(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2022년09월13일
(11) 등록번호 10-2442487
(24) 등록일자 2022년09월06일

(51) 국제특허분류(Int. Cl.)
G06T 1/20 (2018.01) *G06T 15/00* (2006.01)
H04N 13/30 (2020.01)
(52) CPC특허분류
G06T 1/20 (2013.01)
G06T 15/005 (2013.01)
(21) 출원번호 10-2017-0112643
(22) 출원일자 2017년09월04일
심사청구일자 2020년08월27일
(65) 공개번호 10-2018-0027369
(43) 공개일자 2018년03월14일
(30) 우선권주장
1615012.0 2016년09월05일 영국(GB)
(56) 선행기술조사문헌
US20150287165 A1*
(뒷면에 계속)
전체 청구항 수 : 총 18 항

(73) 특허권자
암, 리미티드
영국 캠브리지 씨비1 9엔제이 풀번로드 110
(72) 발명자
마틴, 사무엘
영국, 캠브리지셔 씨비1 9엔제이, 캠브리지, 풀번로드 110, 암, 리미티드 내
(74) 대리인
(유)한양특허법인

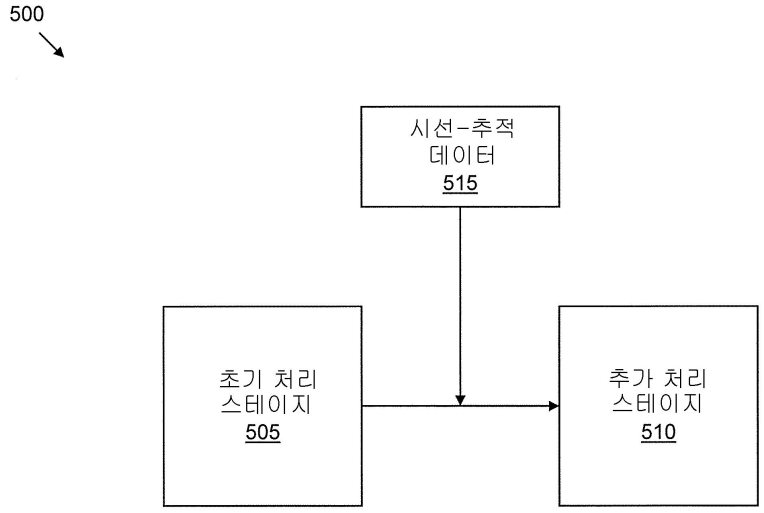
심사관 : 이후락

(54) 발명의 명칭 **그래픽 처리 시스템 및 그래픽 프로세서**

(57) 요 약

그래픽 처리 시스템은, 적어도 초기 처리 스테이지와 추가 처리 스테이지를 포함하는 그래픽 처리 파이프라인을 포함한다. 제1 해상도의 장면을 위한 데이터와 제2 해상도의 장면을 위한 데이터를 초기 처리 스테이지에서 처리한다. 제1 처리 스테이지에서 제1 및 제2 해상도의 장면을 위한 데이터를 처리한 후, 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 시선-추적 데이터를 수신한다. 장면의 적어도 하나의 하위-영역을 시선-추적 데이터로부터 식별한다. 제1 해상도의 장면을 위한 데이터와, 제2 해상도의 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터만 추가 처리 스테이지에서 처리한다. 장면은 추가 처리 스테이지에서 처리된 장면을 위한 데이터를 결합함으로써 렌더링된다.

대 표 도



(52) CPC특허분류
H04N 13/383 (2018.05)
G06T 2200/28 (2013.01)

(56) 선행기술조사문헌
KR1020140014870 A*
US20150287167 A1
US20150287158 A1
US20140247277 A1
KR1020150099463 A
US20040233219 A1
*는 심사관에 의하여 인용된 문헌

명 세 서

청구범위

청구항 1

적어도 초기 처리 스테이지와 추가 처리 스테이지를 포함하는 그래픽 처리 파이프라인을 포함하는 그래픽 처리 시스템을 동작하는 방법으로서,

상기 초기 처리 스테이지에서 제1 해상도로 장면을 위한 데이터를 처리하는 단계;

상기 초기 처리 스테이지에서 제2 해상도로 상기 장면을 위한 데이터를 처리하는 단계;

상기 초기 처리 스테이지에서 상기 제1 및 제2 해상도로 상기 장면을 위한 데이터를 처리한 후, 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 시선-추적 데이터를 수신하는 단계;

수신된 상기 시선-추적 데이터로부터 상기 장면의 적어도 하나의 하위-영역을 식별하는 단계;

상기 추가 처리 스테이지에서 상기 제1 해상도로 상기 장면을 위한 데이터를 처리하는 단계;

상기 추가 처리 스테이지에서 상기 제2 해상도로 상기 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터만 처리하는 단계;

상기 제1 해상도로 상기 추가 처리 스테이지에서 처리되는 상기 장면을 위한 데이터와, 상기 제2 해상도로 상기 추가 처리 스테이지에서 처리되는 상기 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터를 결합함으로써 상기 장면을 렌더링하는 단계를 포함하는, 그래픽 처리 시스템 동작 방법.

청구항 2

청구항 1에 있어서, 상기 그래픽 처리 파이프라인은 상기 초기 처리 스테이지와 상기 추가 처리 스테이지 중간에 중간 처리 스테이지를 포함하며, 상기 방법은:

상기 중간 처리 스테이지에서 상기 제1 해상도로 상기 장면을 위한 데이터를 처리하는 단계; 및

상기 중간 처리 스테이지에서 상기 제2 해상도로 상기 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터만 처리하는 단계를 포함하며,

상기 시선-추적 데이터는 상기 중간 처리 스테이지에서 상기 제1 및 제2 해상도로 상기 장면을 위한 데이터를 처리하기 전에 수신되며, 상기 중간 처리 스테이지에서 상기 제1 및 제2 해상도로 상기 장면을 위한 데이터를 처리하기 전에 상기 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지션에 관한 것인, 그래픽 처리 시스템 동작 방법.

청구항 3

청구항 1에 있어서, 상기 그래픽 처리 파이프라인은 상기 초기 처리 스테이지와 상기 추가 처리 스테이지 중간에 중간 처리 스테이지를 포함하며, 상기 방법은:

상기 중간 처리 스테이지에서 상기 제1 해상도로 상기 장면을 위한 데이터를 처리하는 단계; 및

상기 중간 처리 스테이지에서 상기 제2 해상도로 상기 장면을 위한 데이터를 처리하는 단계를 포함하며,

상기 시선-추적 데이터는 상기 중간 처리 스테이지에서 상기 제1 및 제2 해상도로 상기 장면을 위한 데이터를 처리한 후에 수신되며, 상기 중간 처리 스테이지에서 상기 제1 및 제2 해상도로 상기 장면을 위한 데이터를 처리한 후에 상기 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 것인, 그래픽 처리 시스템 동작 방법.

청구항 4

청구항 2 또는 청구항 3에 있어서, 상기 추가 처리 스테이지를 위한 처리는 상기 중간 처리 스테이지를 위한 처리보다 더 높은 계산 부담을 갖는, 그래픽 처리 시스템 동작 방법.

청구항 5

청구항 2 또는 청구항 3에 있어서, 상기 중간 처리 스테이지는 하나 이상의 지아메트리 동작(geometry operation)을 포함하는, 그래픽 처리 시스템 동작 방법.

청구항 6

청구항 1 내지 청구항 3 중 어느 한 항에 있어서, 상기 추가 처리 스테이지를 위한 처리는 상기 초기 처리 스테이지를 위한 처리보다 더 높은 계산 부담을 갖는, 그래픽 처리 시스템 동작 방법.

청구항 7

청구항 1 내지 청구항 3 중 어느 한 항에 있어서, 상기 추가 처리 스테이지는 하나 이상의 픽셀 처리 동작을 포함하는, 그래픽 처리 시스템 동작 방법.

청구항 8

청구항 1 내지 청구항 3 중 어느 한 항에 있어서, 상기 초기 처리 스테이지는 하나 이상의 데이터 버퍼링 동작을 포함하는, 그래픽 처리 시스템 동작 방법.

청구항 9

청구항 1 내지 청구항 3 중 어느 한 항에 있어서, 상기 그래픽 처리 파이프라인의 각각의 스테이지에서의 장면을 위한 데이터의 처리는 상기 그래픽 처리 파이프라인의 각각의 스테이지에서의 상기 장면을 위한 데이터의 프레임을 처리하는 것을 포함하는, 그래픽 처리 시스템 동작 방법.

청구항 10

청구항 1 내지 청구항 3 중 어느 한 항에 있어서, 상기 제1 해상도는 상기 제2 해상도와 비교하여 상대적으로 높은, 그래픽 처리 시스템 동작 방법.

청구항 11

청구항 1 내지 청구항 3 중 어느 한 항에 있어서, 수신된 상기 시선-추적 데이터는, 상기 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 포비아(fovea)의 현재 고정점에 관한 것이며,

상기 장면의 식별된 적어도 하나의 하위 영역은, 상기 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 포비아의 현재 고정점 주위에 중심을 둔 하위 영역에 관한 것인, 그래픽 처리 시스템 동작 방법.

청구항 12

청구항 1 내지 청구항 3 중 어느 한 항에 있어서, 상기 장면을 위한 데이터는 왼쪽 눈 뷰와 관련된 데이터와 오른쪽 눈 뷰와 관련된 데이터를 포함하고, 상기 그래픽 처리 파이프라인의 각각의 스테이지를 위한 데이터의 처리는 상기 왼쪽 눈 뷰와 관련된 데이터 처리와 상기 오른쪽 눈 뷰와 관련된 데이터 처리를 포함하며,

수신된 상기 시선-추적 데이터는 상기 가상 현실 사용자 디바이스의 사용자의 왼쪽 눈의 포비아의 현재의 포지셔닝과, 상기 가상 현실 사용자 디바이스의 사용자의 오른쪽 눈의 포비아의 현재의 포지셔닝에 관한 것인, 그래픽 처리 시스템 동작 방법.

청구항 13

청구항 12에 있어서, 상기 장면의 식별된 적어도 하나의 하위 영역은 상기 가상 현실 사용자 디바이스의 사용자의 왼쪽 눈의 포비아의 현재의 포지셔닝 주위에 중심을 둔 왼쪽 눈 하위 영역과, 상기 가상 현실 사용자 디바이스의 사용자의 오른쪽 눈의 포비아의 현재의 포지셔닝 주위에 중심을 둔 오른쪽 눈 하위 영역에 관한 것이며,

상기 장면을 렌더링하는 단계는:

상기 제1 해상도로 상기 추가 처리 스테이지에서 처리되는 상기 장면을 위한 왼쪽 눈 데이터와, 상기 제2 해상도로 상기 추가 처리 스테이지에서 처리되는 상기 장면의 식별된 왼쪽 눈 하위 영역에 대응하는 데이터를 결합하는 단계; 및

상기 제1 해상도로 상기 추가 처리 스테이지에서 처리되는 상기 장면을 위한 오른쪽 눈 데이터와, 상기 제2 해상도로 상기 추가 처리 스테이지에서 처리되는 상기 장면의 식별된 오른쪽 눈 하위 영역에 대응하는 데이터를 결합하는 단계를 포함하는, 그래픽 처리 시스템 동작 방법.

청구항 14

청구항 13에 있어서, 식별된 상기 왼쪽 눈 하위 영역의 적어도 일부분이 식별된 상기 오른쪽 눈 하위 영역의 적어도 일부분과 상이한, 그래픽 처리 시스템 동작 방법.

청구항 15

청구항 1 내지 청구항 3 중 어느 한 항에 있어서,

상기 초기 처리 스테이지에서 제3 해상도로 상기 장면을 위한 데이터를 처리하는 단계를 포함하며,

상기 시선-추적 데이터는 상기 초기 처리 스테이지에서 상기 제3 해상도로 상기 장면을 위한 데이터를 처리한 후에 수신되며, 상기 초기 처리 스테이지에서 상기 제3 해상도로 상기 장면을 위한 데이터를 처리한 후에 상기 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 것이며, 상기 방법은:

상기 추가 처리 스테이지에서 상기 제3 해상도로 상기 장면을 위한 데이터를 처리하는 단계; 및

상기 추가 처리 스테이지에서 상기 제3 해상도로 상기 장면의 추가 하위 영역에 대응하는 데이터만 처리하는 단계를 포함하며,

상기 장면을 렌더링하는 단계는 상기 제3 해상도로 상기 추가 처리 스테이지에서 처리되는 장면을 위한 데이터를 결합하는 단계를 더 포함하는, 그래픽 처리 시스템 동작 방법.

청구항 16

청구항 15에 있어서, 수신된 상기 시선-추적 데이터로부터 상기 장면의 추가 하위 영역을 식별하는 단계; 또는

상기 장면의 식별된 적어도 하나의 하위 영역을 기초로 하여 상기 장면의 추가 하위 영역을 계산하는 단계를 포함하는, 그래픽 처리 시스템 동작 방법.

청구항 17

삭제

청구항 18

삭제

청구항 19

컴퓨터로 판독 가능한 저장 매체 상에 저장되며 소프트웨어 코드를 포함하는 컴퓨터 프로그램으로서, 상기 컴퓨터 프로그램이 적어도 초기 처리 스테이지와 추가 처리 스테이지를 포함하는 그래픽 처리 파이프라인을 포함하는 그래픽 처리 시스템에서 작동될 때, 상기 소프트웨어 코드는, 상기 그래픽 처리 시스템을 동작시키는 방법을 수행하도록 되어 있으며, 상기 방법은,

상기 초기 처리 스테이지에서 제1 해상도로 장면을 위한 데이터를 처리하는 단계;

상기 초기 처리 스테이지에서 제2 해상도로 상기 장면을 위한 데이터를 처리하는 단계;

상기 초기 처리 스테이지에서 상기 제1 및 제2 해상도로 장면을 위한 데이터를 처리한 후, 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 시선-추적 데이터를 수신하는 단계;

수신된 상기 시선-추적 데이터로부터 상기 장면의 적어도 하나의 하위-영역을 식별하는 단계;

상기 추가 처리 스테이지에서 상기 제1 해상도로 상기 장면을 위한 데이터를 처리하는 단계;

상기 추가 처리 스테이지에서 상기 제2 해상도로 상기 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터만 처리하는 단계; 및

상기 제1 해상도로 상기 추가 처리 스테이지에서 처리되는 상기 장면을 위한 데이터와, 상기 제2 해상도로 상기 추가 처리 스테이지에서 처리되는 상기 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터를 결합함으로써 상기 장면을 렌더링하는 단계를 포함하는, 컴퓨터 프로그램.

청구항 20

적어도 초기 처리 스테이지와 추가 처리 스테이지를 포함하는 그래픽 처리 파이프라인을 포함하는 그래픽 처리 시스템으로서,

상기 초기 처리 스테이지에서 제1 해상도로 장면을 위한 데이터를 처리하고;

상기 초기 처리 스테이지에서 제2 해상도로 상기 장면을 위한 데이터를 처리하고;

상기 초기 처리 스테이지에서 상기 제1 및 제2 해상도로 상기 장면을 위한 데이터를 처리한 후, 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 시선-추적 데이터를 수신하고;

수신된 상기 시선-추적 데이터로부터 상기 장면의 적어도 하나의 하위-영역을 식별하고;

상기 추가 처리 스테이지에서 상기 제1 해상도로 상기 장면을 위한 데이터를 처리하고;

상기 추가 처리 스테이지에서 상기 제2 해상도로 상기 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터만 처리하며;

상기 제1 해상도로 상기 추가 처리 스테이지에서 처리되는 상기 장면을 위한 데이터와, 상기 제2 해상도로 상기 추가 처리 스테이지에서 처리되는 상기 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터를 결합함으로써 상기 장면을 렌더링하도록 되어 있는, 그래픽 처리 시스템.

청구항 21

삭제

청구항 22

삭제

청구항 23

삭제

청구항 24

삭제

청구항 25

삭제

청구항 26

삭제

청구항 27

삭제

청구항 28

삭제

청구항 29

삭제

청구항 30

삭제

청구항 31

삭제

청구항 32

삭제

청구항 33

삭제

청구항 34

삭제

발명의 설명

기 술 분 야

[0001] 본 발명은 그래픽 처리 시스템 및 그래픽 프로세서에 관한 것이다.

배 경 기 술

[0002] 포비티드 렌더링(Foveated rendering)은, 디스플레이되고 있는 이미지의 일 부분이 고 해상도로 렌더링되며, 이미지의 하나 이상의 다른 부분이 저 해상도로 렌더링되는 렌더링 기술이다. 이것은, 사용자가 직접 보고 있는 이미지의 부분이 시각적 허용 가능성을 위해 고 해상도로 렌더링될 필요가 있을 수 있는 반면, 사용자가 직접 보고 있지 않은 이미지의 주변 영역은 여전히 시각적으로 허용 가능하게 보이면서도 저 해상도로 렌더링될 수 있기 때문이다. 포비티드 렌더링은, 최고 해상도로 전체 이미지를 렌더링하기 보다는, 저 해상도로 이미지의 주변 영역을 렌더링함으로써, 그래픽 처리 유닛(GPU)에 대한 렌더링 부담을 줄이는데 사용될 수 있다.

[0003] 포비티드 렌더링은 이미지의 고 해상도 버전이 렌더링될 하나 이상의 고정점을 식별하며, 고정점 또는 고정점들로부터 더 먼 영역이 저 해상도로 렌더링되는 것을 수반할 수 있다. 이미지의 최고 해상도 영역의 위치, 즉 고정점 또는 고정점들을 결정할 수 있는 상이한 방식들이 있다. 예컨대, 머리-추적 또는 눈-추적 시스템이 사용자가 이미지에서 어디를 보고 있는지를 식별하려고 사용될 수 있다.

[0004] 포비티드 렌더링의 하나의 용도는 예컨대 가상 현실 머리-장착 디스플레이(VR HMDs)와 같은 가상 현실(VR)을 위한 이미지를 렌더링할 때이다. 고 해상도 VR HMDs는, 심각한 배럴 왜곡 특징이 있는 렌즈를 사용할 수 도 있다. 이것의 효과는, 각각의 눈에 대해 디스플레이의 중심을 향한 렌더링된 이미지가 확대되는 반면, 주변 구역은 모두 축소된다는 점이다. 주변 영역은 그러므로, 사용자에 대한 전체적인 시각 효과에 임의의 상당한 손실이 없이도, 중앙의 확대된 영역보다 더 저 품질로 렌더링될 수 있다.

발명의 내용

해결하려는 과제

[0005] 그러나 예컨대 포비티드 렌더링을 실행할 때, 그래픽 프로세서 및 그래픽 처리 시스템의 동작에 대한 개선의 여지가 남아 있다.

과제의 해결 수단

[0006] 본 발명의 제1 양상에 따르면, 적어도 초기 처리 스테이지와 추가 처리 스테이지를 포함하는 그래픽 처리 파이프라인을 포함하는 그래픽 처리 시스템을 동작하는 방법이 제공되며, 이 방법은:

[0007] 초기 처리 스테이지에서 제1 해상도로 장면을 위한 데이터를 처리하는 단계;

[0008] 초기 처리 스테이지에서 제2 해상도로 장면을 위한 데이터를 처리하는 단계;

[0009] 초기 처리 스테이지에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리한 후, 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 시선-추적 데이터를 수신하는 단계;

[0010] 수신된 시선-추적 데이터로부터 장면의 적어도 하나의 하위-영역을 식별하는 단계;

[0011] 추가 처리 스테이지에서 제1 해상도로 장면을 위한 데이터를 처리하는 단계;

[0012] 추가 처리 스테이지에서 제2 해상도로 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터만 처리하는 단계;

[0013] 제1 해상도로 추가 처리 스테이지에서 처리되는 장면을 위한 데이터와, 제2 해상도로 추가 처리 스테이지에서 처리되는 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터를 결합함으로써 장면을 렌더링하는 단계를 포함한다.

[0014] 본 발명의 제2 양상에 따르면, 그래픽 처리 파이프라인을 포함하는 그래픽 처리 시스템을 동작하는 방법이 제공되며, 이 방법은 그래픽 처리 파이프라인에서 늦게 래칭된(late-latched) 눈-추적 데이터를 사용하여 장면의 하나 이상의 다른 영역의 해상도와 비교하여 상대적으로 고 해상도로 렌더링되는 장면의 하나 이상의 영역을 식별하는 단계를 포함한다.

[0015] 본 발명의 제3 양상에 따르면, 적어도 초기 처리 스테이지와 추가 처리 스테이지를 포함하는 그래픽 처리 파이프라인을 포함하는 그래픽 처리 시스템을 동작하는 방법이 제공되며, 이 방법은:

[0016] 제1 시간에 그래픽 처리 작업 아이템을 위한 입력 데이터를 그래픽 처리 파이프라인에 제공하는 단계; 및

[0017] 그래픽 처리 파이프라인이 그래픽 처리 작업 아이템을 처리하고 있는 동안 제2의 추후 시간에 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 눈-추적 데이터를 그래픽 처리 파이프라인에 제공하는 단계를 포함한다.

[0018] 본 발명의 제4 양상에 따르면, 적어도 초기 처리 스테이지와 추가 처리 스테이지를 포함하는 그래픽 처리 파이프라인과 관련되는 그래픽 프로세서를 동작하는 방법이 제공되며, 이 방법은:

[0019] 초기 처리 스테이지에서 제1 해상도로 장면을 위한 데이터를 처리하는 단계;

[0020] 초기 처리 스테이지에서 제2 해상도로 장면을 위한 데이터를 처리하는 단계;

[0021] 초기 처리 스테이지에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리한 후, 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 시선-추적 데이터를 수신하는 단계;

[0022] 수신된 시선-추적 데이터로부터 장면의 적어도 하나의 하위-영역을 식별하는 단계;

[0023] 추가 처리 스테이지에서 제1 해상도로 장면을 위한 데이터를 처리하는 단계;

[0024] 추가 처리 스테이지에서 제2 해상도로 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터만 처리하는 단계; 및

[0025] 제1 해상도로 추가 처리 스테이지에서 처리되는 장면을 위한 데이터와, 제2 해상도로 추가 처리 스테이지에서 처리되는 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터를 결합함으로써 장면을 렌더링하는 단계를 포함한다.

[0026] 본 발명의 제5 양상에 따르면, 컴퓨터 프로그램이 그래픽 처리 시스템에서 실행될 때, 하나 이상의 그러한 방법을 실행하도록 되는 소프트웨어 코드를 포함하는 컴퓨터 프로그램이 제공된다.

[0027] 본 발명의 제6 양상에 따르면, 적어도 초기 처리 스테이지와 추가 처리 스테이지를 포함하는 그래픽 처리 파이프라인을 포함하는 그래픽 처리 시스템이 제공되며, 이 그래픽 처리 시스템은:

[0028] 초기 처리 스테이지에서 제1 해상도로 장면을 위한 데이터를 처리하고;

[0029] 초기 처리 스테이지에서 제2 해상도로 장면을 위한 데이터를 처리하고;

[0030] 초기 처리 스테이지에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리한 후, 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 시선-추적 데이터를 수신하고;

[0031] 수신된 시선-추적 데이터로부터 장면의 적어도 하나의 하위-영역을 식별하고;

[0032] 추가 처리 스테이지에서 제1 해상도로 장면을 위한 데이터를 처리하고;

[0033] 추가 처리 스테이지에서 제2 해상도로 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터만 처리하며;

[0034] 제1 해상도로 추가 처리 스테이지에서 처리되는 장면을 위한 데이터와, 제2 해상도로 추가 처리 스테이지에서 처리되는 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터를 결합함으로써 장면을 렌더링하도록 되어 있다.

[0035] 본 발명의 제7 양상에 따르면, 그래픽 처리 파이프라인을 포함하는 그래픽 처리 시스템이 제공되며, 이 그래픽 처리 시스템은, 그래픽 처리 파이프라인에서 늦게 래칭된 눈-추적 데이터를 사용하여 장면의 하나 이상의 다른 영역의 해상도와 비교하여 상대적으로 고 해상도로 렌더링되는 장면의 하나 이상의 영역을 식별하도록 구성된다.

[0036] 본 발명의 제8 양상에 따르면, 적어도 초기 처리 스테이지와 추가 처리 스테이지를 포함하는 그래픽 처리 파이프라인을 포함하는 그래픽 처리 시스템이 제공되며, 이 그래픽 처리 시스템은:

[0037] 제1 시간에 그래픽 처리 작업 아이템을 위한 입력 데이터를 그래픽 처리 파이프라인에 제공하며;

[0038] 그래픽 처리 파이프라인이 그래픽 처리 작업 아이템을 처리하고 있는 동안 제2의 더 늦은 시간에 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 눈-추적 데이터를 그래픽 처리 파이프라인에 제공하도록 되어 있다.

[0039] 본 발명의 제9 양상에 따르면, 하나 이상의 그러한 그래픽 처리 시스템을 포함하는 가상 현실 사용자 디바이스가 제공된다.

[0040] 본 발명의 제10 양상에 따르면, 적어도 초기 처리 스테이지와 추가 처리 스테이지를 포함하는 그래픽 처리 파이프라인과 관련되는 그래픽 프로세서가 제공되며, 이 그래픽 프로세서는:

[0041] 초기 처리 스테이지에서 제1 해상도로 장면을 위한 데이터를 처리하고;

[0042] 초기 처리 스테이지에서 제2 해상도로 장면을 위한 데이터를 처리하고;

[0043] 초기 처리 스테이지에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리한 후, 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 시선-추적 데이터를 수신하고;

[0044] 수신된 시선-추적 데이터로부터 장면의 적어도 하나의 하위-영역을 식별하고;

[0045] 추가 처리 스테이지에서 제1 해상도로 장면을 위한 데이터를 처리하고;

[0046] 추가 처리 스테이지에서 제2 해상도로 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터만 처리하며;

[0047] 제1 해상도로 추가 처리 스테이지에서 처리되는 장면을 위한 데이터와, 제2 해상도로 추가 처리 스테이지에서 처리되는 장면의 식별된 적어도 하나의 하위-영역에 대응하는 데이터를 결합함으로써 장면을 렌더링하도록 되어 있다.

[0048] 본 발명의 제11 양상에 따르면, 그러한 그래픽 프로세서를 포함하는 가상 현실 사용자 디바이스가 제공된다.

[0049] 추가 특성과 장점이, 수반하는 도면을 참조하여 이루어지며 예를 들어서만 제공되는 다음의 상세한 설명으로부터 분명하게 될 것이다.

도면의 간단한 설명

[0050] 도 1은, 본 발명의 실시예에 따른 그래픽 처리 시스템의 예를 개략적으로 도시한다.

도 2는, 본 발명의 실시예에 따른 그래픽 처리 파이프라인의 예를 개략적으로 도시한다.

도 3은, 본 발명의 실시예에 따른 그래픽 처리 파이프라인의 다른 예를 개략적으로 도시한다.

도 4는, 본 발명의 실시예에 따른 프레임 파이프라이닝의 예를 개략적으로 도시한다.

도 5는, 본 발명의 실시예에 따른 그래픽 처리 파이프라인의 다른 예를 개략적으로 도시한다.

도 6은, 본 발명의 실시예에 따른 그래픽 처리 파이프라인의 다른 예를 개략적으로 도시한다.

도 7은, 본 발명의 실시예에 따른 그래픽 처리 파이프라인의 다른 예를 개략적으로 도시한다.

도 8은 그래픽 처리 파이프라인의 다른 예를 개략적으로 도시한다.

발명을 실시하기 위한 구체적인 내용

[0051] 본 발명의 실시예에 따른 다수의 예를 이제 디스플레이용 컴퓨터 그래픽 처리 환경에서 기재할 것이다.

[0052] 도 1을 참조하면, 그래픽 처리 시스템(100)의 예를 개략적으로 도시한다.

[0053] 이 예에서, 호스트 프로세서(110)에서 수행하는 게임과 같은 애플리케이션(105)이 관련 GPU(115)에 의해 수행될 그래픽 처리 동작을 요청한다. 이를 행하기 위해, 애플리케이션(105)은, GPU(115)를 위해 드라이버(120)에 의해 해석되는 애플케이션 프로그래밍 인터페이스(API) 호출을 생성한다. 드라이버(120)는 호스트 프로세서(110) 상에서 실행한다. 드라이버(120)는 GPU(115)로의 적절한 명령을 생성하여 애플리케이션(105)에 의해 요청되는 그래픽 출력을 생성한다. 명령 세트가 애플리케이션(105)으로부터의 명령에 응답하여 GPU(115)에 제공된다. 명령은 디스플레이 상에 디스플레이될 프레임을 생성하는 것일 수 있다.

[0054] 도 2를 참조하면, 그래픽 처리 파이프라인(200)의 예를 개략적으로 도시한다. 그래픽 처리 파이프라인(200)은, GPU에 의해 실행될 수 있는 액션의 시퀀스를 지시한다.

[0055] 이 예에서, GPU는 타일-기반 렌더러(renderer)이다. GPU는 그러므로 생성될 렌더 출력 데이터 어레이의 타일을 발생시킨다. 렌더 출력 데이터 어레이는 출력 프레임일 수 있다. 타일-기반 렌더링은 즉시 모드 렌더링과, 전체 렌더 출력이 1회의 고우(go)로 처리되기 보다는, 렌더 출력이 다수의 더 작은 하위-영역(또는 "구역")으로 나뉜다는 점에서 상이하다. 이들 하위-영역은 본 명세서에서 타일로 지칭한다. 각각의 타일은 별도로 렌더링된다. 예컨대, 각각의 타일은 차례로 렌더링될 수 있다. 렌더링된 타일은 그 후 재결합되어 디스플레이를 위한 전체 렌더 출력을 제공한다. 타일-기반 렌더링에서, 렌더 출력은 정기적인 크기와 형상의 타일로 나눌 수 있다. 타일은 정사각형 또는 다른 형상일 수 있다.

[0056] "타일링" 및 "타일-기반" 렌더링에 사용될 수 있는 다른 용어는 "청킹(chunking)" - 여기서 렌더링 타일은 "청크"로 지칭함 - 및 "버킷" 렌더링을 포함한다. 용어, "타일" 및 "타일링"은 이후 편의상 사용될 것이지만, 이들 용어는 모든 대안적이며 등가적인 용어 및 기술을 포함하고자 함이 이해되어야 한다.

[0057] 렌더 출력 데이터 어레이는 스크린이나 프린터와 같은 디스플레이 디바이스 상에 디스플레이되고자 하는 출력 프레임일 수 있다. 렌더 출력은 또한 예컨대 추후 렌더링 패스 시 사용하고자 하는 중간 데이터를 포함할 수 있다. 이것의 예는 렌더 투 텍스쳐" 출력이다.

[0058] 컴퓨터 그래픽 이미지가 디스플레이될 때, 이것은 예컨대 일련의 프리미티브(primitive)와 같은 지아메트리(geometry)의 세트로서 먼저 규정될 수 있다. 프리미티브의 예로 다각형이 있다. 지아메트리는 그 후 래스터화 공정에서 그래픽 프래그먼트로 나눈다. 그래픽 렌더링이 그 뒤에 온다. 그래픽 렌더링 동작 동안, 렌더러는 각 프래그먼트와 관련된 데이터를 변경할 수 있어서, 프래그먼트는 정확히 디스플레이될 수 있다. 그러한 데이터의 예는 컬러와 투명도를 포함한다. 프래그먼트가 렌더러를 완전히 횡단하면, 그 관련 데이터 값이 메모리에 저장되어 출력을 준비한다.

[0059] 도 2는, 본 명세서에서 기재한 실시예의 동작에 연관되는 GPU와 관련되는 여러 요소 및 파이프라인 스테이지를 도시한다. 그러나 도 2에 예시하지 않은 그래픽 처리 파이프라인의 다른 요소와 스테이지가 있을 수 있다. 도 2는 단지 개략적이며, 예컨대, 도시한 기능 유닛과 파이프라인 스테이지는 도 2에서 별도의 스테이지로서 개략적으로 도시될지라도 이들은 실제로는 상당한 하드웨어 회로를 공유할 수 있음을 상당히 또한 주목해야 한다. 그래픽 처리 파이프라인(200)의 스테이지, 요소 및 유닛 등 각각은 원하는 대로 구현될 수 있으며 그에 따라 예컨대 적절한 회로 및/또는 처리 로직 등을 포함하여 관련 동작 및 기능을 실행할 것이다.

[0060] 도 2에 도시한 바와 같이, GPU가 수행하는 그래픽 처리 파이프라인(220)은 정점 셰이더(vertex shader)(205), 헐(hull) 셰이더(210), 테셀레이터(tesselator)(215), 도메인 셰이더(220), 지아메트리 셰이더(225), 타일러(230), 래스터화 스테이지(235), 이른 Z(또는 '심도') 및 스텐실 테스트 스테이지(240), 프래그먼트 셰이딩 스테이지(245) 형태의 렌더러, 늦은 Z(또는 '심도') 및 스텐실 테스트 스테이지(250), 혼합 스테이지(255), 타일 버퍼(260) 및 다운샘플링 및 기록 스테이지(265)를 포함한 다수의 스테이지를 포함한다. GPU에 대한 다른 배치가 그러나 가능하다.

[0061] 정점 셰이더(205)는, 생성될 출력에 대해 규정되는 정점과 관련되는 입력 데이터 값을 수신한다. 정점 셰이더(205)는 이들 데이터 값을 처리하여 그래픽 처리 파이프라인(200)의 후속한 스테이지에 의해 사용하기 위한 대응하는, 정점-셰이딩된 출력 데이터 값의 세트를 생성한다.

[0062] 처리될 각각의 프리미티브는 정점 세트에 의해 규정될 수 있고 나타낼 수 있다. 프리미티브에 대한 각각의 정점은 정점과 속성 세트를 관련시키게 될 수 있다. 속성 세트는 정점에 대한 데이터 값 세트이다. 이들 속성은 포지션 데이터 및 기타 비-포지션 데이터(또는 '변수(varying)')를 포함할 수 있다. 비-포지션 데이터는 예컨대 해당 정점에 대한 컬러, 광, 법선 및/또는 텍스쳐 좌표를 규정할 수 있다.

[0063] 정점 세트가 GPU에 의해 생성될 주어진 출력에 대해 규정된다. 출력에 대해 처리될 프리미티브는 정점 세트에서 주어진 정점을 포함한다. 정점 셰이딩 동작은 각각의 정점에 대한 속성을 후속한 그래픽 처리 동작을 위한 원하는 형태로 변환한다. 이것은 예컨대, 그래픽 처리 시스템의 출력이 디스플레이되는 스크린 공간에 이들 정점이 먼저 규정되게 하는 세계 또는 사용자 공간으로부터의 정점 포지션 속성을 변환하는 것을 포함할 수 있다. 이것은 또한 예컨대 렌더링될 이미지의 조명 효과를 고려하기 위해 입력 데이터를 변경하는 것을 포함할 수 있다.

[0064] 헐 셰이더(210)는 패치 제어점의 세트에 관한 동작을 실행하여, 패치 상수로 알려진 추가 데이터를 생성한다.

[0065] 테셀레이션 스테이지(215)는 지아메트리를 세분하여 헐의 고차의 표현을 만든다.

[0066] 도메인 셰이더(220)는, 정점 셰이더(205)에 유사한 방식으로 테셀레이션 스테이지에 의해 출력되는 정점에 관한 동작을 실행한다.

[0067] 지아메트리 셰이더(225)는 삼각형, 점 또는 선과 같은 전체 프리미티브를 처리한다.

[0068] 정점 셰이더(205), 헐 셰이더(210), 테셀레이터(215), 도메인 셰이더(220) 및 지아메트리 셰이더(225)는 변환 및 조명 동작과 같은 프래그먼트 전단 동작과 프리미티브 셋업을 실행하여, GPU에 제공되는 명령과 정점 데이터에 응답하여, 렌더링될 프리미티브를 셋업한다.

[0069] 렌더링될 모든 프리미티브가 적절히 셋업되었으면, 타일러(230)는 그 후, 렌더 출력이 처리 용으로 나눠졌던 각각의 타일에 대해 어떤 프리미티브가 처리될 것인지를 결정한다. 이를 위해, 타일러(230)는 처리될 각 프리미티브의 위치를 타일 포지션과 비교하여, 결정한 바로는 프리미티브가 잠재적으로 그 내부에 속할 수 있는 각각의 타일에 대한 각각의 프리미티브 목록에 프리미티브를 추가한다. 정확한 비닝, 또는 바운딩 박스 비닝 또는 그 사이의 어떤 것과 같은 프리미티브를 정렬하여 타일 목록에 비닝(binning)하기 위한 임의의 적절하며 원하는 기술이 타일링 공정에 사용될 수 있다.

[0070] 렌더링될 프리미티브의 목록(또는 '프리미티브 목록')이 이런 식으로 각각의 렌더링 타일을 준비하였으면, 프리미티브 목록이 사용을 위해 저장된다. 프리미티브 목록으로 인해 시스템은 해당 타일이 렌더링될 때 어떤 프리미티브가 고려되어 렌더링되는지를 식별할 수 있다.

[0071] 타일러(230)가 타일 목록 모두를 준비하였다면, 각각의 타일이 렌더링될 수 있다. 이를 위해, 각각의 타일이, 타일러(230)의 다음에 오는 그래픽 처리 파이프라인 스테이지에 의해 처리된다.

[0072] 주어진 타일이 처리 중일 때, 이 타일에 대해 처리될 각각의 프리미티브가 래스터라이저(rasterizer)(235)에 통과된다. 그래픽 처리 파이프라인(200)의 래스터화 스테이지(235)는 프리미티브를 개별 그래픽 프래그먼트로 래스터화하여 처리하도록 동작한다. 이를 위해, 래스터라이저(235)는 프리미티브를 샘플링점에 래스터화하여 적절한 포지션을 갖는 그래픽 프래그먼트를 생성하여 프리미티브를 렌더링한다. 래스터라이저(235)에 의해 생성되는 프래그먼트는 그 후 파이프라인(200)의 나머지에 계속 전달되어 처리한다.

[0073] 이른 Z 및 스텐실 테스트 스테이지(240)는 래스터라이저(235)로부터 수신한 프래그먼트에 관한 Z(또는 '심도') 테스트를 실행하여, 임의의 프래그먼트가 이 스테이지에서 폐기될 수 있는지(또는 '도태(cull)될 수 있는지')를 알아본다. 이를 위해, 이른 Z 및 스텐실 테스트 스테이지(240)는 래스터라이저(235)에 의해 발행되는 프래그먼트의 심도 값을 이미 렌더링되었던 프래그먼트의 심도 값과 비교한다. 이미 렌더링되었던 프래그먼트의 심도 값이, 타일 버퍼(260)의 일부분인 심도 버퍼에 저장된다. 이른 Z 및 스텐실 테스트 스테이지(240)에 의해 실행되는 비교는, 새로운 프래그먼트가 이미 렌더링되었던 프래그먼트에 의해 가로막힐 것인지의 여부를 결정하는 것이다. 동시에, 이른 스텐실 테스트가 실행된다. 프래그먼트 이른 Z 및 스텐실 테스트 스테이지(240)를 통과한 프래그먼트가 프래그먼트 셰이딩 스테이지(245)에 전달된다. 프래그먼트 셰이딩 스테이지(245)는, 이른 Z 및 스텐실 테스트를 통과하는 프래그먼트에 관한 적절한 프래그먼트 처리 동작을 실행하여 적절한 렌더링된 프래그먼트 데이터를 생성한다. 이러한 프래그먼트 처리는, 적절한 프래그먼트 데이터를 생성하기 위해 프래그먼트에 관

한 프래그먼트 셰이더 프로그램을 수행하는 것, 프래그먼트에 텍스쳐를 적용하는 것, 프래그먼트에 포깅(fogging) 또는 다른 동작을 적용하는 것 등과 같은 임의의 적절한 프래그먼트 셰이딩 공정을 포함할 수 있다. 프래그먼트 셰이딩 스테이지(245)는 프로그램 가능한 프래그먼트 셰이더일 수 있다.

[0074] 이때, 특히 셰이딩된 프래그먼트에 관한 파이프라인 심도 테스트의 끝을 실행하여 렌더링된 프래그먼트가 실제로 최종 이미지에서 보게 될 것인지를 결정하는 늦은 프래그먼트 Z 및 스텐실 테스트 스테이지(250)가 있다. 이 심도 테스트는, 타일 버퍼(260)에서 Z-버퍼에 저장되는 프래그먼트의 포지션에 대한 Z-버퍼 값을 사용하여, 새로운 프래그먼트에 대한 프래그먼트 데이터가 이미 렌더링되었던 프래그먼트의 프래그먼트 데이터를 교체해야 하는지를 결정한다. 이것은, Z 버퍼에 저장된 바와 같이, 프래그먼트 셰이딩 스테이지(245)에 의해 발행되는 프래그먼트의 심도 값을 이미 렌더링되었던 프래그먼트의 심도 값과 비교하는 것을 수반할 수 있다. 이 늦은 프래그먼트 심도 및 스텐실 테스트 스테이지(250)는 또한 프래그먼트에 관한 늦은 알파 및/또는 스텐실 테스트를 실행할 수 있다.

[0075] 늦은 프래그먼트 테스트 스테이지(250)를 통과하는 프래그먼트는 그 후, 타일 버퍼(260)에 이미 저장된 프래그먼트와의 임의의 혼합 동작을 혼합기(255)에서 받게 될 수 있다. (미도시된) 디더(dither) 등과 같은 프래그먼트에 필요한 임의의 다른 남은 동작이 또한 이 스테이지에서 실행된다.

[0076] 마지막으로, 출력 프래그먼트 데이터(또는 '값')가 타일 버퍼(260)에 기록된다. 출력 프래그먼트 데이터가 그 후 디스플레이하기 위해 프레임버퍼(270)에 출력될 수 있다. 출력 프래그먼트에 대한 심도 값이 또한 타일 버퍼(260) 내에 Z-버퍼에 적절히 기록된다. 타일 버퍼(260)는, 버퍼가 나타내는 각각의 샘플링점에 대해 적절한 컬러 등 또는 Z-값을 저장하는 컬러 및 심도 버퍼를 저장한다. 이들 버퍼는, 전체 렌더 출력의 각각의 픽셀에 대응하는 버퍼에서의 샘플 값의 각각의 세트를 갖는 전체 렌더 출력의 부분, 이 예에서는 타일을 나타내는 프래그먼트 데이터의 어레이를 저장한다. 예컨대, 샘플 값의 각각의 2×2 세트는 출력 픽셀에 대응할 수 있으며, 여기서 4×멀티샘플링이 사용된다.

[0077] 타일 버퍼(260)는, 그래픽 처리 파이프라인(200)의 일부인 랜덤 액세스 메모리(RAM)의 부분으로서 제공된다. 다시 말해, 타일 버퍼(260)가 온-칩 메모리에 제공된다.

[0078] 타일 버퍼(260)로부터의 데이터는 다운샘플링 기록 유닛(265)에 입력된 후, 디스플레이 디바이스(미도시)의 프레임버퍼(270)와 같은 외부 메모리 출력 버퍼에 출력(또는 '다시 기록')된다. 디스플레이 디바이스는 예컨대 컴퓨터 모니터 또는 프린터와 같은 픽셀 어레이를 포함하는 디스플레이를 포함할 수 있었다.

[0079] 다운샘플링 및 기록 유닛(265)은 타일 버퍼(260)에 저장되는 프래그먼트 데이터를 출력 버퍼와 디바이스에 대한 적절한 해상도로 다운샘플링하여, 출력 디바이스의 픽셀에 대응하는 픽셀 데이터의 어레이가 생성된다. 이것은 결국 출력 버퍼(270)에 출력하기 위한 픽셀 형태의 출력 값이 된다.

[0080] 렌더 출력의 타일이 처리되었으며 그 데이터가 예컨대 메인 메모리의 프레임 버퍼(270)와 같이 저장을 위한 메인 메모리에 내보내지면, 충분한 타일이 처리되어 전체 렌더 출력을 생성할 때까지 그 다음 타일이 그 후 처리되는 식이 된다. 이 공정은 그 후 그 다음 렌더 출력 등에 대해 반복된다.

[0081] 도 2에서 알 수 있는 바와 같이, 그래픽 처리 파이프라인(200)은 다수의 프로그램 가능한 처리 또는 "셰이더" 스테이지, 즉 정점 셰이더(205), 헐 셰이더(210), 도메인 셰이더(220), 지아메트리 셰이더(225), 및 프래그먼트 셰이더(245)를 포함한다. 이들 프로그램 가능한 셰이더 스테이지가, 하나 이상의 입력 변수를 가지며 출력 변수의 세트를 생성하는 각각의 셰이더 프로그램을 수행한다. 해당 셰이더 프로그램은 예컨대 정점 셰이더(205)의 경우에 각각의 정점에 대해 처리될 각각의 작업 아이템에 대해 수행될 수 있다. 수행 스레드(execution thread)가 처리될 각각의 작업 아이템에 대해 발행될 수 있으며, 스레드는 그 후 셰이더 프로그램에서 명령을 수행하여 원하며 셰이딩된 출력 데이터를 발생시킨다.

[0082] 도 1을 참조하여 앞서 기재한 애플리케이션(105)과 같은 애플리케이션은 OpenGL® 셰이딩 언어(GLSL), 고-레벨 셰이딩 언어(HLSL), 개방 컴퓨팅 언어(Open CL) 등과 같은 고-레벨 셰이더 프로그래밍 언어를 사용하여 수행될 셰이더 프로그램을 제공한다. 이들 셰이더 프로그램은 그 후 셰이더 언어 컴파일러에 의해 타겟 그래픽 처리 파이프라인(200)을 위한 이진 코드로 번역된다. 이것은 컴파일러 내에 프로그램의 하나 이상의 내부 중간 표현을 만드는 것을 포함할 수 있다. 컴파일러는 예컨대 드라이버(120)의 부분일 수 있으며, 컴파일러가 실행되게 하는 특별 API 호출이 있다. 컴파일러 수행은 그에 따라 애플리케이션(105)에 의해 생성되는 API 호출에 응답하여 드라이버에 의해 행해지는 드로우(draw) 호출 준비의 일부분인 것으로 볼 수 있다.

[0083] 이제 포비티드 렌더링으로 돌아가서, 눈-추적으로도 알려져 있는 시선-추적은 포비티드 렌더링을 포함하는 VR의

여러 구역에 충격을 줄 상당한 잠재성을 갖고 있다.

[0084] 사람의 눈은, 선명한 중앙 시각을 담당하는 눈의 부위인 포비아(fovea)에 그 수용자의 대부분을 갖는다. 포비아는 시각의 전 분야에 비교하여 작다. VR 시스템에 사용되는 것과 같은 니어-아이(near-eye) 디스플레이의 경우, 눈은 임의의 순간에 디스플레이 상의 정보의 서브세트를 단지 인지할 수 있다. 시선-추적된 포비티드 렌더링은, 포비아가 볼 수 있는 디스플레이 구역에서 전체 계산을 유지하도록 렌더링을 조정하여 다른 곳에서의 계산을 감소시킨다.

[0085] 도 3으로 돌아가, 그래픽 처리 파이프라인(300)의 예를 개략적으로 도시한다. 그래픽 처리 파이프라인(300)은 그래픽 처리 시스템에 포함된다. 그래픽 처리 파이프라인(300)은 적어도 초기 처리 스테이지(305)와 추가 처리 스테이지(310)를 포함한다. 그래픽 처리 파이프라인(300)은 시선-추적된 포비티드 렌더링을 실행하는데 사용될 수 있다.

[0086] 시선-추적 데이터(315)가 초기 처리 스테이지(305)나 추가 처리 스테이지(310)에서의 임의의 데이터 처리 전에 수신된다. 시선-추적 데이터(315)는 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 포지셔닝에 관한 것이다. 장면의 적어도 하나의 하위 영역이 수신된 시선-추적 데이터로부터 식별된다. 장면의 식별된 적어도 하나의 하위 영역은 예컨대 사용자가 보고 있는 장면의 구역 또는 구역들에 대응할 수 있다. 초기 처리 스테이지(305) 및/또는 추가 처리 스테이지(310)에서 제1 해상도로 장면을 위한 데이터가 처리된다. 그러나 초기 처리 스테이지(305) 및/또는 추가 처리 스테이지(310)에서 제2 해상도로 장면의 식별된 적어도 하나의 하위 영역에 대응하는 데이터만이 처리된다. 제1 해상도는 제2 해상도와 비교하여 상대적으로 높을 수 있다. 장면은 그 후 추가 처리 스테이지(310)에 의해 출력되는 데이터를 결합함으로써 렌더링될 수 있다. 시선-추적된 포비티드 렌더링은 그러므로 제1 해상도가 제2 해상도보다 낮은 곳에 제공될 수 있으며, 이는 시선-추적 데이터(315)는 그래픽 파이프라인(300)에서의 처리에 영향을 미치는데 사용되기 때문이다.

[0087] 그러한 예가 시선-추적된 포비티드 렌더링을 제공하지만, 시선-추적된 포비티드 렌더링의 성공은 여러 요인에 의존한다. 그러한 요인 중 하나로 눈-추적 데이터(315)의 품질이 있다. 그러한 요인 중 다른 하나로, 눈-추적기 디바이스에 의해 획득한 눈-추적 정보가 렌더링 시스템에 통과되어 그에 의해 사용될 때 갖는 응답 레이턴시이다. 눈-추적 데이터가 덜 정확할수록, 레이턴시가 결과를 덜 통용되게 하는 결과로 및/또는 저-품질 추적 데이터로 인해, 포비티드 렌더링의 결과는 덜 성공적이게 된다.

[0088] 레이턴시 면에서, 일부 예에서, GPU는, 예컨대 지아메트리가 초기 프레임에 제출되는지 또는 지아메트리가 그 다음 프레임의 시작까지 연기되는지에 따라 2-프레임 또는 3-프레임 파이프라인을 갖는다. GPU는 보통은 연기 모드로 동작할 수 있으며, 이 경우, 파이프라인은 3개의 프레임일 것이다.

[0089] 도 4를 참조하면, 프레임 파이프라이닝(400)의 예를 개략적으로 도시한다. 이 예에서, GPU는, 중앙처리장치(CPU) 버퍼링(405)이 프레임 0에서 실행되고, 지아메트리 페이즈(phases)(410)가 프레임 1에서 실행되며, 픽셀 페이즈(415)가 프레임 2에서 실행되는 3-프레임 파이프라인을 갖는다. 60Hz 디스플레이 율 시에, 이것은 새로운 데이터가 그래픽 API를 통해 그래픽 파이프라인에 제공되는 시간과 예컨대 디스플레이에 출력 대기중인 결과 사이에 50ms 레이턴시를 제공한다. 진정한 레이턴시는 이것보다 상당히 클 수 있었으며, 이는 이러한 레이턴시가 이미지 센서 캡쳐, 처리 및 트랜스포트를 포함하기 때문이다.

[0090] VR 시스템에 대해 20ms 모션-대-광자 레이턴시의 타겟 인지 스레시홀드가 제공된다면, 50ms를 초과하는 레이턴시는, 사용자 경험에 관한 한, 너무 높을 가능성이 있다.

[0091] 다시 포비티드 렌더링으로 돌아가서, VR의 환경에서 렌더링 내용은, 각각의 눈에 대해 적어도 하나의 뷰로, 장면을 복수의 뷰로 드로잉하는 것을 수반한다. 확장의 OVR_멀티뷰 군(이후 '멀티뷰')이 임베디드 시스템용 OpenGL®(OpenGL®ES)에 사용될 수 있어서 장면의 복수의 뷰를 발생시킬 수 있다. 원칙적으로, 멀티뷰는, 4개의 뷰, 즉 고 해상도 왼쪽 눈 뷰, 고 해상도 오른쪽 눈 뷰, 저 해상도 왼쪽 눈 뷰 및 저 해상도 오른쪽 눈 뷰가 사용되는 포비티드 렌더링을 취급하는데 사용될 수 있었다. 예컨대, 멀티뷰는 4개의 뷰; 각각의 눈에 대해 내부 및 외부 뷰를 동시에 렌더링하는데 사용될 수 있다.

[0092] 그러나 현재의 버전의 멀티뷰에서의 모든 뷰는 서로 동일한 해상도를 가져야 한다. 이것은, 예컨대 비-시선-추적된 구현을 위해, 단일 값이 내부 영역 및 외부 영역 모두에서 구해질 수 있을 때 허용될 수 도 있다. 그러나 현재의 버전의 멀티뷰는 시선-추적된 구현에 사용하기 위해 타이트하게 초점이 맞춰지는 포비티드 영역의 사용을 허용하지 않는다.

[0093] 또한, 현재 버전의 멀티뷰는 최대 4개의 뷰만을 허용한다. 새로운 멀티뷰 확장을 갖고 사용될 수 있는 다수의

뷰를 확장할 수 있지만, 추가 뷰도 단일-해상도 규칙에 따라야 할 가능성이 있다.

[0094] 게다가, 초점 포지션과 관련되는 구성을 포함한, 각각의 뷰의 구성은 그래픽 파이프라인의 시작에서 애플리케이션에 의해 제공된다. 구성은 그러므로 적어도 그래픽 파이프라인만큼 긴 레이턴시를 가질 것이다. 앞서 기재한 예에서, 뷰어의 눈 위치를 추적하는 것과 관련된 그래픽 출력을 갖는 것 사이의 레이턴시는 그러므로 항상 적어도 50ms일 것이다. 그러한 레이턴시를 고려해 두고, 그래픽 파이프라인은, 출력의 고 해상도 영역의 크기를 결정할 때 보수적인 접근을 취할 수 있다. 예컨대, 그래픽 파이프라인은, 그래픽 파이프라인의 시작 시에 뷰어의 초점 포지션에 중심을 둔 상대적으로 큰 영역을, 뷰어가 렌더링이 완료될 때 이 상대적으로 큰 영역의 어딘가를 보고 있을 가능성이 있으며 그에 따라 출력의 고 해상도 부분을 보고 있을 가능성이 있다는 기대 속에서, 사용할 수 있다. 그러나 그러한 보수적인 접근은, 뷰어가 출력이 디스플레이될 때 보고 있게 될 곳 주위의 불확실성으로 인해 고 해상도로 장면의 상대적으로 큰 부분을 렌더링하는 것을 수반한다.

[0095] 멀티뷰와 호환될 수 있지만, 멀티뷰를 확장하여 저 레이턴시를 갖고 변화할 수 있는 포비아 영역(fovea region)에 렌더링을 허용하고 및/또는 포비아 영역이 멀티뷰 해상도에 해상도가 상이하게 허용하는 조치를 제공하는 것이 바람직할 수 있었다.

[0096] 장면의 동일한 뷰를 나타내는 다수의 이미지를 그러나 상이한 해상도로 렌더링함으로써, 그래픽 프로세서가 포비티드 렌더링을 실행하는데 사용되는 예를 이제 기재할 것이다. 상이한 해상도 뷰는 이후 결합되어, 해상도가 이미지에 걸쳐서 변하는 출력 포비티드 이미지를 제공한다. 이들 예에서, 앞서 언급한 파이프라인 레이턴시 및 상이한 해상도로의 렌더링의 간접비용도 감소할 수 있다.

[0097] 이들 예에서, 눈-추적 데이터는 그래픽 파이프라인의 시작 이후 예컨대 그래픽 파이프라인의 중간에 소비된다. 이처럼, 더 이른(즉, 더 최근에 수신되거나 획득된) 눈-추적 데이터가, 그래픽 파이프라인의 시작 시에 사용되는 눈-추적 데이터와 비교하여, 사용될 수 있다. 렌더링 파이프라인에서 눈-추적 데이터를 소비하는 레이턴시를 감소시킴으로써, 포비티드 렌더링의 장점은 증가할 수 도 있다. 더 나아가, 본 명세서에서 기재한 방식으로 그러한 조치를 구현하는 것은 기존의 아키텍처와 쉽게 통합될 수 있어서, 특히 효율적인 구현을 얻을 수 있다.

[0098] 도 5를 참조하면, 그래픽 처리 파이프라인(500)의 예를 개략적으로 도시한다.

[0099] 그래픽 처리 파이프라인(500)은 그래픽 처리 시스템에 포함된다. 그래픽 처리 파이프라인(500)은 적어도 초기 처리 스테이지(505)와 추가 처리 스테이지(510)를 포함한다. 추가 처리 스테이지(510)를 위한 처리는 초기 처리 스테이지(505)를 위한 처리보다 더 큰 계산 부담을 가질 수 있다.

[0100] 일부 실시예에서, 추가 처리 스테이지(510)는 예컨대 래스터화 및/또는 프래그먼트 셰이딩과 같은 하나 이상의 픽셀-처리 동작을 포함하며, 초기 처리 스테이지(505)는 예컨대 정점 셰이딩과 같은 하나 이상의 기하학적 동작을 포함한다. 일부 예에서, 추가 처리 스테이지(510)는 예컨대 래스터화 및/또는 프래그먼트 셰이딩과 같은 하나 이상의 픽셀-처리 동작을 포함하며, 초기 처리 스테이지(505)는 예컨대 CPU 버퍼링과 같은 하나 이상의 버퍼링 동작을 포함한다. 일부 예에서, 추가 처리 스테이지(510)는 예컨대 프래그먼트 셰이딩과 같은 하나 이상의 지아메트리 동작을 포함하며, 초기 처리 스테이지(505)는 예컨대 CPU 버퍼링과 같은 하나 이상의 데이터 버퍼링 동작을 포함한다.

[0101] 초기 처리 스테이지(505)에서 제1 해상도로 장면을 위한 데이터가 처리되며 제2 해상도로 장면을 위한 데이터가 처리된다. 초기 처리 스테이지(505)에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리한 후, 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 시선-추적 데이터(515)가 수신된다. 장면의 적어도 하나의 하위 영역이 수신된 시선-추적 데이터(515)로부터 식별된다.

[0102] 수신된 시선-추적 데이터(515)는 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 포비아의 현재 고정점에 관한 것일 수 있다. 장면의 식별된 적어도 하나의 하위 영역은, 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 포비아의 현재의 고정점 주위에 중심을 둔 하위 영역에 관한 것일 수 있다.

[0103] 추가 처리 스테이지(510)에서 제1 해상도로 장면을 위한 데이터가 처리된다. 그러나 추가 처리 스테이지(510)에서 제2 해상도로 장면의 식별된 적어도 하나의 하위 영역에 대응하는 데이터만이 처리된다.

[0104] 그래픽 처리 파이프라인(500)의 각각의 스테이지에서 장면을 위한 데이터의 처리는 그래픽 처리 파이프라인(500)의 각각의 스테이지에서 장면을 위한 데이터의 프레임을 처리하는 것을 포함할 수 있다.

[0105] 장면을 위한 데이터는 왼쪽 눈 뷰와 관련된 데이터와 오른쪽 눈 뷰와 관련된 데이터를 포함할 수 있다. 그래픽 처리 파이프라인의 각각의 스테이지에 대한 데이터의 처리는 왼쪽 눈 뷰와 관련되는 처리 데이터와 오른쪽 눈

뷰와 관련되는 처리 데이터를 포함할 수 있다. 수신된 시선-추적 데이터(515)는 가상 현실 사용자 디바이스의 사용자의 왼쪽 눈의 포비아의 현재의 포지셔닝과, 가상 현실 사용자 디바이스의 사용자의 오른쪽 눈의 포비아의 현재의 포지셔닝에 관한 것일 수 있다.

[0106] 장면의 식별된 적어도 하나의 하위 영역은 가상 현실 사용자 디바이스의 사용자의 왼쪽 눈의 포비아의 현재의 포지셔닝 주위에 중심을 둔 왼쪽 눈 하위 영역과, 가상 현실 사용자 디바이스의 사용자의 오른쪽 눈의 포비아의 현재의 포지셔닝 주위에 중심을 둔 오른쪽 눈 하위 영역에 관한 것일 수 있다.

[0107] 장면은, 제1 해상도로 추가 처리 스테이지(510)에서 처리되는 장면을 위한 데이터를 제2 해상도로 추가 처리 스테이지(510)에서 처리되는 장면의 식별된 적어도 하나의 하위 영역에 대응하는 데이터를 결합함으로써 렌더링된다.

[0108] 장면을 렌더링하는 것은 첫째, 제1 해상도로 추가 처리 스테이지(510)에서 처리되는 장면을 위한 왼쪽 눈 데이터와 제2 해상도로 추가 처리 스테이지(510)에서 처리되는 장면의 식별된 왼쪽 눈 하위 영역에 대응하는 데이터를, 둘째 제1 해상도로 추가 처리 스테이지(510)에서 처리되는 장면을 위한 오른쪽 눈 데이터와 제2 해상도로 추가 처리 스테이지(510)에 처리되는 장면의 식별된 오른쪽 눈 하위 영역에 대응하는 데이터와 함께 결합하는 것을 포함할 수 있다. 식별된 왼쪽 눈 하위 영역의 적어도 일부분은 식별된 오른쪽 눈 하위 영역의 적어도 일부분과 상이할 수 있다.

[0109] 일부 예에서, 초기 처리 스테이지(505)에서 제3 해상도로 장면을 위한 데이터가 처리된다. 시선-추적 데이터(520)는 초기 처리 스테이지(505)에서 제3 해상도로 장면을 위한 데이터를 처리한 후 수신되며, 초기 처리 스테이지(505)에서 제3 해상도로 장면을 위한 데이터를 처리한 후 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 것이다. 추가 처리 스테이지(510)에서 제3 해상도로 장면을 위한 데이터가 처리된다. 그러나 추가 처리 스테이지(510)에서 제3 해상도로 장면의 추가 하위 영역에 대응하는 데이터만이 처리된다. 장면을 렌더링하는 것은 제3 해상도로 추가 처리 스테이지(510)에서 처리되는 장면을 위한 데이터를 결합하는 것을 더 포함한다. 장면의 추가 하위 영역은 수신된 시선-추적 데이터(515)로부터 식별할 수 도 있다. 장면의 추가 하위 영역은 장면의 식별된 적어도 하나의 하위 영역을 기초로 계산할 수 있다.

[0110] 도 6을 참조하면, 그래픽 처리 파이프라인(600)의 예를 개략적으로 도시한다. 그래픽 처리 파이프라인(600)은 그래픽 처리 시스템에 포함된다. 그래픽 처리 파이프라인(600)은 초기 처리 스테이지(605)와 추가 처리 스테이지(610)를 포함한다.

[0111] 그래픽 처리 파이프라인(600)은, 초기 처리 스테이지(605)와 추가 처리 스테이지(610)를 포함한다는 점에서 앞서 기재한 그래픽 처리 파이프라인(500)과 유사하다.

[0112] 그러나 이 예에서, 그래픽 처리 파이프라인(600)은 초기 처리 스테이지(605)와 추가 처리 스테이지(610) 중간에 중간 처리 스테이지(620)를 또한 포함한다.

[0113] 추가 처리 스테이지(610)를 위한 처리는 중간 처리 스테이지(620)를 위한 처리보다 더 큰 계산 부담을 가질 수 있다. 예컨대, 추가 처리 스테이지(610)는 예컨대 래스터화 및/또는 프래그먼트 셰이딩과 같은 하나 이상의 픽셀 처리 동작을 포함할 수 있으며, 중간 처리 스테이지(620)는 예컨대 정점 셰이딩과 같은 하나 이상의 지아메트리 동작을 포함할 수 있다.

[0114] 중간 처리 스테이지(620)에서 제1 해상도로 장면을 위한 데이터가 처리되며 제2 해상도로 장면을 위한 데이터가 처리된다. 이 예에서, 시선-추적 데이터(615)는 중간 처리 스테이지(620)에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리한 후 수신된다. 이 예에서, 시선-추적 데이터(615)는, 중간 처리 스테이지(620)에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리한 후 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 것이다. 시선-추적 데이터(615)는 그래픽 처리 파이프라인(600)에서 상대적으로 늦게 수신되며, 그에 따라 뷰어의 현재의 초점 또는 초점들을 정확히 반영할 가능성이 더 있다. 그러나 이것은 그래픽 처리 파이프라인(600)에서 상대적으로 늦게 수신되므로, 그래픽 처리 파이프라인(600)에서 진행된 전체 처리에 덜 영향을 미친다.

[0115] 도 7을 참조하면, 그래픽 처리 파이프라인(700)의 예를 개략적으로 도시한다. 그래픽 처리 파이프라인(700)은 그래픽 처리 시스템에 포함된다. 그래픽 처리 파이프라인(700)은 초기 처리 스테이지(705), 중간 처리 스테이지(720) 및 추가 처리 스테이지(710)를 포함한다.

[0116] 그래픽 처리 파이프라인(700)은, 초기 처리 스테이지(705), 중간 처리 스테이지(720) 및 추가 처리 스테이지

(710)를 포함한다는 점에서 도 6을 참조하여 앞서 기재한 그래픽 처리 파이프라인(600)과 유사하다.

[0117] 그러나 그래픽 처리 파이프라인(600)에서, 시선-추적 데이터(615)가 중간 처리 스테이지(620)에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리한 후 수신되는 반면, 그래픽 처리 파이프라인(700)에서, 시선-추적 데이터(715)는 중간 처리 스테이지(720)에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리하기 전 수신된다. 더 나아가, 이 예에서, 시선-추적 데이터(715)는, 중간 처리 스테이지(720)에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리하기 전 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지션에 관한 것이다. 이처럼, 이 예에서, 중간 처리 스테이지(720)에서 제1 해상도로 장면을 위한 데이터가 처리된다. 그러나 중간 처리 스테이지(720)에서 제2 해상도로 장면의 식별된 적어도 하나의 하위 영역에 대응하는 데이터만이 처리된다. 게다가, 앞서 설명한 바와 같이, 추가 처리 스테이지(710)에서 제1 해상도로 장면을 위한 데이터가 처리되며 제2 해상도로 장면의 식별된 적어도 하나의 하위 영역에 대응하는 데이터만이 처리된다. 시선-추적 데이터(715)는 여전히 그래픽 처리 파이프라인(700)의 바로 시작 후 수신되지만, 앞서 기재한 시선-추적 데이터(615)만큼 그래픽 처리 파이프라인(700)에서 늦지 않다. 시선-추적 데이터(715)는 그러므로 시선-추적 데이터(615)와 비교하여 뷰어의 현재의 초점 또는 초점들을 정확히 반영할 가능성이 덜 있다. 그러나 시선-추적 데이터(715)가 시선-추적 데이터(615)보다 그래픽 처리 파이프라인(700)에서 더 일찍 수신되기 때문에, 그래픽 처리 파이프라인(700)에서 진행된 전체 처리에 더 영향을 미칠 수 있다.

[0118] 도 8을 참조하면, 그래픽 처리 파이프라인(800)의 예를 개략적으로 도시한다. 그래픽 처리 파이프라인(800)은 그래픽 처리 시스템에 포함된다. 그래픽 처리 파이프라인(800)은 초기 처리 스테이지(805), 중간 처리 스테이지(820) 및 추가 처리 스테이지(810)를 포함한다.

[0119] 그래픽 처리 파이프라인(800)은, 초기 처리 스테이지(805), 중간 처리 스테이지(820) 및 추가 처리 스테이지(810)를 포함한다는 점에서 도 6 및 도 7을 참조하여 앞서 기재한 그래픽 처리 파이프라인(600, 700)과 유사하다.

[0120] 이 예에서, 제1 시선-추적 데이터(815)가 중간 처리 스테이지(820)에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리하기 전에 수신된다. 장면의 적어도 하나의 제1 하위 영역은 제1 시선-추적 데이터(815)로부터 식별된다. 이 예에서, 시선-추적 데이터(815)는 중간 처리 스테이지(820)에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리하기 전 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지션에 관한 것이다. 이처럼, 이 예에서, 중간 처리 스테이지(820)에서 제1 해상도로 장면을 위한 데이터가 처리된다. 그러나 중간 처리 스테이지(820)에서 제2 해상도로 장면의 식별된 적어도 하나의 제1 하위 영역에 대응하는 데이터만이 처리된다.

[0121] 이 예에서, 제2 시선-추적 데이터(825)가 중간 처리 스테이지(820)에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리한 후에 수신된다. 장면의 적어도 하나의 제2 하위 영역은 제2 시선-추적 데이터(825)로부터 식별된다. 이 예에서, 제2 시선-추적 데이터(825)는 중간 처리 스테이지(820)에서 제1 및 제2 해상도로 장면을 위한 데이터를 처리한 후 가상 현실 사용자 디바이스의 사용자의 적어도 하나의 눈의 현재의 포지셔닝에 관한 것이다.

[0122] 장면의 적어도 하나의 제2 하위 영역은 장면의 적어도 하나의 제1 하위 영역에 포함될 수 있다. 예컨대, 제1 시선-추적 데이터(815)는, 제1 시선-추적 데이터(815)를 기반으로 하여, 뷰어가 보고 있을 가능성이 있는 장면의 하나 이상의 제1 부분으로 처리를 제약하는데 사용될 수 있다. 이것은 중간 처리 스테이지(820)에서 제2 해상도로 장면 모두의 처리를 절약한다. 제2 시선-추적 데이터(825)는, 제2 시선-추적 데이터(825)를 기반으로 하여, 뷰어가 보고 있을 가능성이 있는 장면의 하나 이상의 제2 부분으로 추가 처리를 제한하는데 사용될 수 있으며, 여기서 장면의 하나 이상의 제2 부분은 장면의 하나 이상의 제1 부분보다 작다. 제2 시선-추적 데이터(825)는, 장면이 출력될 때 사용자가 시청하고 있을 장면의 부분을 지시할 가능성이 더 있다.

[0123] 시스템은 2개 이상의 뷰 세트에 동시에 렌더링하도록 구성될 수 있다. 이것은 혼합된 해상도로의 효율적인 렌더링을 용이하게 할 수 있으며 늦게 래칭된 눈-추적 데이터를 허용할 수 있다. 늦게 래칭된 눈-추적 데이터는 그래픽 파이프라인의 초기 스테이지 후 눈-추적 데이터의 소비에 관한 것이다. 본 명세서에서 기재한 예에서, 눈-추적 데이터의 늦게 래칭됨은, 그래픽 파이프라인이 제출된 후, 실행된다.

[0124] 복수의 해상도로 렌더링하기 위한 예시적 기술을, 먼저 눈-추적 없이, 이제 기재할 것이다. 앞서 기재한 바와 같이, 멀티뷰는 동일한 해상도의 단일 뷰 세트에 렌더링을 가능케 한다. 이것은, 컬러, 심도 및 스텐실과 같은 하나 이상의 부착점에서 GL_TEXTURE_2D_ARRAY로부터 GL_DRAW_FRAMEBUFFER 타겟까지의 계속적인 범위를 바인딩

하는 것으로서 OpenGL® ES API에서 기재된다. GL_DRAW_FRAMEBUFFER 타겟이 또한 여기서 드로우 타겟으로 지칭된다.

[0125] 예컨대, 심도 테스팅에 의한 2개의 뷰로의 렌더링은 폭 × 높이 × num_views의 심도 텍스쳐(GL_TEXTURE_2D_ARRAY)를 만들고, 폭 × 높이 × num_views의 컬러 텍스쳐(GL_TEXTURE_2D_ARRAY)를 만들고, 프레임버퍼 오브젝트를 만들고, 프레임버퍼 드로우 타겟의 심도 부착에 심도 범위를 바인딩하고, 프레임버퍼 드로우 타겟의 컬러 부착에 컬러 범위를 바인딩하며, 그 후 num_views에 대해 출력하도록 구성되는 셰이더로 지아메트리를 드로잉하는 애플리케이션을 수반할 수 있으며, 여기서 이 예에서, num_views = 2이다.

[0126] 이것은, GL_DRAW_ALT_FRAMEBUFFER로 표시되며 draw_alt로서 이후 지칭되는 새로운 프레임버퍼 타겟을 도입함으로써 확장될 수 있다. 이 새로운 타겟 타입은, 상이하지만, 관련된 해상도로 드로우 타겟으로 드로잉된 어떤 것의 결과를 복제하는 역할을 한다. draw_alt 프레임버퍼는 사실상 드로우 프레임버퍼를 미러링한다(mirror). 바인딩 공정은 앞서 지시한 것과 동일할 수 있지만, 간략성을 위해, draw_alt 텍스쳐가 원래의 드로우 구성과 매칭할 수 있다. 예컨대, 해상도 X에서 심도와 컬러를 갖는 2개의 뷰가 드로우 타겟에 바인딩되었다면, draw_alt에 바인딩되는 관련된 해상도(Y(X))에서 심도와 컬러를 갖는 2개의 뷰가 있었을 수 있다. 일부 예에서, 1개보다 많은 'draw_alt' 타겟이 사용된다. 일부 그러한 예에서, 각각의 'draw_alt' 타겟은 상이한 관련된 해상도를 갖는다. 이 경우에, 타겟의 어레이를 사용할 수 도 있다.

[0127] 드로우 타겟 각각은 완전히 상이한 해상도를 갖게 될 수 도 있다. 대안적으로, 타겟은 예컨대 2의 멱수(power-of-two) 스케일링을 허용하는 방식으로 관련된 해상도를 가질 수 있었다. 이로 인해 이하에서 이제 기재될 바와 같이 정점 및 타일링 스테이지를 통해 처리되는 지아메트리는 2개의 세트 사이에 재사용될 수 있다. 이처럼, 앞서 지칭되는 제1 및 제2 해상도는 예컨대 1/4와 같이 2의 멱수를 포함하는 미리 결정된 관계를 가질 수 있다. 두 세트는 그러나 프래그먼트-셰이딩될 수 도 있다.

[0128] GPU는, 포비티드 렌더링을 실행할 때 장면의 동일한 뷰의 다수의 상이한 해상도 버전을 렌더링하도록 제어될 수 있다. GPU는 예컨대 장면의 동일 뷰의 3개의 상이한 해상도 버전, 즉 최고 해상도 뷰, 중간 해상도 뷰 및 최저 해상도 뷰를 렌더링할 수 도 있다. 뷰의 이들 상이한 해상도 이미지는 적절히 결합(또는 '혼합')되어, 예컨대 디스플레이를 위해 출력 포비티드 이미지를 제공한다. 상이한 해상도 이미지 각각은 렌더링되는 전체 장면의 뷰의 상이한 필드(또는 '부분')를 보여줄 수 도 있다. 예로, 고정점과 관련되는 포비티드 이미지의 영역은 최고 해상도를 가지며, 고정점에서 먼 이미지의 하나 이상의 영역은 저 해상도로 디스플레이된다.

[0129] 렌더링을 요청하는 애플리케이션은, 장면을 위한 지아메트리가 포비알 뷰와 관련된 스케일 팩터와 함께 복수의 뷰에 렌더링될 것임을 그래픽 프로세서를 위한 드라이버에게 지시할 수 있다. 애플리케이션으로부터의 그러한 명령에 응답하여, 드라이버는 그 후 GPU에 전송하는 적절한 타일링 및 렌더링 작업을 구성하여, 본 명세서에서 기재한 방식으로 포비티드 렌더링을 실행한다.

[0130] 예로, 장면을 위한 지아메트리가 처리되고 타일링되며, 다시 말해 단 한번 렌더링된 이미지의 각각의 렌더링 타일을 위한 목록으로 정렬된다. 이것은, 각각의 해상도 이미지를 렌더링할 때, 즉 각각의 해상도 이미지를 렌더링하기 위한 프래그먼트 처리를 실행할 때, 그 후 공통적으로 사용되는(또는 '공유되는') 타일 지아메트리 목록의 단일 세트를 제공한다. 특히, GPU는, 출력 이미지에 필요한 최고 해상도로, 렌더링되고 있는 장면에 대해 한 번만 정점 포지션 처리와 지아메트리 목록 생성 공정을 실행하도록 제어될 수 도 있다. 지아메트리 목록이 이런 식으로 준비되면, 뷰 중 일부나 모두를 보여주는 각각의 상이한 해상도 이미지를 렌더링할 때 그러한 지아메트리 목록의 단일 세트가 이때 사용될 수 도 있다.

[0131] 최고 해상도 뷰를 위해, 준비된 지아메트리 목록이 그대로 사용될 수 있어서, 이 최고 해상도 이미지를 렌더링할 때 각각의 렌더링 타일을 위해 처리될 지아메트리를 식별할 수 있다. 다른 저 해상도 이미지를 위해, 최고 해상도로 준비되어진 지아메트리 목록과 그러한 목록에서의 지아메트리는, 최고 해상도 지아메트리 목록에서의 지아메트리를 적절한 스케일링 팩터만큼 스케일링 다운함으로써 저 해상도 이미지에 대해 렌더링될 지아메트리를 식별하여 규정하는데 사용된다. 스케일링 팩터는 해당 저 해상도 이미지의 해상도를 예컨대 선형 스케일링을 사용하여 지아메트리 목록이 준비된 최고 해상도에 관련시킨다. 이 다운스케일링은 정점 포지션 로딩 동작의 부분으로서 행해질 수 있지만, 다른 배치가 원하는 경우 가능할 것이다.

[0132] 일부 예에서, 최종 이미지의 일부지만 전부는 아닌 것이 상이한 각각의 해상도로 렌더링된다. 특히, 최종 구성에서 미사용될 이미지의 영역은 렌더링되지 않을 수도 있다. 각각의 상이한 해상도 이미지의 선택된 부분만의 렌더링은 출력 이미지의 각각의 렌더링 타일에 대해 어떤 해상도 이미지나 이미지들이 그 타일 위치에 대해 발

생되어야 하는지를 지시함으로써 달성할 수 도 있다. 이 정보는 렌더링 타일을 위한 지아메트리 목록과 함께 제공될 수 도 있다. 렌더링 작업을 그래픽 프로세서에 제공할 때 및/또는 렌더를 지시할 때 렌더링될 타일의 적절한 목록을 제공하도록 구성되는 GPU를 위한 드라이버와 같은 다른 배치는 그 내부에서 타일이 대신 사용될 수 있는 이미지에 대해 렌더링되지 않아야 하는 구역을 식별하는 영역을 배제한다.

[0133] 각각의 상이한 해상도 이미지의 적절한 부분이 GPU에 의해 렌더링되었다면, 이들 이미지는 그래픽 처리 시스템의 메모리의 적절한 프레임 버퍼에 저장된다. 이들 이미지는 그 후 적절히 결합되어 디스플레이될 최종 출력 포비티드 뷰를 제공한다.

[0134] 최종, 포비티드 출력 뷰를 제공할 뷰의 상이한 해상도 이미지의 구성은, 각각의 상이한 해상도 뷰를 그래픽 텍스쳐로서 취급한 후 텍스쳐 매핑 동작을 사용하는 GPU에 의해 실행될 수 도 있으며, 이러한 매핑 동작은 디스플레이될 최종, 출력 포비티드 뷰에 걸쳐 상이한 해상도 이미지를 적절히 샘플링하고 결합하도록 프래그먼트 셰이퍼의 일부분으로서 구현할 수 도 있다.

[0135] 그러한 포비티드 렌더링을 실행할 때, 최고 해상도 이미지는, 고정점 또는 고정점들의 포지션에 따라 디스플레이되는 최종 출력 이미지의 중앙에 반드시 있기보다는, 뷰의 어딘 가에 오프셋될 수 있으며, 이러한 고정점 또는 고정점들은 예컨대 시선 추적 센서의 사용을 통해 뷰어가 실제 보고 있는 곳을 식별하도록 결정될 수 도 있다.

[0136] 최고 해상도 세트를 렌더링하고 다운샘플링하여 다른 세트를 발생시키는 것이 동일한 결과를 발생시키지 않았을 것이다.

[0137] 멀티샘플링된 셰이딩 율은 세트 사이에서 변경될 수 있다. 멀티뷰에서, 해상도가 타겟에 바인딩되는 모든 텍스쳐 사이에서 바람직하게는 매칭되어야 하는 것과 같이, 멀티샘플 카운트도 바람직하게는 매칭해야 한다. 해상도, 멀티샘플 카운트, 텍스쳐 포맷 및 레이아웃은 그러나 상이한 세트 사이에서 변경될 수 있었다. 렌더링될 수 있는 텍스쳐를 기재하는 것은 변할 수 있다.

[0138] 앞서 지시한 바와 같이, 본 명세서에서 기재한 예는 늦게 래칭된 눈-추적 데이터를 사용한 포비티드 렌더링에 관한 것이다.

[0139] 일부 예에서, 하드웨어 프래그먼트 작업 또는 작업 아이템, 설명자는, 바운딩 박스에 의해 바운딩되는 영역 내로만 프래그먼트 셰이딩을 제한하는 2D 축-정렬된 바운딩 박스를 포함한다. 이것은 최대 해상도 뷰의 셰이딩을 포비아 영역으로만 제한하기 위한 효율적인 메커니즘이며, 포비아 영역은 바운딩 박스와 관련된다. 바운딩 박스 데이터는 프래그먼트 셰이딩이 개시하기 바로 전에 사용되며, 그래픽 파이프라인에서 그에 따라 늦게 변화할 수 있다. 이것이 의미하는 점은, 사용 중인 시선-추적 데이터의 커런시와 관련하여 최악의 경우에 1 프레임의 레이턴시, 즉 60Hz에서 16.7ms라는 점이다. 이것은 전체 그래픽 파이프라인에 대한 50ms와는 대조적이다. 레이턴시는, 지아메트리가 단일 프레임 내에서 제출되고, 타일링 및 셰이딩된다면 더 감소할 수 도 있다.

[0140] 이전에 제출된 하드웨어 프래그먼트 작업 설명자에 복제된 임의의 '늦은' 바운딩 박스 데이터가 이것을 정확히 읽기 위해 필요할 수 도 있다. 새로운 데이터가 주어진다면, 이것은, 드라이버가 프래그먼트 작업이 아직 시작하지 않았음을 안다면, 드라이버 또는 공유된 버퍼로부터 데이터를 복제하는 GPU 계산 셰이더 중 어느 하나에 의해 행해질 수 있다.

[0141] 파이프라인에서 데이터를 늦게 제공하는 것은 OpenGL®ES에서는 흔한 것은 아니지만, 데이터로의 수동으로 동기화된 액세스는 잘 이해된다. OpenGL®ES에서의 계산 셰이더는 디폴트에 의해 비동기화된다. 사용자가 동기화 프리미티브를 사용하여 정확한 동작을 달성한다. 이것은 적절한 시기에 프래그먼트 설명자를 업데이트할 충분한 하부구조를 제공할 것이다. 구현은 아토믹스 및 공유된 버퍼를 이용할 수 있거나 커스텀 API 진입점을 통해 등가의 기능을 노출할 수 있다.

[0142] Vulkan은 OpenGL®ES보다 더 저 레벨이며, 또한 명시적으로 요청받지 않는다면 임의의 동기화를 실행하지 않으며, 따라서 OpenGL®ES에서의 메커니즘과 유사한 메커니즘이 Vulkan에서 작동한다. Vulkan의 하나의 특성은, 그 VkRenderPass 오브젝트가, 명시된 영역으로 셰이딩을 제한하는 2D 축-정렬된 바운딩 박스를 규정하는 명시적 'renderArea' 속성을 이미 갖는다는 점이다. Vulkan 확장을 개발하는 것이, OpenGL®ES 확장을 개발하는 것보다 더 효율적일 수 있으며, 이는 바운딩 박스 개념이 이미 Vulkan API에 존재하기 때문이다. Vulkan은, 사용자가 다른 것을 '미러링'하는 제2 타겟 세트를 규정하게 하는 vkRenderPass 오브젝트의 확장을 도입할 수 있다. 이것은 앞서 기재한 'GL_DRAW_ALT_FRAMEBUFFER' 기능의 등가 기능을 허용할 수 있다.

[0143] 특정 예에서, 그리고 2개의 뷰를 갖는 멀티뷰를 가정하면, 구현은 최대 해상도로 2개의 뷰에 대한 텍스쳐 어레이 세트(컬러, 심도 등)를 만들어 이들 어레이를 프레임버퍼에서 '드로우' 타겟에 바인딩한다. 이들 텍스쳐 어레이에 대한 프래그먼트 바운딩 박스는 업데이트되어 가장 최근에 보고된 눈-추적 데이터와 매칭된다. 이 시스템은 다른 2개의 뷰에 대해 그러나 최대 해상도 뷰의 해상도의 1/4에서 만들어진 텍스쳐 어레이의 매칭 세트를 만들어, 이들 뷰를 'draw_alt' 타겟에 바인딩한다. 장면은 그 후 드로우 및 draw_alt 타겟을 소비함으로써 드로잉된다. 이처럼, 이 예에서, 제1 해상도와 제2 해상도 사이의 미리 결정된 관계는 1/4이다.

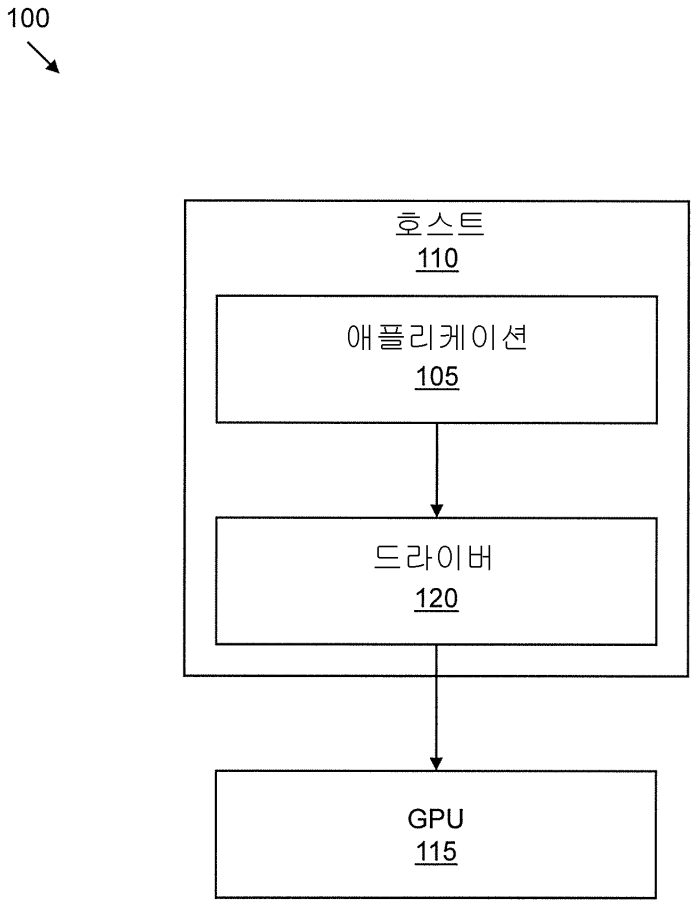
[0144] 이후, 각 프레임의 시작에서나 정상 간격에서 중 어느 하나에서, 모든 관련된 제출되고 여전히 변경 가능한 프래그먼트 작업이 업데이트되어 최근 바운딩 박스를 사용할 것이다. 이를 위한 이상적인 시간은 새로운 눈 추적 데이터가 수신된 후일 것이며, 이것은 주된 렌더링 루프와 비동기적으로 실행될 수 있다.

[0145] 이 특정 예는 작지만 전체 해상도 영역의 쌍 + 스테레오 이미지 모두에 대한 1/4 해상도 출력을 발생시켜서 눈 위치를 매칭할 것이다. 그 후 이들은 중첩되어서 최종 이미지를 생성할 것이다. 이 중첩 공정은, 저-레이턴시 VR 시스템에서 존재하며 다른 용어 중에서 "타입랩(timewarp)", ", "비동기 타입랩", "전자 디스플레이 동기화"로 알려진 것들과 같은 후속하는 렌즈 정정 또는 혼합 공정에 포함시키기에 적절할 것이다. 이것은, 영역들을 추적함으로써 이들의 위치와 크기의 지식을 사용한다. 타입랩 공정은 또한 더 부드러운 전환을 만들기 위해서 또는 예컨대 박스를 구형으로 트리밍하는 것과 같이 최대 해상도를 더 작은 대안 형상으로 트리밍하는 것을 선택하기 위해 2개의 영역 사이에서 혼합할 수 도 있다.

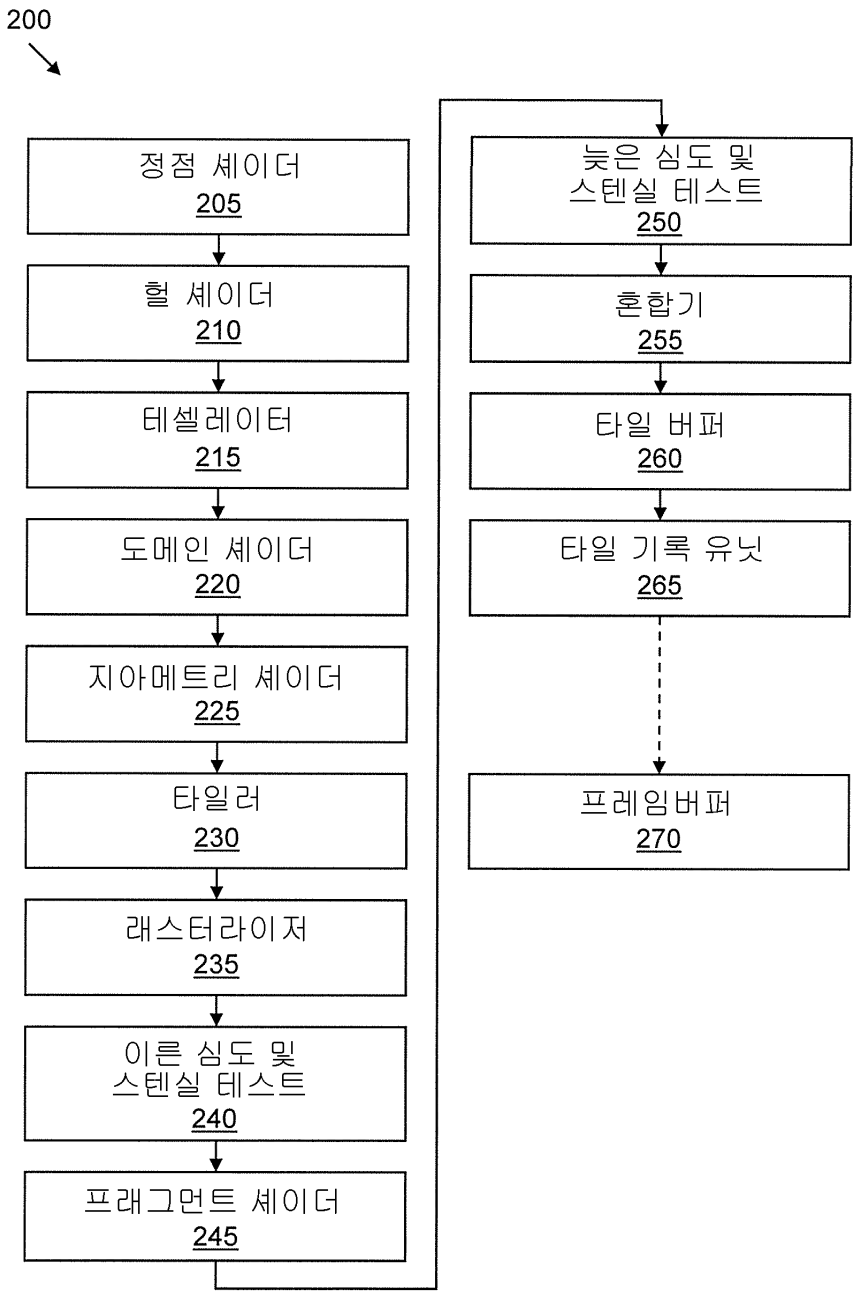
[0146] 앞선 실시예는 예시적인 예로 이해될 것이다. 추가 실시예를 생각해 볼 수 있다. 임의의 하나의 실시예와 관련하여 기재한 임의의 특성은 단독으로 또는 기재한 다른 특성과 조합하여 사용될 수 있으며, 실시예 중 임의의 다른 것 또는 실시예 중 임의의 다른 것의 임의의 조합의 하나 이상의 특성과 조합될 수 있음을 이해해야 한다. 더 나아가, 앞서 기재하지 않은 등가 내용 및 변경도, 수반하는 청구항에 규정되는 본 개시의 실시예의 범위에서 벗어나지 않고 사용될 수 있다.

도면

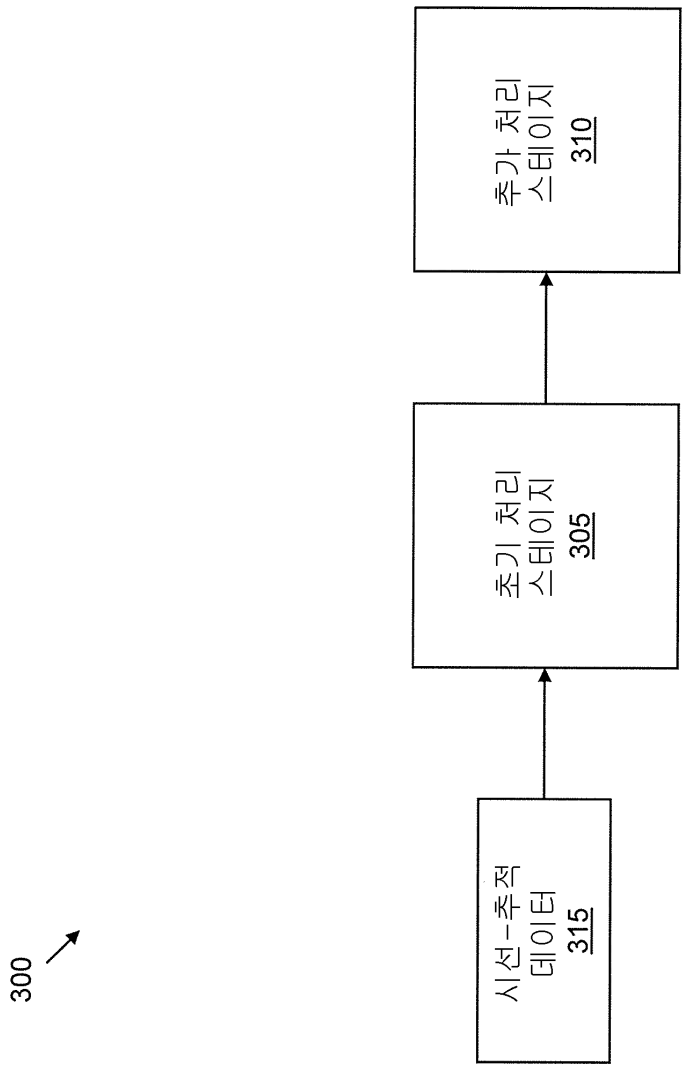
도면1



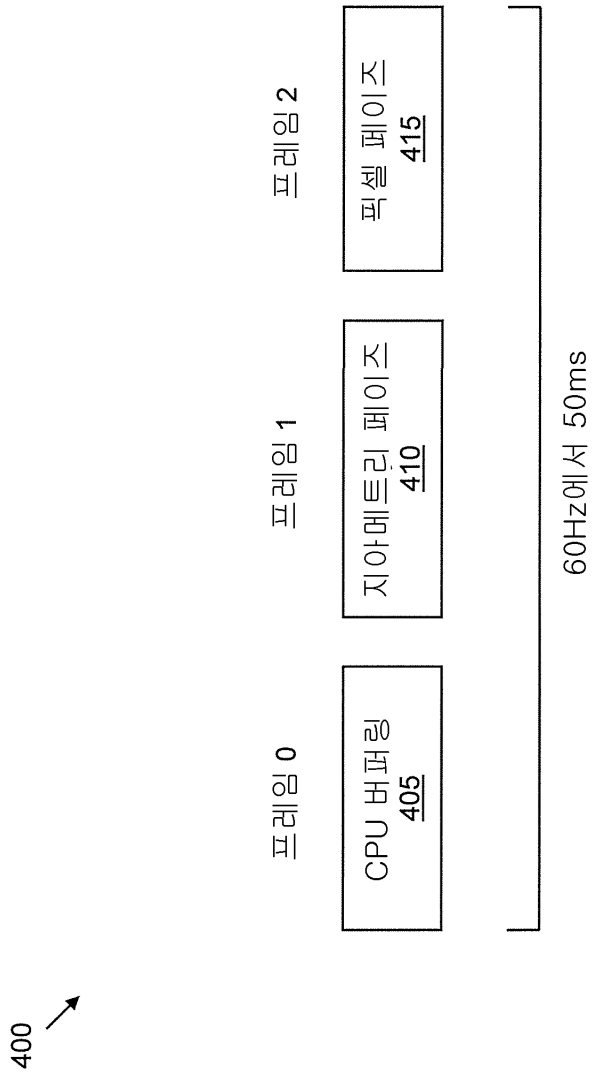
도면2


200
정점 셰이더
205
헐 셰이더
210
테셀레이터
215
도메인 셰이더
220
지아메트리 셰이더
225
타일러
230
래스터라이저
235
이른 심도 및
스텐실 테스트
240
프래그먼트 셰이더
245
늦은 심도 및
스텐실 테스트
250
혼합기
255
타일 버퍼
260
타일 기록 유닛
265
프레임버퍼
270

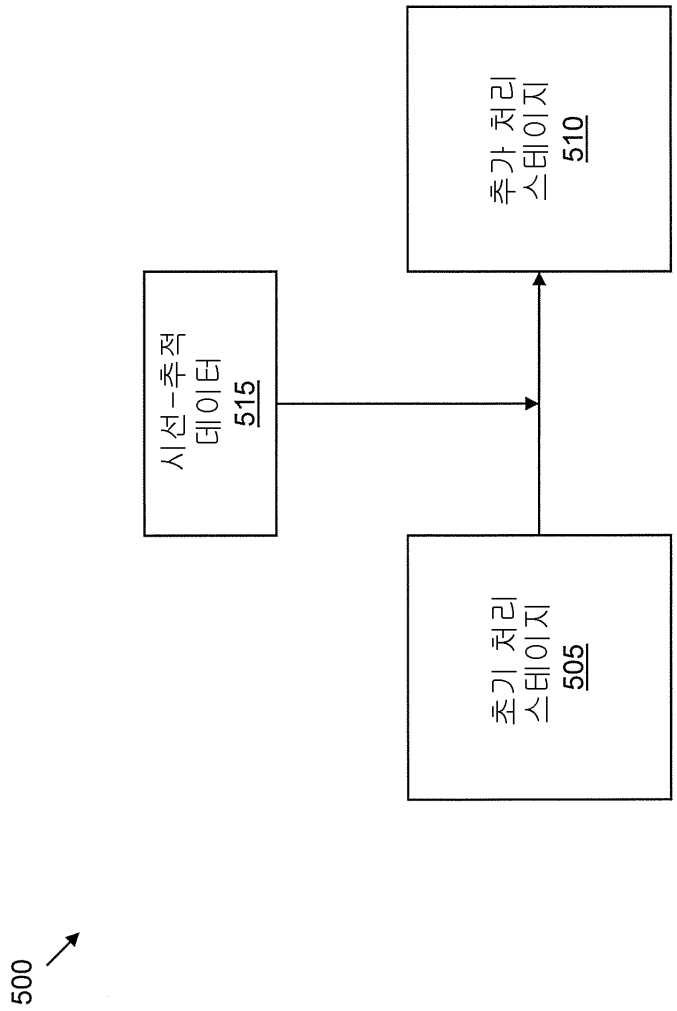
도면3



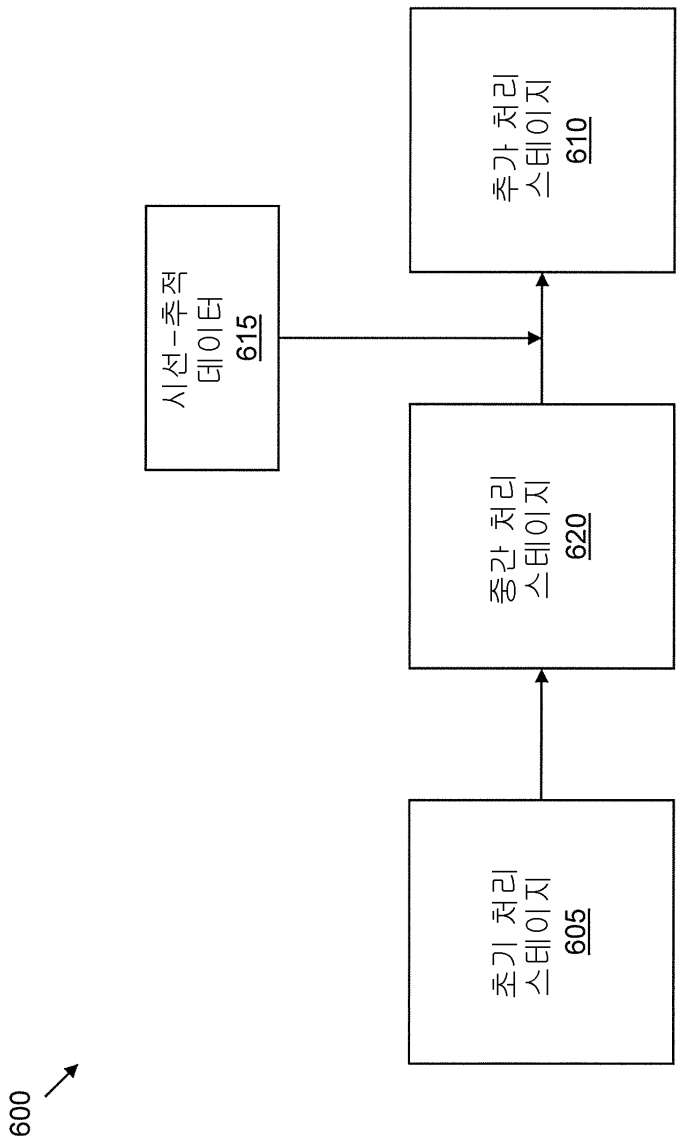
도면4


400
프레임 0
CPU 버퍼링
405
프레임 1
지아메트리 페이즈
410
프레임 2
픽셀 페이즈
415
60Hz에서 50ms

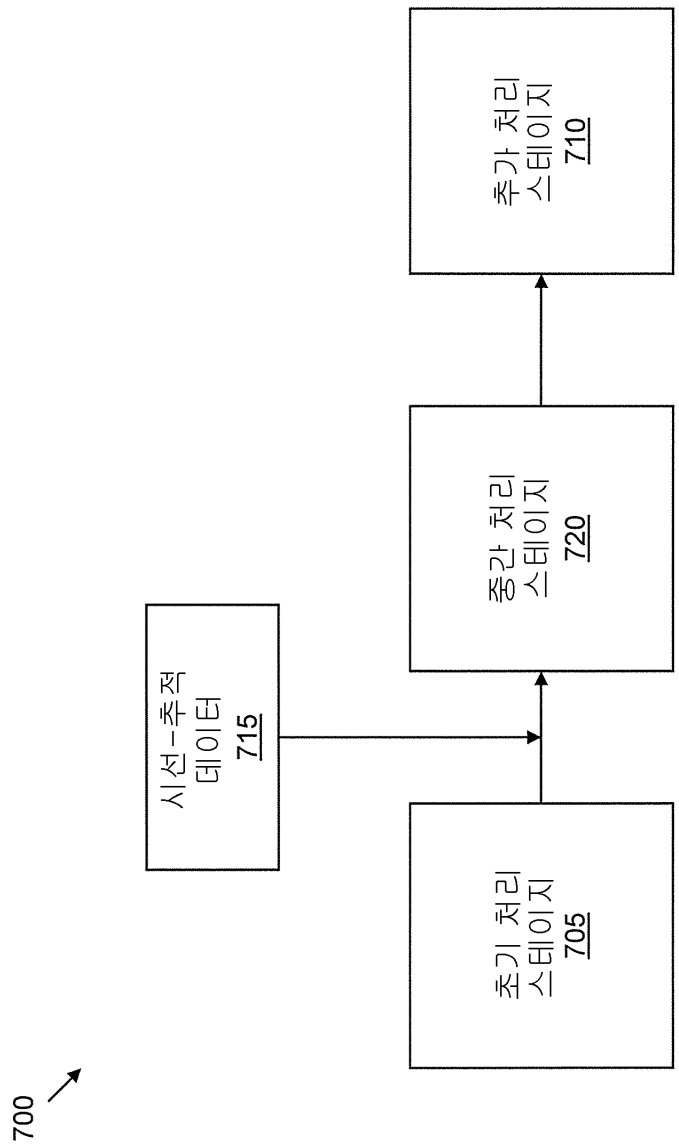
도면5



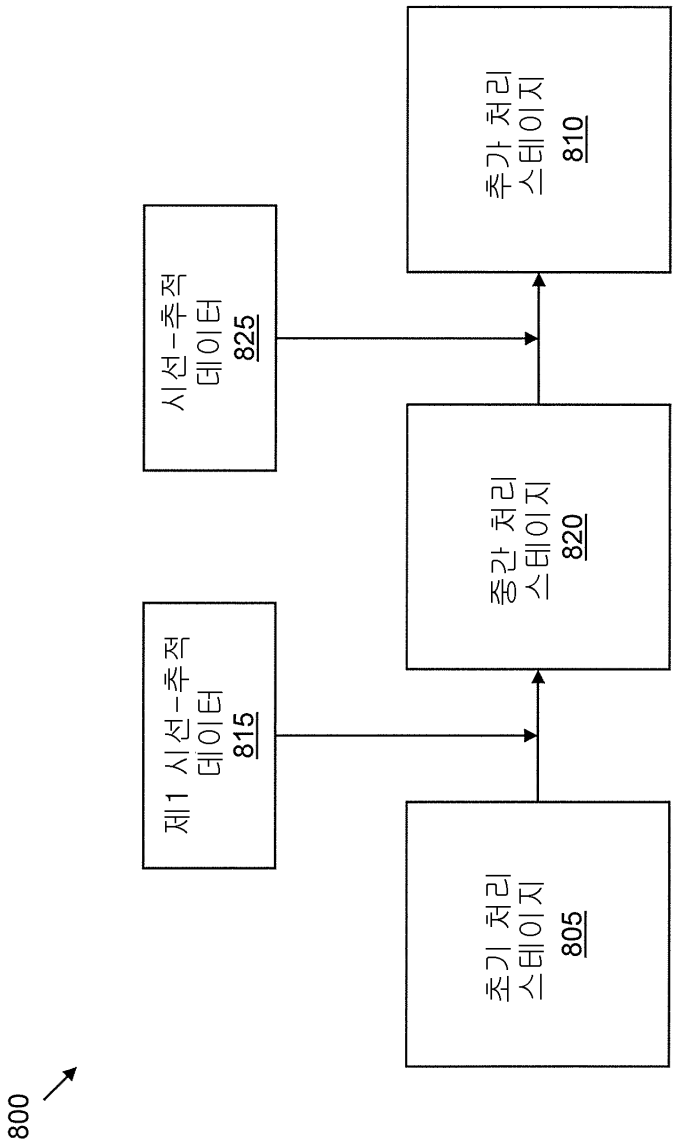
도면6



도면7



도면8


800
초기 처리 스테이지 805
제1 시선-추적 데이터 815
중간 처리 스테이지 820
시선-추적 데이터 825
추가 처리 스테이지 810