

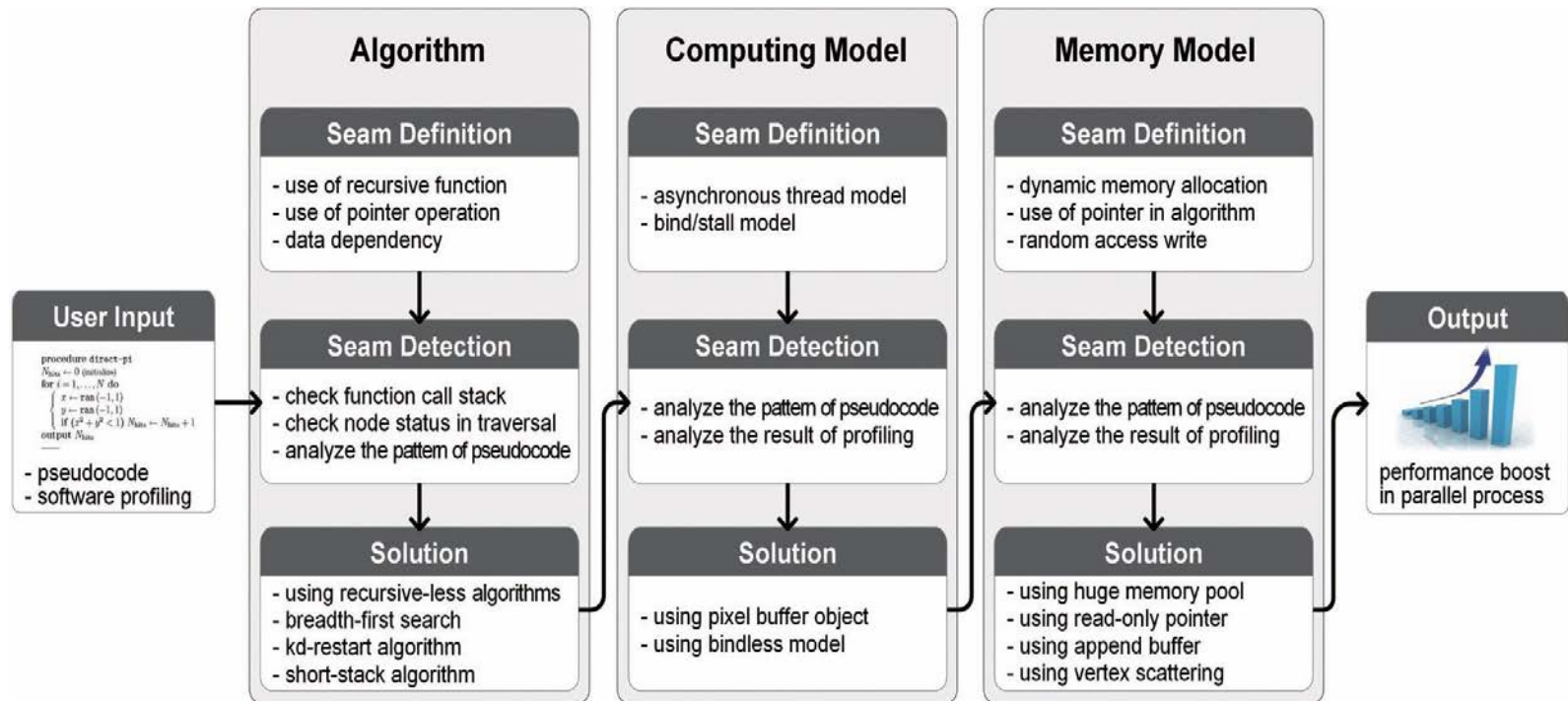
성균관대학교

고성능 GPU 컴퓨팅(GPGPU)



기술 개요

- GPU는 컴퓨팅 모델, 메모리 모델, 하드웨어, 알고리즘에서 CPU와 큰 차이가 있으며, CPU의 특성은 GPU에서 초고성능 컴퓨팅 환경의 성능을 저하시키는 주요 원인으로, 이러한 **간극(Seam)**을 제거하기 위해 문제들을 체계적으로 정형화하고, 이에 대한 최적화/대안기법을 설계함으로써 **고효율/고신뢰할 수 있는 GPU 컴퓨팅 기술**



<고성능 GPU 컴퓨팅을 위한 설계 도식도>

세부 기술 내용

● 3D 모델 로더 및 대용량 텍스트 파싱 기술

• 대용량 텍스트 파일 입력 및 처리 기술

- 최적화된 크기의 청크 설정으로 성능을 극대화
- 메모리 매핑 파일 입출력 기술로 중복된 버퍼 사용 없이 파일을 입력하는 기술

• 청크 기반의 대용량 텍스트파일 이진 변환 기술

- 이진변환에 유효한 정보만 선별하는 태그 인덱스를 생성하는 기술
- 유효한 태그 인덱스 정보를 이용한 고속 병렬 파싱 기술

• 고속 로딩을 위한 청크 기반의 버텍스 인덱싱 기술

- 테이블 기반의 청크 기반 고속 인덱싱 기술
- 정렬 기반으로 쓰기 충돌을 없는 고속 병렬화가 가능

● 효율적인 메모리 사용을 위한 대안기법

• 메모리 할당/접근의 간극 분석/검출 기술

- 성능 저하를 일으키는 합법적인 메모리 접근의 회귀 모델 정의
- 효과적인 메모리 접근의 예상 성능과 비교하여 간극의 여부를 판단하는 기술

• 효율적인 메모리 사용을 위한 대안기법 및 알고리즘 개발

- 현재 메모리가 사용하는 알고리즘에 적합한 패턴인지 분석
- 고성능을 달성할 수 있는 메모리 접근 패턴을 사용자에게 제시하는 기술

고성능 GPU 컴퓨팅(GPGPU)

기술의 특징

- 기존 CPU/GPU 기반의 텍스트 파서는 성능 혹은 메모리 문제로 대용량 파일 처리에 제한이 있음
- 청크 단위 분할 기법으로 기존의 CPU/GPU 보다 월등한 성능과 확장성을 확보
- 간극을 분석/검출하여 GPU 성능을 극대화할 수 있는 가이드라인을 제시

관련 특허

No.	특허번호	발명의 명칭
1	10-2017-0022361	확장성 있는 GPU기반의 대용량 OBJ 파일 처리 방법 및 장치
2	10-1431715	선형회귀분석을 이용한 합산 영역 테이블의 정밀도 향상을 위한 방법 및 장치 및 이를 이용한 영상 이미지 복원을 위한 필터링 방법 및 장치
3	10-2014-0083129	깊이 와핑에 기반한 폐색 커팅을 위한 방법, 장치 및 컴퓨터 판독 가능한 기록 매체

활용분야

- GPU 컴파일러/프로파일러
 - GPU 최적화 컴파일러와 프로파일러, GUI에 활용 가능
- 고해상도 미디어 압축
 - 복잡한 의존성을 가지는 직렬 알고리즘과 새로운 GPU 코덱 개발에 활용 가능
- 빅데이터 분석 및 검색
 - 매일 업데이트되는 소셜 데이터 모니터링 등의 거대한 데이터의 활용에 적용 가능
- 대용량 회귀분석과 기계학습
 - 스트리밍 입력에 대응하는 실시간 회귀분석 기법과 빅데이터에 적용 가능
- Bioinformatics
 - DNA gene sequencing 등의 무수히 많은 패턴과 조합의 대규모 처리에 활용 가능
- 이외에도 고성능의 병렬처리가 요구되는 다양한 분야에 적용 가능



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년06월30일
(11) 등록번호 10-1632992
(24) 등록일자 2016년06월17일

(51) 국제특허분류(Int. Cl.)

G06T 15/20 (2011.01)

(21) 출원번호 10-2014-0083129

(22) 출원일자 2014년07월03일

심사청구일자 2014년07월03일

(65) 공개번호 10-2016-0004605

(43) 공개일자 2016년01월13일

(56) 선행기술조사문헌

KR1020120087084 A*

KR1020100116520 A

KR1020130086976 A

KR1020120014876 A

*는 심사관에 의하여 인용된 문헌

(73) 특허권자

성균관대학교산학협력단

경기도 수원시 장안구 서부로 2066 (천천동, 성균관대학교내)

재단법인 실감교류인체감응솔루션연구단

서울특별시 성북구 화랑로14길 5, 국제협력관 (하월곡동, 한국과학기술연구원)

(72) 발명자

이성길

경기도 수원시 장안구 천천로74번길 35, 812동 904호 (정자동, 대월마을주공아파트)

김기혁

경기도 성남시 분당구 정자일로 248, 610동 2602호 (정자동, 파크뷰)

김영옥

경기도 수원시 장안구 일월로90번길 26-2, 306호 (천천동, 대현빌라)

(74) 대리인

특허법인 수

전체 청구항 수 : 총 15 항

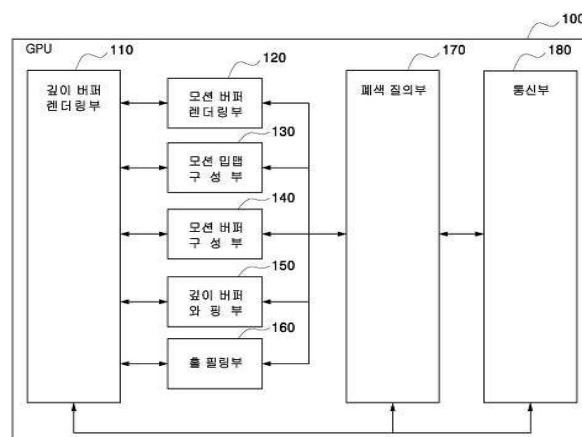
심사관 : 조우연

(54) 발명의 명칭 깊이 와평에 기반한 폐색 컬링을 위한 방법, 장치 및 컴퓨터 판독 가능한 기록 매체

(57) 요약

본 발명의 일 태양에 따르면, 폐색 질의를 수행하기 위한 방법으로서, (a) 그래픽 처리 장치가, 제1 프레임의 제1 깊이 버퍼를 사용하여 제2 프레임의 제2 깊이 버퍼를 생성하는 단계; 및 (b) 상기 그래픽 처리 장치가, 상기 제2 깊이 버퍼를 사용하여 상기 제2 프레임에 대한 폐색 질의를 수행하는 단계를 포함하고, 상기 제1 프레임은 상기 제2 프레임에 시간적으로 선행하는 프레임인 것을 특징으로 하는 방법이 제공된다. 본 발명에 의하면, 폐색 컬링의 성능 및 정확도가 향상되면서도 기존의 하드웨어 및 소프트웨어에 실용적으로 쉽게 통합될 수 있는 효과가 달성된다.

대표도 - 도1



이 발명을 지원한 국가연구개발사업

과제고유번호 1345200203
 부처명 미래창조과학부
 연구관리전문기관 한국연구재단
 연구사업명 글로벌프론티어
 연구과제명 실시간 3D 그래픽스 구현을 위한 병렬처리형 소프트웨어 라이브러리 개발
 기 여 율 1/2
 주관기관 성균관대학교 산학협력단
 연구기간 2013.09.01 ~ 2014.08.31

이 발명을 지원한 국가연구개발사업

과제고유번호 1711000784
 부처명 미래창조과학부
 연구관리전문기관 한국연구재단
 연구사업명 중견연구자지원
 연구과제명 하이레벨 GPU 소프트웨어 모델링 및 디자인 인터페이스
 기 여 율 1/2
 주관기관 성균관대학교 산학협력단
 연구기간 2013.09.01 ~ 2014.08.31

명세서

청구범위

청구항 1

폐색 질의를 수행하기 위한 방법으로서,

(a) 그래픽 처리 장치가, 제1 프레임의 제1 깊이 버퍼를 사용하여 제2 프레임의 제2 깊이 버퍼를 예측하는 단계; 및

(b) 상기 그래픽 처리 장치가, 상기 제2 깊이 버퍼를 사용하여 상기 제2 프레임에 대한 폐색 질의를 수행하는 단계를 포함하고,

상기 제1 프레임은 상기 제2 프레임에 시간적으로 선행하는 프레임이고,

상기 (a) 단계에서,

상기 그래픽 처리 장치는, 상기 제1 깊이 버퍼에 대한 와핑에 기반하여 상기 제2 깊이 버퍼를 예측하며,

상기 제1 프레임의 스크린 공간 및 객체 공간 중 적어도 하나의 공간에서의 모션을 이용하여 상기 제1 깊이 버퍼에 대한 와핑을 수행하는 것을 특징으로 하는 방법.

청구항 2

제1항에 있어서,

(c) 상기 폐색 질의의 결과를 중앙 처리 장치로 전송하는 단계

를 더 포함하는 것을 특징으로 하는 방법.

청구항 3

제1항에 있어서,

상기 (b) 단계에서,

상기 그래픽 처리 장치는, 상기 제2 깊이 버퍼를 오클루더 맵으로 이용함으로써 상기 폐색 질의를 수행하는 것을 특징으로 하는 방법.

청구항 4

제1항에 있어서,

상기 (b) 단계에서,

상기 폐색 질의는 상기 제2 프레임의 하나 이상의 객체들을 오클루더 및 오클루더로 분류하지 않고 수행되는 것을 특징으로 하는 방법.

청구항 5

삭제

청구항 6

삭제

청구항 7

폐색 질의를 수행하기 위한 방법으로서,

(a) 그래픽 처리 장치가, 제1 프레임의 제1 깊이 버퍼를 사용하여 제2 프레임의 제2 깊이 버퍼를 예측하는

단계; 및

(b) 상기 그래픽 처리 장치가, 상 기 제2 깊이 버퍼를 사용하여 상기 제2 프레임에 대한 폐색 질의를 수행하는 단계를 포함하고,

상기 제1 프레임은 상기 제2 프레임에 시간적으로 선행하는 프레임이고,

상기 (a) 단계에서,

상기 그래픽 처리 장치는, 상기 제1 깊이 버퍼에 대한 와핑에 기반하여 상기 제2 깊이 버퍼를 예측하며,

상기 와핑은 후향 와핑이고,

상기 그래픽 처리 장치는 상기 제1 프레임의 hierarchical data structure를 이용함으로써 상기 후향 와핑을 수행하는 것을 특징으로 하는 방법.

청구항 8

제1항에 있어서,

상기 (a) 단계에서,

상기 그래픽 처리 장치는 상기 와핑 후 상기 제2 깊이 버퍼에서 발생한 홀에 대하여 최대의 깊이로 상기 홀을 채우는 것을 특징으로 하는 방법.

청구항 9

제1항에 있어서,

상기 제2 프레임은 하나 이상의 객체들을 포함하고,

상기 (b) 단계에서,

상기 그래픽 처리 장치는 상기 하나 이상의 객체들에 대해 상기 폐색 질의를 병렬로 처리하는 것을 특징으로 하는 방법.

청구항 10

제1항 내지 제4항, 제7항 내지 제9항 중 어느 한 항에 따른 방법을 실행하기 위한 컴퓨터 프로그램을 기록한 컴퓨터 판독 가능한 기록 매체.

청구항 11

그래픽 처리 장치로서,

제1 프레임의 제1 깊이 버퍼를 사용하여 제2 프레임의 제2 깊이 버퍼를 예측하는 깊이 버퍼 렌더링부;

상기 제2 깊이 버퍼를 사용하여 상기 제2 프레임에 대한 폐색 질의를 수행하는 폐색 질의부; 및

상기 제1 깊이 버퍼에 대한 와핑에 기반하여 상기 제2 깊이 버퍼를 예측하는 깊이 버퍼 와핑부를 포함하고,

상기 제1 프레임은 상기 제2 프레임에 시간적으로 선행하는 프레임이고,

상기 깊이 버퍼 와핑부는 상기 제1 프레임의 스크린 공간 및 객체 공간 중 적어도 하나의 공간에서의 모션을 이용하여 상기 제1 깊이 버퍼에 대한 와핑을 수행하는 것을 특징으로 하는 장치.

청구항 12

제11항에 있어서,

상기 폐색 질의의 결과를 중앙 처리 장치로 전송하는 통신부

를 더 포함하는 것을 특징으로 하는 장치.

청구항 13

제11항에 있어서,

상기 폐색 질의부는 상기 제2 깊이 버퍼를 오클루더 맵으로 이용함으로써 상기 폐색 질의를 수행하는 것을 특징으로 하는 장치.

청구항 14

제11항에 있어서,

상기 폐색 질의는 상기 제2 프레임의 하나 이상의 객체들을 오클루더 및 오클루더로 분류하지 않고 수행되는 것을 특징으로 하는 장치.

청구항 15

삭제

청구항 16

삭제

청구항 17

그래픽 처리 장치로서,

제1 프레임의 제1 깊이 버퍼를 사용하여 제2 프레임의 제2 깊이 버퍼를 예측하는 깊이 버퍼 렌더링부;

상기 제2 깊이 버퍼를 사용하여 상기 제2 프레임에 대한 폐색 질의를 수행하는 폐색 질의부; 및

상기 제1 깊이 버퍼에 대한 와핑에 기반하여 상기 제2 깊이 버퍼를 예측하는 깊이 버퍼 와핑부

를 포함하고,

상기 제1 프레임은 상기 제2 프레임에 시간적으로 선행하는 프레임이고,

상기 와핑은 후향 와핑이고,

상기 깊이 버퍼 와핑부는 상기 제1 프레임의 hierarchical data structure를 이용함으로써 상기 후향 와핑을 수행하는 것을 특징으로 하는 장치.

청구항 18

제11항에 있어서,

상기 와핑 후 상기 제2 깊이 버퍼에서 발생한 홀에 대하여 최대의 깊이로 상기 홀을 채우는 홀 필링부

를 더 포함하는 것을 특징으로 하는 장치.

청구항 19

제11항에 있어서,

상기 제2 프레임은 하나 이상의 객체들을 포함하고,

상기 폐색 질의부는 상기 하나 이상의 객체들에 대해 상기 폐색 질의를 병렬로 처리하는 것을 특징으로 하는 장치.

발명의 설명

기술 분야

[0001]

본 발명은 프레임의 폐색을 제거하기 위한 방법, 장치 및 컴퓨터 판독 가능한 기록 매체에 관한 것으로, 보다 상세하게는, 현재의 프레임의 깊이 버퍼에 기반하여 다음의 프레임의 깊이 버퍼를 생성하고, 다음의 프레임의 깊이 버퍼를 사용하여 다음의 프레임에 대한 폐색 질의를 수행하는 방법, 장치, 시스템 및 컴퓨터 판독 가능한

기록 매체에 관한 것이다.

배경 기술

- [0002] 컴퓨터 그래픽스의 렌더링 과정은 장면(Scene)에 존재하는 모든 객체들에 대하여, 빛 및 물체 간의 상호작용을 물리적으로 시뮬레이션한다. 그러나, 모든 객체들이 한 화면에 보이지 않을 수 있으므로, 보이지 않을 객체들에 대한 시뮬레이션을 생략함으로써 렌더링 성능이 현격히 향상될 수 있다.
- [0003] 객체의 가시도(Visibility)를 렌더링 과정 이전에 미리 판단하여, 객체의 렌더링 여부를 판단하는 기법을 컬링(Culling) 또는 폐색(Occlusion) 컬링이라 칭한다. 말하자면, 컬링은 다른 객체들에 의해 가려져 보이지 않는 객체를 판단하는 기법이다.
- [0004] 컬링은 크게 뷰 절두체 컬링(VFC: View Frustum Culling, 이하 "VFC"라고 함) 및 폐색 컬링(OC: Occlusion Culling, 이하 "OC"라고 한다)으로 분류될 수 있다.
- [0005] VFC는 가상 카메라의 현재의 화각의 범위를 완전히 벗어나는 객체들을 렌더링하지 않는 기법인 반면, OC는 화각의 범위 내에 있어 보일 가능성이 있지만, 다른 객체 의해 가려져(즉, 폐색되어) 보이지 않는 객체들을 렌더링하지 않는 기법이다.
- [0006] VFC는 객체가 뷰 절두체의 내부 및 외부 중 어디에 있는 지만 판단하면 객체의 렌더링 여부를 결정할 수 있으므로, 이미 많은 렌더링 엔진 및 소프트웨어에서 사용되고 있다.
- [0007] 반면, OC는 객체들 간의 가림 여부를 판단한 후에야 객체의 렌더링 여부를 결정할 수 있는데, 가림 여부를 판단하기 위해서는 비교적 복잡한 계산이 요구된다. 가림 여부의 판단을 위한 추가 계산은 높은 부하를 발생시키며, 기존의 렌더링 엔진 및 소프트웨어에 추가되기가 어렵다는 문제를 갖는다. 따라서, 일반적으로 OC는 렌더링 엔진 및 소프트웨어에서 부분적으로만 적용된다.
- [0008] OC의 기본적인 처리 과정은 다음과 같다. 먼저, OC는 다른 객체를 가릴 가능성이 있는 객체인 오클루더(Occluder) 및 다른 객체에 의해 가려질 가능성이 있는 객체인 오클루디(Occludee)를 구분한다. 다음으로, OC는 오클루더를 렌더링하고, 렌더링된 오클루더의 깊이 버퍼(Depth Buffer)를 생성한다. 다음으로, OC는 오클루더를 완전히 감싸는 기하 객체인 바운딩 객체(Bounding object)(예를 들면, 바운딩 박스(Bounding Box) 및 바운딩 스페어(Bounding Sphere) 등)를 렌더링하거나 사영(Projection)한다. 다음으로, OC는 렌더링 또는 사영된 바운딩 객체들의 이미지 영역 내의 모든 깊이들이 오클루더의 깊이들보다 더 멀리 있으면, 오클루더가 가려졌다고 판단한다. 상술된 것과 같은 개념은 대부분의 OC에서 공통적으로 적용된다.
- [0009] 나아가, OC는 세부적으로 오클루더의 폐색 여부를 판단하는 기법에 따라, 다음에 설명하는 두 종류의 기법들로 분류될 수 있다.
- [0010] 첫 번째의 방법은 전통적인 방법으로, 계층적 폐색 맵(Hierarchical Occlusion Map)을 생성하고 이용하는 기법이다. 상기의 기법은 오클루더들만을 렌더링한 오클루더 깊이 버퍼를 계층적으로 (예를 들면, 밍맵(Mipmap) 또는 엔-버퍼(N-Buffer)들을 사용하여) 구성하고, 계층적으로 구성된 오클루더 깊이 버퍼를 오클루더들의 바운딩 객체의 깊이와 비교하는 방식이다. 이러한 방식은 계층 구조 생성의 부하의 문제, 폐색 판단의 부하의 문제 및 중앙 처리 장치(CPU: Central Processing Unit; 이하 "CPU"라고 함)의 메모리로 폐색 검사 후의 결과를 효율적으로 가져오지 못하는 문제 등을 갖기 때문에 널리 쓰이지는 못하고 있다.
- [0011] 두 번째의 방법은 최근 그래픽스 하드웨어에서 제공하는 하드웨어 폐색 질의(HOQ: Hardware Occlusion Query, 이하 "HOQ"라고 한다)를 이용하여 가시도를 판단하는 방법이다. 이러한 방법에서, 그래픽스 하드웨어는 계층 구조를 갖지 않은 채 폐색 깊이 버퍼에 대해 오클루더를 렌더링하고, 렌더링된 오클루더의 픽셀들 중 오클루더의 깊이 버퍼보다 더 가까이 있는 픽셀의 개수를 센다. 그래픽스 하드웨어는 오클루더 깊이 버퍼보다 더 가까이 있는 픽셀이 존재하지 않을 때 오클루더가 완전히 가려졌다고 판단하고, 판단의 결과를 반환한다. 판단의 결과는 렌더링 과정에서 객체를 그리지 않게 하기 위해 반드시 CPU 쪽으로 전송되어야 한다. 그러나 HOQ는 하나의 객체의 단위로 수행되고, 하나의 객체의 단위로 판단의 결과를 전송하기 위해서 상당한 부하를 요구한다. 따라서, HOQ를 이용하더라도 렌더링의 성능이 향상되는 것이 쉽지 않다는 문제가 있으며, 따라서 HOQ는 보편적으로 사용되기 어렵다.
- [0012] HOQ를 개선하는 진보된 기법 중 하나로서, 일관적 계층적 컬링(CHC: Coherent Hierarchical Culling, 이하 "CHC"라고 한다) 및 CHC가 개선된 기법인 CHC++가 있다. CHC++는 오클루더들의 결과를 즉시 그래픽 처리 장치

(GPU: Graphic Processing Unit, 이하 "GPU"라고 한다)로부터 CPU로 전송하지 않고, 상기의 전송을 다음의 프레임에 처리할 때로 지연시킨다. 또한, CHC++는 객체들의 렌더링 장면 그래프(예를 들면, 씬그래프(Scenegraph))에서 반드시 요구되는 객체들만 전송함으로써 전송의 부하를 감소시킨다. 그러나, 이러한 CHC++의 처리에 있어서도, 객체가 다수인 경우 전송의 부하 및 폐색 판단의 부하가 커진다는 문제가 발생한다. 또한, CHC는 구조적으로 렌더링 및 킬링을 중첩하여 처리해야 하기 때문에, 모든 객체에 대한 킬링을 수행한 후 별도로 렌더링을 수행하는 기존의 렌더링 엔진 및 소프트웨어에게 현격한 구조의 변화를 요구한다는 문제를 갖는다. 따라서, CHC가 실용적으로 사용되는 것은 어려우며, 또한 CHC가 다중 GPU 렌더링에 적용되는 것은 매우 어렵다는 문제가 있다.

[0013] 이에, 본 발명자는 폐색 여부의 판단에 드는 추가 부하를 감소시키면서도, 기존의 렌더링 엔진 또는 소프트웨어에 용이하게 추가될 수 있는 폐색 킬링 기술을 개발하기에 이르렀다.

발명의 내용

해결하려는 과제

[0014] 본 발명은 상술한 문제점을 모두 해결하는 것을 그 목적으로 한다.

[0015] 또한, 본 발명은 기존의 방법들과는 달리, 폐색 킬링에 추가적으로 요구되는 계산 및 전송의 부하를 감소시키면서도 폐색 킬링의 정확도가 향상시킬 수 있도록 하는 것을 다른 목적으로 한다.

[0016] 특히, 다음의 세 가지가 중요한 차이점이며 개선점이다.

[0017] 첫 번째로, 본 발명은 폐색 질의를 위한 오클루더를 선택하는 과정을 수행하지 않고, 다음의 프레임에 그려질 깊이 버퍼를 생성하는 것으로 오클루더의 선택을 대체할 수 있다. 이러한 방식은 오클루더 선택에 있어서의 어려움을 감소시키면서도, 최적화된 오클루더를 선택하는 방식과 유사하다. 따라서, 이러한 방식을 통해 폐색 질의의 정확도가 개선되면서도 오클루더 선택을 위한 추가의 계산 부하가 발생하지 않는다.

[0018] 두 번째로, 본 발명은 현재의 프레임에 대한 렌더링의 결과인 깊이 버퍼 및 모션 벡터를 이용하여 상기의 깊이 버퍼에 대한 와핑을 수행함으로써 다음의 프레임의 깊이 버퍼를 생성할 수 있다. 이러한 방식은 현재의 프레임에 대한 처리 중 다음의 프레임의 렌더링의 결과를 미리 알 수 있게 하므로, 다음의 프레임에 대한 폐색 질의를 미리 수행할 수 있게 한다. 폐색 질의가 미리 수행되고, 폐색 질의의 결과를 바로 CPU로 전송함으로써 GPU로부터 CPU로의 전송 시간으로 인한 추가적인 부하가 현격하게 감소될 수 있다.

[0019] 세 번째로, 본 발명은 계층적인 구조에 기반한 계층적인 폐색 맵을 이용함으로써 프레임의 객체들에 대한 폐색 질의를 병렬로 후처리할 수 있다. 이러한 방식은 하드웨어 폐색 질의와는 달리 킬링 및 렌더링의 과정을 완전히 분리시킬 수 있다. 이러한 방식은 기존의 렌더링 소프트웨어에 쉽게 통합될 수 있고, 다중 GPU로 용이하게 확장될 수 있다.

과제의 해결 수단

[0020] 상기 목적을 달성하기 위한 본 발명의 대표적인 구성은 다음과 같다.

[0021] 본 발명의 일 태양에 따르면, 폐색 질의를 수행하기 위한 방법으로서, (a) 그래픽 처리 장치가, 제1 프레임의 제1 깊이 버퍼를 사용하여 제2 프레임의 제2 깊이 버퍼를 생성하는 단계; 및 (b) 상기 그래픽 처리 장치가, 상기 제2 깊이 버퍼를 사용하여 상기 제2 프레임에 대한 폐색 질의를 수행하는 단계를 포함하고, 상기 제1 프레임은 상기 제2 프레임에 시간적으로 선행하는 프레임인 것을 특징으로 하는 방법이 제공된다.

[0022] 본 발명의 다른 태양에 따르면, 그래픽 처리 장치로서, 제1 프레임의 제1 깊이 버퍼를 사용하여 제2 프레임의 제2 깊이 버퍼를 생성하는 깊이 버퍼 렌더링부; 및 상기 제2 깊이 버퍼를 사용하여 상기 제2 프레임에 대한 폐색 질의를 수행하는 폐색 질의부를 포함하고, 상기 제1 프레임은 상기 제2 프레임에 시간적으로 선행하는 프레임인 것을 특징으로 하는 장치가 제공된다.

[0023] 이 외에도, 본 발명을 구현하기 위한 다른 방법, 장치, 시스템 및 상기 방법을 실행하기 위한 컴퓨터 프로그램을 기록하기 위한 컴퓨터 판독 가능한 기록 매체가 더 제공된다.

발명의 효과

[0024] 본 발명에 의하면, 폐색 킬링의 성능 및 정확도가 향상되면서도 기존의 하드웨어 및 소프트웨어에 실용적으로

쉽게 통합될 수 있는 효과가 달성된다.

- [0025] 본 발명에 의하면, 오클루더 및 오클루더의 구분을 위한 구성이 요구되지 않는다는 효과가 달성된다.
- [0026] 본 발명에 의하면, 다음의 프레임에 대한 폐색 질의가 결과가 현재의 프레임에 대한 처리 과정에서 미리 획득된다는 효과가 달성된다.
- [0027] 본 발명에 의하면, 폐색 질의의 결과를 GPU로부터 CPU로 전송하는 시간의 부하가 감소한다는 효과가 달성된다.
- [0028] 본 발명에 의하면, CHC++ 등과 같이, 객체들의 계층적인 구조가 반드시 요구되는 방식에 비해, 객체들의 계층적인 구조와 무관하게 폐색 질의가 병렬로 수행된다는 효과가 달성된다.
- [0029] 본 발명에 의하면, 폐색 질의의 후처리 구조에 의해 폐색질의가 기존의 렌더링 엔진 및 소프트웨어에 용이하게 통합될 수 있고, 멀티-GPU 렌더링의 구조에 쉽게 통합될 수 있다는 효과가 달성된다.
- [0030] 본 발명에 의하면, 깊이 버퍼의 와핑을 위해 요구되는 깊이 버퍼 및 모션 벡터 버퍼는 근래 실시간 렌더링 구조에서 보편적인 지(G)-버퍼에서 이미 가용한 것이기 때문에, 깊이 버퍼의 와핑에 의해 추가되는 부하가 높지 않다는 효과가 달성된다.

도면의 간단한 설명

- [0031] 도 1은 본 발명에 따라 프레임을 생성하는 GPU의 구성을 나타내는 도면이다.
- 도 2는 본 발명의 일 실시예에 따른 폐색 질의를 수행하기 위한 방법의 흐름도이다.
- 도 3a 내지 3c는 본 발명의 일 실시예에 따른 현재의 프레임의 RGB 이미지, 현재의 프레임의 깊이 버퍼 및 다음의 프레임의 깊이 버퍼를 나타낸다.
- 도 3a는 본 발명의 일 실시예에 따른 현재의 프레임의 RGB 이미지를 나타낸다.
- 도 3b는 본 발명의 일 실시예에 따른 현재의 프레임의 깊이 버퍼를 나타낸다.
- 도 3c는 본 발명의 일 실시예에 따른 다음의 프레임의 깊이 버퍼를 나타낸다.

발명을 실시하기 위한 구체적인 내용

- [0032] 후술하는 본 발명에 대한 상세한 설명은, 본 발명이 실시될 수 있는 특정 실시예를 예시로서 도시하는 첨부 도면을 참조한다. 이들 실시예는 당업자가 본 발명을 실시할 수 있기에 충분하도록 상세히 설명된다. 본 발명의 다양한 실시예는 서로 다르지만 상호 배타적일 필요는 없음이 이해되어야 한다. 예를 들어, 여기에 기재되어 있는 특정 형상, 구조 및 특성은 일 실시예에 관련하여 본 발명의 정신 및 범위를 벗어나지 않으면서 다른 실시예로 구현될 수 있다. 또한, 각각의 개시된 실시예 내의 개별 구성요소의 위치 또는 배치는 본 발명의 정신 및 범위를 벗어나지 않으면서 변경될 수 있음이 이해되어야 한다. 따라서, 후술하는 상세한 설명은 한정적인 의미로서 취하려는 것이 아니며, 본 발명의 범위는, 적절하게 설명된다면, 그 청구항들이 주장하는 것과 균등한 모든 범위와 더불어 첨부된 청구항에 의해서만 한정된다. 도면에서 유사한 참조부호는 여러 측면에 걸쳐서 동일하거나 유사한 기능을 지칭한다.
- [0033] 이하에서는, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자가 본 발명을 용이하게 실시할 수 있도록 하기 위하여, 본 발명의 바람직한 실시예들에 관하여 첨부된 도면을 참조하여 상세히 설명하기로 한다.
- [0034] GPU의 구성
- [0035] 도 1은 본 발명에 따라 프레임을 생성하는 GPU(100)의 구성을 나타내는 도면이다.
- [0036] 본 발명의 일 실시예에 따르면, GPU(100)는 컴퓨터 시스템에서 컴퓨터 그래픽을 담당하는 부품일 수 있다. 그러나, 본 발명에서 GPU(100)는 컴퓨터 그래픽을 담당하는 부품으로 한정되지 않고, 그래픽과 관련된 처리를 수행하는 부품, 독립적인 장치 및 독립적인 장치의 일부를 의미할 수 있다.
- [0037] 도 1에 도시되어 있는 바와 같이, GPU(100)는 깊이 버퍼 렌더링부(Depth Buffer Rendering Unit)(110), 모션 벡터 버퍼 렌더링부(Motion Vector Buffer Rendering Unit)(120), 모션 밍맵 구성부(Motion Mipmap Construction Unit)(130), 모션 버퍼 구성부(Motion Buffer Construction Unit)(140), 깊이 버퍼 와핑부(Depth Buffer Warping Unit)(150), 홀 필링부(Hole Filling Unit)(160), 폐색 질의부(Occlusion Query Unit)(170)

및 통신부(Communication Unit)(180)를 포함할 수 있다.

- [0038] 본 발명의 일 실시예에 따르면, 깊이 버퍼 렌더링부(110), 모션 벡터 버퍼 렌더링부(120), 모션 맵 구성부(130), 모션 버퍼 구성부(140), 깊이 버퍼 와핑부(150), 홀 필링부(160), 폐색 질의부(170) 및 통신부(180)는 그 중 적어도 일부가 프로그램 모듈들일 수 있다. 이러한 프로그램 모듈들은 운영 시스템, 응용 프로그램 모듈, 기타 프로그램 모듈 및 펌웨어 등의 GPU(100)에 포함될 수 있으며, 물리적으로는 여러 가지 공지의 기억 장치 상에 저장될 수 있다. 또한, 이러한 프로그램 모듈 중 적어도 일부는 GPU(100)와 통신 가능한 원격 기억 장치에 저장될 수도 있다. 여기에서, 통신은 네트워크를 통한 외부의 다른 장치와의 통신일 수 있으며, 버스(Bus)를 통한 컴퓨터 시스템 내부에서의 통신일 수 있다. 한편, 이러한 프로그램 모듈들은 본 발명에 따라 후술할 특정 업무를 수행하거나 특정 추상 데이터 유형을 실행하는 루틴, 서브루틴, 프로그램, 오브젝트, 컴포넌트, 데이터 구조 등을 포괄하지만, 이에 제한되지는 않는다.
- [0039] 먼저, 본 발명의 일 실시예에 따르면, GPU(100)는 시간적으로 연속된 복수의 프레임들을 생성할 수 있다. 복수의 프레임들을 생성함에 있어서, GPU(100)는 현재의 프레임의 깊이 버퍼를 사용하여 다음의 프레임의 깊이 버퍼를 생성할 수 있고, 생성된 다음의 프레임의 깊이 버퍼를 사용하여 다음의 프레임에 대한 폐색 질의를 수행할 수 있다. 현재의 프레임이 시점 t 에서의 프레임이라면, 다음의 프레임은 시점 $t+1$ 에서의 프레임일 수 있다.
- [0040] 본 발명의 일 실시예에 따르면, 깊이 버퍼는 프레임의 픽셀들의 깊이(또는, 깊이를 나타내는 깊이 값들)들을 포함할 수 있다. 픽셀들은 2차원으로 배치될 수 있으며, 픽셀 및 픽셀에 대응하는 깊이 또한 2차원의 좌표를 가질 수 있다. 깊이 버퍼는 프레임의 좌표들의 깊이들을 포함할 수 있다.
- [0041] 본 발명의 일 실시예에 따르면, GPU(100)는 폐색 컬링의 성능을 향상시키기 위해 현재의 프레임의 모든 보이는 객체들에 대한 깊이 버퍼를 와핑함으로써 다음의 프레임의 깊이 버퍼를 생성할 수 있으며, 생성된 다음의 프레임의 깊이 버퍼를 다음의 프레임의 오클루더로서 사용할 수 있다. 현재의 프레임에서 보이는 객체들은 다음의 프레임에서 보이는 위치로 이동되었을 때 최적의 오클루더가 될 수 있다. 따라서, 본 발명의 일 실시예에서는 다음의 프레임의 객체들 중에서 오클루더를 선택하는 과정이 제거될 수 있다.
- [0042] 본 발명의 일 실시예에 따르면, 폐색 질의는 계층적 폐색 맵을 사용하여 수행될 수 있다. 예를 들면, 폐색 질의는 N -버퍼들을 깊이 버퍼의 계층적인 구조화를 위해 사용할 수 있다.
- [0043] 구성 요소들의 기능
- [0044] 아래에서, 본 발명의 일 실시예에 따른 GPU(100)의 구성 요소들의 기능이 설명된다.
- [0045] 본 발명의 일 실시예에 따르면, 깊이 버퍼 렌더링부(110)는 프레임의 깊이 버퍼를 생성할 수 있다.
- [0046] 폐색 컬링을 수행하기 위해서는 프레임의 하나 이상의 객체들을 렌더링하였을 때, 깊이들이 저장될 깊이 버퍼가 요구된다. 여기에서, 깊이들은 프레임의 픽셀들의 깊이들일 수 있다. 깊이 버퍼는 프레임의 깊이를 포함할 수 있다.
- [0047] 일반적으로, 렌더링의 결과로서, 빛 및 객체 간의 시뮬레이션이 완료됨에 따른 프래그먼트(또는, 픽셀)의 색(예를 들면, 적녹청(RGB; Red Green Blue, 이하 "RGB"라고 함))만 요구되고, 프래그먼트의 깊이는 반드시 요구되지는 않는다. 그러나, 특히 실시간 렌더링의 경우, 많은 경우에 있어서 더 앞에 위치하는 프래그먼트만을 유지하기 위해 깊이 버퍼를 사용할 수 있다. GPU(100)는 깊이 버퍼를 외부로부터 입력 받거나, 외부로 출력할 수 있다. 따라서, 전술된 프레임의 깊이 버퍼의 생성 및 처리는 일반적인 그래픽 처리 장치에 비해 추가적인 구성 또는 부하를 요구하지 않거나, 단지 적은 정도만을 요구할 수 있다. 또한, 실시간 렌더링에 있어서, 깊이 버퍼는 지원된(deferred) 셰이딩을 위해 기본적으로 출력될 수 있다.
- [0048] 본 발명의 일 실시예에 따르면, 모션 벡터 버퍼 렌더링부(120)는 모션 벡터 버퍼를 생성할 수 있다.
- [0049] 깊이 버퍼의 와핑을 수행하기 위해서는, 프레임의 하나 이상의 객체들의 모션을 나타내는 모션 벡터들의 계산이 요구되고, 계산된 모션 벡터들에 따라 하나 이상의 객체들을 이동시키는(즉, 와핑) 처리가 요구된다. 모션 벡터 버퍼는 이러한 프레임의 모션 벡터를 포함할 수 있다.
- [0050] 모션 벡터 버퍼 렌더링부(120)는 하나 이상의 객체들에 대한 카메라의 모션을 프레임 별로 저장하고, 프레임 별로 저장된 모션을 사용하여 모션 벡터를 선형 또는 비선형으로 예측하는 수학적 모델을 이용함으로써 모션 벡터를 획득할 수 있다. 모션 벡터는 프레임의 픽셀이 이동한 거리 및 방향을 나타낼 수 있다.
- [0051] 본 발명의 일 실시예에 따르면, 모션 벡터 버퍼 렌더링부(120)는 스크린 공간(Screen Space)에서의 모션 벡터를

획득할 수 있고, 객체 공간(Object Space)에서의 모션 벡터를 획득할 수 있다. 또한, 모션 벡터 버퍼 렌더링부(120)는 스크린 공간 및 객체 공간에 제한되지 않는 다양한 방식에 따라 모션 벡터를 획득할 수 있다.

- [0052] 본 발명의 일 실시예에 따르면, 스크린 공간에서의 모션 벡터를 획득할 경우, 모션 벡터 버퍼 렌더링부(120)는 이전의 프레임 및 현재의 프레임에서의 카메라의 뷰들 및 사영 행렬(projection matrix)들을 각각 이용함으로써 스크린의 하나 이상의 픽셀들의 각 픽셀의 모션 벡터를 계산할 수 있다.
- [0053] 지연된 셰이딩에서는, 모션 블러(Motion blur) 효과를 시뮬레이션하기 위한 모션 벡터가 출력될 수 있다. 따라서, 전송된 모션 벡터 버퍼의 생성 및 처리는 일반적인 그래픽 처리 장치에 비해 추가적인 구성을 요구하지 않거나 단지 적은 정도만을 요구할 수 있다.
- [0054] 본 발명의 일 실시예에 따르면, 모션 맵핑 구성부(130)는 프레임의 맵핑을 생성할 수 있다.
- [0055] 모션 맵핑은 모션 벡터의 최대값 및 최소값을 사용하여 $1/n$ 으로 감축된 이미지일 수 있다. 예를 들면, n 은 4일 수 있다. 모션 맵핑의 생성에 따라, 모션 벡터 버퍼의 크기는 $1/n$ 로 감소될 수 있다. 말하자면, 모션 맵핑에 의해 이미지의 해상도가 $1/n$ 로 감소될 수 있다. 스크린 공간에서 모션 벡터는 2차원(Dimension)일 수 있다.
- [0056] 본 발명의 일 실시예에 따르면, 모션 맵핑 구성부(130)는 모션 벡터 버퍼를 이용함으로써 모션 맵핑을 생성할 수 있다. 모션 맵핑 구성부(130)는 $1/n$ 로 모션 벡터 버퍼의 크기가 감소될 때마다, n 개의 픽셀들의 최소값 및 최대값을 검출할 수 있고, 검출된 최소값 및 최대값을 사용하여 상기의 n 개의 픽셀들을 1×1 의 픽셀로 감소시킬 수 있다. n 개의 픽셀들은 사각(squared) 내의 n 개의 픽셀들일 수 있다.
- [0057] 본 발명의 일 실시예에 따르면, 생성된 맵핑은 하기에서 설명될 후방 매핑(Backward mapping)에서, 프레임의 한 픽셀의 주변의 픽셀들을 검출할 때, 상기의 검출을 수행할 범위의 최대값을 찾기 위해 이용될 수 있다.
- [0058] 본 발명의 일 실시예에 따르면, 맵핑은 모션의 전역적인 최소값 및 최대값을 나타낼 수 있다. 전역적인 모션의 최소값 및 최대값에 비해, 아래에서 설명될 모션 벡터의 N -버퍼들은 모션의 범위를 국지적으로 감축시킴으로써 국지적인 검출이 이루어지게 할 수 있다.
- [0059] 본 발명의 일 실시예에 따르면, 모션 버퍼 구성부(140)는 모션 벡터 버퍼를 이용함으로써 모션 N -버퍼들을 생성할 수 있다. 모션 N -버퍼들은 모션 벡터에 대한 계층적인 구조의 이미지들을 나타낼 수 있다.
- [0060] 모션 N -버퍼들은, 맵핑과는 달리, 계층적인 구조의 이미지의 해상도(또는, 크기)를 $1/n$ 로 감소시키지 않고, 모션 벡터의 해상도를 그대로 유지한다. 해상도를 그대로 유지하기 때문에, 일반적으로 N -버퍼들을 생성하기 위한 비용은 맵핑을 생성하기 위한 비용에 비해 더 높을 수 있으나, 국지적인 영역에서의 최소값 및 최대값을 검출하기 위한 질의에 대해서는 일반적으로 N -버퍼들이 맵핑에 비해 더 정확한 결과를 생성할 수 있다.
- [0061] 본 발명의 일 실시예에 따르면, 깊이 버퍼 와핑부(150)는 깊이 버퍼에 대한 와핑을 수행할 수 있다.
- [0062] 본 발명의 일 실시예에 따르면, 깊이 버퍼 와핑부(150)는 깊이 버퍼에 대한 와핑을 수행하기 위해 후향 와핑을 이용할 수 있다. 전향(Forward) 와핑 방식이 픽셀을 모션 벡터에 따라 이동시키는 방식임에 비해, 후향 와핑 방식은 와핑에 의해 생성될 픽셀에 대해서, 소정의 대상의 픽셀들 중 와핑을 통해 이동될 원본의 픽셀을 검출하는 방식일 수 있다. 후향 와핑 방식은 전향 와핑 방식에 비해 와핑의 성능을 향상시킬 수 있으나, 원본의 픽셀을 검출하기 위해 스크린의 전체가 아닌 지역적인 범위로 검색의 범위를 축소시킬 것을 요구한다.
- [0063] 후향 와핑 방식에 따라, 깊이 버퍼 와핑부(150)는 깊이 버퍼의 각 픽셀에 대해서 와핑에 의해 옮겨질 픽셀을 선택할 수 있다. 말하자면, 깊이 버퍼 와핑부(150)는 깊이 버퍼의 픽셀들을 소정의 순서에 따라 현재의 픽셀로서 선택할 수 있고, 현재의 픽셀로 와핑될 픽셀을 선택할 수 있다. 본 발명의 일 실시예에서, 깊이 버퍼 와핑부(150)는 검색의 범위의 축소를 위해 움직임 벡터 전체의 모션에 대해서는 맵핑을 이용하고, 검색의 범위에 대한 지역적인 조정에 대해서는 N -버퍼들을 이용함으로써, 현재의 픽셀의 주위에서 와핑에 적용될 수 있는 모션들을 검출할 수 있다.
- [0064] 깊이 버퍼 와핑부(150)는 모션 맵핑 및 모션 N -버퍼들을 이용하여, 현재의 픽셀의 주위에서 모션에 의해 현재의 픽셀로 이동될 가능성이 있는 픽셀들의 최대의 범위를 검출할 수 있다. 여기서, 범위는 현재의 픽셀로부터 소정의 거리 내에 있는 픽셀들의 영역을 의미할 수 있다. 또는, 범위는 원형 및 사각형 등 소정의 형태를 갖는 영역일 수 있다.
- [0065] 깊이 버퍼 와핑부(150)는 우선, 모션 맵핑을 사용하여 현재의 픽셀의 모션의 전역적인 제1 최대값 및 제1 최소값을 검출할 수 있고, 상기의 제1 최대값 및 제1 최소값에 해당하는 스크린의 영역 내에서 N -버퍼들을 사용하여

다시 제2 최대값 및 제2 최소값을 검출할 수 있다. 깊이 버퍼 와핑부(150)는 검출된 실제의 제2 최대값 및 제2 최소값을 밀착(tight)하여 감싸는 이미지의 영역을 검출할 수 있다.

[0066] 이미지의 영역이 검출된 후, 깊이 버퍼 와핑부(150)는 깊이 버퍼에 대한 와핑을 수행하기 위해 고정-소수점(Fixed-point) 반복(Iteration)을 이용할 수 있다.

[0067] 반복의 결과는, 반복의 초기값에 따라 서로 상이할 수 있다. 깊이 버퍼 와핑부(150)는 무작위로 선택된 복수의 초기값들을 사용함으로써 와핑의 성공 가능성을 상승시킬 수 있다. 깊이 버퍼 와핑부(150)는 검출된 이미지의 영역의 픽셀들 중, 현재의 픽셀과의 거리가 모션 벡터와 동일한 하나 이상의 픽셀들을 검출할 수 있다. 깊이 버퍼 와핑부(150)는 검출된 하나 이상의 픽셀들 중 최저의 작은 깊이를 갖는 픽셀을 현재의 픽셀에 와핑될 픽셀로서 선택할 수 있다.

[0068] 본 발명의 일 실시예에 따르면, 홀 필터링부(160)는 깊이 버퍼 내의 홀을 채울 수 있다. 와핑이 수행된 후에, 깊이 버퍼 내에는 홀이 있을 수 있다. 홀은 깊이가 정해지지 않은 픽셀일 수 있다. 홀을 채운다는 것은 홀의 깊이를 소정의 절차에 따라 결정하는 것을 의미할 수 있다.

[0069] 특정한 픽셀에 있어서, 상기의 특정한 픽셀의 주위의 픽셀들 중 와핑에 의해 상기의 특정한 픽셀의 위치로 이동하는 픽셀이 없을 경우에 상기의 특정한 픽셀은 홀이 될 수 있다. 깊이 버퍼에서는 픽셀에 대해 와핑이 수행되면 상기의 픽셀의 원래의 깊이가 보존되지 않을 수 있기 때문에, 일반적으로 홀의 깊이를 정확하게 결정하는 것은 어려울 수 있다.

[0070] 본 발명의 일 실시예에서, 홀 필터링부(160)는 기존의 다양한 홀 필터링 방법을 사용할 수 있다. 예를 들면, 효율적인 근사로서, 홀 필터링부(160)는 홀의 주변의 픽셀의 깊이를 홀의 깊이로 결정할 수 있다. 말하자면, 홀 필터링부(160)는 홀을 채우기 위해 홀의 주변의 픽셀의 깊이를 사용할 수 있다. 주변의 픽셀의 깊이를 사용하는 방식은 최선의 근사법 중 하나이지만, 채워진 깊이가 반드시 정확하다고 보장될 수는 없다. 프레임의 색상의 와핑에 있어서는, 홀 필터링에서의 오차는 큰 문제가 될 수 있다. 그러나, 본 발명의 일 실시예에서, 깊이 버퍼는 페색 질의를 위해 와핑되기 때문에, 홀 필터링에서 오차가 발생하는 문제에 대해서 가능한 보수적으로 대응하는 것이 적합할 수 있다. 예를 들면, 보수적인 대응으로서, 홀 필터링부(160)는 가용한 최대의 깊이로 홀을 채울 수 있다. 말하자면, 홀 필터링부(160)는 홀 필터링에 있어서, 발생할 수 있는 최악의 경우를 가정함으로써 홀이 채워진 깊이 버퍼를 이용하는 페색 질의가 잘못된 결과를 생성하는 것을 최대한 방지할 수 있다.

[0071] 본 발명의 일 실시예에서, 페색 질의부(170)는 와핑된 깊이 버퍼를 사용하여 페색 결링을 수행할 수 있다.

[0072] 페색 질의부(170)는 와핑된 깊이 버퍼를 사용하여 계층적인 깊이 버퍼를 생성할 수 있고, 계층적인 깊이 버퍼를 사용하여 페색 결링을 수행할 수 있다. 페색 질의부(170)는 계층적인 깊이 버퍼로서, 깊이의 N-버퍼들을 사용할 수 있다. 본 발명의 일 실시예에서, N-버퍼는 깊이 버퍼의 계층적인 구조로서 사용될 수 있으며, 모션 벡터의 계층적인 구조로서 사용될 수 있다.

[0073] 본 발명의 일 실시예에서, 페색 질의부(170)는 계층적인 구조의 깊이 버퍼를 사용하여, 프레임의 하나 이상의 객체들의 각 객체에 대한 월드(World) 변환, 카메라 변환 및 사영 변환을 순차적으로 수행함으로써 상기의 각 객체의 스크린 좌표를 생성할 수 있다. 객체의 바운딩 박스는 8 개의 꼭지점들로 표현되므로, 페색 질의부(170)는 바운딩 박스의 꼭지점들의 각 꼭지점의 최대 화면 좌표 및 최소 화면 좌표를 계산함으로써 객체를 완전히 포함하는 스크린 상의 사각의 영역을 획득할 수 있다. 페색 질의부는 획득된 사각의 영역에 대하여, 깊이의 N-버퍼들에서의 최대값을 획득할 수 있다. 페색 질의부(170)는 객체 내의 최소의 깊이가 깊이의 N-버퍼들에서 검출된 최대값보다 더 크면 상기의 객체가 페색된 것으로 판단할 수 있다. 전술된 것과 같이, 페색 질의부(170)는 깊이 버퍼를 사용하여 이미지 기반의 페색 질의를 수행할 수 있다.

[0074] 본 발명의 일 실시예에서, 바운딩 박스가 객체를 감싸기에 충분히 밀착되어(tight) 생성되지 않을 수 있기 때문에, 페색 질의부(170)는 바운딩 박스를 세분화하여 페색 질의를 수행할 수 있고, 바운딩 박스의 세분화를 통해 페색 질의의 효율을 더 향상시킬 수 있다.

[0075] 통신부(170)는 페색 질의의 결과를 출력할 수 있다. 통신부(170)는 페색 질의의 결과를 비동기적으로 CPU로 전송할 수 있다. 페색 질의의 결과는 GPU(100)의 메모리에 저장되어 있기 때문에, 이후의 렌더링에서 페색된 객체를 제외하기 위해서는, 페색 질의의 결과가 CPU의 메모리로 전송될 필요가 있다.

[0076] 페색 질의의 결과가 한 프레임의 내에서 동기적으로 수행되거나, 프레임의 하나 이상의 객체들 중 일부의 객체에 대해서만 수행될 경우, 전송의 방식이 복잡해지거나, 비효율적으로 이루어질 수 있다. 본 발명의 일 실시예

에서, 폐색 질의의 결과는 하나의 이미지 내에 포함될 수 있기 때문에, 폐색 질의의 결과는 GPU(100)로부터 CPU로 미리 전송될 수 있다. 상기의 전송은 보통 1 내지 2 프레임의 처리 시간을 요구하기 때문에, GPU(100)가 다음의 프레임을 처리할 때, CPU는 현재의 프레임에 대한 폐색 질의의 결과를 획득할 수 있다. 따라서, 통신부(170)는 하나의 이미지를 통해 프레임의 하나 이상의 객체들의 모두에 대한 폐색 질의의 결과를 즉시 가용하게 할 수 있다.

[0077] 폐색 질의를 위한 구체적인 방법.

[0078] 도 2는 본 발명의 일 실시예에 따른 폐색 질의를 수행하기 위한 방법의 흐름도이다.

[0079] 하기의 단계들(S210 내지 S)에서, GPU(100)는 프레임의 폐색 제거 및 폐색 제거를 위한 폐색 질의를 수행할 수 있다.

[0080] 우선, 이미 깊이 버퍼가 생성된 프레임을 제1 프레임으로 명명한다. 또한, 와핑에 의해 깊이 버퍼가 생성될 프레임을 제2 프레임으로 명명한다. 본 발명의 일 실시예에 따르면, 제1 프레임은 제2 프레임에 시간적으로 선행하는 프레임일 수 있다. 말하자면, 제1 프레임이 시각 t 에서의 프레임이라면, 제2 프레임은 시각 $t+x$ 에서의 프레임일 수 있다. x 는 1 이상의 정수일 수 있다. 또는, 제1 프레임 및 제2 프레임은 시간적으로 잇따른(consecutive) 프레임들일 수 있다. 제1 프레임의 깊이 버퍼를 제1 깊이 버퍼로 명명하고, 제2 프레임의 깊이 버퍼를 제2 깊이 버퍼로 명명한다. 예를 들면, 제1 프레임이 현재의 처리 대상인 현재의 프레임이라면, 제2 프레임은 제1 프레임의 다음으로 생성될 다음의 프레임일 수 있다.

[0081] (i) 깊이 버퍼 렌더링부(110)는 제1 프레임의 제1 깊이 버퍼를 사용하여 제2 프레임의 제2 깊이 버퍼를 생성할 수 있다(S210 내지 S250). 깊이 버퍼 렌더링부(110)는 모션 벡터 버퍼 렌더링부(120), 모션 맵 구성부(130), 모션 버퍼 구성부(140), 깊이 버퍼 와핑부(150) 및 홀 필링부(160)를 사용하여 제2 깊이 버퍼를 생성할 수 있다.

[0082] (i-1) 제2 깊이 버퍼의 생성을 위해, 모션 벡터 버퍼 렌더링부(120)는 제1 프레임에 대한 모션 벡터 버퍼를 생성할 수 있다(S210). 여기서, 제1 프레임은 도 1을 참조하여 설명된 현재의 프레임일 수 있다.

[0083] (i-2) 다음으로, 모션 맵 구성부(130)는 모션 맵을 생성할 수 있다(S220). 모션 맵 구성부(130)는 모션 벡터 버퍼를 이용함으로써 제1 프레임의 모션 맵을 생성할 수 있다.

[0084] (i-3) 다음으로, 모션 버퍼 구성부(140)는 모션 N-버퍼들을 생성할 수 있다(S230). 모션 버퍼 구성부(140)는 모션 벡터 버퍼를 이용함으로써 제1 프레임의 모션 N-버퍼들을 생성할 수 있다.

[0085] (i-4) 다음으로, 깊이 버퍼 와핑부(150)는 제1 깊이 버퍼에 대한 와핑에 기반하여 제2 깊이 버퍼를 생성할 수 있다(S240). 말하자면, 제2 깊이 버퍼는 제1 깊이 버퍼가 와핑된 깊이 버퍼일 수 있다.

[0086] 도 1을 참조하여 전송된 것과 같이, 모션 벡터 버퍼 렌더링부(120)는 스크린 공간에서의 모션 벡터 또는 객체 공간에서의 모션 벡터를 획득할 수 있다. 깊이 버퍼 와핑부(150)는 제1 프레임의 스크린 공간 및 객체 공간 중 적어도 하나의 공간에서의 모션을 이용하여 제1 깊이 버퍼에 대한 와핑을 수행할 수 있다. 깊이 버퍼 와핑부(150)는 모션 맵 및 모션 N-버퍼들을 이용하여 제1 깊이 버퍼에 대한 와핑을 수행함으로써 제2 깊이 버퍼를 생성할 수 있다.

[0087] 폐색 컬링의 성능의 향상을 위해, 깊이 버퍼 와핑부(150)는 제1 프레임의 하나 이상의 보이는 객체의 깊이 버퍼에 대한 와핑을 수행할 수 있고, 와핑된 깊이 버퍼는 이후 오클루더로서 사용될 수 있다.

[0088] 본 발명의 일 실시예에 따르면, 상기의 와핑은 후향 와핑일 수 있고, 모션 벡터 버퍼 렌더링부(120)는 모션 맵 및 모션 N-버퍼들을 이용함으로써 후향 와핑을 수행할 수 있다.

[0089] (i-5) 다음으로, 홀 필링부(160)는 와핑 후 제2 깊이 버퍼에서 발행한 홀에 대해 홀 필링을 수행할 수 있다(S2350). 홀 필링부(160)는 와핑 후 제2 깊이 버퍼에서 발생한 홀에 대해 최대의 깊이로 홀을 채울 수 있다.

[0090] (ii) 폐색 질의부(170)는 제2 깊이 버퍼를 사용하여 제2 프레임에 대한 폐색 질의를 수행할 수 있다(S260).

[0091] 본 발명의 일 실시예에 따르면, 현재의 프레임인 제1 프레임의 제1 깊이 버퍼에 대한 와핑을 수행함으로써 다음에 그려질 프레임인 제2 프레임의 제2 깊이 버퍼가 생성되면, 폐색 질의부(170)는 오클루더를 별도로 선택하지 않고 제2 프레임의 제2 깊이 버퍼를 오클루더로서 이용할 수 있다. 말하자면, 종래의 기술에서의 오클루더의 선택은 본 발명의 일 실시예에서는 다음의 프레임인 제2 프레임에 그려질 깊이 버퍼를 생성하는 것으로 대체될

수 있다. 패색 질의부(170)는 제2 깊이 버퍼를 오클루더 맵으로 이용함으로써 패색 질의를 수행할 수 있다.

- [0092] 전술된 것과 같이, 제1 깊이 버퍼에 대한 와핑을 통해 제2 깊이 버퍼가 생성될 수 있다. 말하자면, 현재의 프레임인 제1 프레임에서 보이는 객체들이 다음의 프레임인 제2 프레임에서 보이는 위치로 이동되었을 때 최적의 오클루더가 될 수 있다. 따라서, 패색 질의부(170)는 오클루더를 선택하는 별도의 과정이 없이도 최적의 오클루더를 생성할 수 있고, 또한 다음의 프레임인 제2 프레임의 패색 검사를 미리 수행할 수 있다. 상기와 같은 패색 질의부(170)의 기능으로 인해, 패색 질의에 요구되는 추가의 부하가 감소될 수 있으며, 높은 성능의 패색 질의가 수행될 수 있다. 또한, 본 발명의 일 실시예에 따르면, 패색 질의는 제2 프레임의 하나 이상의 객체들을 오클루더 및 오클루더로 분류하지 않고 수행될 수 있다.
- [0093] 또한, 본 발명의 일 실시예에 따르면, 제2 프레임은 하나 이상의 객체들을 포함할 수 있고, 패색 질의부(170)는 제2 프레임의 하나 이상의 객체들에 대해 패색 질의를 병렬로 처리할 수 있다. 패색 질의부(170)는 제2 프레임의 하나 이상의 객체들에 대한 패색 검사를 단일-패스(single-pass)에 수행할 수 있다.
- [0094] 본 발명의 일 실시예에 따르면, 패색 질의부(170)는 깊이 버퍼의 계층적인 구조화를 위해 N-버퍼들을 사용할 수 있고, N-버퍼들을 사용하여 패색 질의를 수행할 수 있다. 패색 질의부(170)는 계층적인 패색 맵을 이용하여 제2 프레임의 하나 이상의 객체들에 대한 패색 질의를 병렬로 처리할 수 있고, 상기의 패색 질의는 후처리로서 수행될 수 있다.
- [0095] 또한, 본 발명의 일 실시예에 따르면, 패색 질의부(170)는 후처리(post-processing)의 접근 방식으로 제2 프레임에 대한 패색 검사를 수행하기 때문에, 컬링 및 렌더링의 과정이 완전히 분리될 수 있다. 상기의 분리를 통해, 패색 질의부(170)의 패색 질의는 다른 하드웨어 기반의 패색 질의에 비해 렌더링 엔진에 더 용이하게 합쳐질 수 있으며, 기존의 렌더링 소프트웨어와 용이하게 통합될 수 있고, 다중-GPU(Multi-GPU) 렌더링에도 용이하게 확장될 수 있다.
- [0096] (iii) 통신부(180)는 패색 질의의 결과를 CPU로 전송할 수 있다(S270). 통신부(180)는 패색 질의 후 바로 패색 질의의 결과를 CPU로 전송할 수 있다. 전술된 것과 같이, 다음의 프레임인 제2 프레임의 제2 깊이 버퍼는 현재의 프레임인 제1 프레임에 대한 렌더링의 결과인 제1 깊이 버퍼 및 모션 버퍼를 사용하여 생성될 수 있다. 말하자면, 현재의 프레임인 제1 프레임에 대한 처리 중 다음의 프레임인 제2 프레임의 렌더링의 결과가 미리 획득될 수 있고, 제2 프레임에 대한 패색 질의가 미리 수행될 수 있다. 통신부(180)는 미리 수행된 패색 질의의 결과를 바로 CPU로 전송함으로써 GPU(100)로부터 CPU로의 전송 시간으로 인한 추가적인 부하를 감소시킬 수 있다.
- [0097] 도 3a 내지 3c는 본 발명의 일 실시예에 따른 현재의 프레임의 RGB 이미지, 현재의 프레임의 깊이 버퍼 및 다음의 프레임의 깊이 버퍼를 나타낸다.
- [0098] 도 3a는 본 발명의 일 실시예에 따른 현재의 프레임의 RGB 이미지를 나타낸다. 현재의 프레임은 시각 t에서의 프레임일 수 있고, RGB 이미지는 시각 t에서의 이미지인 RGB_t 일 수 있다.
- [0099] 도 3b는 본 발명의 일 실시예에 따른 현재의 프레임의 깊이 버퍼를 나타낸다. 깊이 버퍼는 시각 t에서의 깊이 버퍼인 Z_t 일 수 있다.
- [0100] 도 3c는 본 발명의 일 실시예에 따른 다음의 프레임의 깊이 버퍼를 나타낸다. 깊이 버퍼는 시각 t+1에서의 깊이 버퍼인 Z_{t+1} 일 수 있다.
- [0101] GPU(100)는 현재 프레임에 대한 처리 과정 중 깊이 버퍼 Z_t 에 대한 와핑을 수행함으로써 다음 프레임의 깊이 버퍼 Z_{t+1} 를 미리 생성할 수 있고, 미리 생성된 깊이 버퍼 Z_{t+1} 를 사용하여 이미지 기반의 패색 컬링을 수행할 수 있다. GPU(100)는 패색 컬링의 결과를 사용하여 다음 프레임에 대한 처리 과정에서는 패색된 객체들을 그리지 않을 수 있다.
- [0102] 도 3c에서, 도 3b와 유사한 형태를 갖는 부분은 와핑에 의해 생성된 부분이고, 흰색으로 표시된 부분들은 홀이 발생한 부분이다. 홀은 최대의 깊이인 1로 채워졌다.
- [0103] 이상 설명된 본 발명에 따른 실시예들은 다양한 컴퓨터 구성요소를 통하여 수행될 수 있는 프로그램 명령어의 형태로 구현되어 컴퓨터 판독 가능한 기록 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능한 기록 매체는 프로그램 명령어, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 컴퓨터 판독 가능한 기록 매체에 기록되는 프로그램 명령어는 본 발명을 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어

트웨어 분야의 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능한 기록 매체의 예에는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM, DVD와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 ROM, RAM, 플래시 메모리 등과 같은 프로그램 명령어를 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령어의 예에는, 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드도 포함된다. 상기 하드웨어 장치는 본 발명에 따른 처리를 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0104] 이상에서 본 발명이 구체적인 구성요소 등과 같은 특정 사항들과 한정된 실시예 및 도면에 의해 설명되었으나, 이는 본 발명의 보다 전반적인 이해를 돕기 위해서 제공된 것일 뿐, 본 발명이 상기 실시예들에 한정되는 것은 아니며, 본 발명이 속하는 기술분야에서 통상적인 지식을 가진 자라면 이러한 기재로부터 다양한 수정 및 변형을 꾀할 수 있다.

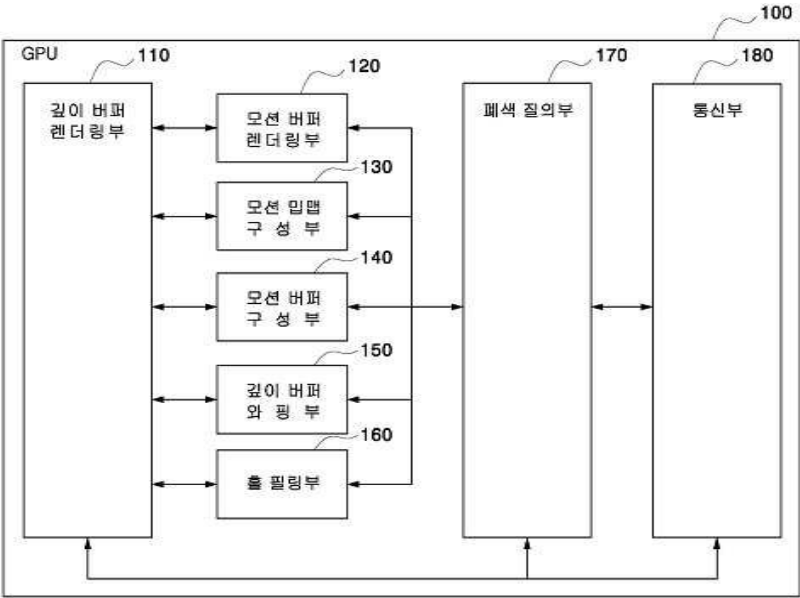
[0105] 따라서, 본 발명의 사상은 상기 설명된 실시예에 국한되어 정해져서는 아니 되며, 후술하는 특허청구범위뿐만 아니라 이 특허청구범위와 균등하게 또는 등가적으로 변형된 모든 것들은 본 발명의 사상의 범주에 속한다고 할 것이다.

부호의 설명

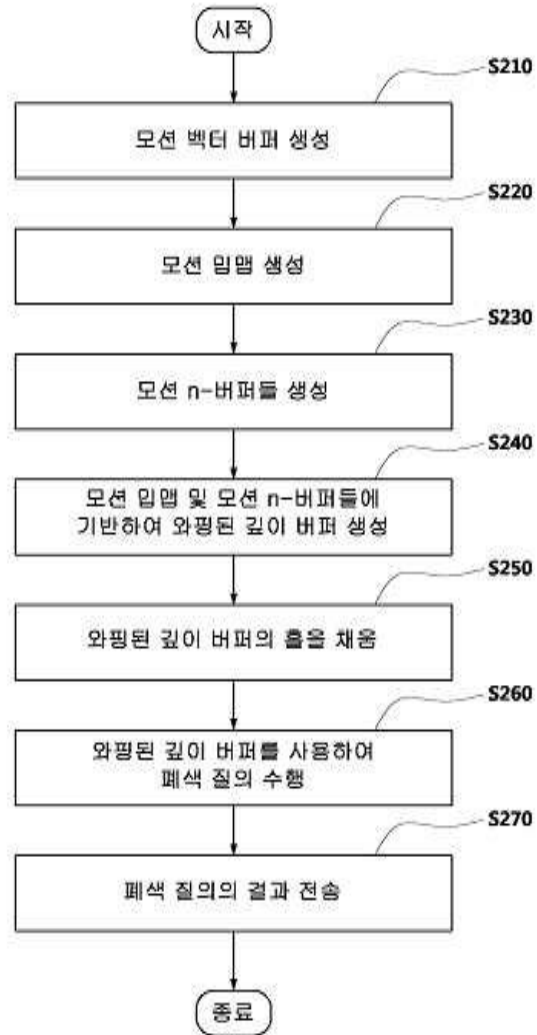
[0106] 100: GPU
 110: 깊이 버퍼 렌더링부
 120: 모션 벡터 버퍼 렌더링부
 130: 모션 맵 구성부
 140: 모션 버퍼 구성부
 150: 깊이 버퍼 와핑부
 160: 홀 필링부
 170: 폐색 질의부
 180: 통신부

도면

도면1



도면2



도면3a



(a) RGB_t

도면3b



(b) Z_t

도면3c



(c) Z_{t+1}



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년08월22일
(11) 등록번호 10-1431715
(24) 등록일자 2014년08월12일

(51) 국제특허분류(Int. Cl.)
G06T 1/00 (2006.01) G06T 5/00 (2006.01)
(21) 출원번호 10-2013-0094771
(22) 출원일자 2013년08월09일
심사청구일자 2013년08월09일
(56) 선행기술조사문헌
KR1020060031519 A
KR1020070121440 A
KR101101441 B1

(73) 특허권자
성균관대학교산학협력단
경기도 수원시 장안구 서부로 2066, 성균관대학교
내 (천천동)
(72) 발명자
이성길
경기도 수원시 장안구 정자동
정주현
서울 강서구 우장산로4길 25, 1004동 407호 (내발
산동, 청솔샤르망아파트)
(74) 대리인
에스앤아이피특허법인

전체 청구항 수 : 총 15 항

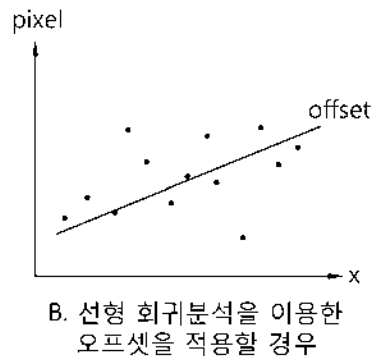
심사관 : 이주미

(54) 발명의 명칭 선형 회귀 분석을 이용한 합산 영역 테이블의 정밀도 향상을 위한 방법 및 장치 및 이를 이용한 영상 이미지 복원을 위한 필터링 방법 및 장치

(57) 요약

본 발명의 영상의 현재 픽셀 주변의 임의의 블록에 대한 산술적 연산을 수행하기 위한 합산 영역 테이블(SAT: Summed Area Table)의 정밀도를 향상시키기 위한 방법은 선형 회귀 분석을 통해 생성된 다항식을 이용하여 오프셋을 상기 블록 내의 각 픽셀마다 적용하는 오프셋 적용 단계 및 상기 적용된 오프셋을 기반으로 상기 합산 영역 테이블을 생성하는 테이블 생성 단계를 포함한다. 따라서, 합산 영역 테이블 생성시, 부동 소수점 표현 방식의 정밀도가 향상이 되어 원본 이미지와 복원한 이미지 사이에 오차를 줄일 수 있다.

대표도 - 도2



이 발명을 지원한 국가연구개발사업

과제고유번호	1345192513
부처명	교육과학기술부
연구사업명	중견연구자지원
연구과제명	하이레벨 GPU 소프트웨어 모델링 및 디자인 인터페이스
기 여 율	70/100
주관기관	성균관대학교(자연과학캠퍼스)
연구기간	2012.09.01 ~ 2013.08.31

이 발명을 지원한 국가연구개발사업

과제고유번호	1345180738
부처명	교육과학기술부
연구사업명	글로벌프론티어사업
연구과제명	실시간 3D 그래픽스 구현을 위한 병렬처리형 소프트웨어 라이브러리 개발
기 여 율	30/100
주관기관	성균관대학교(자연과학캠퍼스)
연구기간	2012.09.01 ~ 2013.08.31

특허청구의 범위

청구항 1

영상의 현재 픽셀 주변의 임의의 블록에 대한 산술적 연산을 수행하기 위한 합산 영역 테이블(SAT: Summed Area Table)의 정밀도를 향상시키기 위한 방법에 있어서,

영상 내의 픽셀 값에 대한 선형 회귀 분석을 통해 생성된 다항식을 이용하여 오프셋을 상기 블록 내의 각 픽셀마다 적용하는 오프셋 적용 단계; 및

상기 적용된 오프셋을 기반으로 상기 합산 영역 테이블을 생성하는 테이블 생성 단계를 포함하는 것을 특징으로 하는 합산 영역 테이블의 정밀도 향상 방법.

청구항 2

제 1 항에 있어서, 상기 테이블 생성 단계는

상기 오프셋이 적용된 상기 블록 내 각 픽셀의 값을 기반으로 재귀적 곱셈 연산(recursive doubling) 알고리즘을 이용하여 합산 영역 테이블을 생성하는 단계를 포함하는 것을 특징으로 하는 합산 영역 테이블의 정밀도 향상 방법.

청구항 3

제 1 항에 있어서, 상기 테이블 생성 단계는

상기 블록 내의 각 행마다 수평 방향으로 제 1 프리픽스 합산(prefix sum)을 수행하여 저장하는 수평 단계; 및

상기 블록 내의 각 열마다 수직 방향으로 제 2 프리픽스 합산을 수행하여 상기 합산 영역 테이블을 생성하는 단계를 포함하는 것을 특징으로 하는 합산 영역 테이블의 정밀도 향상 방법.

청구항 4

제 1 항에 있어서, 상기 오프셋 적용 단계는

상기 현재 픽셀 주변의 임의의 블록을 정의하는 단계;

상기 정의된 블록 내에 포함된 픽셀들에 대해 선형 회귀 분석을 통해 생성된 다항식을 오프셋으로 적용하는 단계를 포함하는 것을 특징으로 하는 합산 영역 테이블의 정밀도 향상 방법.

청구항 5

제 1 항에 있어서,

상기 다항식은 1차 또는 2차 이상의 고차 다항식을 포함하는 것을 특징으로 하는 합산 영역 테이블의 정밀도 향상 방법.

청구항 6

제 1 항에 있어서, 상기 오프셋 적용 단계는

상기 블록 내의 각 픽셀의 R(Red) 정보, G(Green) 정보 및 B(Blue) 정보에 대해 각각 서로 다른 다항식을 적용하는 단계를 포함하는 것을 특징으로 하는 합산 영역 테이블의 정밀도 향상 방법.

청구항 7

영상의 현재 픽셀 주변의 임의의 블록에 대한 산술적 연산을 수행하기 위한 합산 영역 테이블(SAT: Summed Area Table)의 정밀도를 향상시키기 위한 장치에 있어서,

영상 내의 픽셀 값에 대한 선형 회귀 분석을 통해 생성된 다항식을 이용하여 오프셋을 상기 블록 내의 각 픽셀마다 적용하는 오프셋 적용부; 및

상기 적용된 오프셋을 기반으로 상기 합산 영역 테이블을 생성하는 테이블 생성부를 포함하는 것을 특징으로 하는 합산 영역 테이블의 정밀도 향상 장치.

청구항 8

영상 이미지를 복원하기 위한 필터링 방법에 있어서,

상기 영상 이미지의 현재 픽셀 주변의 임의의 블록에 대한 산술적 연산을 수행하기 위한 합산 영역 테이블(SAT: Summed Area Table)을 생성하는 합산 영역 테이블 생성 단계; 및

상기 합산 영역 테이블을 기반으로 상기 블록 내 픽셀 값의 평균을 계산하여 필터링을 수행하는 필터링 수행 단계를 포함하되, 상기 합산 영역 테이블 생성 단계는,

영상 내의 픽셀 값에 대한 선형 회귀 분석을 통해 생성된 다항식을 이용하여 오프셋을 상기 블록 내의 각 픽셀마다 적용하는 오프셋 적용 단계; 및

상기 적용된 오프셋을 기반으로 상기 합산 영역 테이블을 생성하는 테이블 생성 단계를 포함하는 것을 특징으로 하는 영상 이미지 복원을 위한 필터링 방법.

청구항 9

제 8 항에 있어서, 상기 필터링 수행 단계는

상기 블록과 관련된 합산 영역 테이블의 우측 최상단 픽셀값, 우측 최하단 픽셀값, 좌측 최상단 픽셀값 및 좌측 최하단 픽셀값을 이용하여 상기 블록 내 픽셀 값의 평균을 계산하는 단계를 포함하는 것을 특징으로 하는 영상 이미지 복원을 위한 필터링 방법.

청구항 10

제 9 항에 있어서, 상기 테이블 생성 단계는

상기 오프셋이 적용된 상기 블록 내 각 픽셀의 값을 기반으로 재귀적 곱셈 연산(recursive doubling) 알고리즘을 이용하여 합산 영역 테이블을 생성하는 단계를 포함하는 것을 특징으로 하는 영상 이미지 복원 방법.

청구항 11

제 9 항에 있어서, 상기 테이블 생성 단계는

상기 블록 내의 각 행마다 수평 방향으로 제 1 프리픽스 합산(prefix sum)을 수행하여 저장하는 수평 단계; 및

상기 블록 내의 각 열마다 수직 방향으로 제 2 프리픽스 합산을 수행하여 상기 합산 영역 테이블을 생성하는 단계를 포함하는 것을 특징으로 하는 영상 이미지 복원을 위한 필터링 방법.

청구항 12

제 9 항에 있어서, 상기 오프셋 적용 단계는

상기 현재 픽셀 주변의 임의의 블록을 정의하는 단계;

상기 정의된 블록 내에 포함된 픽셀들에 대해 선형 회귀 분석을 통해 생성된 다항식을 오프셋으로 적용하는 단계를 포함하는 것을 특징으로 하는 영상 이미지 복원을 위한 필터링 방법.

청구항 13

제 9 항에 있어서,

상기 다항식은 1차 또는 2차 이상의 고차 다항식을 포함하는 것을 특징으로 하는 영상 이미지 복원을 위한 필터링 방법.

청구항 14

제 9 항에 있어서, 상기 오프셋 적용 단계는

상기 블록 내의 각 픽셀의 R(Red) 정보, G(Green) 정보 및 B(Blue) 정보에 대해 각각 서로 다른 다항식을 적용하는 단계를 포함하는 것을 특징으로 하는 영상 이미지 복원을 위한 필터링 방법.

청구항 15

영상 이미지를 복원하기 위한 필터링 장치에 있어서,

상기 영상 이미지의 현재 픽셀 주변의 임의의 블록에 대한 산술적 연산을 수행하기 위한 합산 영역 테이블(SAT: Summed Area Table)을 생성하는 합산 영역 테이블 생성부; 및

상기 합산 영역 테이블을 기반으로 상기 블록 내 픽셀 값의 평균을 계산하여 필터링을 수행하는 필터링 수행부를 포함하되, 상기 합산 영역 테이블 생성부는,

영상 내의 픽셀 값에 대한 선형 회귀 분석을 통해 생성된 다항식을 이용하여 오프셋을 상기 블록 내의 각 픽셀마다 적용하는 오프셋 적용부; 및

상기 적용된 오프셋을 기반으로 상기 합산 영역 테이블을 생성하는 테이블 생성부를 포함하는 것을 특징으로 하는 영상 이미지 복원을 위한 필터링 장치.

명세서

기술분야

[0001] 본 발명은 합산 영역 테이블의 정밀도 향상 방법 및 장치에 관한 것으로, 보다 상세하게는 임의의 사각 영역의 평균을 픽셀을 모두 읽지 않고 표시할 수 있는 합산 영역 테이블의 정밀도를 향상시키기 위한 방법 및 장치에 관한 것이다.

배경기술

[0002] 합산 영역 테이블(Summed Area Table)은 이미지 처리에서 사각 필터링(filtering)을 효율적으로 적용하기 위한 자료구조이다. 현재 픽셀 주변으로 임의의 사각 영역의 평균을 모든 픽셀을 읽을 필요 없이, 단 4번의 픽셀 값의 합과 차로 표시할 수 있다. 합산 영역 테이블은 사각 필터링을 필요로 하는 이미지 블러(blur), 필터링 등의 분야에 쓰이고 있다.

[0003] 그러나, 합산 영역 테이블은 좌측 아래(또는 우측 위)의 방향으로 영역의 합을 재귀적으로 더해서 저장하는데,

이 경우 많은 픽셀의 값이 누적되는 경우 보통 쓰이는 IEEE 754 방식의 32비트 부동 소수점 표현의 정밀도가 떨어지는 문제점이 있을 수 있다. 이에, 색상의 강도가 높거나 해상도가 높은 이미지를 이용해 합산 영역 테이블을 생성할 경우, 데이터가 손실되어 이미지를 복원할 시 일부 픽셀이 원본과 다른 문제가 발생한다.

- [0004] 이런 문제를 극복하기 위해 종래 연구에서는 이미지에 오프셋(offset)을 적용하여 합산 영역 테이블을 생성하는 방법이 제안되었다. 그러나, 모든 픽셀에 대해 같은 오프셋을 사용하기 때문에 여전히 고해상도 이미지나 색상 분포에 따른 문제점을 해결하지 못해 모든 이미지에 대해 정밀도를 보장해주지 못하는 문제점이 있다.

발명의 내용

해결하려는 과제

- [0005] 상술한 문제점을 해결하기 위한 본 발명의 목적은 선형 회귀분석(linear regression)을 이용하여 오프셋을 사용함으로써 합산 영역 테이블의 정밀도를 향상시킬 수 있는 방법 및 장치, 및 이를 이용한 영상 이미지 복원을 위한 필터링 방법 및 장치를 제공하는 것이다.

과제의 해결 수단

- [0006] 상기한 목적을 달성하기 위한 본 발명의 영상의 현재 픽셀 주변의 임의의 블록에 대한 산술적 연산을 수행하기 위한 합산 영역 테이블(SAT: Summed Area Table)의 정밀도를 향상시키기 위한 방법은 선형 회귀 분석을 통해 생성된 다항식을 이용하여 오프셋을 상기 블록 내의 각 픽셀마다 적용하는 오프셋 적용 단계 및 상기 적용된 오프셋을 기반으로 상기 합산 영역 테이블을 생성하는 테이블 생성 단계를 포함할 수 있다.
- [0007] 상기 테이블 생성 단계는 상기 오프셋이 적용된 상기 블록 내 각 픽셀의 값을 기반으로 재귀적 곱셈 연산(recursive doubling) 알고리즘을 이용하여 합산 영역 테이블을 생성하는 단계를 포함할 수 있다.
- [0008] 상기 테이블 생성 단계는 상기 블록 내의 각 행마다 수평 방향으로 제 1 프리픽스 합산(prefix sum)을 수행하여 저장하는 수평 단계 및 상기 블록 내의 각 열마다 수직 방향으로 제 2 프리픽스 합산을 수행하여 상기 합산 영역 테이블을 생성하는 단계를 포함할 수 있다.
- [0009] 상기 오프셋 적용 단계는 상기 현재 픽셀 주변의 임의의 블록을 정의하는 단계 및 상기 정의된 블록 내에 포함된 픽셀들에 대해 선형 회귀 분석을 통해 생성된 다항식을 오프셋으로 적용하는 단계를 포함할 수 있다.
- [0010] 상기 다항식은 1차 또는 2차 이상의 고차 다항식을 포함할 수 있다.
- [0011] 상기 오프셋 적용 단계는 상기 블록 내의 각 픽셀의 R(Red) 정보, G(Green) 정보 및 B(Blue) 정보에 대해 각각 서로 다른 다항식을 적용하는 단계를 포함할 수 있다.
- [0012] 상기한 목적을 달성하기 위한 본 발명의 영상의 현재 픽셀 주변의 임의의 블록에 대한 산술적 연산을 수행하기 위한 합산 영역 테이블(SAT: Summed Area Table)의 정밀도를 향상시키기 위한 장치는 선형 회귀 분석을 통해 생성된 다항식을 이용하여 오프셋을 상기 블록 내의 각 픽셀마다 적용하는 오프셋 적용부 및 상기 적용된 오프셋을 기반으로 상기 합산 영역 테이블을 생성하는 테이블 생성부를 포함할 수 있다.
- [0013] 상기한 목적을 달성하기 위한 본 발명의 영상 이미지를 복원하기 위한 필터링 방법은 상기 영상 이미지의 현재 픽셀 주변의 임의의 블록에 대한 산술적 연산을 수행하기 위한 합산 영역 테이블(SAT: Summed Area Table)을 생성하는 합산 영역 테이블 생성 단계 및 상기 합산 영역 테이블을 기반으로 상기 블록 내 픽셀 값의 평균을 계산하여 필터링을 수행하는 필터링 수행 단계를 포함하되, 상기 합산 영역 테이블 생성 단계는 선형 회귀 분석을 통해 생성된 다항식을 이용하여 오프셋을 상기 블록 내의 각 픽셀마다 적용하는 오프셋 적용 단계 및 상기 적용된 오프셋을 기반으로 상기 합산 영역 테이블을 생성하는 테이블 생성 단계를 포함할 수 있다.
- [0014] 상기 필터링 수행 단계는 상기 블록과 관련된 합산 영역 테이블의 우측 최상단 픽셀값, 우측 최하단 픽셀값, 좌측 최상단 픽셀값 및 좌측 최하단 픽셀값을 이용하여 상기 블록 내 픽셀 값의 평균을 계산하는 단계를 포함할 수 있다.
- [0015] 상기 테이블 생성 단계는 상기 오프셋이 적용된 상기 블록 내 각 픽셀의 값을 기반으로 재귀적 곱셈 연산

(recursive doubling) 알고리즘을 이용하여 합산 영역 테이블을 생성하는 단계를 포함할 수 있다.

- [0016] 상기 테이블 생성 단계는 상기 블록 내의 각 행마다 수평 방향으로 제 1 프리픽스 합산(prefix sum)을 수행하여 저장하는 수평 단계 및 상기 블록 내의 각 열마다 수직 방향으로 제 2 프리픽스 합산을 수행하여 상기 합산 영역 테이블을 생성하는 단계를 포함할 수 있다.
- [0017] 상기 오프셋 적용 단계는 상기 현재 픽셀 주변의 임의의 블록을 정의하는 단계 및 상기 정의된 블록 내에 포함된 픽셀들에 대해 선형 회귀 분석을 통해 생성된 다항식을 오프셋으로 적용하는 단계를 포함할 수 있다.
- [0018] 상기 다항식은 1차 또는 2차 이상의 고차 다항식을 포함할 수 있다.
- [0019] 상기 오프셋 적용 단계는 상기 블록 내의 각 픽셀의 R(Red) 정보, G(Green) 정보 및 B(Blue) 정보에 대해 각각 서로 다른 다항식을 적용하는 단계를 포함할 수 있다.
- [0020] 상기한 목적을 달성하기 위한 본 발명의 영상 이미지를 복원하기 위한 필터링 장치는 상기 영상 이미지의 현재 픽셀 주변의 임의의 블록에 대한 산술적 연산을 수행하기 위한 합산 영역 테이블(SAT: Summed Area Table)을 생성하는 합산 영역 테이블 생성부 및 상기 합산 영역 테이블을 기반으로 상기 블록 내 픽셀 값의 평균을 계산하여 필터링을 수행하는 필터링 수행부를 포함하되, 상기 합산 영역 테이블 생성부는 선형 회귀 분석을 통해 생성된 다항식을 이용하여 오프셋을 상기 블록 내의 각 픽셀마다 적용하는 오프셋 적용부 및 상기 적용된 오프셋을 기반으로 상기 합산 영역 테이블을 생성하는 테이블 생성부를 포함할 수 있다.

발명의 효과

- [0021] 본 발명의 선형 회귀 분석을 이용한 합산 영역 테이블의 정밀도 향상을 위한 방법 및 장치, 및 이를 이용한 영상 이미지 복원을 위한 필터링 방법 및 장치에 따르면, 합산 영역 테이블 생성시, 부동 소수점 표현 방식의 정밀도가 향상이 되어 원본 이미지와 복원한 이미지 사이에 오차를 줄이는 효과가 있다.

도면의 간단한 설명

- [0022] 도 1은 종래 방식에 따라 고정 오프셋을 적용한 경우를 나타내는 그래프,
- 도 2는 본 발명의 일 실시예에 따라서, 선형 회귀분석을 이용한 오프셋을 적용한 경우를 나타내는 그래프,
- 도 3은 본 발명의 일 실시예에 따른 영상 이미지 복원을 위한 필터링 장치를 개략적으로 나타낸 블록도,
- 도 4는 본 발명의 일 실시예에 따른 영상 이미지 복원을 위한 필터링 장치의 합산 영역 테이블 생성부를 구체적으로 나타낸 상세블록도,
- 도 5는 본 발명의 일 실시예에 따른 영상 이미지 복원을 위한 필터링 장치의 오프셋 적용부를 구체적으로 나타낸 상세블록도,
- 도 6은 1차원 배열에 대한 재귀적 곱셈 연산 알고리즘을 설명하기 위한 개념도,
- 도 7은 합산 영역 테이블을 이용하여 현재 블록의 픽셀값의 평균을 구하는 방법을 설명하기 위한 도면,
- 도 8은 원본 이미지와 본 발명의 일 실시예에 따른 영상 이미지 복원을 위한 필터링 방법을 사용하여 복원한 이미지를 비교한 도면이다.

발명을 실시하기 위한 구체적인 내용

- [0023] 본 발명은 다양한 변형을 가할 수 있고 여러 가지 실시예를 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 상세하게 설명하고자 한다.
- [0024] 그러나, 이는 본 발명을 특정한 실시 형태에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.
- [0025] 제 1, 제 2 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 사용

된다. 예를 들어, 본 발명의 권리 범위를 벗어나지 않으면서 제 1 구성요소는 제 2 구성요소로 명명될 수 있고, 유사하게 제 2 구성요소도 제 1 구성요소로 명명될 수 있다. 및/또는이라는 용어는 복수의 관련된 기재된 항목들의 조합 또는 복수의 관련된 기재된 항목들 중의 어느 항목을 포함한다.

- [0026] 어떤 구성요소가 다른 구성요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성요소가 다른 구성요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는, 중간에 다른 구성요소가 존재하지 않는 것으로 이해되어야 할 것이다.
- [0027] 본 출원에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 출원에서, "포함하다" 또는 "가지다" 등의 용어는 명세서상에 기재된 특징, 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.
- [0028] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가지고 있다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 가진 것으로 해석되어야 하며, 본 출원에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.
- [0029] 이하, 첨부한 도면들을 참조하여, 본 발명의 바람직한 실시예를 보다 상세하게 설명하고자 한다. 본 발명을 설명함에 있어 전체적인 이해를 용이하게 하기 위하여 도면상의 동일한 구성요소에 대해서는 동일한 참조부호를 사용하고 동일한 구성요소에 대해서 중복된 설명은 생략한다.
- [0030] 본 발명의 일 실시예에 따르면, 영상의 현재 픽셀을 복원할 때, 픽셀 복원의 정밀도를 높이기 위해 합산 영역 테이블을 이용할 수 있다. 합산 영역 테이블(Summed Area Table)은 이미지 처리에서 사각 필터링을 효율적으로 적용하기 위한 자료구조로서, 이를 이용하면 효율적으로 특정 블록의 평균 픽셀 값을 구할 수 있고, 상기 블록의 평균 픽셀 값을 통해 현재 픽셀의 픽셀 값을 산출할 수 있다.
- [0031] 도 1은 종래 방식에 따라 고정 오프셋을 적용한 경우를 나타내는 그래프이다.
- [0032] 도 1을 참조하면, 1차원 배열의 경우에 대해 같은 오프셋을 적용한 경우를 나타낸다. 즉, 가로축은 픽셀들의 위치이고, 세로축은 픽셀 값을 나타낸다. 픽셀들에게 모두 동일한 오프셋 값을 적용하였고, 이에 따라 고정된 오프셋으로부터 멀리 떨어진 픽셀의 경우 픽셀 값이 커지고, 이를 그대로 부동 소수점 표현으로 가져갔을 시 정밀도가 떨어지게 된다.
- [0033] 도 2는 본 발명의 일 실시예에 따라서, 선형 회귀분석을 이용한 오프셋을 적용한 경우를 나타내는 그래프이다.
- [0034] 도 2를 참조하면, 도 1의 고정 오프셋을 사용한 경우와는 다르게 각 픽셀의 값을 기반으로 하는 다항식을 이용하여 각 픽셀별로 서로 다른 오프셋을 제공한다. 이때, 선형 회귀 분석(linear regression)이 이용할 수 있다. 본 발명의 일 실시예에 따르면, 각 픽셀별로 해당 값에 최대한 가까운 값을 오프셋으로 적용할 수 있다. 이러한 방식으로 오프셋을 적용할 경우, 모든 픽셀에 대해 동일한 값을 적용하는 방식보다 더 많은 픽셀이 0에 수렴하기 될 수 있다. 이를 통해 합산 영역 테이블을 생성할 때 총합이 부동소수점 표현 방식의 한계 범위를 초과하지 않는 효과를 기대할 수 있다. 또한, 상기한 방식으로 픽셀마다 각각 다른 오프셋을 적용하면, 별도의 텍스처를 생성할 필요가 없어 텍스처 샘플링 없이 원본 픽셀을 빠르게 복원할 수 있다.
- [0035] 도 3은 본 발명의 일 실시예에 따른 영상 이미지 복원을 위한 필터링 장치를 개략적으로 나타낸 블록도이다. 도 3에 도시된 바와 같이, 본 발명의 일 실시예에 따른 필터링 장치는 합산 영역 테이블 생성부(310) 및 필터링 수행부(320)를 포함할 수 있다.
- [0036] 도 3을 참조하면, 합산 영역 테이블 생성부(310)는 영상 이미지의 현재 픽셀 주변의 임의의 블록에 대해 산술적 연산을 수행하기 위해 합산 영역 테이블(SAT)을 생성한다. 전술한 바와 같이, 합산 영역 테이블 생성시 선형 회귀 분석을 통한 다항식이 이용될 수 있다.
- [0037] 필터링 수행부(320)는 합산 영역 테이블 생성부(310)를 통해 생성된 합산 영역 테이블을 기반으로 블록 내의 픽

셀 값의 평균을 계산하여 필터링을 수행한다. 필터링 수행부(320)는 현재 픽셀 주변의 블록에 대한 합산 영역 테이블의 특정 위치의 테이블 값만을 이용하여 블록의 평균 픽셀 값을 산출할 수 있고, 이를 통해 필터링을 수행할 수 있다.

[0038] 도 4는 본 발명의 일 실시예에 따른 영상 이미지 복원을 위한 필터링 장치의 합산 영역 테이블 생성부(310)를 구체적으로 나타낸 상세블록도이다. 도 4에 도시된 바와 같이, 본 발명의 일 실시예에 따른 합산 영역 테이블 생성부(310)는 오프셋 적용부(410) 및 테이블 생성부(420)를 포함할 수 있다.

[0039] 오프셋 적용부(410)는 현재 픽셀 주변 블록의 각각의 픽셀에 대해 선형 회귀 분석을 통해 생성된 다항식을 오프셋으로 적용한다. 이때, 각 픽셀마다 채널별로 선형 회귀 분석을 통해 생성된 다항식을 오프셋으로 적용할 수 있다.

[0040] 도 5는 본 발명의 일 실시예에 따른 영상 이미지 복원을 위한 필터링 장치의 오프셋 적용부(410)를 구체적으로 나타낸 상세블록도이다. 도 5에 도시된 바와 같이, 오프셋 적용부(410)는 블록 정의부(510) 및 다항식 생성부(520)를 포함할 수 있다.

[0041] 도 5를 참조하면, 블록 정의부(510)는 현재 픽셀 주변의 임의의 블록을 정의한다. 임의의 블록은 현재 픽셀을 중심으로 하여 크기가 다른 여러 개의 블록을 의미할 수 있다. 또한, 블록은 현재 픽셀을 중심으로 정사각형 또는 직사각형 블록일 수 있다.

[0042] 다항식 생성부(520)는 정의된 블록 내에 포함된 복수 개의 픽셀에 대해, 각 픽셀마다 적용될 수 있는 오프셋 관련 다항식을 생성한다. 각 픽셀에 대해 적용되는 다항식을 다음과 같은 수학식을 이용하여 생성될 수 있다.

수학식 1

$$r(x, y) = a_r x + b_r y + c_r$$

$$g(x, y) = a_g x + b_g y + c_g$$

$$b(x, y) = a_b x + b_b y + c_b$$

[0043]

[0044] 여기서, x와 y는 각각의 이미지의 x축 및 y축 좌표이다. 각 픽셀별로 오프셋을 적용시키기 위해 픽셀마다 채널별로 선형 회귀 분석을 통해 생성되는 다항식을 오프셋으로 적용할 수 있다.

[0045] 본 발명의 일 실시예에 따르면, 다항식 생성부(520)는 각 블록에 포함된 픽셀들의 각각의 좌표에 대한 픽셀값을 기준으로 선형 회귀 분석을 수행하여 다항식을 도출한다. 다항식 생성부(520)는 각각의 휘도 값 또는 색상 값에 대해 서로 다른 다항식을 도출할 수 있다. 또한, 색상 값에 대해서도, R(Red), G(Green) 및 B(Blue) 색상 값에 대해 서로 다른 다항식을 도출할 수 있다. 또한, 다항식 생성부(520)는 각각의 픽셀에 대한 오프셋 관련 다항식을 1차 다항식뿐만 아니라 고차 다항식을 통해 수행할 수 있다. 즉, 1차 다항식으로 회귀 분석을 하는 것뿐만 아니라 고차 다항 회귀 분석을 통해 오프셋을 적용할 수도 있다.

[0046] 다항식 생성부(520)는 생성된 다항식을 각 픽셀에 적용시킬 수 있다. 이에 따라 각 픽셀마다 변경된 픽셀 값을 가질 수 있다. 그리고, 다항과 관련된 오프셋 정보는 이후, 테이블 생성부(420) 및 필터링 수행부(320)에 전송될 수 있다.

[0047] 다시 도 4로 돌아가서, 테이블 생성부(420)는 오프셋 적용부(410)에서 적용된 오프셋을 기반으로 합산 영역 테이블을 생성한다. 테이블 생성부(420)는 오프셋을 적용한 이미지를 통해 합산 영역 테이블을 생성하는데, 이때 재귀적 곱셈(recursive doubling) 알고리즘을 사용할 수 있다.

[0048] 도 6은 1차원 배열에 대한 재귀적 곱셈 연산 알고리즘을 설명하기 위한 개념도이다. 도 6은 1차원 배열에 대한 재귀적 곱셈 알고리즘을 도시한다.

[0049] 도 6을 참조하면, 테이블 생성부(420)는 오프셋이 적용된 각각의 픽셀의 픽셀 값을 픽셀 배치에 맞춰 가지고 있고, 이를 기반으로 수평 및 수직 단계를 수행하여 테이블을 생성한다. 테이블 생성부(420)는 텍스처 샘플링의 효율성을 위해 수평 단계에선 각 행마다 수평 방향으로 프리픽스 썸(prefix sum)을 수행하여 저장한다. 그리고는 수직 단계에서 각 열마다 수직 방향으로 프리픽스 썸을 수행한다. 각 단계의 패스 수는 수평 단계에서

$\log_2(\text{width})$ 이고, 수직 단계에서는 $\log_2(\text{height})$ 이다. 즉, 프리픽스 섬의 반복 수행을 통해 2^n 크기의 테이블을 생성할 수 있다. 테이블 생성부(420)는 본 방식을 이용하여 픽셀마다 서로 다른 오프셋을 적용하면, 별도의 텍스처를 생성할 필요가 없어 텍스처 샘플링 없이 원본 픽셀을 빠르게 복원할 수 있다.

[0050] 다시 도 3로 돌아가서, 합산 영역 테이블 생성부(310)에서의 테이블 생성이 완료되면, 필터링 수행부(320)는 합산 영역 테이블을 이용하여 필터링을 수행한다. 이때, 임의의 블록 내의 픽셀 값의 평균을 합산 영역 테이블을 이용하여 쉽게 산출할 수 있다.

[0051] 도 7은 합산 영역 테이블을 이용하여 현재 블록의 픽셀값의 평균을 구하는 방법을 설명하기 위한 도면이다.

[0052] 도 7을 참조하면, 필터링 수행부(320)는 다음의 수학적식을 이용하여, 필터링을 수행할 수 있다.

수학적식 2

$$average = \frac{UR - UL - LR + LL}{w * h}$$

[0053]

[0054] 여기서, average는 해당 블록 내의 픽셀값의 평균, UR은 우측 상단 코너의 값(710), UL은 좌측 상단 코너의 값(740), LR은 우측 하단 코너의 값(720), LL은 좌측 하단 코너의 값(730)을 의미할 수 있다. w는 블록의 폭을, h는 블록의 높이를 의미할 수 있다. 이렇게 4번의 텍스처 룩업을 통해 해당 블록의 픽셀 값 평균 값을 구할 수 있고, 이에 많은 픽셀 값이 누적되어도 4개의 텍스처 룩업만을 수행하여 평균 값을 산출할 수 있으므로, 정밀도가 떨어지지 않을 수 있다.

[0055] 필터링 수행부(320)는 이렇게 생성된 평균 값을 이용하여 이미지 블러, 필터링 등의 사각 영역 필터링을 수행할 수 있다. 필터링 수행부(320)는 본 방식을 이용하여 픽셀마다 서로 다른 오프셋을 적용하여 생성된 테이블을 이용하면, 별도의 텍스처를 생성할 필요가 없어 텍스처 샘플링 없이 원본 픽셀을 빠르게 복원할 수 있다. 이때, 추가적인 계산을 줄이기 위해, 정적분을 이용하여 임의의 블록(즉, 사각 영역)의 오프셋을 빠르게 복구할 수 있다.

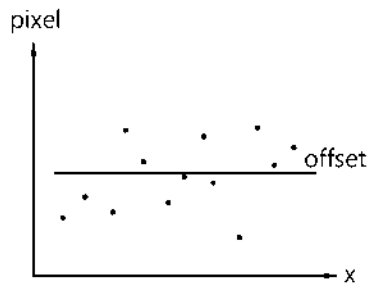
[0056] 도 8은 원본 이미지와 본 발명의 일 실시예에 따른 영상 이미지 복원을 위한 필터링 방법을 사용하여 복원한 이미지를 비교한 도면이다.

[0057] 도 8을 참조하면, 본 발명의 필터링 방법의 성능 시험과 관련하여, 정밀도 오차를 측정하기 위해, 원본 이미지와 합산 영역 테이블을 이용하여 복원한 이미지를 비교하였다. 측정 오차는 PSNR, SSIM의 두 대표적 방법을 이용하여 측정되었다. 결과는 PSNR은 93.87 dB이고, SSIM은 1.0이다. 복원 결과, 오차가 매우 작은 것을 알 수 있고, 특히 SSIM은 1에 매우 가까움을 확인할 수 있다.

[0058] 이상 도면 및 실시예를 참조하여 설명하였지만, 본 발명의 보호범위가 상기 도면 또는 실시예에 의해 한정되는 것을 의미하지는 않으며 해당 기술 분야의 숙련된 당업자는 하기의 특허 청구의 범위에 기재된 본 발명의 사상 및 영역으로부터 벗어나지 않는 범위 내에서 본 발명을 다양하게 수정 및 변경시킬 수 있음을 이해할 수 있을 것이다.

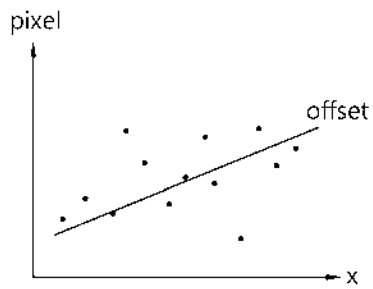
도면

도면1



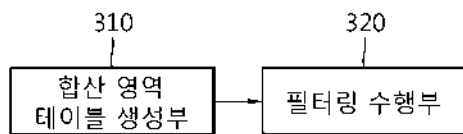
A. 고정 오프셋을 적용할 경우

도면2

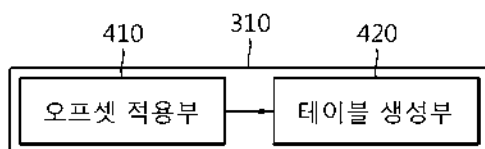


B. 선형 회귀분석을 이용한
오프셋을 적용할 경우

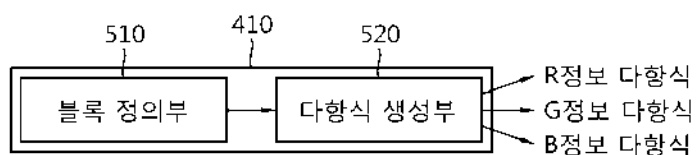
도면3



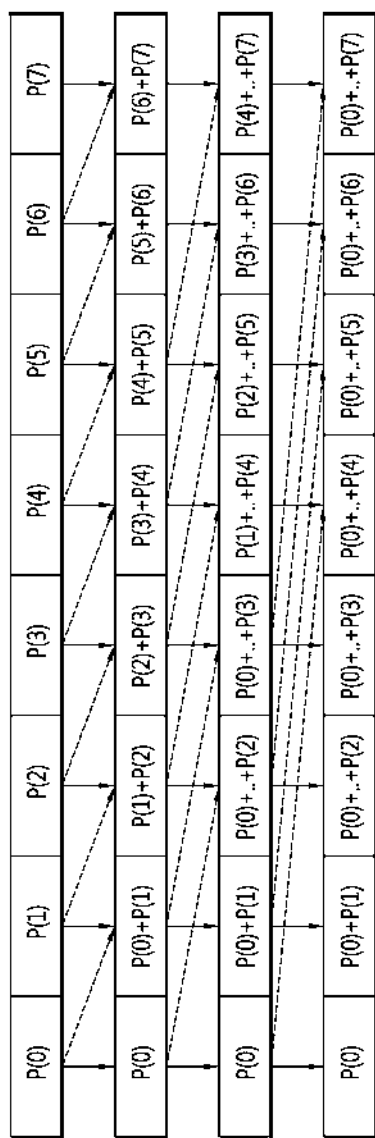
도면4



도면5



도면6



도면7

740	2	1	2	3	0	710
	4	3	-2	0	-1	
	-1	-2	4	-3	-1	
	1	3	-2	2	1	
730	-3	0	0	1	3	720

도면8

Original

From SAT



PSNR: 93.87 dB SSIM: 1.0



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2018년02월08일
(11) 등록번호 10-1827395
(24) 등록일자 2018년02월02일

(51) 국제특허분류(Int. Cl.)
G06F 9/54 (2018.01) G06F 12/02 (2018.01)
G06F 17/30 (2006.01) G06T 1/20 (2018.01)
(52) CPC특허분류
G06F 9/54 (2013.01)
G06F 12/023 (2013.01)
(21) 출원번호 10-2017-0022361
(22) 출원일자 2017년02월20일
심사청구일자 2017년02월20일
(56) 선행기술조사문헌
KR101465447 B1
(뒷면에 계속)

(73) 특허권자
성균관대학교 산학협력단
경기도 수원시 장안구 서부로 2066 (천천동, 성균관대학교내)
재단법인 실감교류인체감응솔루션연구단
서울특별시 성북구 화랑로14길 5, 국제협력관 (하월곡동, 한국과학기술연구원)
(72) 발명자
이성길
경기도 수원시 장안구 서부로 2066 천천동 300 성균관대학교자연과학캠퍼스 27328호
조성훈
경기도 수원시 장안구 서부로 2066 천천동 300 성균관대학교자연과학캠퍼스 27336호
(뒷면에 계속)
(74) 대리인
김인철

전체 청구항 수 : 총 13 항

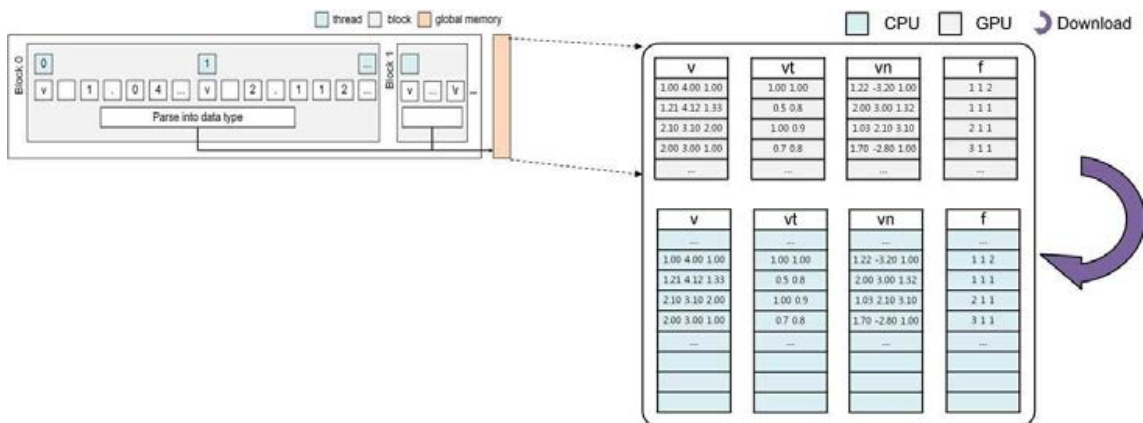
심사관 : 유진태

(54) 발명의 명칭 확장성 있는 GPU기반의 대용량 OBJ 파일 처리 방법 및 장치

(57) 요약

본 명세서에 따른 OBJ 파일 처리 방법은 out-of-core 방식으로 대용량 파일을 일정 단위 크기로 나누고, GPU 병렬 처리를 통해 순차적으로 파싱(parsing)할 수 있다.

대표도 - 도4



- (52) CPC특허분류
G06F 17/30159 (2013.01)
G06T 1/20 (2013.01)
- (72) 발명자
정유나
 경기도 수원시 장안구 서부로 2066 천천동 300 성
 균관대학교자연과학캠퍼스 27336호
- 유범재**
 서울특별시 서초구 서초중앙로 188, B동 2003호 (서초동, 아크로비스타)
- (56) 선행기술조사문헌
 KR1020160087127 A
 Aidan L. Possemiers 외 1명. 'Fast OBJ file importing and parsing in CUDA'. Computational Visual Media, Vol. 1, No. 3, 2015.09, pp.229-238.
 JP2007133576 A
 US6184908 B2
 KR1020090007926 A
- 이 발명을 지원한 국가연구개발사업
 과제고유번호 1711041012
 부처명 미래창조과학부
 연구관리전문기관 한국연구재단
 연구사업명 글로벌프론티어연구개발사업
 연구과제명 실감교류확장공간을 위한 초고속 전역조명 기술 개발
 기 여 율 34/100
 주관기관 성균관대학교 산학협력단
 연구기간 2016.07.01 ~ 2017.04.30
- 이 발명을 지원한 국가연구개발사업
 과제고유번호 1711037174
 부처명 미래창조과학부
 연구관리전문기관 한국연구재단-[EZBARO]이공분야 기초연구사업
 연구사업명 중견연구자지원사업(핵심개인연구)2/3
 연구과제명 [EZ]간극없는 GPU 컴퓨팅을 위한 소프트웨어 분석/향상 기법
 기 여 율 33/100
 주관기관 성균관대학교 산학협력단
 연구기간 2016.05.01 ~ 2017.04.30
- 이 발명을 지원한 국가연구개발사업
 과제고유번호 1711039593
 부처명 미래창조과학부
 연구관리전문기관 정보통신기술진흥센터
 연구사업명 대학ICT연구 센터 육성지원 사업
 연구과제명 모바일 플랫폼 기반 엔터테인먼트 VR 기술 연구
 기 여 율 33/100
 주관기관 세종대학교 산학협력단
 연구기간 2017.01.01 ~ 2017.12.31
-

명세서

청구범위

청구항 1

- (a) CPU가 OBJ 파일을 미리 설정된 크기의 청크(chunk)로 나누고, 상기 청크들을 순차적으로 GPU의 전역 메모리에 로딩하는 단계;
- (b) 상기 GPU가 상기 전역 메모리에 로딩된 청크를 순차적으로 파싱하여 v , vt , vn 및 f 어레이를 각각 생성하고, 메인 메모리로 순차적으로 다운로드하는 단계;
- (c) 상기 CPU가 상기 메인 메모리에 저장된 f 어레이를 미리 설정된 크기의 f 청크로 나누어서 순차적으로 상기 GPU의 전역 메모리에 로딩하는 단계;
- (d) 상기 GPU가 상기 전역 메모리에 순차적 로딩된 f 청크를 이용하여 인덱스 버퍼를 생성하는 단계; 및
- (e) 상기 GPU가 상기 인덱스 버퍼를 이용하여 버텍스 버퍼를 생성하는 단계;를 포함하는 것을 특징으로 하는 OBJ 파일 처리 방법.

청구항 2

청구항 1에 있어서,

상기 (b) 단계는,

- (b-1) 상기 GPU가 청크 내 태그를 찾고, 그 위치를 상기 전역 메모리에 저장하는 단계;
- (b-2) 상기 GPU가 각 태그의 라인 길이를 계산하고, 이를 상기 전역 메모리에 저장하는 단계;
- (b-3) 상기 GPU가 각 태그의 종류별 개수를 세고, 이를 상기 전역 메모리에 저장하는 단계; 및
- (b-4) 상기 GPU가 각 태그 별로 파싱하여 v , vt , vn 및 f 어레이를 각각 생성하는 단계;를 포함하는 것을 특징으로 하는 OBJ 파일 처리 방법.

청구항 3

청구항 2에 있어서,

상기 (b-1) 단계는, 상기 GPU가 태그에 해당하지 않는 위치 정보는 제외하여, 태그에 해당하는 위치 정보만으로 압축하여 저장하는 것을 특징으로 하는 OBJ 파일 처리 방법.

청구항 4

청구항 2에 있어서,

상기 (b-3) 단계는, 상기 GPU가 원자적 덧셈 연산을 통해 각 태그의 종류별 개수를 세는 것을 특징으로 하는 OBJ 파일 처리 방법.

청구항 5

청구항 1에 있어서,

상기 (d) 단계는,

- (d-1) 상기 GPU가 상기 전역 메모리에 순차적 로딩된 f 청크를 태그 v 를 기준으로 정렬하는 단계;
- (d-2) 상기 GPU가 태그 v 가 같은 데이터가 시작되는 지점에 스레드를 할당하는 단계; 및
- (d-3) 상기 GPU가 태그 v 를 기준으로 인덱스 테이블을 작성하는 단계;를 포함하는 것을 특징으로 하는 OBJ 파일 처리 방법.

청구항 6

청구항 5에 있어서,

(d-4) 상기 GPU가 모든 f 청크에 대해서 (d-1) 내지 (d-3)를 실행하여 작성된 상기 인덱스 테이블을 이용하여 지표를 생성하는 단계;를 더 포함하는 것을 특징으로 하는 OBJ 파일 처리 방법.

청구항 7

컴퓨터에서 청구항 1 내지 청구항 6 중 어느 한 청구항에 따른 OBJ 파일 처리 방법의 각 단계들을 수행하도록 작성되어 컴퓨터로 독출 가능한 기록 매체에 기록된 컴퓨터프로그램.

청구항 8

CPU, 메인 메모리, GPU, 전역 메모리를 포함하여 OBJ 파일 처리하는 컴퓨터 시스템에 있어서,

상기 CPU는, OBJ 파일을 미리 설정된 크기의 청크(chunk)로 나누고, 상기 청크들을 순차적으로 상기 GPU의 상기 전역 메모리에 로딩하고, 상기 메인 메모리에 저장된 f 어레이를 미리 설정된 크기의 f 청크로 나누어서 순차적으로 상기 GPU의 상기 전역 메모리에 로딩하며,

상기 GPU는, 상기 전역 메모리에 로딩된 청크를 순차적으로 파싱하여 v , vt , vn 및 f 어레이를 각각 생성하고, 상기 메인 메모리로 순차적으로 다운로드하고, 상기 전역 메모리에 순차적 로딩된 f 청크를 이용하여 인덱스 버퍼를 생성하며, 상기 인덱스 버퍼를 이용하여 버텍스 버퍼를 생성하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 9

청구항 8에 있어서,

상기 GPU는 상기 전역 메모리에 로딩된 청크를 순차적으로 파싱하여 v , vt , vn 및 f 어레이를 각각 생성함에 있어서,

청크 내 태그를 찾고, 그 위치를 상기 전역 메모리에 저장하고, 각 태그의 라인 길이를 계산하고, 이를 상기 전역 메모리에 저장하고, 각 태그의 종류별 개수를 세고, 이를 상기 전역 메모리에 저장하며, 각 태그 별로 파싱하여 v , vt , vn 및 f 어레이를 각각 생성하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 10

청구항 9에 있어서,

상기 GPU는, 청크 내 태그를 찾고, 그 위치를 상기 전역 메모리에 저장함에 있어서,

태그에 해당하지 않는 위치 정보는 제외하여, 태그에 해당하는 위치 정보만으로 압축하여 저장하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 11

청구항 9에 있어서,

상기 GPU는, 상기 GPU가 각 태그의 종류별 개수를 세고, 이를 상기 전역 메모리에 저장함에 있어서,

원자적 덧셈 연산을 통해 각 태그의 종류별 개수를 세는 것을 특징으로 하는 컴퓨터 시스템.

청구항 12

청구항 9에 있어서,

상기 GPU는, 상기 GPU가 상기 전역 메모리에 순차적 로딩된 f 청크를 이용하여 상기 인덱스 버퍼를 생성함에 있어서,

상기 전역 메모리에 순차적 로딩된 f 청크를 태그 v 를 기준으로 정렬하고, 태그 v 가 같은 데이터가 시작되는 지점에 스트레드를 할당하며, 태그 v 를 기준으로 인덱스 테이블을 작성하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 13

청구항 12에 있어서,

상기 GPU는, 상기 인덱스 테이블을 이용하여 지표를 더 생성하는 것을 특징으로 하는 컴퓨터 시스템.

발명의 설명

기술 분야

[0001] 본 발명은 파일 처리 방법에 관한 것으로서, 구체적으로는 대용량 OBJ 파일을 GPU를 이용하여 효율적으로 처리하는 방법에 관한 것이다.

배경 기술

[0002] OBJ 파일의 파싱(parsing) 방법은 CPU 구조에 최적화 되어있으므로, CPU 기반한 처리를 주로 해왔다. 이러한 CPU 기반한 처리는 OBJ 파일 크기가 증가할수록, 연산량이 선형적으로 증가하므로 대용량 크기의 경우 로딩은 많은 시간을 소비한다. 하드웨어의 성능이 개선되면서 앞서 언급한 한계를 극복 하고자 병렬처리를 통한 자체적인 최적화에 대한 시도가 있었다. 멀티 코어 CPU 기반의 XML parsing 연구는 기존의 단일 코어 CPU 기반에 비해 성능의 향상이 있었다. 어느 정도의 성능 향상이 있었지만, 대규모 크기의 OBJ 파일 처리에는 어려움이 있었다.

[0003] 한편, GPU는 다수의 코어를 지원하므로 병렬 처리에 최적화된 성능을 가지고 있으며, 대용량 데이터의 처리에 적합하다. 이에 따라 GPU에 기반한 파일 로더(loader)가 등장하여 기존 CPU에 비해 월등히 가속화 되었지만, 여전히 대용량 OBJ 파일의 경우 GPU의 메모리 부족 문제로 어려움이 있었다.

[0004] 따라서 대용량 OBJ 파일을 처리하기 위해 GPU 메모리에 파일을 로딩함에 있어서 보다 효율적으로 방법이 필요하다.

선행기술문헌

특허문헌

[0006] (특허문헌 0001) 대한민국 등록특허공보 제10-1415200호

발명의 내용

해결하려는 과제

[0007] 본 명세서는 대용량 OBJ 파일을 GPU를 통해 효율적으로 처리할 수 있는 방법을 제시하고 한다.

[0008] 본 명세서에 기재된 해결과제는 이상에서 언급한 것들에 한정되지 않으며, 언급되지 아니한 다른 해결과제들은 아래의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0009] 상술한 과제를 해결하기 위한 본 명세서에 따른 OBJ 파일 처리 방법은, (a) CPU가 OBJ 파일을 미리 설정된 크기의 청크(chunk)로 나누고, 상기 청크들을 순차적으로 GPU의 전역 메모리에 로딩하는 단계; (b) 상기 GPU가 상기 전역 메모리에 로딩된 청크를 순차적으로 파싱하여 v, vt, vn 및 f 어레이를 각각 생성하고, 메인 메모리로 순차적으로 다운로드하는 단계; (c) 상기 CPU가 상기 메인 메모리에 저장된 f 어레이를 미리 설정된 크기의 f 청크로 나누어서 순차적으로 상기 GPU의 상기 전역 메모리에 로딩하는 단계; (d) 상기 GPU가 상기 전역 메모리에 순차적 로딩된 f 청크를 이용하여 인덱스 버퍼를 생성하는 단계; 및 (e) 상기 GPU가 상기 인덱스 버퍼를 이용하여 버텍스 버퍼를 생성하는 단계;를 포함할 수 있다.

[0010] 본 명세서의 일 실시예에 따르면, 상기 (b) 단계는, (b-1) 상기 GPU가 청크 내 태그를 찾고, 그 위치를 상기 전역 메모리에 저장하는 단계; (b-2) 상기 GPU가 각 태그의 라인 길이를 계산하고, 이를 상기 전역 메모리에 저장하는 단계; (b-3) 상기 GPU가 각 태그의 종류별 개수를 세고, 이를 상기 전역 메모리에 저장하는 단계; 및 (b-4) 상기 GPU가 각 태그 별로 파싱하여 v, vt, vn 및 f 어레이를 각각 생성하는 단계;를 포함할 수 있다.

[0011] 본 명세서의 일 실시예에 따르면, 상기 (b-1) 단계는, 상기 GPU가 태그에 해당하지 않는 위치 정보는 제외하여,

태그에 해당하는 위치 정보만으로 압축하여 저장할 수 있다.

- [0012] 본 명세서의 일 실시예에 따르면, 상기 (b-3) 단계는, 상기 GPU가 원자적 덧셈 연산을 통해 각 태그의 종류별 개수를 셀 수 있다.
- [0013] 본 명세서의 일 실시예에 따르면, 상기 (d) 단계는, (d-1) 상기 GPU가 상기 전역 메모리에 순차적 로딩된 f 청크를 태그 v를 기준으로 정렬하는 단계; (d-2) 상기 GPU가 태그 v가 같은 데이터가 시작되는 지점에 스레드를 할당하는 단계; 및 (d-3) 상기 GPU가 태그 v를 기준으로 인덱스 테이블을 작성하는 단계;를 포함할 수 있다.
- [0014] 본 명세서의 일 실시예에 따르면, (d-4) 상기 GPU가 모든 f 청크에 대해서 (d-1) 내지 (d-3)를 실행하여 작성된 상기 인덱스 테이블을 이용하여 지표(indices)를 생성하는 단계;를 더 포함할 수 있다.
- [0015] 본 명세서에 따른 OBJ 파일 처리 방법은 각 단계들이 수행하도록 작성되어 컴퓨터로 독출 가능한 기록 매체에 기록된 컴퓨터프로그램을 형태를 가질 수 있다.
- [0016] 상술한 과제를 해결하기 위한 본 명세서에 따른 컴퓨터 시스템은 CPU, 메인 메모리, GPU, 전역 메모리를 포함하여 OBJ 파일 처리하는 컴퓨터 시스템으로서, 상기 CPU는, OBJ 파일을 미리 설정된 크기의 청크(chunk)로 나누고, 상기 청크들을 순차적으로 상기 GPU의 상기 전역 메모리에 로딩하고, 상기 메인 메모리에 저장된 f 어레이를 미리 설정된 크기의 f 청크로 나누어서 순차적으로 상기 GPU의 상기 전역 메모리에 로딩하며, 상기 GPU는, 상기 전역 메모리에 로딩된 청크를 순차적으로 파싱하여 v, vt, vn 및 f 어레이를 각각 생성하고, 상기 메인 메모리로 순차적으로 다운로드하고, 상기 전역 메모리에 순차적 로딩된 f 청크를 이용하여 인덱스 테이블을 생성 후 인덱스 버퍼를 생성하며, 상기 인덱스 버퍼를 이용하여 버텍스 버퍼를 생성할 수 있다.

발명의 효과

- [0017] 본 명세서에 따른 OBJ 파일 처리 방법은 대용량 OBJ 파일의 처리 속도를 상당히 향상시킬 수 있다.
- [0018] 본 명세서에 기재된 효과는 이상에서 언급된 것들에 한정되지 않으며, 언급되지 아니한 다른 효과들은 아래의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

- [0019] 도 1은 OBJ 청크 내 태그의 위치를 찾는 개념도이다.
- 도 2는 태그의 라인 길이를 계산하는 개념도이다.
- 도 3은 각 태그의 개수를 세는 개념도이다.
- 도 4는 각 태그 별로 파싱하는 개념도이다.
- 도 5는 인덱스 테이블을 바탕으로 인덱스 버퍼를 생성하는 과정을 도시한 개념도이다.
- 도 6은 인덱스 테이블을 이용하여 버텍스 버퍼를 생성하는 과정을 도시한 개념도이다.

발명을 실시하기 위한 구체적인 내용

- [0020] 이하, 첨부한 도면을 참조하여, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자가 용이하게 실시할 수 있도록 본 발명의 실시예를 설명한다. 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자가 용이하게 이해할 수 있는 바와 같이, 후술하는 실시예는 본 발명의 개념과 범위를 벗어나지 않는 한도 내에서 다양한 형태로 변형될 수 있다. 가능한 한 동일하거나 유사한 부분은 도면에서 동일한 도면부호를 사용하여 나타낸다.
- [0021] 본 명세서에서 사용되는 전문용어는 단지 특정 실시예를 언급하기 위한 것이며, 본 발명을 한정하는 것을 의도하지는 않는다. 여기서 사용되는 단수 형태들은 문구들이 이와 명백히 반대의 의미를 나타내지 않는 한 복수 형태들도 포함한다.
- [0022] 본 명세서에서 사용되는 "포함하는"의 의미는 특정 특성, 영역, 정수, 단계, 동작, 요소 및/또는 성분을 구체화하며, 다른 특정 특성, 영역, 정수, 단계, 동작, 요소, 성분 및/또는 군의 존재나 부가를 제외시키는 것은 아니다.
- [0023] 본 명세서에서 사용되는 기술용어 및 과학용어를 포함하는 모든 용어들은 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자가 일반적으로 이해하는 의미와 동일한 의미를 가진다. 사전에 정의된 용어들은 관련기술문헌

과 현재 개시된 내용에 부합하는 의미를 가지는 것으로 추가 해석되고, 정의되지 않는 한 이상적이거나 매우 공식적인 의미로 해석되지 않는다.

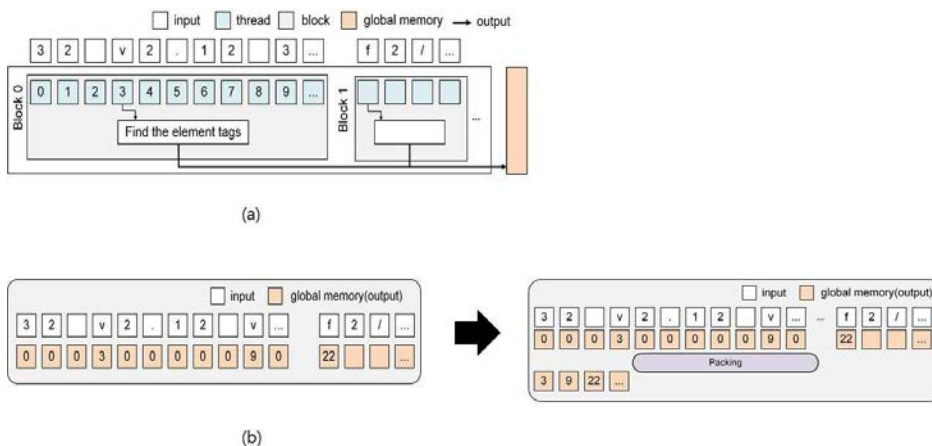
- [0024] 본 명세서에 따른 OBJ 파일은 렌더링 분야에 사용되는 파일을 의미한다.
- [0025] 본 명세서에 따른 OBJ 파일 처리 방법은 out-of-core 방식으로 대용량 파일을 일정 단위 크기로 나누고, GPU(Graphics Processing Unit) 병렬 처리를 통해 순차적으로 파싱(parsing)할 수 있다. 본 명세서에서 파싱이란 문자열 타입의 OBJ 파일을 integer 혹은 float 타입의 데이터로 변환하는 것을 의미한다. 본 명세서에 따른 OBJ 파일 처리 방법은 크게 두 부분으로 나누어진다. OBJ 파일을 문자열 타입에서 데이터 타입으로 파싱(parsing) 부분과 파싱한 데이터에서 버텍스 버퍼(vertex buffer)와 인덱스 버퍼(index buffer)를 생성하는 부분이다.
- [0026] 종래 OBJ 파일의 처리 방법은 CPU(Central Processing Unit, 중앙 처리 장치) 기반한 처리 방법이며, OBJ 파일 크기가 증가할수록 연산량이 선형적으로 증가하므로 OBJ 파일의 로딩은 많은 시간을 소비한다. GPU가 병렬처리 통하여 대용량 데이터에 대해 최적화된 성능을 지원하여도, 과도한 크기의 데이터가 GPU 메모리에 로딩되었을 때 오히려 GPU에 부담을 주며 성능을 하락시키는 원인이된다. 따라서 본 명세서에 따른 OBJ 파일 처리 방법은 OBJ 파일을 일정 단위로 나누어 GPU에 로딩하여 연산 부하를 줄일 수 있어, 대용량의 파일도 처리가 가능하다.
- [0027] 한편, 이하에서 상세히 설명될 본 명세서에 따른 OBJ 파일 처리 방법은 컴퓨터 프로그램 형태로 구현될 수 있다. 상기 컴퓨터 프로그램은 본 발명이 속한 기술분야에 알려진 프로세서, ASIC(application-specific integrated circuit), 다른 칩셋, 논리 회로, 레지스터, 통신 모듈, 데이터 처리 장치 등이 독출할 있는 메모리 장치에 저장될 수 있으며, 상기 프로세서 등이 상기 메모리에 저장된 프로그램을 독출하여 본 명세서에 따른 메모리 로딩 방법을 실행할 수 있다. 이 경우, 상기 프로세서 등은 본 명세서에 따른 메모리 로딩 장치가 될 수 있음은 자명하다.
- [0029] 이하 도면을 참조하여 본 명세서에 따른 OBJ 파일 처리 방법을 설명하도록 하겠다.
- [0031] 먼저, CPU가 OBJ 파일을 미리 설정된 크기의 청크(chunk)로 나누고, 상기 청크들을 순차적으로 GPU의 전역 메모리(Global Memory)에 로딩할 수 있다.
- [0032] 상기 '청크(chunk)'란 OBJ 파일을 임의의 크기로 나누었을 때, 임의의 크기만큼 읽어드린 문자열을 의미한다. 상기 청크의 크기는 GPU의 전역 메모리의 크기에 따라 다양하게 설정될 수 있다. 본 명세서의 출원 당시 상용 판매 중인 NVIDIA GPU 980 Ti 기준 상기 청크의 크기는 64MB일때 최대 성능을 가진다.
- [0034] 다음으로, 상기 GPU는 상기 전역 메모리에 로딩된 청크를 순차적으로 파싱하여 v, vt, vn 및 f 어레이를 각각 생성할 수 있다.
- [0035] 먼저, 상기 GPU는 각 청크 내 태그(v, vt, vn 및 f) 위치를 찾을 수 있다.
- [0036] 도 1은 OBJ 청크 내 태그의 위치를 찾는 개념도이다.
- [0037] 도 1을 참조하면, 하나의 청크 내에서 태그(v, vt, vn, f)를 병렬적으로 찾는 과정을 확인할 수 있다. 각 청크 내의 모든 태그(v, vt, vn, f)들의 위치(index) 정보를 알아야 다음 과정에서 이 위치(index)를 통해 각 태그에 접근이 가능하기 때문에 위치를 찾을 필요가 있다.
- [0038] 도 1의 (a)를 보면, 각 스레드(thread)가 문자 하나하나에 할당된 것을 확인할 수 있다. 각 스레드들은 자신에게 할당된 문자가 v 혹은 f 로 시작하는 지를 판단하여, 태그로 판단되면 위치를 별도의 전역 메모리에 저장할 수 있다. 이 때 각각의 스레드들은 동시다발적으로 진행되고, 그 순서를 제어할 수 없다. 그렇지만 스레드ID를 이용하면(파란색 블록) 동시 다발적으로 진행되어도, 순차적으로는 저장할 수 있다.
- [0039] 한편, 도 1의 (a)를 거치면, 도 1의 (b) 왼쪽에 도시된 것과 같이 결과물(output)이 생성된다. 그런데 도 1의 (b) 왼쪽에 보이는 대로, 위치(index) 데이터 분포가 드문드문(sparse)하기 때문에, 이를 압축(packing)하는 과정이 필요하다.
- [0040] 도 1의 (b) 오른쪽에 도시된 것과 같이, 위치 정보를 압축할 수 있다. 이때 태그에 해당하지 않는 위치 정보는 제외하여, 태그에 해당하는 위치 정보만으로 압축하여 저장한다. 즉, 도 1의 (b)에서 '3, 9, 22 ...' 가 최종적인 결과물(output)이다. 이 과정을 통해 전역 메모리의 낭비를 방지할 수 있다.
- [0041] 다음으로, 상기 GPU는 각 태그의 라인 길이를 계산하고 이를 전역 메모리에 저장할 수 있다.

- [0042] 도 2는 태그의 라인 길이를 계산하는 개념도이다.
- [0043] 도 2를 참조하면, GUP는 상기 위치 정보의 개수만큼 스레드의 수를 할당하고 할당된 각 스레드는 캐리지 리턴(carriage return)을 만날 때까지 병렬적으로 센 후, 이 값을 전역 메모리에 저장할 수 있다. 각 태그 별로 위치 정보는 알지만, 어디까지 처리해야하는 지에 대한 정보가 없기 때문에, 각 태그의 라인 길이를 미리 계산하면, 태그의 위치와 태그 길이를 모두 알고 있으므로, 각 스레드는 병렬로 처리가 가능하다.
- [0044] 다음으로, 상기 GPU는 각 태그의 종류별 개수를 세고 이를 전역 메모리에 저장할 수 있다.
- [0045] 도 3은 각 태그의 개수를 세는 개념도이다.
- [0046] 도 3을 참조하면, 한 청크내 각 태그별로 개수가 파악 태그(v=33, vt=32, vn=20, f=35)되어 이에 대한 정보가 전역 메모리에 저장된 예시를 확인할 수 있다. 각 태그(v, vt, vn, f)의 수 정보가 필요한 이유는 사용자에게 모델에 대한 정보 제시 혹은 메모리 할당을 위해서 정보가 필요하므로 OBJ 파일내의 태그 정보의 각 개수를 파악하는 것이 필요하다. 이를 위해, 위치 정보에 스레드를 할당하고, 원자적 덧셈 연산(atomic Add)으로 개수를 셀 수 있다. 각 스레드는 태그의 종류(v, vt, vn, f)를 파악하고 카운트(count)를 담당하는 변수에 1을 더하는 식으로 처리할 수 있다. 원자적 덧셈 연산(atomic Add)를 하는 이유는 스레드간의 동시적으로 쓰기(write)를 하는 것은 기본적으로 문제가 발생한다. 예를 들어, 어느 한 스레드가 데이터를 쓰고 있는 시점에 다른 스레드가 접근을 해서 써버리면 이상한 값이 발생하는 등의 문제가 발생한다. 그러므로 스레드가 쓰기(write)를 하는 동안에 다른 스레드가 접근하지 못하는 기능을 지원하는 'atomic 함수'를 사용하고 그 중에서 덧셈을 하는 'atomic Add'를 사용할 수 있다.
- [0047] 다음으로, 상기 GPU는 각 태그 별로 파악하여 v, vt, vn 및 f 어레이를 각각 생성할 수 있다.
- [0048] 도 4는 각 태그 별로 파악하는 개념도이다.
- [0049] 도 4를 참조하면, 각 태그 별로 파악되어 생성된 v 어레이(array), vt 어레이(array), vn 어레이(array) 및 f 어레이(array)를 확인할 수 있다. 좀 더 자세하게는, 각 태그 별로 스레드를 할당하고, 각 파악이 끝나는 지점까지 이동하면서 문자 정보를 숫자 정보로 변환한다. 문자에서 숫자 변환은 아스키 코드(ascii code)값을 이용하여, 문자 정보에서 '48'만큼 빼서 숫자 정보로 변환할 수 있다.
- [0050] 다음으로, 상기 GPU는 상기 v, vt, vn 및 f 어레이를 메인 메모리로 순차적으로 다운로드할 수 있다. 하나의 청크에 대해서 태그 별로 파악이 완료되면 전역 메모리에는 파악된 v, vt, vn, f 어레이가 있다. GPU의 전역 메모리는 일반적으로 CPU의 메인 메모리보다 저장 용량이 적기 때문에(예: Intel Core i7-4790 CPU 기준 메인 메모리는 '16GB' / NVIDIA GPU 980 Ti 기준 GPU의 전역 메모리는 '6GB') 다음 차례의 청크를 처리하기 위해서는 GPU에 처리된 데이터를 메인 메모리에 다운로드 해야 한다. 도 4의 우측에 도시된 내용이 이에 대한 개념도이다.
- [0052] 상기 GPU는 도 1 내지 도 4에 도시된 과정을 모든 청크가 파악 완료될 때까지 반복한다. 따라서 모든 청크에 대한 파악이 끝나면, 태그 별로 파악된 정보가 메인 메모리에 저장된다.
- [0054] 다음으로, 모든 청크에 대한 파악이 완료되었으므로, 로딩을 위해서 버텍스 버퍼(vertex buffer)와 인덱스 버퍼(index buffer)를 생성하는 것이 필요하다. 상기 메인 메모리에 저장된 v, vt, vn, f 어레이로부터 버텍스 버퍼(vertex buffer)와 인덱스 버퍼(index buffer)를 생성하는 과정은 다음과 같다.
- [0056] 대용량의 크기의 OBJ 파일일수록 삼각형의 개수도 비례하여 증가하며 f 태그의 개수는 비례하여 증가함은 자명하다. 그러므로 f 어레이 전부를 GPU의 전역 메모리에 로딩하여 처리할 경우, GPU에 부하가 많이 발생한다. 따라서 본 명세서에 따른 CPU는 상기 메인 메모리에 저장된 f 어레이를 미리 설정된 크기의 f 청크로 나누어서 순차적으로 상기 GPU의 전역 메모리에 로딩할 수 있다. 상기 f 청크의 크기는 GPU의 전역 메모리의 크기에 따라 다양하게 설정될 수 있다. 본 명세서 출원 당시 상용 판매 중인 NVIDIA GPU 980 Ti 기준 상기 청크의 크기는 256KB 일 때 최대 성능을 가진다.
- [0058] 한편, 본 명세서에 따른 OBJ 파일 처리 방법은, 종래 기술과 달리, 버텍스 버퍼(vertex buffer)를 먼저 생성하는 것이 아닌, 인덱스 버퍼(index buffer)를 먼저 생성한다. 상기 f 어레이에 의해 인덱싱(indexing)된 버텍스(vertex)보다 f의 메모리 사이즈가 더 적기 때문에, 연산 비용이 상대적으로 적고 인덱스 버퍼(index buffer)를 먼저 생성하면 중복된 버텍스(vertex) 정보가 제거되므로, f를 이용하여 인덱스 버퍼(index buffer)를 먼저 생성할 수 있다.

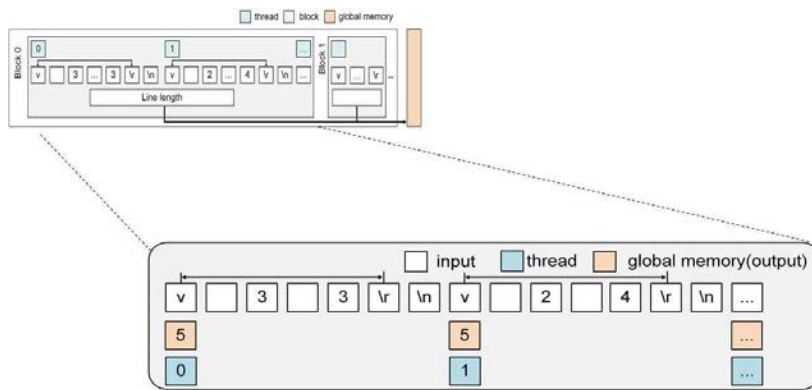
- [0059] 도 5는 인덱스 버퍼를 생성하는 과정을 도시한 개념도이다.
- [0060] 도 5를 참조하면, 먼저 상기 GPU는 상기 전역 메모리에 순차적 로딩된 f 청크를 태그 v를 기준으로 정렬할 수 있다. 그 결과, 태그 v가 같은 데이터끼리 모여있다.
- [0061] 다음으로, 상기 GPU는 태그 v가 같은 데이터가 시작되는 지점에 스레드를 할당할 수 있다. 이 과정은 인덱스 테이블(index table)을 만들기 위한 사전 작업으로서, 태그 v를 키(key)로 간주하고 병렬로 인덱스 테이블(index table)에 저장하기 때문에, 정렬을 하지 않으면, 쓰기 충돌(write conflict) 문제가 발생할 수 있다. 그리고 같은 태그 v를 가진 클러스터(cluster)가 시작되는 지점(v가 1, 2, 3 시작되는 지점)에 스레드를 할당해야 병렬로 처리가 가능하다. 구체적으로, GPU는 전역 메모리에 같은 태그 v를 가진 클러스터(cluster)가 시작되는 곳에 1을 저장하고, f의 수만큼 스레드를 할당하고 상기 '1'이 마킹(marking)된 스레드만 작업을 하도록 하면 클러스터(cluster)시작되는 지점에 접근이 가능하다.
- [0062] 다음으로, 상기 GPU는 태그 v를 기준으로 인덱스 테이블을 작성할 수 있다. 구체적으로, 각 스레드는 태그 v를 키(key)로 보고 해당위치에 vt/vn을 저장할 수 있다(도면 5 참조). 그리고 각 스레드는 저장 시에 버텍스(vertex)가 인덱스 (index table)에 해당 값이 있는지를 확인하고, 이미 존재할 경우 그 값은 인덱스 테이블에 입력하지 않는다.
- [0063] 상기 도 5에 도시된 과정은 모든 f 청크가 처리될 때까지 반복한다. 따라서 f 청크가 처리 될 때마다 인덱스 테이블 내의 f 정보는 누적된다. 모든 f의 버텍스(vertex)에 대해서 인덱스 테이블에 등록하면, 인덱스 테이블에는 중복된 데이터는 전부 제거가 되기 때문에 그 크기는 매우 작아진다. 그리고 상기 GPU는 상기 인덱스 테이블을 참조하여 지표(indices)를 생성할 수 있다. 이 경우, 지표(indices)를 생성하는 작업도 f 청크를 나누어서 처리할 수 있다. 상기 CPU는 f 청크를 다시 GPU의 전역 메모리에 로딩하고, 상기 GPU는 미리 생성된 인덱스 테이블(index table)을 참조하여 지표(indices)를 분할하여 생성할 수 있다. 상기 과정을 모든 f 청크에 대하여 진행하면 지표 버퍼(indices buffer) 생성이 가능하다.
- [0065] 마지막으로 상기 GPU는 상기 인덱스 테이블을 이용하여 버텍스 버퍼를 생성할 수 있다.
- [0066] 도 6은 인덱스 버퍼를 이용하여 버텍스 버퍼를 생성하는 과정을 도시한 개념도이다.
- [0067] 도 6을 참조하면, 인덱스 버퍼를 통해 버텍스 버퍼가 생성된 것을 확인할 수 있다. 본 명세서에 따른 OBJ 파일 처리 방법은 중복된 버텍스(vertex) 정보가 확인하게 줄어들었기 때문에, GPU가 전역 메모리에서 인메모리(in memory)방식으로 한번에 처리할 수 있다.
- [0069] 본 명세서에서 설명되는 실시예와 첨부된 도면은 본 발명에 포함되는 기술적 사상의 일부를 예시적으로 설명하는 것에 불과하다. 따라서, 본 명세서에 개시된 실시예들은 본 발명의 기술적 사상을 한정하기 위한 것이 아니라 설명하기 위한 것이므로, 이러한 실시예에 의하여 본 발명의 기술 사상의 범위가 한정되는 것은 아님은 자명하다. 본 발명의 명세서 및 도면에 포함된 기술적 사상의 범위 내에서 당업자가 용이하게 유추할 수 있는 변형예와 구체적인 실시 예는 모두 본 발명의 권리범위에 포함되는 것으로 해석되어야 할 것이다.

도면

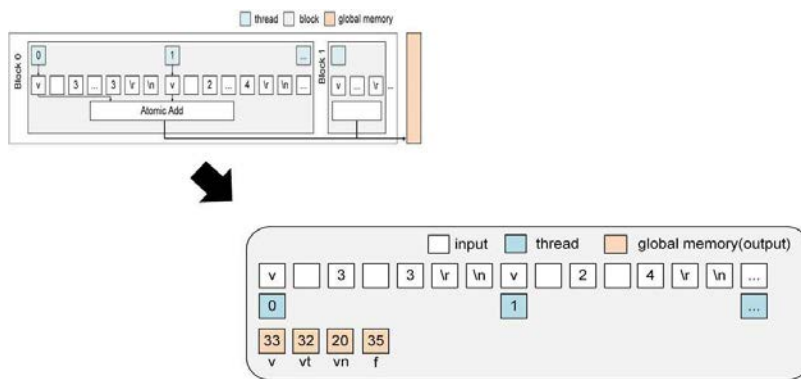
도면1



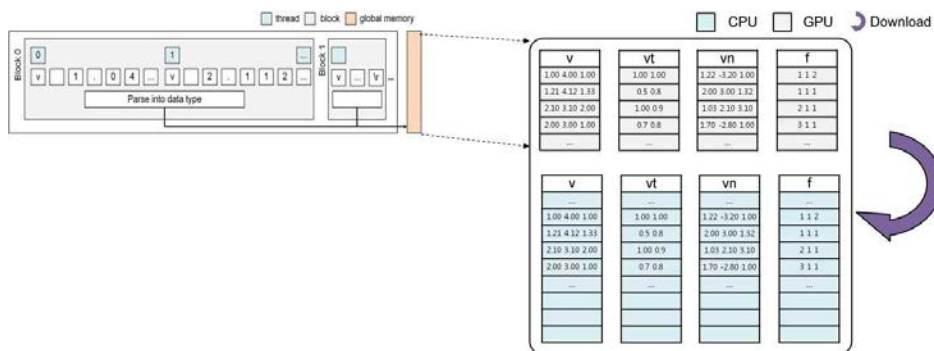
도면2



도면3

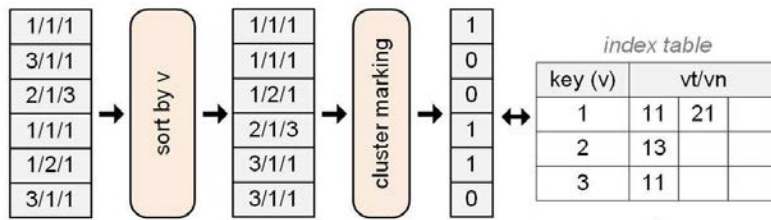


도면4

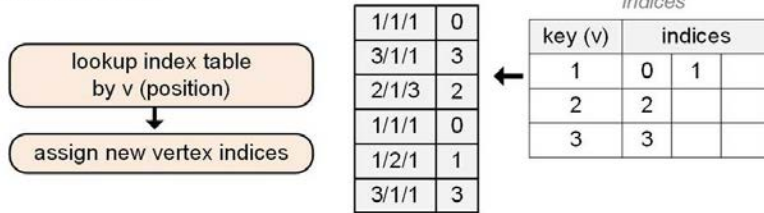


도면5

for each f chunk



for each f chunk



도면6

index table				parsed v, vt, vn		
key (v)	vt/vn			v	vt	vn
1	11	21		0.1 0.2 0.3	0.1 0.3	0.1 0.2 -0.3
2	13			1.1 0.2 0.3	0.1 0.2	-0.1 0.2 0.3
3	11			2.1 0.2 0.3	0.1 0.3	0.1 -0.2 0.3

vertex
v (0.1 0.2 0.3) vt (0.1 0.3) vn(0.1 0.2 -0.3)
v (0.1 0.2 0.3) vt (0.1 0.2) vn(0.1 0.2 -0.3)
v (1.1 0.2 0.3) vt (0.1 0.3) vn(0.1 -0.2 0.3)
v (2.1 0.2 0.3) vt (0.1 0.3) vn(0.1 0.2 -0.3)