(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2023년04월24일
(11) 등록번호 10-2525084
(24) 등록일자 2023년04월19일

(51) 국제특허분류(Int. Cl.)
*G06T 15/06* (2011.01) *G06T 1/20* (2018.01)
*G06T 1/60* (2006.01) *G06T 15/00* (2006.01)
(52) CPC특허분류
*G06T 15/06* (2013.01)
*G06T 1/20* (2013.01)
(21) 출원번호 10-2020-0143658
(22) 출원일자 2020년10월30일
심사청구일자 2020년10월30일
(65) 공개번호 10-2022-0058183
(43) 공개일자 2022년05월09일
(56) 선행기술조사문헌
KR101807172 B1*
KR1020150114767 A*
Jinyoung Lee, et. al., "Load Balancing Algorithm for Real-Time Ray Tracing of Dynamic Scenes", IEEE Access ( Volume: 8), 2020.08.24.*
Elena Vasiou, et. al., "Mach-RT: A Many Chip Architecture for HighPerformance Ray Tracing", IEEE Transactions on Visualization and Computer Graphics ( Early Access ), 2020.09.01.*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
**세종대학교산학협력단**
서울특별시 광진구 능동로 209 (군자동, 세종대학교)
(72) 발명자
**박우찬**
서울특별시 광진구 능동로 209, 대양AI센터 723호(군자동, 세종대학교)
**이진영**
서울특별시 광진구 능동로 209, 대양AI센터 827호(군자동, 세종대학교)
(74) 대리인
**정부연**

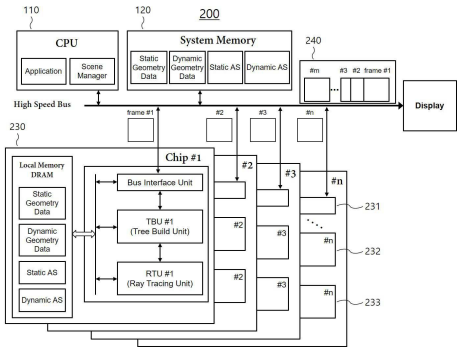
전체 청구항 수 : 총 13 항 심사관 : 박상철

(54) 발명의 명칭 **프레임 분할을 이용한 멀티칩 기반의 레이 트레이싱 장치 및 방법**

## (57) 요 약

본 발명은 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 장치 및 방법에 관한 것으로, 상기 장치는 장면(scene) 생성을 위한 기하데이터(geometry data)와 가속 구조(AS, Acceleration Structure) 저장하는 시스템 메모리; 각이 상기 기하데이터와 가속 구조를 기초로 개별 프레임에 관한 독립된 레이 트레이싱을 수행하는 복수의 레이 트레이싱 코어들; 및 레이 트레이싱(ray tracing) 어플리케이션 및 장면 매니저(scene manager)를 실행하고 관리하며 상기 복수의 레이 트레이싱 코어들에게 상기 기하데이터와 가속 구조를 전달하는 중앙 처리 유닛을 포함한다.

*대 표 도* - 도4



(52) CPC특허분류
*G06T 1/60* (2013.01)
*G06T 15/005* (2013.01)

이 발명을 지원한 국가연구개발사업

| | |
|---|---|
| 과제고유번호 | 1711103244 |
| 과제번호 | 2016-0-00204-005 |
| 부처명 | 과학기술정보통신부 |
| 과제관리(전문)기관명 | 정보통신기획평가원 |
| 연구사업명 | ICT융합산업원천기술개발(R&D) |
| 연구과제명 | 극사실적인 실시간 가상현실을 위한 모바일 GPU 하드웨어 개발 |
| 기 여 율 | 1/1 |
| 과제수행기관명 | 세종대학교 산학협력단 |
| 연구기간 | 2020.01.01 ~ 2020.12.31 |

공지예외적용 : 있음

# 명 세 서

## 청구범위

### 청구항 1

장면(scene) 생성을 위한 기하데이터(geometry data)와 가속 구조(AS, Acceleration Structure) 저장하는 시스템 메모리;

각각이 상기 기하데이터와 가속 구조를 기초로 개별 프레임에 관한 독립된 레이 트레이싱을 수행하는 복수의 레이 트레이싱 코어들; 및

레이 트레이싱(ray tracing) 어플리케이션 및 장면 매니저(scene manager)를 실행하고 관리하며 상기 복수의 레이 트레이싱 코어들에게 상기 기하데이터와 가속 구조를 전달하는 중앙 처리 유닛을 포함하는 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 장치.

### 청구항 2

제1항에 있어서, 상기 시스템 메모리는

원시 정적 장면(Primitive Static Scene)을 저장하는 PSS 영역, 동적 장면(Primitive Dynamic Scene)을 저장하는 PDS 영역 및 정적 가속 구조와 동적 가속 구조들을 각각 저장하는 AS 영역들을 포함하는 것을 특징으로 하는 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 장치.

### 청구항 3

제1항에 있어서, 상기 복수의 레이 트레이싱 코어들 각각은

데이터 송·수신을 처리하는 버스 인터페이스 유닛(Bus Interface Unit);

가속 구조(AS)를 구축하는 트리 빌드 유닛(TBU, Tree Build Unit);

상기 가속 구조(AS)를 기초로 레이 트레이싱을 수행하는 레이 트레이싱 유닛(RTU, Ray Tracing Unit); 및

상기 레이 트레이싱을 위해 상기 기하데이터와 상기 가속 구조를 임시 저장하는 로컬 메모리(Local Memory)를 포함하는 것을 특징으로 하는 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 장치.

### 청구항 4

제1항에 있어서,

상기 복수의 레이 트레이싱 코어들로부터 수신된 프레임들을 순서에 따라 정렬하여 출력하는 프레임 유닛을 더 포함하는 것을 특징으로 하는 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 장치.

### 청구항 5

제4항에 있어서, 상기 프레임 유닛은

상기 복수의 레이 트레이싱 코어들 각각에 할당되어 상기 프레임들을 처리 순서에 따라 저장하는 복수의 프레임 버퍼들; 및

상기 복수의 프레임 버퍼들로부터 수신한 프레임들을 상기 처리 순서와 상관없이 프레임 넘버에 따라 저장하는 프레임 큐(queue)를 포함하는 것을 특징으로 하는 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레

이싱 장치.

**청구항 6**

제5항에 있어서,

상기 복수의 프레임 버퍼들 각각은 동일 크기로 형성되고,

상기 프레임 큐의 크기는 상기 프레임 버퍼의 개수 및 크기에 따라 결정되는 것을 특징으로 하는 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 장치.

**청구항 7**

제5항에 있어서, 상기 프레임 유닛은

상기 프레임 버퍼에 저장된 특정 프레임에 대해 해당 프레임 넘버를 큐 인덱스에 대응시켜 상기 특정 프레임을 상기 프레임 큐에 저장하는 것을 특징으로 하는 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 장치.

**청구항 8**

제7항에 있어서, 상기 프레임 유닛은

현재의 드로우 넘버(draw number)를 읽는 단계;

상기 드로우 넘버와 상기 프레임 큐에 저장된 프레임들의 프레임 넘버를 각각 비교하는 단계;

상기 드로우 넘버가 상기 프레임 넘버와 동일하면 해당 프레임을 출력하는 단계;

상기 출력에 성공한 경우 상기 드로우 넘버를 1 증가시키는 단계; 및

상기 프레임 큐가 공백이 될 때까지 반복하는 단계를 통해 동작하는 것을 특징으로 하는 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 장치.

**청구항 9**

특정 장면(scene)을 구성하는 복수의 프레임들에 대한 레이 트레이싱을 위해 복수의 레이 트레이싱 코어들의 총 개수를 결정하는 단계;

상기 복수의 프레임들을 프레임 단위로 분할하여 상기 복수의 레이 트레이싱 코어들 각각에 할당하는 단계;

상기 복수의 레이 트레이싱 코어들 각각에게 시스템 메모리의 기하데이터를 전송하는 단계;

상기 복수의 레이 트레이싱 코어들 각각에서 프레임 단위로 레이 트레이싱의 완료 여부를 결정하는 단계;

특정 레이 트레이싱 코어에서 레이 트레이싱이 완료된 경우 해당 프레임을 해당 프레임 버퍼에 저장하는 단계;

상기 해당 프레임을 프레임 큐(queue)에 저장하는 단계; 및

상기 프레임 큐의 프레임들을 순차적으로 출력하는 단계를 포함하는 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 방법.

**청구항 10**

제9항에 있어서,

상기 복수의 레이 트레이싱 코어들의 총 개수를 결정하는 단계는 드로우 넘버(draw number)를 초기화 하는 단계

를 포함하고,

상기 출력하는 단계는 상기 프레임 큐에서 현재의 드로우 넘버와 동일한 프레임 넘버의 프레임을 출력하는 단계를 포함하는 것을 특징으로 하는 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 방법.

**청구항 11**

제10항에 있어서, 상기 출력하는 단계는

상기 출력에 성공한 경우 상기 현재의 드로우 넘버를 1만큼 증가시키는 단계를 포함하는 것을 특징으로 하는 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 방법.

**청구항 12**

제9항에 있어서,

상기 복수의 레이 트레이싱 코어들 각각에 할당하는 단계는 할당된 프레임에 프레임 넘버를 부여하는 단계를 포함하고,

상기 프레임 넘버는 레이 트레이싱 코어 별로 설정되고 이전의 프레임 넘버를 기준으로 상기 총 개수 만큼의 간격으로 설정되는 것을 특징으로 하는 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 방법.

**청구항 13**

제12항에 있어서,

상기 프레임 넘버가 상기 복수의 프레임들의 총 개수보다 큰 경우 상기 특정 장면(scene)에 대한 렌더링(rendering)이 종료되는 것을 특징으로 하는 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 방법.

## *발명의 설명*

### *기 술 분 야*

[0001] 본 발명은 3차원 그래픽 처리 기술에 관한 것으로, 보다 상세하게는 레이 트레이싱을 독립적으로 수행하는 다수의 칩을 이용하여 향상된 성능의 그래픽 처리를 수행할 수 있는 프레임 분할을 이용한 멀티칩 기반의 레이 트레이싱 장치 및 방법에 관한 것이다.

### *배 경 기 술*

[0003] 3차원 그래픽 기술은 컴퓨팅에 저장된 기하학적 데이터(Geometric data)의 3차원 표현을 사용하는 그래픽 기술로, 오늘날 미디어 산업과 게임 산업을 포함하는 다양한 산업에서 널리 사용되고 있다. 일반적으로 3차원 그래픽 기술은 많은 연산량으로 인하여 별개의 고성능 그래픽 프로세서를 요구한다.

[0004] 최근 프로세서의 발전에 따라 매우 현실적인 3차원 그래픽을 생성할 수 있는 레이 트레이싱(Ray Tracing) 기술이 연구되고 있다.

[0005] 레이 트레이싱(Ray Tracing) 기술은 전역 조명(Global Illumination)에 따른 렌더링(Rendering)방식으로, 다른 물체에서 반사되거나 굴절된 빛이 현재 물체의 영상에 미치는 영향을 고려하여 반사, 굴절, 그림자 효과가 자연적으로 제공되기 때문에 현실감 있는 3D 영상을 생성할 수 있다.

### *선행기술문헌*

#### *특허문헌*

[0007] (특허문헌 0001) 한국공개특허 제10-2015-0039493호 (2015.04.10)

## 발명의 내용

### 해결하려는 과제

[0008] 본 발명의 일 실시예는 레이 트레이싱을 독립적으로 수행하는 다수의 칩을 이용하여 향상된 성능의 그래픽 처리를 수행할 수 있는 프레임 분할을 이용한 멀티칩 기반의 레이 트레이싱 장치 및 방법을 제공하고자 한다.

[0009] 본 발명의 일 실시예는 트리 빌드 유닛을 독립적으로 포함하도록 구현된 멀티칩을 기반으로 시스템에서 사용되는 칩의 개수에 비례하여 레이 트레이싱에 관한 성능 향상을 제공할 수 있는 프레임 분할을 이용한 멀티칩 기반의 레이 트레이싱 장치 및 방법을 제공하고자 한다.

### 과제의 해결 수단

[0011] 실시예들 중에서, 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 장치는 장면(scene) 생성을 위한 기하데이터(geometry data)와 가속 구조(AS, Acceleration Structure) 저장하는 시스템 메모리; 각이 상기 기하데이터와 가속 구조를 기초로 개별 프레임에 관한 독립된 레이 트레이싱을 수행하는 복수의 레이 트레이싱 코어들; 및 레이 트레이싱(ray tracing) 어플리케이션 및 장면 매니저(scene manager)를 실행하고 관리하며 상기 복수의 레이 트레이싱 코어들에게 상기 기하데이터와 가속 구조를 전달하는 중앙 처리 유닛을 포함한다.

[0012] 제1항에 있어서, 상기 시스템 메모리는 원시 정적 장면(Primitive Static Scene)을 저장하는 PSS 영역, 동적 장면(Primitive Dynamic Scene)을 저장하는 PDS 영역 및 정적 가속 구조와 동적 가속 구조들을 각각 저장하는 AS 영역들을 포함할 수 있다.

[0013] 상기 복수의 레이 트레이싱 코어들 각각은 데이터 송·수신을 처리하는 버스 인터페이스 유닛(Bus Interface Unit); 가속 구조(AS)를 구축하는 트리 빌드 유닛(TBU, Tree Build Unit); 상기 가속 구조(AS)를 기초로 레이 트레이싱을 수행하는 레이 트레이싱 유닛(RTU, Ray Tracing Unit); 및 상기 레이 트레이싱을 위해 상기 기하데이터와 상기 가속 구조를 임시 저장하는 로컬 메모리(Local Memory)를 포함할 수 있다.

[0014] 상기 레이 트레이싱 장치는 상기 복수의 레이 트레이싱 코어들로부터 수신된 프레임들을 순서에 따라 정렬하여 출력하는 프레임 유닛을 더 포함할 수 있다.

[0015] 상기 프레임 유닛은 상기 복수의 레이 트레이싱 코어들 각각에 할당되어 상기 프레임들을 처리 순서에 따라 저장하는 복수의 프레임 버퍼들; 및 상기 복수의 프레임 버퍼들로부터 수신한 프레임들을 상기 처리 순서와 상관없이 프레임 넘버에 따라 저장하는 프레임 큐(queue)를 포함할 수 있다.

[0016] 상기 복수의 프레임 버퍼들 각각은 동일 크기로 형성되고, 상기 프레임 큐의 크기는 상기 프레임 버퍼의 개수 및 크기에 따라 결정될 수 있다.

[0017] 상기 프레임 유닛은 상기 프레임 버퍼에 저장된 특정 프레임에 대해 해당 프레임 넘버를 큐 인덱스에 대응시켜 상기 특정 프레임을 상기 프레임 큐에 저장할 수 있다.

[0018] 상기 프레임 유닛은 현재의 드로우 넘버(draw number)를 읽는 단계; 상기 드로우 넘버와 상기 프레임 큐에 저장된 프레임들의 프레임 넘버를 각각 비교하는 단계; 상기 드로우 넘버가 상기 프레임 넘버와 동일하면 해당 프레임을 출력하는 단계; 상기 출력에 성공한 경우 상기 드로우 넘버를 1 증가시키는 단계; 및 상기 프레임 큐가 공백이 될 때까지 반복하는 단계를 통해 동작할 수 있다.

[0019] 실시예들 중에서, 프레임 분할을 이용한 멀티칩(multi-chip) 기반의 레이 트레이싱 방법은 특정 장면(scene)을 구성하는 복수의 프레임들에 대한 레이 트레이싱을 위해 복수의 레이 트레이싱 코어들의 총 개수를 결정하는 단계; 상기 복수의 프레임들을 프레임 단위로 분할하여 상기 복수의 레이 트레이싱 코어들 각각에 할당하는 단계; 상기 복수의 레이 트레이싱 코어들 각각에게 시스템 메모리의 기하데이터를 전송하는 단계; 상기 복수의 레이 트레이싱 코어들 각각에서 프레임 단위로 레이 트레이싱의 완료 여부를 결정하는 단계; 특정 레이 트레이싱 코어에서 레이 트레이싱이 완료된 경우 해당 프레임을 해당 프레임 버퍼에 저장하는 단계; 상기 해당 프레임을 프레임 큐(queue)에 저장하는 단계; 및 상기 프레임 큐의 프레임들을 순차적으로 출력하는 단계를 포함한다.

[0020] 상기 복수의 레이 트레이싱 코어들의 총 개수를 결정하는 단계는 드로우 넘버(draw number)를 초기화 하는 단계

를 포함하고, 상기 출력하는 단계는 상기 프레임 큐에서 현재의 드로우 넘버와 동일한 프레임 넘버의 프레임을 출력하는 단계를 포함할 수 있다.

[0021] 상기 출력하는 단계는 상기 출력에 성공한 경우 상기 현재의 드로우 넘버를 1만큼 증가시키는 단계를 포함할 수 있다.

[0022] 상기 복수의 레이 트레이싱 코어들 각각에 할당하는 단계는 할당된 프레임에 프레임 넘버를 부여하는 단계를 포함하고, 상기 프레임 넘버는 레이 트레이싱 코어 별로 설정되고 이전의 프레임 넘버를 기준으로 상기 총 개수만큼의 간격으로 설정될 수 있다.

[0023] 상기 프레임 넘버가 상기 복수의 프레임들의 총 개수보다 큰 경우 상기 특정 장면(scene)에 대한 렌더링(rendering)이 종료될 수 있다.

## 발명의 효과

[0025] 개시된 기술은 다음의 효과를 가질 수 있다. 다만, 특정 실시예가 다음의 효과를 전부 포함하여야 한다거나 다음의 효과만을 포함하여야 한다는 의미는 아니므로, 개시된 기술의 권리범위는 이에 의하여 제한되는 것으로 이해되어서는 아니 될 것이다.

[0026] 본 발명의 일 실시예에 따른 프레임 분할을 이용한 멀티칩 기반의 레이 트레이싱 장치 및 방법은 레이 트레이싱을 독립적으로 수행하는 다수의 칩을 이용하여 향상된 성능의 그래픽 처리를 수행할 수 있다.

[0027] 본 발명의 일 실시예에 따른 프레임 분할을 이용한 멀티칩 기반의 레이 트레이싱 장치 및 방법은 트리 빌드 유닛을 독립적으로 포함하도록 구현된 멀티칩을 기반으로 시스템에서 사용되는 칩의 개수에 비례하여 레이 트레이싱에 관한 성능 향상을 제공할 수 있다.

## 도면의 간단한 설명

[0029] 도 1은 레이 트레이싱 과정의 일 실시예를 설명하는 도면이다.

도 2는 레이 트레이싱 과정에서 사용되는 가속 구조로서 KD 트리의 일 실시예를 설명하는 도면이다.

도 3은 화면 분할 방식의 멀티칩 레이 트레이싱 시스템을 설명하는 도면이다.

도 4는 본 발명에 따른 프레임 분할 방식의 멀티칩 레이 트레이싱 시스템을 설명하는 도면이다.

도 5는 디스플레이 순서에 따른 프레임 정렬 동작을 설명하는 도면이다.

도 6은 본 발명에 따른 프레임 분할 방식의 멀티칩 레이 트레이싱 방법의 동작 과정을 설명하는 순서도이다.

## 발명을 실시하기 위한 구체적인 내용

[0030] 본 발명에 관한 설명은 구조적 내지 기능적 설명을 위한 실시예에 불과하므로, 본 발명의 권리범위는 본문에 설명된 실시예에 의하여 제한되는 것으로 해석되어서는 아니 된다. 즉, 실시예는 다양한 변경이 가능하고 여러 가지 형태를 가질 수 있으므로 본 발명의 권리범위는 기술적 사상을 실현할 수 있는 균등물들을 포함하는 것으로 이해되어야 한다. 또한, 본 발명에서 제시된 목적 또는 효과는 특정 실시예가 이를 전부 포함하여야 한다거나 그러한 효과만을 포함하여야 한다는 의미는 아니므로, 본 발명의 권리범위는 이에 의하여 제한되는 것으로 이해되어서는 아니 될 것이다.

[0031] 한편, 본 출원에서 서술되는 용어의 의미는 다음과 같이 이해되어야 할 것이다.

[0032] "제1", "제2" 등의 용어는 하나의 구성요소를 다른 구성요소로부터 구별하기 위한 것으로, 이들 용어들에 의해 권리범위가 한정되어서는 아니 된다. 예를 들어, 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소도 제1 구성요소로 명명될 수 있다.

[0033] 어떤 구성요소가 다른 구성요소에 "연결되어"있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결될 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성요소가 다른 구성요소에 "직접 연결되어"있다고 언급된 때에는 중간에 다른 구성요소가 존재하지 않는 것으로 이해되어야 할 것이다. 한편, 구성요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.

[0034] 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한 복수의 표현을 포함하는 것으로 이해되어야 하고, "포함하다"또는 "가지다" 등의 용어는 실시된 특징, 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이며, 하나 또는 그 이상의 다른 특징이나 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0035] 각 단계들에 있어 식별부호(예를 들어, a, b, c 등)는 설명의 편의를 위하여 사용되는 것으로 식별부호는 각 단계들의 순서를 설명하는 것이 아니며, 각 단계들은 문맥상 명백하게 특정 순서를 기재하지 않는 이상 명기된 순서와 다르게 일어날 수 있다. 즉, 각 단계들은 명기된 순서와 동일하게 일어날 수도 있고 실질적으로 동시에 수행될 수도 있으며 반대의 순서대로 수행될 수도 있다.

[0036] 본 발명은 컴퓨터가 읽을 수 있는 기록매체에 컴퓨터가 읽을 수 있는 코드로서 구현될 수 있고, 컴퓨터가 읽을 수 있는 기록 매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록 장치를 포함한다. 컴퓨터가 읽을 수 있는 기록 매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피 디스크, 광 데이터 저장 장치 등이 있다. 또한, 컴퓨터가 읽을 수 있는 기록 매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어, 분산 방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수 있다.

[0037] 여기서 사용되는 모든 용어들은 다르게 정의되지 않는 한, 본 발명이 속하는 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 것으로 해석되어야 하며, 본 출원에서 명백하게 정의하지 않는 한 이상적이거나 과도하게 형식적인 의미를 지니는 것으로 해석될 수 없다.

[0039] 도 1은 레이 트레이싱 과정의 일 실시예를 설명하는 도면이다.

[0040] 도 1을 참조하면, 레이 트레이싱 장치에서 수행되는 레이 트레이싱 방식은 전역 조명(global illumination)에 따른 렌더링(rendering) 방식에 해당할 수 있다. 이는 다른 물체에서 반사되거나 굴절된 빛(Light)도 현재 물체의 영상에 영향을 준다는 것을 의미할 수 있다. 이로 인하여 반사, 굴절, 그림자 효과가 자연적으로 제공되기 때문에 현실감 있는 3D 영상을 생성할 수 있다.

[0041] 레이 트레이싱 장치는 먼저 각 픽셀(pixel) 당 카메라(Camera) 위치로부터 프라이머리 레이(primary ray, P)를 생성하여 해당 레이와 만나는 물체를 찾기 위한 계산을 수행할 수 있다. 레이 트레이싱 장치는 레이와 만나게 된 물체가 반사나 굴절의 성질이 있으면 레이와 물체가 만난 위치에서 반사 효과를 위한 반사 레이(reflection ray, R)나 굴절 효과를 위한 굴절 레이(refraction ray, F)를 생성할 수 있고, 또한 그림자 효과를 위하여 빛(Light) 방향으로 그림자 레이(shadow ray, S)를 생성할 수 있다.

[0042] 이 때, 해당 빛(Light) 위치로 향한 그림자 레이와 어떤 물체가 만나면 그림자가 생성이 되며 그렇지 않을 경우는 그림자가 생성되지 않는다. 반사 레이와 굴절 레이는 세컨더리 레이(secondary ray)라고 불리며 레이 트레이싱 장치는 각각의 레이에 대해 해당 레이와 만나는 물체를 찾기 위한 계산을 수행할 수 있다. 이러한 과정은 레이 트레이싱 장치에 의해 반복적(recursive)으로 수행될 수 있다.

[0044] 도 2는 레이 트레이싱 과정에서 사용되는 가속 구조로서 KD 트리의 일 실시예를 설명하는 도면이다.

[0045] 도 2를 참조하면, 레이 트레이싱을 수행하기 위하여 전체 지오메트리 데이터(geometry data)(triangle의 좌표들로 구성)를 기초로 생성된 KD 트리(K-Dimensional Tree)나 BVH(Bounding Volume Hierarchy)와 같은 가속 구조(Acceleration Structure, AS)가 필수적으로 요구된다. 따라서, 레이 트레이싱을 수행하기 이전에 AS를 구축(build)할 필요가 있다. 이러한 가속 구조 구축(AS build)에는 연산량이 많이 필요하기 때문에 시간이 많이 소요될 수 있다.

[0046] 도 2에서, kd-트리의 전체 구성도를 설명하고 있다. kd-트리는 분할한 공간에 대하여 계층적(hierarchy) 구조를 갖는 이진 트리(binary tree)에 해당할 수 있다. kd-트리는 내부 노드(inner node)(top node 포함)와 리프 노드(leaf node)로 구성될 수 있으며, 리프 노드는 해당 노드와 교차(intersection)되는 객체(object)들을 포함하고 있는 공간에 대응될 수 있다. 즉, kd-트리는 공간 분할 트리(spatial partitioning tree)로서 공간 분할 구조체(spatial partitioning structure)의 일종에 해당할 수 있다.

[0047] 반면, 내부 노드는 바운딩 박스(bounding box) 기반의 공간 영역을 가질 수 있으며 해당 공간 영역은 다시 2개의 영역들로 나뉘어서 두 개의 하위 노드에 할당될 수 있다. 결과적으로 내부 노드는 분할 평면과 이를 통해 분할된 두 개의 영역의 서브-트리(sub-tree)로 구성될 수 있고, 리프 노드는 일련의 삼각형(triangle)들 만을 포함할 수 있다. 예를 들어, 리프 노드는 기하학적 데이터에 포함된 적어도 하나의 삼각형 정보를 포인팅하기 위

한 삼각형 리스트를 포함할 수 있으며, 삼각형 정보는 삼각형의 세 점에 대한 정점 좌표, 법선 벡터 및/또는 텍스처 좌표를 포함할 수 있다. 만약, 기하학적 데이터에 포함된 삼각형 정보가 배열로 구현된 경우에는 리프 노드에 포함된 삼각형 리스트는 배열 인덱스에 상응할 수 있다.

[0048] 한편, 공간을 분할하는 위치 p는 임의의 레이와 충돌(hit)하는 삼각형을 찾기 위한 비용(cost)(노드 방문 횟수, 삼각형과 교차되는지 계산하는 횟수 등)이 최소가 되는 지점에 해당할 수 있고, 해당 위치 p를 찾기 위해 현재 가장 많이 사용되는 방법은 SAH(Surface area heuristic)에 해당할 수 있다.

[0050] 도 3은 화면 분할 방식의 멀티칩 레이 트레이싱 시스템을 설명하는 도면이다.

[0051] 도 3을 참조하면, 화면 분할 방식의 멀티칩 레이 트레이싱 시스템(100)은 n개의 멀티칩(multi-chip)(ASIC or FPGA)을 활용하여 장면(scene) 렌더링을 수행할 수 있다. 화면 분할 방식의 멀티칩 레이 트레이싱 시스템(100)은 중앙처리유닛(CPU)(110), 시스템 메모리(System Memory)(120) 및 멀티칩들을 포함하여 구성될 수 있다. CPU(110)는 레이 트레이싱 어플리케이션(ray tracing application)과 장면 매니저(scene manager)를 실행하고 관리할 수 있다. 또한, CPU(110)는 시스템 메모리의 기하데이터(geometry data)와 가속 구조 데이터(AS data)를 각 멀티칩(130)들에게 전달하는 역할도 수행할 수 있다.

[0052] 멀티칩(130)들은 버스 인터페이스 유닛(bus interface unit), 레이 트레이싱 유닛(RTU, Ray Tracing Unit), 전달받은 기하데이터와 가속 구조 데이터를 위한 로컬 메모리(local memory)로 구성될 수 있다. 특히, 멀티칩 중 어느 하나의 칩(chip)은 (도 3의 경우, chip #1) 화면의 영역을 나누어 분배해주는 역할을 수행할 수 있으며, kd트리(kd-tree) 생성을 위한 트리 빌드 유닛(TBU, Tree Build Unit)을 포함하여 구성될 수 있다.

[0053] 화면 분할 방식의 멀티칩 레이 트레이싱 시스템(100)에서의 동작 과정은 다음과 같을 수 있다. Chip #1의 트리 빌드 유닛(TBU)은 렌더링(rendering) 될 프레임에 대한 기하데이터를 전달받아 공간 분할 구조체로서 kd트리를 구축할 수 있다.

[0054] 화면 분할 방식의 멀티칩 레이 트레이싱 시스템(100) Chip #1에서의 kd트리 구축이 완료되면, kd트리 정보를 레이 트레이싱 유닛(RTU) #1 내지 #n으로 각각 전송할 수 있으며, 각 레이 트레이싱 유닛(RTU)은 이를 기초로 레이 트레이싱을 수행할 수 있다. 이때, Chip #1은 각 칩의 레이 트레이싱 유닛(RTU)이 레이 트레이싱 할 영역을 할당하는 로드 마스터(load master)의 역할을 수행할 수 있으며, 렌더링(rendering) 될 프레임을 k*k 픽셀(예를 들어, 8*8) 단위의 블록으로 프레임 영역을 나누어 각 칩에 분배할 수 있다. 각 chip의 레이 트레이싱 유닛(RTU)은 할당된 영역에 대한 레이 트레이싱을 수행할 수 있고, 생성된 색상 결과는 메모리 컨트롤러를 통해 chip #1의 프레임 버퍼에 저장될 수 있다. 최종적으로 각 칩은 프레임의 1/n에 해당하는 영역에 대한 레이 트레이싱을 수행할 수 있다.

[0055] 즉, 화면 분할 방식은 프레임 1장을 다수의 레이 트레이싱 유닛들(RTU)이 영역을 나누어서 렌더링 하는 방식에 해당할 수 있다. 이때, 모든 레이 트레이싱 유닛(RTU)은 해당 프레임에 대한 kd트리 정보가 필요하므로 대규모의 데이터 전송이 발생할 수 있다. 트리 빌드 유닛(TBU)을 탑재한 칩은 해당 프레임에 대한 kd트리를 구축하여 모든 레이 트레이싱 유닛(RTU)에 전송해야 하기 때문이다.

[0056] 이러한 대규모의 데이터 전송량은 시스템에서 활용하는 칩의 수가 많아질수록 증가할 수 있다. 칩의 수가 n개라면, 프레임 1장당 kd트리에 대한 데이터 전송이 n번 발생할 수 있다. 결과적으로, m개의 프레임으로 이루어진 장면(scene)에 대해서 총 n*m만큼의 데이터 전송량이 발생할 수 있다.

[0058] 도 4는 본 발명에 따른 프레임 분할 방식의 멀티칩 레이 트레이싱 시스템을 설명하는 도면이다.

[0059] 도 4를 참조하면, 프레임 분할 방식의 멀티칩 레이 트레이싱 시스템(200)은 화면 분할 방식의 멀티칩 레이 트레이싱 시스템(100)과 동일하게 중앙처리유닛(CPU)(110) 및 시스템 메모리(System Memory)(120)를 포함하여 구성될 수 있다. CPU(110)는 레이 트레이싱(ray tracing) 어플리케이션 및 장면 매니저(scene manager)를 실행하고 관리하며 복수의 레이 트레이싱 코어들(230)에게 기하데이터와 가속 구조를 전달할 수 있다. 시스템 메모리(120)는 장면(scene) 생성을 위한 기하데이터(geometry data)와 가속 구조(AS, Acceleration Structure) 저장할 수 있다.

[0060] 일 실시예에서, 시스템 메모리(120)는 원시 정적 장면(Primitive Static Scene)을 저장하는 PSS 영역, 동적 장면(Primitive Dynamic Scene)을 저장하는 PDS 영역 및 정적 가속 구조와 동적 가속 구조들을 각각 저장하는 AS 영역들을 포함할 수 있다.

[0061] 일 실시예에서, 프레임 분할 방식의 멀티칩 레이 트레이싱 시스템(200)은 각각이 기하데이터와 가속 구조를 기

초로 개별 프레임에 관한 독립된 레이 트레이싱을 수행하는 복수의 레이 트레이싱 코어들(230)을 포함할 수 있다.

[0062] 일 실시예에서, 복수의 레이 트레이싱 코어들(230) 각각은 데이터 송·수신을 처리하는 버스 인터페이스 유닛(Bus Interface Unit)(231), 가속 구조(AS)를 구축하는 트리 빌드 유닛(TBU, Tree Build Unit)(232), 가속 구조(AS)를 기초로 레이 트레이싱을 수행하는 레이 트레이싱 유닛(RTU, Ray Tracing Unit)(233) 및 레이 트레이싱을 위해 기하데이터와 가속 구조를 임시 저장하는 로컬 메모리(Local Memory)를 포함할 수 있다.

[0063] 보다 구체적으로, 트리 빌드 유닛(TBU)(232)은 공간 분할 구조체(Spatial Partitioning Structure)로서 가속 구조(AS)를 구축하는 동작을 수행할 수 있다. 예를 들어, 트리 빌드 유닛(232)은 시스템 메모리(120) 상에 저장된 기하데이터를 기초로 BVH(Bounding Volume Hierarchy) 또는 KD 트리(K-Dimensional Tree) 등의 가속 구조를 생성할 수 있으며, 로컬 메모리 상에 저장할 수 있다.

[0064] 보다 구체적으로, 트리 빌드 유닛(232)은 3D 게임 엔진과 같은 어플리케이션(application)의 구동에 따라 레이 트레이싱 과정에 필요한 정적 장면(static scene) 및 동적 장면(dynamic scene)에 관한 가속 구조를 생성할 수 있다. 이때, 정적 장면(static scene)의 경우 3D 어플리케이션 구동 시 1번의 트리 빌드를 통해 정적 가속 구조(static AS)가 생성될 수 있고, 동적 장면(dynamic scene)의 경우 매 프레임(frame) 마다 프리미티브 정보가 변하기 때문에 매 프레임 마다 트리 빌드가 수행되어 동적 가속 구조(dynamic AS)가 생성될 수 있다. 트리 빌드 유닛(TBU)에 의해 생성된 정적 및 동적 가속 구조들은 각각 해당 레이 트레이싱 코어의 로컬 메모리에 저장되어 이후 레이 트레이싱 과정에서 사용될 수 있다.

[0065] 레이 트레이싱 유닛(RTU)(233)은 공간 분할 구조체(Spatial Partitioning Structure), 즉 가속 구조를 기초로 레이 트레이싱(Ray Tracing)을 수행할 수 있다. 보다 구체적으로, 레이 트레이싱 유닛(RTU)(233)은 트리 빌드 유닛(TBU)(232)에 의해 생성된 동적 및 정적 가속 구조들을 이용하여 레이 트레이싱을 수행할 수 있으며, 동적 및 정적 가속 구조들은 각각 로컬 메모리(local memory) 상에 저장될 수 있다.

[0066] 본 발명에 따른 프레임 분할 방식의 멀티칩 레이 트레이싱 시스템(200)은 도 3의 화면 분할 방식과 달리 모든 칩들, 즉 레이 트레이싱 코어들(230) 각각이 독립된 트리 빌드 유닛(TBU)을 포함하여 구현될 수 있다. 즉, 프레임 분할 방식의 멀티칩 레이 트레이싱 시스템(200)은 하나의 칩에 해당하는 레이 트레이싱 코어에서 1장의 프레임(frame)에 대한 렌더링(rendering)을 독립적으로 수행하는 방식으로 동작할 수 있다. 따라서, 모든 칩들이 RTU 및 TBU를 탑재하기 때문에, 할당받은 프레임에 대한 kd트리 구축과 레이 트레이싱을 독립적으로 수행할 수 있으며, 화면 분할 방식과 달리 TBU가 없는 칩으로 kd트리 정보를 전송하거나 RTU의 레이 트레이싱 수행 결과값을 특정 칩(도 3의 경우, Chip #1)의 프레임 버퍼로 전송하는 등의 추가적인 데이터 전송이 필요하지 않을 수 있다.

[0067] 또한, 시스템에서 활용하는 칩의 수가 늘어나는 경우, 화면 분할 방식은 총 데이터 전송량이 증가할 수 있는 반면, 프레임 분할 방식은 총 데이터 전송량이 변함없이 기존 수준을 그대로 유지할 수 있다. 즉, 프레임 분할 방식은 시스템에 활용한 칩의 수가 n개일 때, m개의 프레임으로 이루어진 장면(scene)에 대한 총 데이터 전송량은 항상 m개로 일정하게 유지될 수 있다.

[0068] 한편, 렌더링(rendering)에 있어서, 화면 분할 방식은 RTU의 수가 늘어날수록 성능이 향상되는 반면, kd트리 구축의 경우 알고리즘 상으로 병렬화가 매우 어려울 수 있으며, TBU의 수가 늘어나도 성능 향상을 달성하기 어려울 수 있다. 즉, 도 3에서, 화면 분할 방식은 다수의 칩들이 한 장의 프레임을 생성할 수 있으며, 칩의 수가 늘어날수록 RTU의 수가 증가하여 렌더링 성능은 향상되는 반면, TBU의 성능은 변화없이 그대로일 수 있다. 결과적으로, RTU와 TBU의 성능 차이가 커질 수 있고, 이는 전체 시스템의 성능 향상에 제약으로 작용할 수 있다.

[0069] 반면, 도 4의 프레임 분할 방식은 다수의 칩들이 각각 프레임을 생성하는 방식에 해당할 수 있고, 각 칩은 RTU와 TBU를 포함하도록 구현될 수 있다. 즉, 칩의 수가 늘어날수록 RTU와 TBU가 증가할 수 있고, 동시에 생성 가능한 프레임의 수가 비례적으로 증가할 수 있다. 프레임 분할 방식은 시스템에 장착된 칩의 수가 많을수록 그에 비례하는 성능 향상을 기대할 수 있다.

[0070] 일 실시예에서, 프레임 분할 방식의 멀티칩 레이 트레이싱 시스템(200)은 복수의 레이 트레이싱 코어들(230)로부터 수신된 프레임들을 순서에 따라 정렬하여 출력하는 프레임 유닛(240)을 더 포함하여 구현될 수 있다. 프레임 유닛(240)은 프레임 분할 방식의 멀티칩 레이 트레이싱 시스템(200)의 논리적인 구성에 해당할 수 있으며, 해당 동작을 수행하는 독립적인 모듈로 구현되거나 또는 다른 모듈들에서 수행되는 기능들의 논리적 집합에 해당할 수 있다. 따라서, 도 4에서는 프레임 유닛(240)을 독립적인 유닛으로 표현하고 있으나, 반드시 이에 한정

되지 않고, 기능적으로 분산되어 다른 유닛들의 내부 동작 또는 다른 유닛들 간의 연계 동작으로 구체화될 수도 있음은 물론이다.

[0072] 도 5는 디스플레이 순서에 따른 프레임 정렬 동작을 설명하는 도면이다.

[0073] 도 5를 참조하면, 프레임 분할 방식의 멀티칩 레이 트레이싱 시스템(200)은 프레임 유닛(240)을 통해 복수의 레이 트레이싱 코어들(230)로부터 수신된 프레임들을 순서에 따라 정렬하여 출력할 수 있다.

[0074] 일 실시예에서, 프레임 유닛(240)은 복수의 레이 트레이싱 코어들(230) 각각에 할당되어 프레임들을 처리 순서에 따라 저장하는 복수의 프레임 버퍼(241)들 및 복수의 프레임 버퍼(241)들로부터 수신한 프레임들을 처리 순서와 상관없이 프레임 넘버에 따라 저장하는 프레임 큐(queue)(242)를 포함할 수 있다. 프레임 분할 방식은 동시에 많은 프레임을 생성하여 이를 화면에 디스플레이 하는 방식에 해당할 수 있다. 이때, 프레임들의 생성이 완료되는 순서가 디스플레이 순서와 반드시 일치하는 것이 아니기 때문에, 생성된 프레임들을 후처리 없이 프레임 버퍼(241)에 그대로 저장하게 되면, 디스플레이 순서가 어긋날 수 있다. 이를 방지하기 위해서는 프레임 버퍼(241)에 저장되는 프레임의 순서를 디스플레이 순서와 맞춰줄 필요가 있다. 즉, 프레임 유닛(240)은 복수의 레이 트레이싱 코어들(230) 각각이 포함하는 독립적인 프레임 버퍼(241)들과 해당 버퍼들과 연동하는 프레임 큐(242)를 논리적으로 포함하여 구현될 수 있다.

[0075] 일 실시예에서, 복수의 프레임 버퍼(241)들 각각은 동일 크기로 형성되고, 프레임 큐(242)의 크기는 프레임 버퍼(241)의 개수 및 크기에 따라 결정될 수 있다. 예를 들어, 도 5의 경우, 각 칩의 프레임 버퍼(241)는 3개의 버퍼들로 구성될 수 있고, 각 칩들의 프레임 버퍼(241)와 연동하는 프레임 큐(242)의 크기는 프레임 버퍼(241)의 개수(예를 들어, 3)와 크기(예를 들어, 3)에 비례하여 9로 결정될 수 있다. 즉, 프레임 유닛(240)은 각 칩에 형성된 프레임 버퍼(241)들과 이와 연동하는 프레임 큐(242)를 통해 다양한 칩들에서 처리된 프레임들을 실제 순서에 따라 정렬할 수 있다.

[0076] 일 실시예에서, 프레임 유닛(240)은 프레임 버퍼(241)에 저장된 특정 프레임에 대해 해당 프레임 넘버를 큐 인덱스에 대응시켜 특정 프레임을 프레임 큐(242)에 저장할 수 있다. 여기에서, 프레임 넘버는 디스플레이 순서와 일치할 수 있고, 초기값은 칩 넘버에 의해 결정될 수 있으며, 칩 넘버는 레이 트레이싱 코어마다 부여된 식별번호에 해당할 수 있다. 예를 들어, Chip #1은 프레임 #1을 생성할 수 있고, Chip #2는 프레임 #2를 생성할 수 있으며, Chip #3은 프레임 #3을 생성할 수 있다. 이후 각 칩이 할당받는 프레임의 프레임 넘버는 처음 할당받은 프레임의 프레임 넘버에 시스템에서 사용되는 칩들의 수를 더하여 계산될 수 있다.

[0077] 또한, 각 칩에서 할당받은 프레임에 대한 생성이 끝나면 이를 프레임 버퍼(241)에 차례로 저장할 수 있다. 예를 들어, Chip #1의 프레임 버퍼 1에는 프레임 #1이 저장되게 되고, 차례로 프레임 버퍼 2에는 프레임 넘버 #(1(칩 넘버)+n(총 칩의 수))이 저장되며, 프레임 버퍼 3에서는 프레임 넘버 #(1+2n)이 저장될 수 있다.

[0078] 또한, 프레임 버퍼(241)에 저장된 프레임은 프레임 큐(242)에 저장될 수 있다. 이 때, 프레임 유닛(240)은 프레임 버퍼(241)에 저장된 특정 프레임에 대해 해당 프레임 넘버를 큐 인덱스에 대응시켜 특정 프레임을 프레임 큐(242)에 저장할 수 있다. 예를 들어, Chip #1의 프레임 버퍼 #1, #2, #3에 저장된 프레임 #1, #(1+n), #(1+2n)은 각각 프레임 큐 #1, #(1+n), #(1+2n)에 저장될 수 있다.

[0079] 일 실시예에서, 프레임 유닛(240)은 현재의 드로우 넘버(draw number)를 읽는 단계, 드로우 넘버와 프레임 큐(242)에 저장된 프레임들의 프레임 넘버를 각각 비교하는 단계, 드로우 넘버가 프레임 넘버와 동일하면 해당 프레임을 출력하는 단계, 출력에 성공한 경우 드로우 넘버를 1 증가시키는 단계 및 프레임 큐(242)가 공백이 될 때까지 해당 동작들을 반복하는 단계를 통해 동작할 수 있다.

[0080] 즉, 프레임 유닛(240)은 프레임 큐(242)에 저장된 프레임들을 드로우(draw)를 통해 화면에 출력할 수 있다. 이때, 드로우(draw) 순서는 드로우 넘버(draw number)에 의해 결정될 수 있다. 여기에서, 드로우 넘버(draw number)는 드로우 순서를 결정하는데 사용되는 식별번호에 해당할 수 있다. 프레임 유닛(240)은 드로우 넘버가 1로 초기화된 상태에서 드로우 동작을 개시할 수 있으며, 드로우가 수행될 때마다 드로우 넘버를 1씩 증가시킬 수 있다. 결과적으로, 레이 트레이싱 시스템에 의해 수행되는 드로우 동작은 드로우 넘버와 일치하는 프레임 넘버를 가진 프레임이 프레임 큐(242)에 있으면 해당 프레임을 화면에 뿌려주고, 없으면 대기하는 동작에 해당할 수 있다. 따라서, 프레임 #1부터 차례대로 드로우가 수행될 수 있으며, 이는 디스플레이 순서와 동일하게 처리될 수 있다.

[0081] 도 5에서, Chip#1, #2, #3은 각각 frame #1, #2, #3을 생성하고, 이후부터는 총 chip 수가 더해진 #4, #5, #6을 생성할 수 있다. 각 칩(레이 트레이싱 코어)은 프레임 버퍼를 3개씩 가지고 있으며, 생성된 프레임들은 프레

임 버퍼에 차례로 저장될 수 있다. Chip #1의 프레임 버퍼 1, 2, 3에는 각각 프레임 #1, #4, #7이 저장되며, chip #2의 프레임 버퍼 1, 2, 3에는 프레임 #2, #5, #8이 저장되고, chip #3의 프레임 버퍼 1, 2, 3에는 프레임 #3, #6, #9가 저장될 수 있다.

[0082] 각 칩의 프레임 버퍼에 저장된 프레임들은 프레임 넘버와 같은 순번의 프레임 큐(242)에 저장될 수 있다. Chip #1의 프레임 버퍼 1, 2, 3에 저장된 프레임 #1, #4, #7은 각각 프레임 큐 1, 4, 7에 저장되고, chip #2, #3의 프레임 버퍼에 저장된 프레임들도 같은 방식으로 프레임 큐(242)에 저장될 수 있다. 프레임 큐(242)에 저장된 프레임들은 드로우(draw)에 의해 차례대로 화면에 뿌려지게 되며, 드로우는 프레임 큐 1부터 드로우 넘버와 프레임 넘버를 비교해가며 차례로 수행될 수 있다. 이때, 드로우 넘버는 드로우(draw)를 수행하면 1씩 증가될 수 있다.

[0084] 도 6은 본 발명에 따른 프레임 분할 방식의 멀티칩 레이 트레이싱 방법의 동작 과정을 설명하는 순서도이다.

[0085] 도 6을 참조하면, 본 발명에 따른 프레임 분할 방식의 멀티칩 레이 트레이싱 방법은 다음과 같이 수행될 수 있다.

[0086] 멀티칩 수 설정(MaxCardNum check) 단계(S610)에서 멀티칩(multi-chip)(ASIC or FPGA) 시스템에 사용되는 총 칩(레이 트레이싱 코어)의 수를 확인하고 드로우 넘버(draw number)를 1로 설정할 수 있다.

[0087] 프레임 할당(Frame assignment) 단계(S620)는 각 칩에 렌더링(rendering) 할 프레임을 할당하는 단계에 해당할 수 있다. 활용한 칩이 #1, #2, ~#n인 경우, 프레임 #1, #2, ~#n이 각 칩에 할당될 수 있다.

[0088] 기하데이터 전송(Geometry data transmission) 단계(S630)에서 각 칩은 할당받은 프레임에 대한 기하(geometry) 정보를 전송받을 수 있다. 이후 렌더링 완료(rendering done) 단계(S640)에서는 전송받은 기하(geometry) 정보를 바탕으로 렌더링(rendering)을 수행할 수 있다. 렌더링이 완료되면 결과 이미지를 프레임 버퍼(frame buffer)에 저장하는 프레임 버퍼 저장 단계(S650)와 다음에 렌더링할 프레임의 프레임 넘버를 설정하는 프레임 넘버 셋팅(frame number setting) 단계(S690)가 진행될 수 있다.

[0089] 프레임 버퍼 저장(Store to frame buffer) 단계(S650)에서는 렌더링이 완료된 프레임을 해당 칩에 할당된 프레임 버퍼에 저장할 수 있다. 프레임 큐 저장(Store to frame queue) 단계(S660)는 각 칩의 프레임 버퍼에 저장된 프레임들을 프레임 큐로 저장하는 단계이며, 프레임 큐의 크기는 총 프레임 버퍼의 수와 같을 수 있다. 프레임 큐에 프레임들이 저장될 때, 프레임 넘버를 확인하여 그에 맞는 위치의 프레임 큐에 저장될 수 있다. 프레임이 저장될 프레임 큐의 위치는 프레임 넘버를 프레임 큐의 크기로 나눈 결과의 나머지 값으로 결정될 수 있다.

[0090] 프레임 넘버와 드로우 넘버 비교 단계(S670)에서는 프레임 큐에 저장된 프레임의 프레임 넘버와 드로우 넘버가 같은지를 확인할 수 있다. 두 값이 같지 않으면 대기하고, 같다면 드로우 프레임(draw frame) 단계(S680)를 통해 해당 프레임을 드로우하여 디스플레이 할 수 있다. 드로우가 진행되면 드로우 넘버는 1 증가하고, 프레임 넘버와 비교하는 단계(S670)로 돌아가 저장된 프레임이 드로우 넘버와 같은지 확인할 수 있다.

[0091] 프레임 넘버 셋팅(Frame number setting) 단계(S690)에서는 칩에 재할당될 프레임에 대한 프레임 넘버를 설정할 수 있다. 이는 총 칩의 수에 의해 결정되며, 최초에 받은 프레임 넘버에 총 칩의 수를 더하여 계산될 수 있다. Chip #1이 최초에 프레임 #1을 할당받고, 총 칩의 총 개수가 3이라면, 이후부터 프레임 넘버는 #4, #7, #10으로 설정될 수 있다. Chip #2가 최초에 프레임 #2를 할당받았다면, 이후부터 프레임 넘버는 #5, #8, #11로 설정될 수 있다.

[0092] 프레임 넘버와 m(total frame number)의 비교 단계(S691)는 프레임 넘버 셋팅 단계(S690)에서 설정된 프레임 넘버와 장면(scene)의 총 프레임 수를 비교하는 단계에 해당할 수 있다. 설정된 프레임 넘버가 m보다 작으면, 각 칩은 해당 프레임에 대한 렌더링을 수행하며, m보다 크면 해당 장면에 대한 렌더링은 종료될 수 있다.

[0094] 상기에서는 본 발명의 바람직한 실시예를 참조하여 설명하였지만, 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.

## *부호의 설명*

[0096] 100: 화면 분할 방식의 멀티칩 레이 트레이싱 시스템

110: 중앙처리유닛 120: 시스템 메모리

130: 멀티칩

200: 프레임 분할 방식의 멀티칩 레이 트레이싱 시스템

230: 레이 트레이싱 코어들 231: 버스 인터페이스 유닛

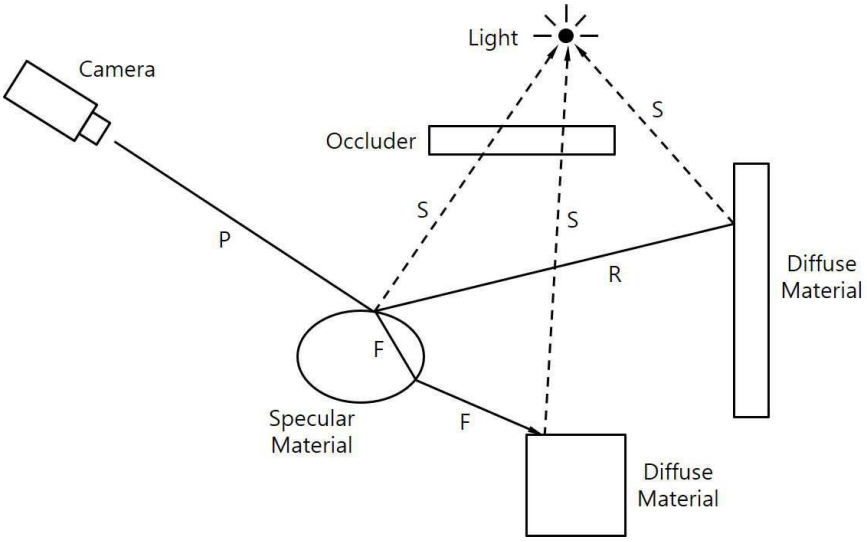
232: 트리 빌드 유닛 233: 레이 트레이싱 유닛

240: 프레임 유닛 241: 프레임 버퍼

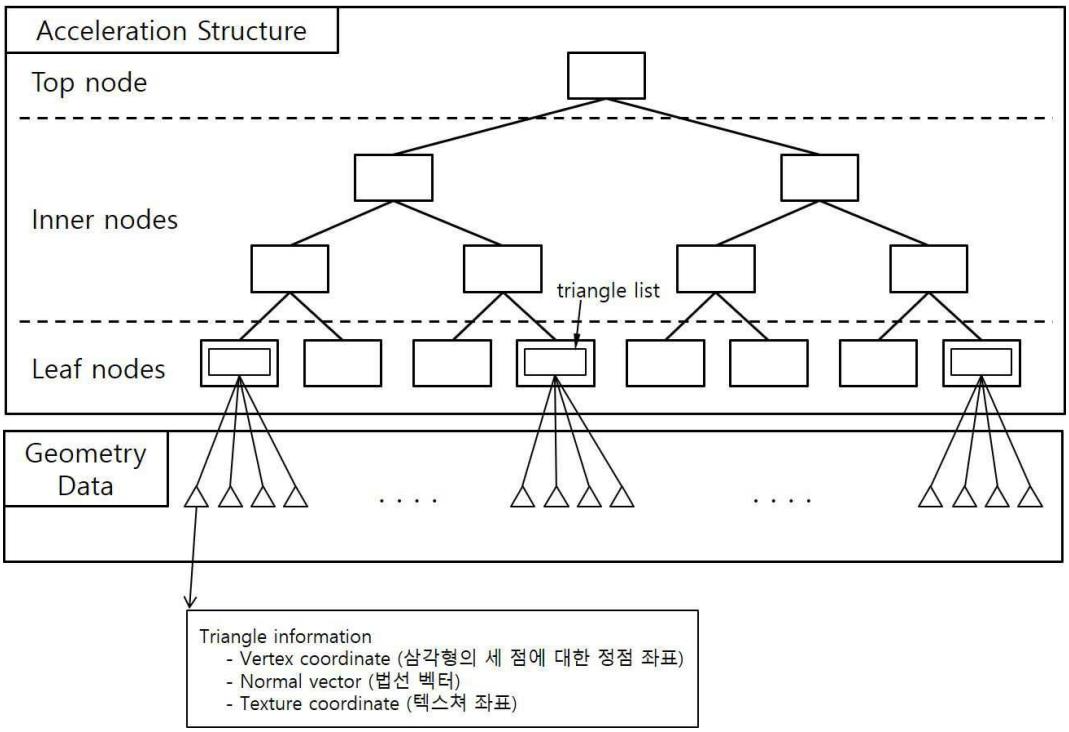
242: 프레임 큐

## 도면

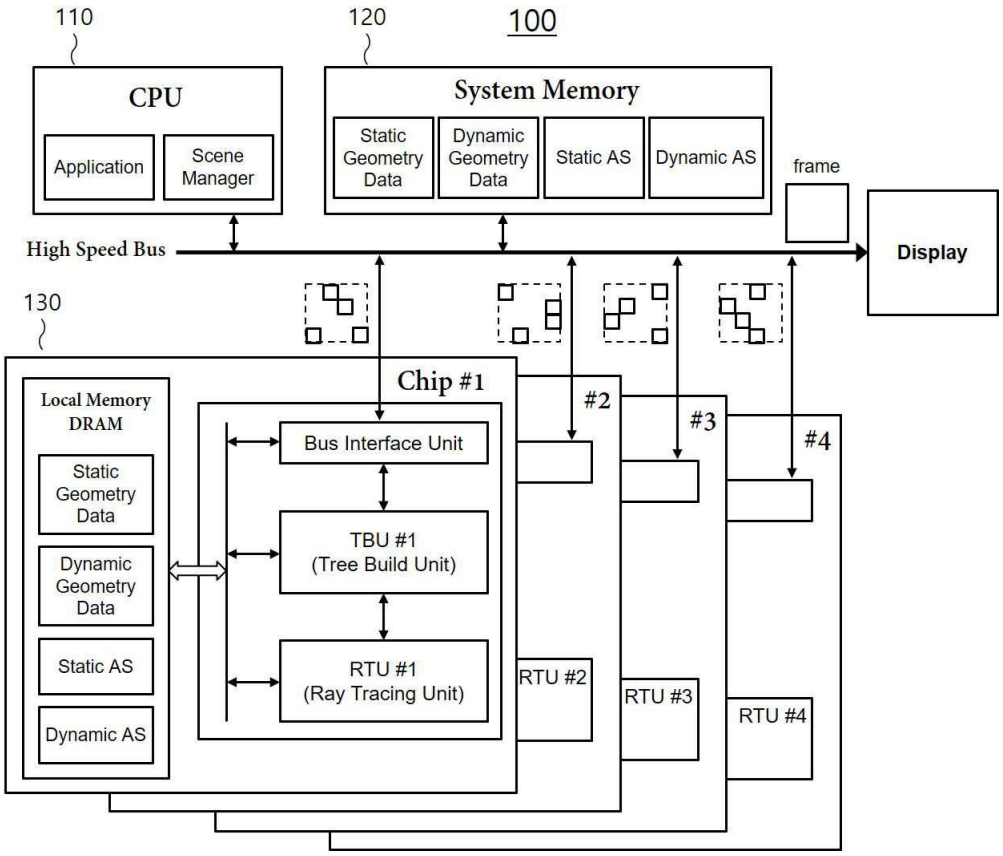
### 도면1



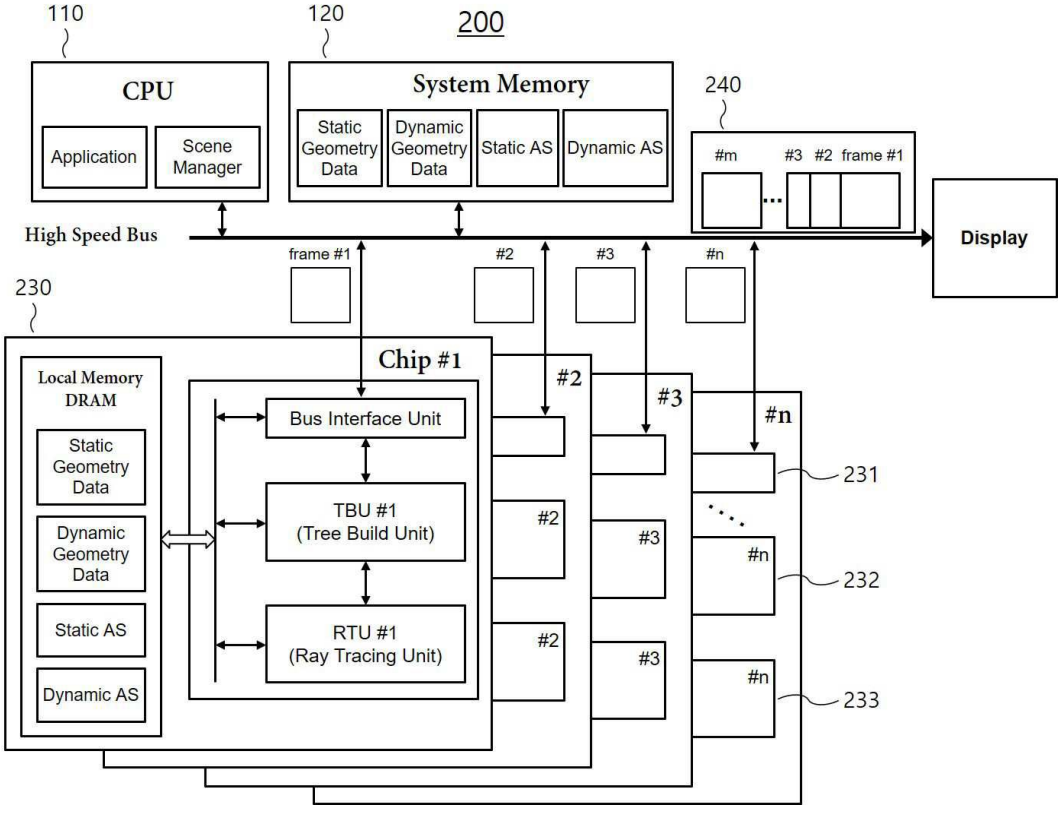

### 도면2



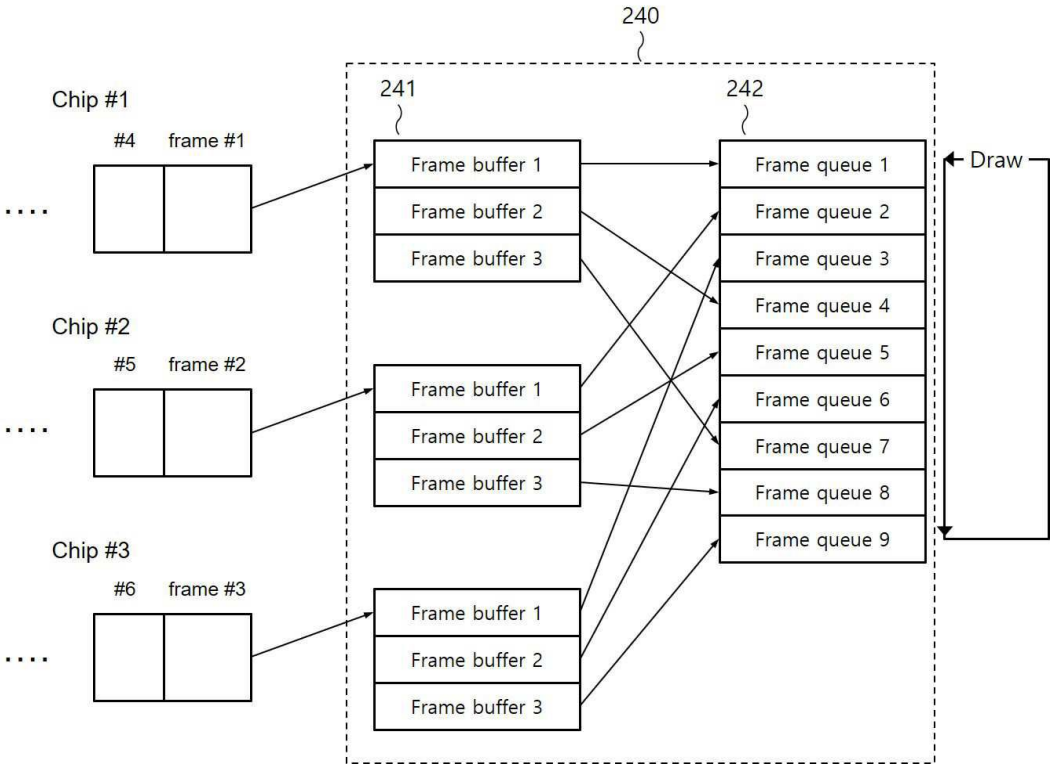
도면3


110
120
100
CPU
Application
Scene Manager
System Memory
Static Geometry Data
Dynamic Geometry Data
Static AS
Dynamic AS
frame
High Speed Bus
Display
130
Chip #1
Local Memory DRAM
Static Geometry Data
Dynamic Geometry Data
Static AS
Dynamic AS
Bus Interface Unit
TBU #1 (Tree Build Unit)
RTU #1 (Ray Tracing Unit)
#2
#3
#4
RTU #2
RTU #3
RTU #4

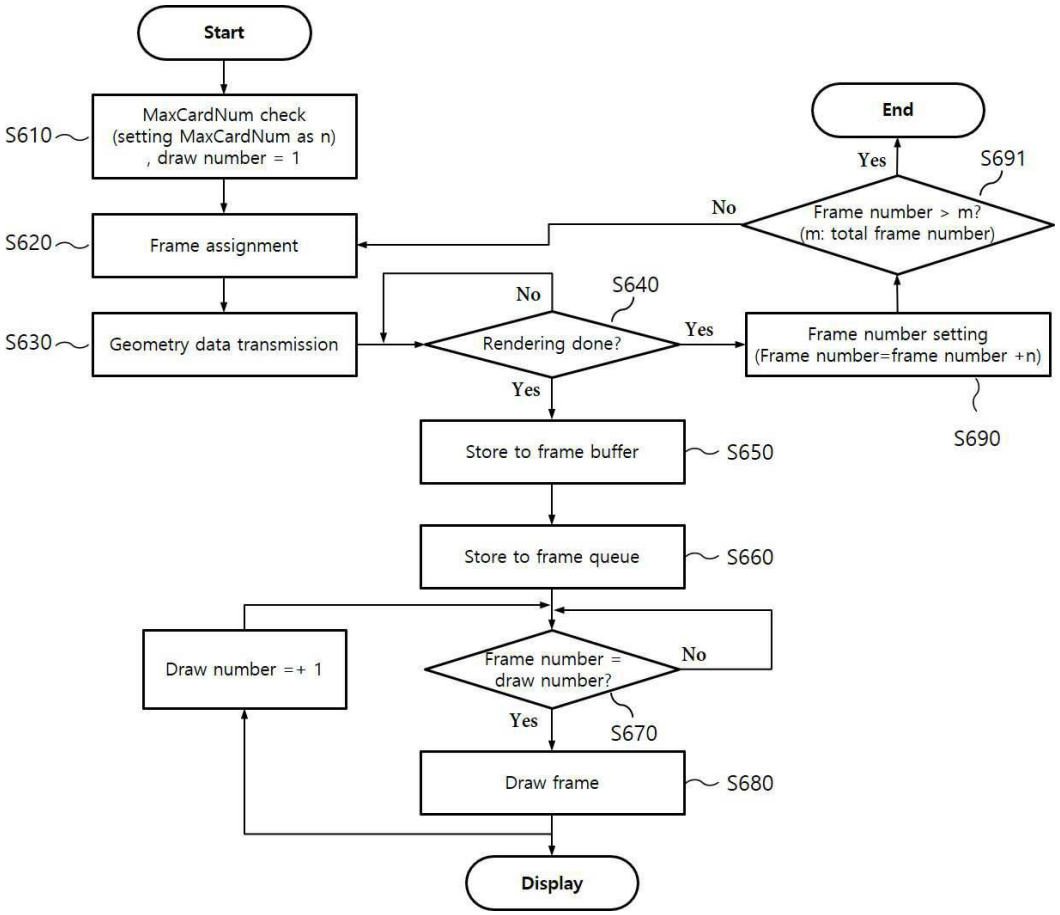
도면4


110
120
200
CPU
Application
Scene Manager
System Memory
Static Geometry Data
Dynamic Geometry Data
Static AS
Dynamic AS
240
#m
#3
#2
frame #1
High Speed Bus
Display
frame #1
#2
#3
#n
230
Chip #1
Local Memory DRAM
Static Geometry Data
Dynamic Geometry Data
Static AS
Dynamic AS
Bus Interface Unit
TBU #1 (Tree Build Unit)
RTU #1 (Ray Tracing Unit)
231
232
233

도면5


240
241
242
Chip #1
#4
frame #1
Chip #2
#5
frame #2
Chip #3
#6
frame #3
Frame buffer 1
Frame buffer 2
Frame buffer 3
Frame buffer 1
Frame buffer 2
Frame buffer 3
Frame buffer 1
Frame buffer 2
Frame buffer 3
Frame queue 1
Frame queue 2
Frame queue 3
Frame queue 4
Frame queue 5
Frame queue 6
Frame queue 7
Frame queue 8
Frame queue 9
Draw

도면6


Start
S610
MaxCardNum check
(setting MaxCardNum as n)
, draw number = 1
S620
Frame assignment
S630
Geometry data transmission
No
S640
Rendering done?
Yes
Yes
Store to frame buffer
S650
Store to frame queue
S660
Draw number =+ 1
Frame number =
draw number?
No
Yes
S670
Draw frame
S680
Display
End
Yes
S691
No
Frame number > m?
(m: total frame number)
Frame number setting
(Frame number=frame number +n)
S690