 임의로 결정될 수 있다.

<34> 이와 같은 깊이 값과 관련된 문제를 해결하기 위해서, 본 발명에서는 새로운 형태의 속성인 "평균 위치 값"을 추가하여 사용한다. 여기서, 평균 위치 값이란 하나의 픽셀로 블렌딩되는 여러 프래그먼트들의 시각-공간 위치

에 대한 가중 평균으로 정의할 수 있다. 평균 위치 맵은 이와 같은 평균 위치 값에 대응하는 일종의 속성 맵으로서 가시성 맵과는 다른 것이다. 다른 속성들과 마찬가지로 블렌딩될 프래그먼트들의 시각-공간 위치는 평균 위치 맵에 누적되며, 다음 쉐이딩 단계(S30)에서 사용된다.

<35> 2. 렌더링(rendering)

<36> 본 발명의 일 실시예에서는 다중-경로 렌더링(multi-pass rendering)을 채택하고 있는바, 다중-경로 렌더링에 대하여 자세히 설명하기로 한다.

<37> 본 발명의 특징에 따른 다중-경로 렌더링은, 가시성 계산 경로(단계), 속성 계산 경로(단계), 쉐이딩 경로(단계)로 이루어진다. 가시성 계산 경로에서는, 가시 프래그먼트(visible fragment; 시점을 기준으로 가장 앞쪽에 있는 프래그먼트)의 시각-공간 위치 값을 갱신한다. 속성 계산 경로에서는 가시성 계산 경로에서 만들어진 가시성 맵을 이용하여 블랜딩될 프래그먼트를 결정하고 그들의 속성을 누적하여 속성 맵을 만든다. 마지막 쉐이딩 경로에서는 속성 맵에서 얻어낸 속성 값을 정규화하고 프래그먼트에 대한 쉐이딩 계산을 수행한다. 이와 같이 다중-경로 렌더링을 이용할 경우, 깊이 값을 이용하는 경우에 나타나는 문제점과 쉐이딩이 잘못되는 문제점을 해결할 수 있다.

<38> 다중-경로 렌더링의 가시성 계산 경로 및 속성 계산 경로에서는 렌더링 타깃(render target)을 텍스처 메모리에 텍스처 맵으로 저장하고, 저장한 맵들을 쉐이딩 경로의 입력 텍스처로 이용한다.

<39> 가시성 계산 경로(단계)에서는 모든 스플랫들이 화상 평면상에 래스터화 된다. 생성된 프래그먼트들의 시각-공간 위치 값들은 가시성 맵에 순차적으로 저장되는데, z-버퍼가 활성화되어 있으므로 가시성 맵에는 각 단계의 마지막에서 최전방에 있는 프래그먼트(가시 프래그먼트)의 시각-공간 위치 값이 저장된다. 각 프래그먼트의 깊이 값은 대응하는 시각-공간 위치 값에 따라 보정된다. 이 단계에서 알파 블렌딩(alpha blending)은 꺼지고 z-버퍼 기능은 활성화된다. 정점 쉐이더 프로그램(vertex shader program)에서 각 스플랫들의 법선 벡터 n, 중심 위치 c, 반지름 r에 대한 값들이 픽셀 쉐이더 프로그램으로 전달된다.

<40> 속성 계산 경로(단계)에서 스플랫들은 다시 한 번 래스터화 되는데, 이때 가시성 계산 경로에서 만들어진 가시성 맵을 사용하여 프래그먼트들의 속성을 대응하는 속성 맵들에 저장한다. 예를 들어, 법선 벡터는 법선 맵에, 위치 값은 평균 위치 맵에, 컬러 값은 컬러 맵에 저장된다. 깊이 검사 대신에 거리 검사를 하기 때문에 z-버퍼 기능은 꺼야 한다. 화상 평면의 각 픽셀에 대하여 픽셀에 대한 스플랫들의 기여도는 가우시안 함수를 이용하여 계산된다. 이 기여도 값을 렌더링 타깃의 알파 채널에 저장함으로써 각 픽셀에 대한 속성 값들의 가중 평균을 구할 수 있다. 컬러 채널에 대해서는 다음 수학식 1을, 알파 채널에 대해서는 다음 수학식 2를 적용하여 합산형 알파 블렌딩(additive alpha blending)을 수행한다. 수학식 1 및 2는 순서에 무관하므로 누적된 속성 값과 가중치를 안전하게 계산할 수 있다.

수학식 1

<41> $$C_{out}^{new} = \alpha_i n \cdot C_i n + C_{out}^{old}$$

수학식 2

<42> $$\alpha_{out}^{new} = \alpha_i n + \alpha_{out}^{old}$$

<43> 쉐이딩 경로(단계)에서는 목표 영상 크기의 사각형을 GPU로 보내주고 픽셀 쉐이더 프로그램에서 모든 픽셀들을 처리한다. 각 픽셀에 대하여, 먼저 속성 계산 경로에서 생성된 속성 값들을 샘플링하고 그것을 알파 값(가중합)으로 나눠 줌으로써 정규화한다. 이러한 속성들을 이용하여, Phong 조명 처리와 자기 그림자를 적용할 수 있다. 쉐이딩 계산은 한 픽셀 당 한 번씩만 수행하기 때문에 계산 시간을 줄일 수 있다.

<44> 3. 최적화(optimization)

<45> 스플랫의 투영된 크기가 픽셀의 크기보다 작은 경우에는, 픽셀 쉐이더 프로그램에서 상당수의 계산을 줄일 수 있다. 즉, 도 5에 도시된 바와 같이, 스플랫의 투영된 크기가 픽셀 하나의 크기보다 작을 경우, 프래그먼트의 시각-공간 위치에 대한 계산 및 타원형 스플랫 래스터화에 대한 계산이 모두 생략될 수 있다.

<46> 이와 같이 작은 스플랫들은 타원형을 가질 수 없으므로 타원형 스플랫의 래스터화 작업이나 (EWA 필터링에서 사용되는) 타원 가중치를 적용하지 않아도 무방하다. 또한 앞서 언급한 프래그먼트의 시각-공간 위치 계산도 생

략된다. 즉, 스플랫의 중심점을 해당 프래그먼트의 시각-공간 위치 값으로 사용하면 된다. 이렇게 단순화시키면 속도도 빨라지고 뷰포트 매핑 시에 생길 수 있는 오류도 줄어든다. 작은 스플랫의 경우 뷰포트 위치가 다운 샘플링 되기 때문에 계산된 시각-공간 위치가 잘못될 수 있다. 도 6에 이와 같은 적응형 프래그먼트 처리 기법의 일 예를 보이고 있다.

<47> 4. 실험 결과

<48> 대화식 처리 속도를 얻어내기 위하여 CPU 상에서 LOD를 수행하였다. 입력 점에 대한 집합으로부터 bounding volume hierarchy를 생성한다. 렌더링 과정에서 각 점의 투영된 스플랫 크기에 따라서 모델의 LOD 단계를 정한다. 메모리 전송 부하를 줄이기 위하여 후면 선별(back face culling)과 시가 절두체 선별 기법을 적용하여 불필요한 점들은 미리 제거한다.

<49> 비교 분석을 위하여 다음과 같은 3가지 서로 다른 버전의 알고리즘을 구현하였다. 3가지 버전은, 각각 깊이 검사를 하는 지연 쉐이딩(DSD) 버전, 거리 검사를 하는 지연 쉐이딩(DSV) 버전, 거리 검사와 적응형 프래그먼트 처리를 하는 지연 쉐이딩(ADS) 버전이다. 상기 3가지 버전을 2.0GB의 시스템 메모리와 NVidia GeForce 6800 그래픽스 하드웨어가 장착된 컴퓨터상에서 구현하였다. 또한 Microsoft사의 PIX(DirectX용 성능 분석) 도구를 이용하여 실험 결과를 측정하고 분석하였다.

<50> 쉐이딩 단계(S30)에서, 단순화된 자기 그림자 처리 기능을 가진 Phong 조명 처리 모델을 적용하였다. 이것은 27번의 산술 연산을 필요로 한다. 화상의 해상도는 640 * 480이고, 64bit 부동 소수점수의 렌더링 타깃을 사용하며, 3개의 속성 맵(위치 맵, 법선 맵, 컬러 맵)은 도합 7.5MB의 텍스처 메모리를 사용한다. 가시성 맵은 2.5MB의 추가적인 텍스처 메모리를 필요로 하는데 이것은 최근에 소개된 그래픽 하드웨어에서는 큰 부담이 되지 않는다.

<51> 제안한 방법에서 가시성 맵의 효율성은 많은 프래그먼트들이 하나의 픽셀로 블렌딩될 때 가장 높아진다. 그러므로 우리는 각 픽셀에 대하여 겹치는 프래그먼트의 수(오버랩 수)를 측정하였다. 도 7은 본 발명에서 제안한 방법을 적용하여 Lucy 모델을 렌더링한 결과를 나타내는 도면이다. 도 7에서, 밝은 회색 부분은 1-5, 회색 부분은 6-10, 어두운 회색 부분은 11-15, 검은색 부분은 16 이상의 오버랩 수를 각각 나타낸다. 도 7에서 좌측 영상에 대한 평균 오버랩 수는 8.70이고, 우측 영상에 대한 평균 오버랩 수는 10.19이다. 도면으로부터 확인할 수 있는 바와 같이, 모델을 시점에서부터 멀리 이동시킬수록 평균적으로 겹치는 프래그먼트의 수(오버랩 수)는 증가한다.

<52> 다양한 모델들에 대하여 겹치는 프래그먼트들의 평균값이 다음 표 1에 나타나 있다.

표 1

<53>

모델	픽셀 수	프래그먼트 수	평균 오버랩 수
Bunny	110,659	435,166	3.93
Lion	72,276	280,328	3.88
Buddha	68,399	298,064	4.36
Dragon	109,993	464,106	4.22
Lucy	54,665	475,394	8.70

<54> 상기 결과로부터, 본 발명에서 제안하고 있는 평균 위치 맵 이용 방법이 기존 방법에 비하여 좋은 화질의 영상을 만들어 낸다는 사실을 분명하게 확인할 수 있다. 마지막 쉐이딩 단계에서 각 픽셀로 블렌딩되는 프래그먼트들의 평균 위치를 사용함으로써 좀 더 정확한 쉐이딩 계산과 앨리어싱이 제거된 좋은 화질의 영상을 얻을 수 있다.

<55> 깊이 값 기반의 가시성 검사와 거리 값 기반의 가시성 검사에 대한 보다 자세한 비교를 위하여 각 프래그먼트의 깊이 값과 거리 값의 수치적 차이를 측정하였다. 각 프래그먼트 f와 대응되는 가시성 맵에서의 프래그먼트 f_v에 대한 수치적 차이를 $|dp_f - ds_f|$로 계산할 수 있는데, 여기서 dp_f는 f와 f_v사이의 깊이 값에 대한 차이이고, ds_f는 f와 f_v사이의 거리이다.

<56> 도 8은 깊이 간격(depth interval)과 거리 사이의 차이를, Buddha 모델에 대한 영상으로 시각화한 것을 나타내는 도면이다. 보다 밝은 픽셀은 깊이 갓과 거리 값 사이의 차이가 큰 것을 의미한다. 상대적인 비료를

위하여, 각 픽셀의 음영 값(gray value)은 스플랫의 크기에 대한 차이 값의 비율로서 결정되었다. 깊이 값과 거리 값의 차이가 급격하게 발생하는 부분은 주로 모델의 경계에서 관측된다. 모델이 시점으로부터 멀어질수록 이러한 차이는 점점 더 커진다. 도 8에서 음영 값이 1.0 부근이면 그 차이가 거의 스플랫 크기 정도 된다는 것을 의미하는데, 이것은 무시할 수 없을 정도로 큰 값이다. 우측 영상은 관측 거리가 더 멀어졌을 때의 영상이다. 관측자와 물체 사이의 거리가 멀어질수록 깊이 값과 거리 값의 차이에 의한 효과는 심각할 정도로 커진다. 본 발명에서 제안하고 있는 방법을 사용하면 한 픽셀에 다수의 스플랫들이 겹쳐질 때 화질이 좋아진다. 두 번째 실험은 깊이 값이 아닌 거리 값에 기초한 가시성 계산이 좀 더 정확한 가시성 검사를 가능하게 하며, 깊이 값과 관련된 문제들도 동시에 해결할 수 있음을 보여준다.

<57> 도 9는 DSD(깊이 검사를 하는 지연 쉐이딩) 기법과 DSV(거리 검사를 하는 지연 쉐이딩) 기법에 의해 만들어진 화상들의 차이를 보여주는 도면이다. 확대된 영상은 Buddha 및 Lucy 모델에 대한 특정 영역의 상세 표면을 나타낸다. 확대된 영상 중 좌측 영상은 DSD 기법에 의한 것이고, 우측 영상은 DSV 기법에 의한 것이다. 도 9에서 확인할 수 있는 바와 같이, DSD 기법을 사용할 경우 잘못된 깊이 값 때문에 물체 표면에서 여러 개의 시각적 오류들이 발생하는 반면에 DSV 기법을 이용하면 그러한 오류들이 상당히 감소한다.

<58> 앞서 언급한 바와 같이, 적응형 프래그먼트 처리 기법은 성능을 향상시킬 뿐 아니라 화질도 개선한다. 도 10은 DSV 기법과 ADS 기법을 이용하여 만든 렌더링 결과를 나타내는 도면이다. 좌측 영상은 DSV 기법으로부터의 결과를 나타내고, 우측 영상은 ADS 기법으로부터의 결과를 나타낸다. 또한, 가운데 영상은 좌측 및 우측 두 영상 간의 차이를 보여준다. 도 10에서 확인할 수 있는 바와 같이, 모델의 실루엣 부분에서 차이가 많이 나는 것을 확인할 수 있다. 가운데 그림에서 볼 수 있듯이 깊이 값에 기초한 방법으로 만든 영상은 실루엣 에지와 주름 에지 부분에서 여러 가지 시각적 오류들이 나타나는 반면에 본 발명에서 제안한 방법으로 만든 영상은 (오른쪽) 그러한 오류들이 많이 사라진 것을 알 수 있다.

<59> 3 가지 구현 방법에 따른 렌더링 성능이 다음 표 2에 나타나 있다.

표 2

<60>

모델	픽셀-cut off	점의 수	GPU 수	DSD (fps)	DSV (fps)	ADS (fps)	DSV/DSD (%)	ADS/DSD (%)
Bunny	1.0	35,130	15,754	51.60	49.90	53.58	96.71	103.84
Lion	1.0	181,371	86,245	42.58	41.12	43.74	96.57	102.72
Buddha	1.0	1,052,607	512,629	15.63	15.11	16.45	96.67	105.25
Buddha	2.0	1,052,607	327,793	22.22	21.46	23.36	96.58	105.13
Dragon	1.0	1,267,577	586,249	13.37	12.92	14.09	96.63	105.39
Dragon	2.0	1,267,577	442,216	16.67	16.12	17.59	96.70	105.52
Lucy	1.0	9,990,850	817,497	10.94	10.55	11.07	96.44	101.19
Lucy	2.0	9,990,850	224,324	29.60	28.55	31.09	96.45	105.03

<61> 표 2로부터, DSV 구현 방법은 거리 기반의 가시성 계산 과정 때문에 성능이 약간 떨어지지만 여기에 적응형 프래그먼트 처리 기법을 더하면 성능이 다시 좋아지는 것을 확인할 수 있다. 이러한 결과는 본 발명에서 제안하고 있는 방법이 성능의 손실 없이 화질을 개선할 수 있음을 의미한다.

<62> 이상 설명한 본 발명은 본 발명이 속한 기술분야에서 통상의 지식을 가진 자에 의하여 다양한 변형이나 응용이 가능하며, 본 발명에 따른 기술적 사상의 범위는 아래의 특허청구범위에 의하여 정해져야 할 것이다.

도면의 간단한 설명

<63> 도 1은 본 발명의 일 실시예에 따른, 점-기반 렌더링 방법의 구성을 나타내는 도면.

<64> 도 2는 각각의 스플랫이 화상 평면에 투영되어 정사각형을 이루는 일련의 프래그먼트들을 생성하는 것을 나타내는 도면.

<65> 도 3은 기존의 오프셋 깊이 방법을 이용하여 블렌딩될 프래그먼트들을 결정하는 경우의 문제점을 나타내는 도면.

<66> 도 4는 깊이 값으로부터 직접 얻어지는 시각-공간상의 위치를 이용하는 경우에 발생할 수 있는 z-경합 문제점을

나타내는 도면.

<67> 도 5는 스플랫의 투영된 크기가 픽셀의 크기보다 작은 경우를 나타내는 도면.

<68> 도 6은 적응형 프래그먼트 처리 기법의 일예를 나타내는 도면.

<69> 도 7은 본 발명에서 제안한 방법을 적용하여 Lucy 모델을 렌더링한 결과를 나타내는 도면.

<70> 도 8은 깊이 간격(depth interval)과 거리 사이의 차이를, Buddha 모델에 대한 영상으로 시각화한 것을 나타내는 도면.

<71> 도 9는 DSD(깊이 검사를 하는 지연 쉐이딩) 기법과 DSV(거리 검사를 하는 지연 쉐이딩) 기법에 의해 만들어진 화상들의 차이를 보여주는 도면.

<72> 도 10은 DSV 기법과 ADS 기법을 이용하여 만든 렌더링 결과를 나타내는 도면.

<73> <도면 중 주요 부분에 대한 부호의 설명>

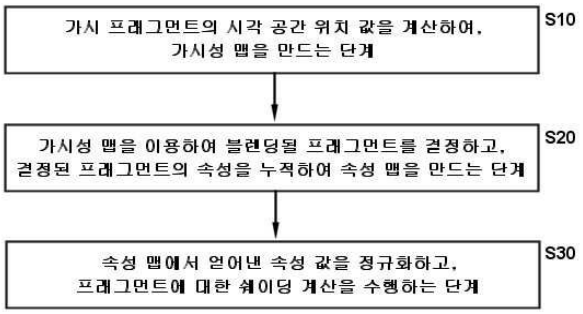
<74> S10: 가시 프래그먼트의 시각-공간 위치 값을 계산하여 가시성 맵을 만드는 단계

<75> S20: 가시성 맵을 이용하여 블렌딩될 프래그먼트를 결정하고, 결정된 프래그먼트의 속성을 누적하여 속성 맵을 만드는 단계

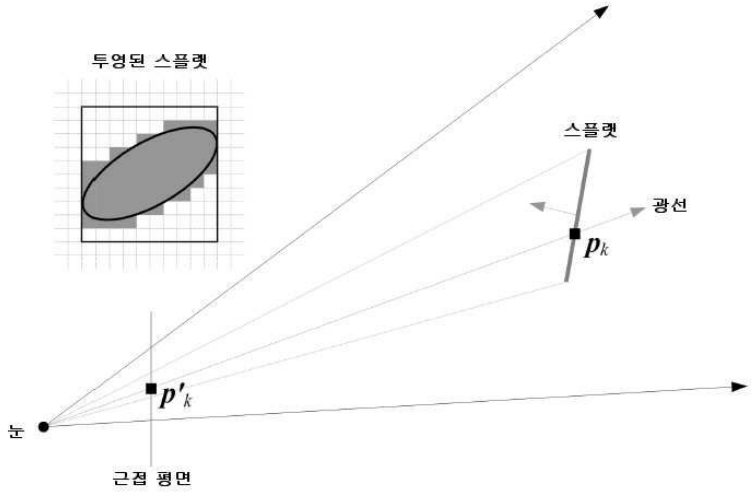
<76> S30: 속성 맵에서 얻어낸 속성 값을 정규화하고 프래그먼트에 대한 쉐이딩 계산을 수행하는 단계

도면

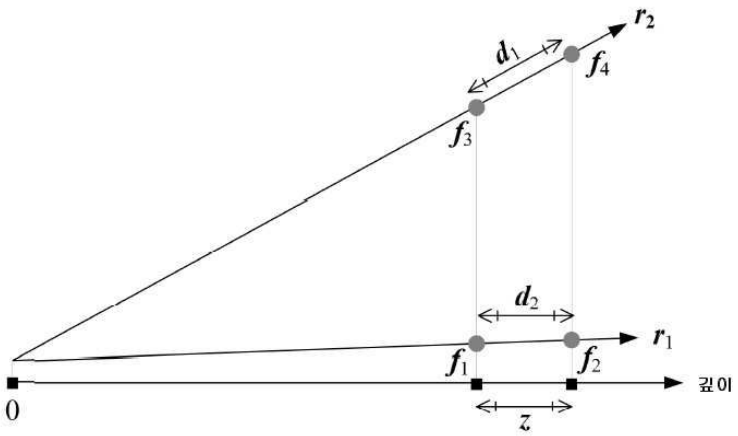
도면1



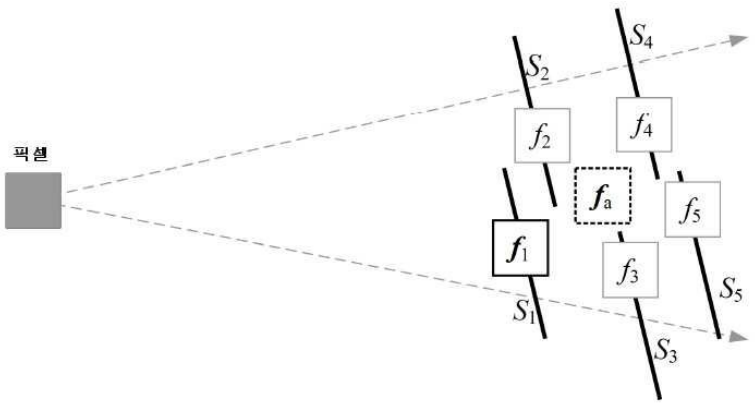
도면2


투영된 스플랫
스플랫
광선
p_k
p'_k
눈
근접 평면

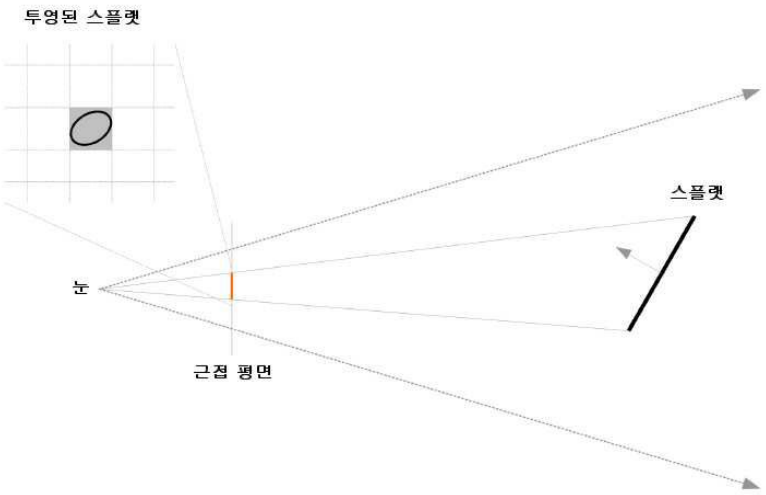
도면3


r_2
d_1
f_4
f_3
d_2
r_1
f_1
f_2
깊이
0
z

도면4


픽셀
S_4
S_2
f_2
f_4
f_a
f_5
f_1
f_3
S_5
S_1
S_3

도면5



도면6

V = SampleTexture(VisibilityMap)

if (r_v > 1.0) { // according to splat's size

Q = ComputeViewSpacePosition(p_v, n, c)

D = ComputeDistance(Q, p_v)

W = ComputeWeight(D)

} else { // for small splats

$Q = p_v$; $D = 0$; $W = 1$

}

ClipOutIf(δ < Distance(Q, V)) // distance-test

ClipOutIf($D > r$) // clip if it's outside ellipse

/* Output here */

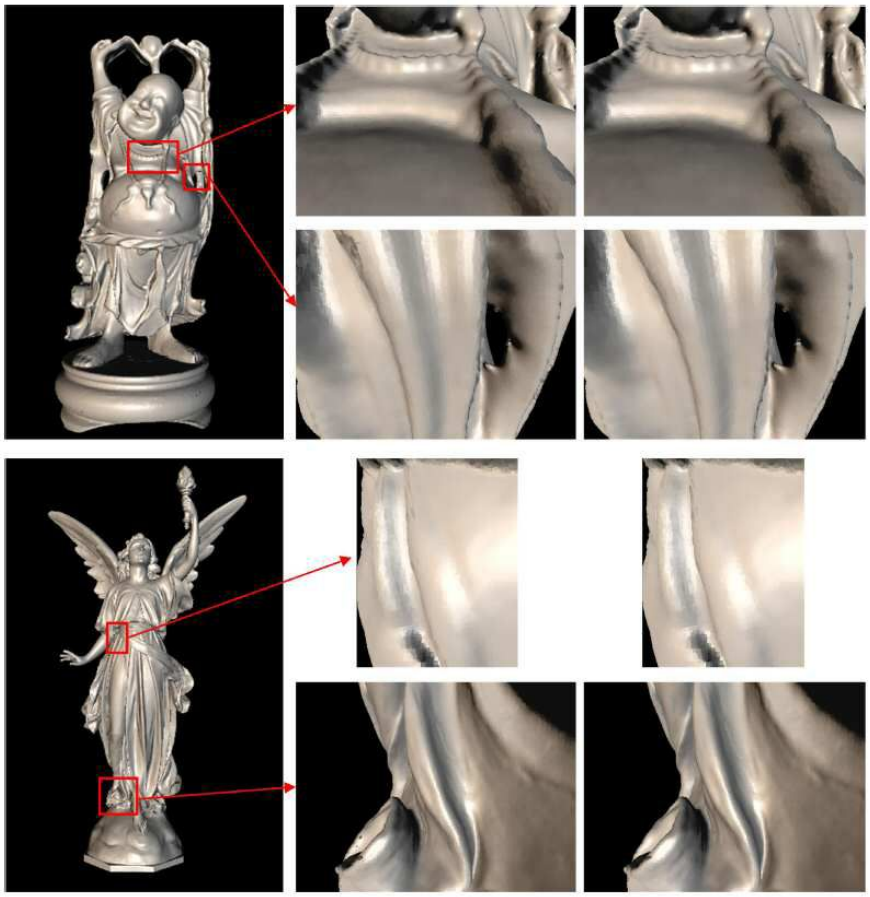
도면7



도면8



도면9



도면10

