(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2020년07월08일
(11) 등록번호 10-2132040
(24) 등록일자 2020년07월02일

(51) 국제특허분류(Int. Cl.)
G06T 15/00 (2006.01)
(21) 출원번호 10-2014-7030816
(22) 출원일자(국제) 2013년03월15일
심사청구일자 2018년02월27일
(85) 번역문제출일자 2014년10월31일
(65) 공개번호 10-2015-0002742
(43) 공개일자 2015년01월07일
(86) 국제출원번호 PCT/US2013/032123
(87) 국제공개번호 WO 2013/151750
국제공개일자 2013년10월10일
(30) 우선권주장
13/830,075 2013년03월14일 미국(US)
(뒷면에 계속)
(56) 선행기술조사문헌
KR1020100036183 A*
KR1020110112828 A*
WO2007049610 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
퀄컴 인코포레이티드
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(72) 발명자
고엘 비니트
미국 92121 캘리포니아주 샌디에고 모어하우스 드라이브 5775
그루버 앤드류 이
미국 92121 캘리포니아주 샌디에고 모어하우스 드라이브 5775
김 동현
미국 92121 캘리포니아주 샌디에고 모어하우스 드라이브 5775
(74) 대리인
특허법인코리아나

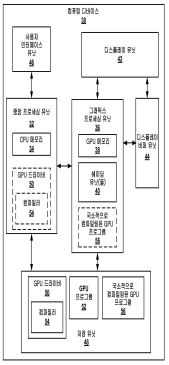
전체 청구항 수 : 총 37 항 심사관 : 이현주

(54) 발명의 명칭 **그래픽스 프로세싱에서의 패치된 쉐이딩**

(57) 요 약

이 개시물의 양태들은, 버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛 (GPU) 의 하드웨어 유닛으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작을 수행하는 것을 포함하는 그래픽스를 렌더링하는 프로세스에 관한 것이며, 여기서 하드웨어 유닛은 단일 버텍스를 입력으로서 수신하고 단일 버텍스를 출력으로서 발생시키는 인터페이스를 고수한다. 프로세스는 또한, 버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 유닛으로, 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 기초하여 하나 이상의 제어 포인트들을 발생시키기 위하여 헐 쉐이딩 동작을 수행하는 것을 포함하며, 여기서 하나 이상의 헐 쉐이딩 동작들은 하나 이상의 제어 포인트들을 출력하기 위하여 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 작용한다.

대 표 도



(30) 우선권주장
61/620,333 2012년04월04일 미국(US)
61/620,340 2012년04월04일 미국(US)
61/620,358 2012년04월04일 미국(US)

명 세 서

청구범위

청구항 1

그래픽스를 렌더링하는 방법으로서,

버텍스 쉐이딩 (vertex shading) 을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 유닛으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 복수의 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작을 수행하는 단계로서, 상기 버텍스 쉐이딩 동작을 수행하는 단계는, 상기 하드웨어 유닛에 의해, 상기 복수의 입력 버텍스들의 각각의 입력 버텍스 각각에 대해 단일 버텍스를 출력하는 단계를 포함하는, 상기 버텍스 쉐이딩 동작을 수행하는 단계; 및

상기 버텍스 쉐이딩을 위해 지정된 상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로, 상기 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 대해 하나 이상의 테셀레이션 동작들을 수행하는 단계로서, 상기 하나 이상의 테셀레이션 동작들을 수행하는 단계는 하나 이상의 제어 포인트들을 출력하기 위하여 상기 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대한 헐 쉐이딩 동작을 수행하는 단계를 포함하는, 상기 하나 이상의 테셀레이션 동작들을 수행하는 단계를 포함하는, 그래픽스를 렌더링하는 방법.

청구항 2

제 1 항에 있어서,

상기 버텍스 쉐이딩 동작을 수행하는 단계 및 상기 헐 쉐이딩 동작을 수행하는 단계는 제 1 렌더링 패스를 수행하는 단계와 연관되고,

제 2 렌더링 패스를 수행하는 단계를 더 포함하며,

상기 제 2 렌더링 패스를 수행하는 단계는,

버텍스 쉐이딩을 위해 지정된 상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로, 상기 제어 포인트들에 적어도 부분적으로 기초하여 버텍스 값들을 발생시키는 것을 포함하는 도메인 쉐이딩 동작을 수행하는 단계; 및

버텍스 쉐이딩을 위해 지정된 상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로, 하나 이상의 새로운 버텍스들을 생성하기 위하여 하나 이상의 도메인 쉐이딩된 버텍스들에 대한 지오메트리 쉐이딩 동작을 수행하는 단계를 포함하며,

상기 지오메트리 쉐이딩 동작은 상기 하나 이상의 새로운 버텍스들을 출력하기 위하여 상기 하나 이상의 도메인 쉐이딩된 버텍스들 중 적어도 하나에 작용하는, 그래픽스를 렌더링하는 방법.

청구항 3

제 2 항에 있어서,

상기 그래픽스 프로세싱 유닛의 하나 이상의 컴포넌트들이 상기 제 1 렌더링 패스와 상기 제 2 렌더링 패스 간에 유휴하도록, 상기 제 2 렌더링 패스를 수행하기 전에 상기 제 1 렌더링 패스를 완료하는 단계를 더 포함하는, 그래픽스를 렌더링하는 방법.

청구항 4

제 2 항에 있어서,

상기 버텍스 쉐이딩 동작, 상기 헐 쉐이딩 동작, 상기 도메인 쉐이딩 동작, 및 상기 지오메트리 쉐이딩 동작은 드로우 콜 (draw call) 과 연관되고,

상기 그래픽스 프로세싱 유닛의 메모리의 사이즈에 기초하여 상기 드로우 콜을 복수의 서브-드로우 콜들로 분할하는 단계를 더 포함하며,

상기 복수의 서브-드로우 콜들 중 서브-드로우 콜들 각각은 상기 제 1 렌더링 패스의 동작들 및 상기 제 2 렌더링 패스의 동작들을 포함하는, 그래픽스를 렌더링하는 방법.

청구항 5

제 2 항에 있어서,

상기 헐 쉐이딩 동작과 연관된 명령들을 상기 버텍스 쉐이딩 동작과 연관된 명령들에 부가하고 상기 지오메트리 쉐이딩 동작과 연관된 명령들을 상기 도메인 쉐이딩 동작과 연관된 명령들에 부가하여, 상기 버텍스 쉐이딩 동작과 상기 헐 쉐이딩 동작이 순차적으로 실행되게 하고 상기 도메인 쉐이딩 동작과 상기 지오메트리 쉐이딩 동작이 순차적으로 실행되게 하는 단계를 더 포함하는, 그래픽스를 렌더링하는 방법.

청구항 6

제 1 항에 있어서,

상기 헐 쉐이딩 동작을 수행하는 단계는,

상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로 헐 쉐이더 프로그램의 제 1 인스턴스를 실행하는 단계;

상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로 상기 헐 쉐이더 프로그램의 제 2 인스턴스를 실행하는 단계;

상기 하드웨어 유닛의 단일 입력 단일 출력 인터페이스를 고수하도록 상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스로부터 단일 제어 포인트를 출력하는 단계; 및

상기 하드웨어 유닛의 상기 단일 입력 단일 출력 인터페이스를 고수하도록 상기 헐 쉐이더 프로그램의 상기 제 2 인스턴스로부터 제 2 의 단일 제어 포인트를 출력하는 단계를 포함하는, 그래픽스를 렌더링하는 방법.

청구항 7

제 6 항에 있어서,

상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스를 실행하는 단계는, 상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로 상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스와 상기 헐 쉐이더 프로그램의 상기 제 2 인스턴스를 동시에 실행하는 단계를 포함하는, 그래픽스를 렌더링하는 방법.

청구항 8

제 6 항에 있어서,

상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스에는 제 1 헐 쉐이더 출력 식별자가 할당되고,

상기 헐 쉐이더 프로그램의 상기 제 2 인스턴스에는 제 2 헐 쉐이더 출력 식별자가 할당되며,

상기 단일 제어 포인트를 출력하는 단계는 상기 제 1 헐 쉐이더 출력 식별자의 제 1 제어 포인트 식별자와의 비교에 기초하여 상기 단일 제어 포인트를 출력하는 단계를 포함하고, 그리고

상기 제 2 의 단일 제어 포인트를 출력하는 단계는 상기 제 2 헐 쉐이더 출력 식별자의 제 2 제어 포인트 식별자와의 비교에 기초하여 상기 제 2 의 단일 제어 포인트를 출력하는 단계를 포함하는, 그래픽스를 렌더링하는 방법.

청구항 9

제 1 항에 있어서,

상기 헐 쉐이딩 동작을 수행하기 전에, 상기 헐 쉐이딩 동작에 대한 프로그램 카운터와 하나 이상의 지원 포인터들을 스위칭하는 단계를 더 포함하는, 그래픽스를 렌더링하는 방법.

청구항 10

그래픽스를 렌더링하기 위한 그래픽스 프로세싱 유닛으로서,

복수의 입력 버텍스들을 저장하도록 구성된 메모리; 및

버텍스 쉐이딩을 위해 지정된 상기 그래픽스 프로세싱 유닛의 하드웨어 유닛으로서, 상기 하드웨어 유닛은:

버텍스 쉐이딩된 버텍스들을 출력하도록 상기 복수의 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작을 수행하는 것으로서, 상기 버텍스 쉐이딩 동작을 수행하기 위해 상기 하드웨어 유닛은 상기 복수의 입력 버텍스들의 각각의 입력 버텍스 각각에 대해 단일 버텍스를 출력하도록 구성되는, 상기 버텍스 쉐이딩 동작을 수행하고; 그리고

상기 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 대해 하나 이상의 테셀레이션 동작들을 수행하는 것으로서, 상기 하나 이상의 테셀레이션 동작들을 수행하기 위해 상기 하드웨어 유닛은 상기 하나 이상의 제어 포인트들을 출력하기 위하여 상기 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대한 헐 쉐이딩 동작을 수행하도록 구성되는, 상기 하나 이상의 테셀레이션 동작들을 수행하도록

구성되는, 상기 하드웨어 유닛을 포함하는, 그래픽스를 렌더링하기 위한 그래픽스 프로세싱 유닛.

청구항 11

제 10 항에 있어서,

상기 버텍스 쉐이딩 동작 및 상기 헐 쉐이딩 동작은 제 1 렌더링 패스와 연관되고,

상기 하드웨어 유닛은 추가로, 제 2 렌더링 패스를 수행하도록 구성되며,

상기 하드웨어 유닛은,

상기 제어 포인트들에 적어도 부분적으로 기초하여 버텍스 값들을 발생시키는 것을 포함하는 도메인 쉐이딩 동작을 수행하고; 그리고

하나 이상의 새로운 버텍스들을 발생시키기 위하여 하나 이상의 도메인 쉐이딩된 버텍스들에 대한 지오메트리 쉐이딩 동작을 수행하도록

구성되며,

상기 지오메트리 쉐이딩 동작은 상기 하나 이상의 새로운 버텍스들을 출력하기 위하여 상기 하나 이상의 도메인 쉐이딩된 버텍스들 중 적어도 하나에 작용하는, 그래픽스를 렌더링하기 위한 그래픽스 프로세싱 유닛.

청구항 12

제 11 항에 있어서,

상기 하드웨어 유닛은 추가로, 상기 그래픽스 프로세싱 유닛의 하나 이상의 컴포넌트들이 상기 제 1 렌더링 패스와 상기 제 2 렌더링 패스 간에 유휴하도록, 상기 제 2 렌더링 패스를 수행하기 전에 상기 제 1 렌더링 패스를 완료하도록 구성되는, 그래픽스를 렌더링하기 위한 그래픽스 프로세싱 유닛.

청구항 13

제 11 항에 있어서,

상기 버텍스 쉐이딩 동작, 상기 헐 쉐이딩 동작, 상기 도메인 쉐이딩 동작, 및 상기 지오메트리 쉐이딩 동작은 드로우 콜과 연관되고,

상기 하드웨어 유닛은 추가로, 상기 그래픽스 프로세싱 유닛의 메모리의 사이즈에 기초하여 상기 드로우 콜을 복수의 서브-드로우 콜들로 분할하도록 구성되며,

상기 복수의 서브-드로우 콜들 중 서브-드로우 콜들 각각은 상기 제 1 렌더링 패스의 동작들 및 상기 제 2 렌더링 패스의 동작들을 포함하는, 그래픽스를 렌더링하기 위한 그래픽스 프로세싱 유닛.

청구항 14

제 11 항에 있어서,

상기 하드웨어 유닛은 추가로, 상기 헐 쉐이딩 동작과 연관된 명령들을 상기 버텍스 쉐이딩 동작과 연관된 명령

들에 부가하고 상기 지오메트리 쉐이딩 동작과 연관된 명령들을 상기 도메인 쉐이딩 동작과 연관된 명령들에 부가하여, 상기 버텍스 쉐이딩 동작과 상기 헐 쉐이딩 동작이 순차적으로 실행되게 하고 상기 도메인 쉐이딩 동작과 상기 지오메트리 쉐이딩 동작이 순차적으로 실행되게 하도록 구성되는, 그래픽스를 렌더링하기 위한 그래픽스 프로세싱 유닛.

청구항 15

제 10 항에 있어서,

상기 헐 쉐이딩 동작을 수행하기 위해, 상기 하드웨어 유닛은,

상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로 헐 쉐이더 프로그램의 제 1 인스턴스를 실행하고;

상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로 상기 헐 쉐이더 프로그램의 제 2 인스턴스를 실행하고;

상기 하드웨어 유닛의 단일 입력 단일 출력 인터페이스를 고수하도록 상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스로부터 단일 제어 포인트를 출력하며; 그리고

상기 하드웨어 유닛의 상기 단일 입력 단일 출력 인터페이스를 고수하도록 상기 헐 쉐이더 프로그램의 상기 제 2 인스턴스로부터 제 2 의 단일 제어 포인트를 출력하도록

구성되는, 그래픽스를 렌더링하기 위한 그래픽스 프로세싱 유닛.

청구항 16

제 15 항에 있어서,

상기 하드웨어 유닛은, 상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로 상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스와 상기 헐 쉐이더 프로그램의 상기 제 2 인스턴스를 동시에 실행하도록 구성되는, 그래픽스를 렌더링하기 위한 그래픽스 프로세싱 유닛.

청구항 17

제 15 항에 있어서,

상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스에는 제 1 헐 쉐이더 출력 식별자가 할당되고,

상기 헐 쉐이더 프로그램의 상기 제 2 인스턴스에는 제 2 헐 쉐이더 출력 식별자가 할당되며,

상기 단일 제어 포인트를 출력하기 위하여, 상기 하드웨어 유닛은 상기 제 1 헐 쉐이더 출력 식별자의 제 1 제어 포인트 식별자와의 비교에 기초하여 상기 단일 제어 포인트를 출력하도록 구성되고, 그리고

상기 제 2 의 단일 제어 포인트를 출력하기 위하여, 상기 하드웨어 유닛은 상기 제 2 헐 쉐이더 출력 식별자의 제 2 제어 포인트 식별자와의 비교에 기초하여 상기 제 2 의 단일 제어 포인트를 출력하도록 구성되는, 그래픽스를 렌더링하기 위한 그래픽스 프로세싱 유닛.

청구항 18

제 10 항에 있어서,

상기 하드웨어 유닛은 상기 헐 쉐이딩 동작을 수행하기 전에, 상기 헐 쉐이딩 동작에 대한 프로그램 카운터와 하나 이상의 지원 포인터들을 스위칭하도록 구성되는, 그래픽스를 렌더링하기 위한 그래픽스 프로세싱 유닛.

청구항 19

그래픽스를 렌더링하기 위한 장치로서,

버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 유닛으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 복수의 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작을 수행하는 수단으로서, 상기 버텍스 쉐이딩 동작을 수행하는 수단은 상기 복수의 입력 버텍스들의 각각의 입력 버텍스 각각에 대해 단일 버텍스를 출

력하는 수단을 포함하는, 상기 버텍스 쉐이딩 동작을 수행하는 수단을 포함하고, 그리고

상기 버텍스 쉐이딩 동작을 수행하는 수단은 상기 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 대해 하나 이상의 테셀레이션 동작들을 수행하는 수단을 더 포함하고, 상기 하나 이상의 테셀레이션 동작들을 수행하는 수단은 하나 이상의 제어 포인트들을 출력하기 위하여 상기 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대한 헐 쉐이딩 동작을 수행하는 수단을 포함하는, 그래픽스를 렌더링하기 위한 장치.

청구항 20

제 19 항에 있어서,

상기 버텍스 쉐이딩 동작 및 상기 헐 쉐이딩 동작은 제 1 렌더링 패스에서 수행되고,

상기 버텍스 쉐이딩 동작을 수행하는 수단은:

제 2 렌더링 패스에서, 상기 제어 포인트들에 적어도 부분적으로 기초하여 버텍스 값들을 발생시키는 것을 포함하는 도메인 쉐이딩 동작을 수행하는 수단; 및

상기 제 2 렌더링 패스에서, 하나 이상의 새로운 버텍스들을 생성하기 위하여 하나 이상의 도메인 쉐이딩된 버텍스들에 대한 지오메트리 쉐이딩 동작을 수행하는 수단을 더 포함하며,

상기 지오메트리 쉐이딩 동작은 상기 하나 이상의 새로운 버텍스들을 출력하기 위하여 상기 하나 이상의 도메인 쉐이딩된 버텍스들 중 적어도 하나에 작용하는, 그래픽스를 렌더링하기 위한 장치.

청구항 21

제 20 항에 있어서,

상기 그래픽스 프로세싱 유닛의 하나 이상의 컴포넌트들이 상기 제 1 렌더링 패스와 상기 제 2 렌더링 패스 간에 유휴하도록, 상기 제 2 렌더링 패스를 수행하기 전에 상기 제 1 렌더링 패스를 완료하는 수단을 더 포함하는, 그래픽스를 렌더링하기 위한 장치.

청구항 22

제 20 항에 있어서,

상기 버텍스 쉐이딩 동작, 상기 헐 쉐이딩 동작, 상기 도메인 쉐이딩 동작, 및 상기 지오메트리 쉐이딩 동작은 드로우 콜과 연관되고,

상기 그래픽스 프로세싱 유닛의 메모리의 사이즈에 기초하여 상기 드로우 콜을 복수의 서브-드로우 콜들로 분할하는 수단을 더 포함하며,

상기 복수의 서브-드로우 콜들 중 서브-드로우 콜들 각각은 상기 제 1 렌더링 패스의 동작들 및 상기 제 2 렌더링 패스의 동작들을 포함하는, 그래픽스를 렌더링하기 위한 장치.

청구항 23

제 20 항에 있어서,

상기 헐 쉐이딩 동작과 연관된 명령들을 상기 버텍스 쉐이딩 동작과 연관된 명령들에 부가하고 상기 지오메트리 쉐이딩 동작과 연관된 명령들을 상기 도메인 쉐이딩 동작과 연관된 명령들에 부가하여, 상기 버텍스 쉐이딩 동작과 상기 헐 쉐이딩 동작이 순차적으로 실행되게 하고 상기 도메인 쉐이딩 동작과 상기 지오메트리 쉐이딩 동작이 순차적으로 실행되게 하는 수단을 더 포함하는, 그래픽스를 렌더링하기 위한 장치.

청구항 24

제 19 항에 있어서,

상기 헐 쉐이딩 동작을 수행하는 수단은,

상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로 헐 쉐이더 프로그램의 제 1 인스턴스를 실행하는 수단;

상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로 상기 헐 쉐이더 프로그램의 제 2 인스턴스를 실행하는

수단;

상기 하드웨어 유닛의 단일 입력 단일 출력 인터페이스를 고수하도록 상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스로부터 단일 제어 포인트를 출력하는 수단; 및

상기 하드웨어 유닛의 상기 단일 입력 단일 출력 인터페이스를 고수하도록 상기 헐 쉐이더 프로그램의 상기 제 2 인스턴스로부터 제 2 의 단일 제어 포인트를 출력하는 수단을 포함하는, 그래픽스를 렌더링하기 위한 장치.

청구항 25

제 24 항에 있어서,

상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스를 실행하는 수단은, 상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로 상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스와 상기 헐 쉐이더 프로그램의 상기 제 2 인스턴스를 동시에 실행하는 수단을 포함하는, 그래픽스를 렌더링하기 위한 장치.

청구항 26

제 24 항에 있어서,

상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스에는 제 1 헐 쉐이더 출력 식별자가 할당되고,

상기 헐 쉐이더 프로그램의 상기 제 2 인스턴스에는 제 2 헐 쉐이더 출력 식별자가 할당되며,

상기 단일 제어 포인트를 출력하는 수단은 상기 제 1 헐 쉐이더 출력 식별자의 제 1 제어 포인트 식별자와의 비교에 기초하여 상기 단일 제어 포인트를 출력하는 수단을 포함하고, 그리고

상기 제 2 의 단일 제어 포인트를 출력하는 수단은 상기 제 2 헐 쉐이더 출력 식별자의 제 2 제어 포인트 식별자와의 비교에 기초하여 상기 제 2 의 단일 제어 포인트를 출력하는 수단을 포함하는, 그래픽스를 렌더링하기 위한 장치.

청구항 27

제 19 항에 있어서,

상기 헐 쉐이딩 동작을 수행하기 전에, 상기 헐 쉐이딩 동작에 대한 프로그램 카운터와 하나 이상의 지원 포인터들을 스위칭하는 수단을 더 포함하는, 그래픽스를 렌더링하기 위한 장치.

청구항 28

명령들을 저장한 비-일시적인 컴퓨터-판독가능한 매체로서,

상기 명령들은, 실행될 경우, 그래픽스 프로세싱 유닛으로 하여금,

버텍스 쉐이딩을 위해 지정된 상기 그래픽스 프로세싱 유닛의 하드웨어 유닛으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 복수의 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작을 수행하게 하는 것으로서, 상기 버텍스 쉐이딩 동작을 수행하기 위해 상기 명령들은, 상기 하드웨어 유닛으로 하여금, 상기 복수의 입력 버텍스들의 각각의 입력 버텍스 각각에 대해 단일 버텍스를 출력하게 하는, 상기 버텍스 쉐이딩 동작을 수행하게 하고; 그리고

버텍스 쉐이딩을 위해 지정된 상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로, 상기 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 대해 하나 이상의 테셀레이션 동작들을 수행하게 하는 것으로서, 상기 하나 이상의 테셀레이션 동작들을 수행하기 위해 상기 명령들은, 상기 하드웨어 유닛으로 하여금, 상기 하나 이상의 제어 포인트들을 출력하기 위하여 상기 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대한 헐 쉐이딩 동작을 수행하게 하는, 상기 하나 이상의 테셀레이션 동작들을 수행하게 하는, 비-일시적인 컴퓨터-판독가능한 매체.

청구항 29

제 28 항에 있어서,

상기 버텍스 쉐이딩 동작 및 상기 헐 쉐이딩 동작은 제 1 렌더링 패스와 연관되고,

상기 하드웨어 유닛으로 하여금 제 2 렌더링 패스를 수행하게 하는 명령들을 더 포함하며,

상기 명령들은 상기 하드웨어 유닛으로 하여금,

상기 제어 포인트들에 적어도 부분적으로 기초하여 버텍스 값들을 발생시키는 것을 포함하는 도메인 쉐이딩 동작을 수행하게 하고; 그리고

하나 이상의 새로운 버텍스들을 발생시키기 위하여 하나 이상의 도메인 쉐이딩된 버텍스들에 대한 지오메트리 쉐이딩 동작을 수행하게 하며,

상기 지오메트리 쉐이딩 동작은 상기 하나 이상의 새로운 버텍스들을 출력하기 위하여 상기 하나 이상의 도메인 쉐이딩된 버텍스들 중 적어도 하나에 작용하는, 비-일시적인 컴퓨터-판독가능한 매체.

청구항 30

제 29 항에 있어서,

상기 하드웨어 유닛으로 하여금, 상기 제 2 렌더링 패스를 수행하기 전에 상기 제 1 렌더링 패스를 완료하게 하고, 상기 제 1 렌더링 패스와 상기 제 2 렌더링 패스 간에 유휴를 위해 대기하게 하는 명령들을 더 포함하는, 비-일시적인 컴퓨터-판독가능한 매체.

청구항 31

제 29 항에 있어서,

상기 버텍스 쉐이딩 동작, 상기 헐 쉐이딩 동작, 상기 도메인 쉐이딩 동작, 및 상기 지오메트리 쉐이딩 동작은 드로우 콜과 연관되고,

상기 그래픽스 프로세싱 유닛으로 하여금, 상기 그래픽스 프로세싱 유닛의 메모리의 사이즈에 기초하여 상기 드로우 콜을 복수의 서브-드로우 콜들로 분할하게 하는 명령들을 더 포함하며,

상기 복수의 서브-드로우 콜들 중 서브-드로우 콜들 각각은 상기 제 1 렌더링 패스의 동작들 및 상기 제 2 렌더링 패스의 동작들을 포함하는, 비-일시적인 컴퓨터-판독가능한 매체.

청구항 32

제 29 항에 있어서,

상기 그래픽스 프로세싱 유닛으로 하여금, 상기 헐 쉐이딩 동작과 연관된 명령들을 상기 버텍스 쉐이딩 동작과 연관된 명령들에 부가하게 하고 상기 지오메트리 쉐이딩 동작과 연관된 명령들을 상기 도메인 쉐이딩 동작과 연관된 명령들에 부가하게 하여, 상기 버텍스 쉐이딩 동작과 상기 헐 쉐이딩 동작이 순차적으로 실행되게 하고 상기 도메인 쉐이딩 동작과 상기 지오메트리 쉐이딩 동작이 순차적으로 실행되게 하는 명령들을 더 포함하는, 비-일시적인 컴퓨터-판독가능한 매체.

청구항 33

제 28 항에 있어서,

상기 헐 쉐이딩 동작을 수행하기 위해, 상기 명령들은 상기 하드웨어 유닛으로 하여금,

상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로 헐 쉐이더 프로그램의 제 1 인스턴스를 실행하게 하고;

상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로 상기 헐 쉐이더 프로그램의 제 2 인스턴스를 실행하게 하고;

상기 하드웨어 유닛의 단일 입력 단일 출력 인터페이스를 고수하도록 상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스로부터 단일 제어 포인트를 출력하게 하며; 그리고

상기 하드웨어 유닛의 상기 단일 입력 단일 출력 인터페이스를 고수하도록 상기 헐 쉐이더 프로그램의 상기 제 2 인스턴스로부터 제 2 의 단일 제어 포인트를 출력하게 하는, 비-일시적인 컴퓨터-판독가능한 매체.

청구항 34

제 33 항에 있어서,

상기 명령들은 상기 하드웨어 유닛으로 하여금, 상기 그래픽스 프로세싱 유닛의 상기 하드웨어 유닛으로 상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스와 상기 헐 쉐이더 프로그램의 상기 제 2 인스턴스를 동시에 실행하게 하는, 비-일시적인 컴퓨터-판독가능한 매체.

청구항 35

제 33 항에 있어서,

상기 헐 쉐이더 프로그램의 상기 제 1 인스턴스에는 제 1 헐 쉐이더 출력 식별자가 할당되고,

상기 헐 쉐이더 프로그램의 상기 제 2 인스턴스에는 제 2 헐 쉐이더 출력 식별자가 할당되며,

상기 단일 제어 포인트를 출력하기 위하여, 상기 명령들은 상기 하드웨어 유닛으로 하여금, 상기 제 1 헐 쉐이더 출력 식별자의 제 1 제어 포인트 식별자와의 비교에 기초하여 상기 단일 제어 포인트를 출력하게 하고, 그리고

상기 제 2 의 단일 제어 포인트를 출력하기 위하여, 상기 명령들은 상기 하드웨어 유닛으로 하여금, 상기 제 2 헐 쉐이더 출력 식별자의 제 2 제어 포인트 식별자와의 비교에 기초하여 상기 제 2 의 단일 제어 포인트를 출력하게 하는, 비-일시적인 컴퓨터-판독가능한 매체.

청구항 36

제 28 항에 있어서,

상기 명령들은 상기 하드웨어 유닛으로 하여금, 상기 헐 쉐이딩 동작을 수행하기 전에, 상기 헐 쉐이딩 동작에 대한 프로그램 카운터와 하나 이상의 지원 포인터들을 스위칭하게 하는, 비-일시적인 컴퓨터-판독가능한 매체.

청구항 37

제 1 항에 있어서,

상기 버텍스 쉐이딩된 버텍스들을 출력하도록 입력 버텍스들을 쉐이딩하기 위하여 상기 버텍스 쉐이딩 동작을 수행하는 단계는, 패치 제어 포인트들을 출력하도록 입력 버텍스들을 쉐이딩하기 위하여 상기 버텍스 쉐이딩 동작을 수행하는 단계를 포함하고,

상기 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 대한 상기 헐 쉐이딩 동작을 수행하는 단계는, 상기 하나 이상의 제어 포인트들 및 하나 이상의 테셀레이션 인자들을 생성하기 위하여 상기 패치 제어 포인트들에 대한 상기 헐 쉐이딩 동작을 수행하는 단계를 포함하는, 그래픽스를 렌더링하는 방법.

발명의 설명

기 술 분 야

[0001] 이 출원은 **2012 년 4 월 4 일**자로 출원된 미국 가출원 제 **61/620,340** 호, **2012 년 4 월 4 일**자로 출원된 미국 가출원 제 **61/620,358** 호, 및 **2012 년 4 월 4 일**자로 출원된 미국 가출원 제 **61/620,333** 호의 이익을 주장하고, 이들 모두의 전체 내용들은 참조를 위해 본원에 통합된다.

[0002] 이 개시물은 컴퓨터 그래픽스에 관한 것이다.

배 경 기 술

[0003] 시각적 제시를 위한 컨텐츠를 제공하는 디바이스는 일반적으로 그래픽스 프로세싱 유닛 (GPU) 을 포함한다. GPU 는 디스플레이 상에 컨텐츠를 나타내는 픽셀들을 렌더링한다. GPU 는 제시를 위한 각각의 픽셀을 렌더링하기 위하여 디스플레이 상의 각각의 픽셀에 대한 하나 이상의 픽셀 값들을 발생시킨다.

[0004] 일부의 인스턴스들에서, GPU 는 그래픽스를 렌더링하기 위한 통합된 쉐이더 아키텍처를 구현할 수도 있다. 이러한 인스턴스들에서, GPU 는 상이한 쉐이딩 동작들의 파이프라인을 실행하도록 복수의 유사한 컴퓨팅 유닛들

을 구성할 수도 있다. 컴퓨팅 유닛들은 통합된 쉐이딩 유닛들 또는 통합된 쉐이더 프로세서들이라고 지칭될 수도 있다.

발명의 내용

해결하려는 과제

과제의 해결 수단

[0005] 이 개시물의 양태들은 그래픽스 렌더링 파이프라인의 쉐이더 스테이지들과 연관된 쉐이딩 동작들을 수행하는 것에 관한 것이다. 예를 들어, 그래픽스 프로세싱 유닛 (GPU) 은 그래픽스 렌더링 파이프라인의 쉐이더 스테이지와 연관된 쉐이딩 동작들을 수행하기 위하여 하나 이상의 쉐이딩 유닛들을 호출할 수도 있다. 이 개시물의 양태들에 따르면, 다음으로, GPU 는 제 1 쉐이딩 동작들을 수행하도록 지정되는 쉐이딩 유닛들로, 그래픽스 렌더링 파이프라인의 제 2 의 상이한 쉐이더 스테이지와 연관된 쉐이딩 동작들을 수행할 수도 있다. 예를 들어, GPU 는 제 1 쉐이더 스테이지와 연관된 입력/출력 인터페이스를 고수 (adhere) 하면서 제 2 스테이지와 연관된 쉐이딩 동작들을 수행할 수도 있다. 이러한 방법으로, GPU 는 동일한 쉐이딩 유닛으로 다수의 쉐이딩 동작들을 수행함으로써 더 큰 쉐이딩 자원들을 갖는 GPU 를 에뮬레이팅할 수도 있다.

[0006] 일 예에서, 이 개시물의 양태들은, 버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 쉐이딩 유닛으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 (vertex shading) 동작들을 수행하는 단계로서, 여기서, 하드웨어 유닛은 단일 버텍스를 입력으로서 수신하고 단일 버텍스를 출력으로서 발생시키도록 구성되는, 상기 버텍스 쉐이딩 동작들을 수행하는 단계, 및 그래픽스 프로세싱 유닛의 하드웨어 쉐이딩 유닛으로, 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 기초하여 하나 이상의 새로운 버텍스들을 발생시키기 위하여 지오메트리 쉐이딩 동작을 수행하는 단계로서, 여기서, 지오메트리 쉐이딩 동작은 하나 이상의 새로운 버텍스들을 출력하기 위하여 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대해 작용하는, 상기 지오메트리 쉐이딩 동작을 수행하는 단계를 포함하는, 그래픽스를 렌더링하는 방법에 관한 것이다.

[0007] 또 다른 예에서, 이 개시물의 양태들은, 버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 쉐이딩 유닛으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작들을 수행하도록 구성된 것으로서, 여기서, 하드웨어 유닛은 단일 버텍스를 입력으로서 수신하고 단일 버텍스를 출력으로서 발생시키도록 구성되는, 상기 버텍스 쉐이딩 동작들을 수행하도록 구성되고, 그래픽스 프로세싱 유닛의 하드웨어 쉐이딩 유닛으로서, 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 기초하여 하나 이상의 새로운 버텍스들을 발생시키기 위하여 지오메트리 쉐이딩 동작을 수행하도록 구성된 것으로서, 여기서, 지오메트리 쉐이딩 동작은 하나 이상의 새로운 버텍스들을 출력하기 위하여 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대해 작용하는, 상기 지오메트리 쉐이딩 동작을 수행하도록 구성된 하나 이상의 프로세서들을 포함하는, 그래픽스를 렌더링하기 위한 그래픽스 프로세싱 유닛에 관한 것이다.

[0008] 또 다른 예에서, 이 개시물의 양태들은, 버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 쉐이딩 유닛으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작들을 수행하는 수단으로서, 여기서, 하드웨어 유닛은 단일 버텍스를 입력으로서 수신하고 단일 버텍스를 출력으로서 발생시키도록 구성되는, 상기 버텍스 쉐이딩 동작들을 수행하는 수단, 및 그래픽스 프로세싱 유닛의 하드웨어 쉐이딩 유닛으로, 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 기초하여 하나 이상의 새로운 버텍스들을 발생시키기 위하여 지오메트리 쉐이딩 동작을 수행하는 수단으로서, 여기서, 지오메트리 쉐이딩 동작은 하나 이상의 새로운 버텍스들을 출력하기 위하여 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대해 작용하는, 상기 지오메트리 쉐이딩 동작을 수행하는 수단을 포함하는, 그래픽스를 렌더링하기 위한 장치에 관한 것이다.

[0009] 또 다른 예에서, 이 개시물의 양태들은, 실행될 경우, 하나 이상의 프로세서들로 하여금, 버텍스 쉐이딩을 위해 지정된 하드웨어 쉐이딩 유닛으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작들을 수행하게 하는 것으로서, 여기서, 하드웨어 유닛은 단일 버텍스를 입력으로서 수신하고 단일 버텍스를 출력으로서 발생시키도록 구성되는, 상기 버텍스 쉐이딩 동작들을 수행하게 하고, 버텍스 쉐이딩을 위해 지정되는 하드웨어 쉐이딩 유닛으로, 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 기초하여 하나 이상

의 새로운 버텍스들을 발생시키기 위하여 지오메트리 쉐이딩 동작을 수행하게 하는 것으로서, 여기서, 지오메트리 쉐이딩 동작은 하나 이상의 새로운 버텍스들을 출력하기 위하여 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대해 작용하는, 상기 지오메트리 쉐이딩 동작을 수행하게 하는 명령들을 저장한 비-일시적인 컴퓨터-판독가능한 매체에 관한 것이다.

[0010] 또 다른 예에서, 이 개시물의 양태들은, 버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 유닛으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작을 수행하는 단계로서, 여기서, 하드웨어 유닛은 단일 버텍스를 입력으로서 수신하고 단일 버텍스를 출력으로서 발생시키는 인터페이스를 고수하는, 상기 버텍스 쉐이딩 동작을 수행하는 단계, 및 버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 유닛으로, 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 기초하여 하나 이상의 제어 포인트들을 발생시키기 위하여 헐 쉐이딩 (hull shading) 동작을 수행하는 단계로서, 여기서, 하나 이상의 헐 쉐이딩 동작들은 하나 이상의 제어 포인트들을 출력하기 위하여 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대해 작용하는, 상기 헐 쉐이딩 동작을 수행하는 단계를 포함하는, 그래픽스를 렌더링하기 위한 방법에 관한 것이다.

[0011] 또 다른 예에서, 이 개시물의 양태들은, 버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 유닛으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작을 수행하도록 구성된 것으로서, 여기서, 하드웨어 유닛은 단일 버텍스를 입력으로서 수신하고 단일 버텍스를 출력으로서 발생시키는 인터페이스를 고수하는, 상기 버텍스 쉐이딩 동작을 수행하도록 구성되고, 버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 유닛으로, 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 기초하여 하나 이상의 제어 포인트들을 발생시키기 위하여 헐 쉐이딩 동작을 수행하도록 구성된 것으로서, 여기서, 하나 이상의 헐 쉐이딩 동작들은 하나 이상의 제어 포인트들을 출력하기 위하여 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대해 작용하는, 상기 헐 쉐이딩 동작을 수행하도록 구성된 하나 이상의 프로세서들을 포함하는, 그래픽스를 렌더링하기 위한 그래픽스 프로세싱 유닛에 관한 것이다.

[0012] 또 다른 예에서, 이 개시물의 양태들은, 버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 유닛으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작을 수행하는 수단으로서, 여기서, 하드웨어 유닛은 단일 버텍스를 입력으로서 수신하고 단일 버텍스를 출력으로서 발생시키는 인터페이스를 고수하는, 상기 버텍스 쉐이딩 동작을 수행하는 수단, 및 버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 유닛으로, 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 기초하여 하나 이상의 제어 포인트들을 발생시키기 위하여 헐 쉐이딩 동작을 수행하는 수단으로서, 여기서, 하나 이상의 헐 쉐이딩 동작들은 하나 이상의 제어 포인트들을 출력하기 위하여 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대해 작용하는, 상기 헐 쉐이딩 동작을 수행하는 수단을 포함하는, 그래픽스를 렌더링하기 위한 장치에 관한 것이다.

[0013] 또 다른 예에서, 이 개시물의 양태들은, 실행될 경우, 하나 이상의 프로세서들로 하여금, 버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 유닛으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작을 수행하게 하는 것으로서, 여기서, 하드웨어 유닛은 단일 버텍스를 입력으로서 수신하고 단일 버텍스를 출력으로서 발생시키는 인터페이스를 고수하는, 상기 버텍스 쉐이딩 동작을 수행하게 하고, 버텍스 쉐이딩을 위해 지정된 그래픽스 프로세싱 유닛의 하드웨어 유닛으로, 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 기초하여 하나 이상의 제어 포인트들을 발생시키기 위하여 헐 쉐이딩 동작을 수행하게 하는 것으로서, 여기서, 하나 이상의 헐 쉐이딩 동작들은 하나 이상의 제어 포인트들을 출력하기 위하여 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대해 작용하는, 상기 헐 쉐이딩 동작을 수행하게 하는 명령들을 갖는 비-일시적인 컴퓨터-판독가능한 매체에 관한 것이다.

[0014] 일 예에서, 이 개시물의 양태들은, 렌더링 파이프라인의 제 1 쉐이더 스테이지와 연관된 제 1 쉐이딩 동작들을 수행하도록 그래픽스 프로세싱 유닛의 하드웨어 쉐이딩 유닛을 지정하는 단계, 제 1 쉐이딩 동작들의 완료 시에 하드웨어 쉐이딩 유닛의 동작 모드들을 스위칭하는 단계, 및 제 1 쉐이딩 동작들을 수행하도록 지정된 그래픽스 프로세싱 유닛의 하드웨어 쉐이딩 유닛으로, 렌더링 파이프라인의 제 2 의 상이한 쉐이더 스테이지와 연관된 제 2 쉐이딩 동작들을 수행하는 단계를 포함하는, 그래픽스를 렌더링하는 방법에 관한 것이다.

[0015] 또 다른 예에서, 이 개시물의 양태들은, 렌더링 파이프라인의 제 1 쉐이더 스테이지와 연관된 제 1 쉐이딩 동작들을 수행하도록 그래픽스 프로세싱 유닛의 하드웨어 쉐이딩 유닛을 지정하고, 제 1 쉐이딩 동작들의 완료 시에 하드웨어 쉐이딩 유닛의 동작 모드들을 스위칭하고, 제 1 쉐이딩 동작들을 수행하도록 지정된 그래픽스 프로세

싱 유닛의 하드웨어 쉐이딩 유닛으로, 렌더링 파이프라인의 제 2 의 상이한 쉐이더 스테이지와 연관된 제 2 쉐이딩 동작들을 수행하도록 구성된 하나 이상의 프로세서들을 포함하는, 그래픽스를 렌더링하기 위한 그래픽스 프로세싱 유닛에 관한 것이다.

[0016] 또 다른 예에서, 이 개시물의 양태들은, 렌더링 파이프라인의 제 1 쉐이더 스테이지와 연관된 제 1 쉐이딩 동작들을 수행하도록 그래픽스 프로세싱 유닛의 하드웨어 쉐이딩 유닛을 지정하는 수단, 제 1 쉐이딩 동작들의 완료 시에 하드웨어 쉐이딩 유닛의 동작 모드들을 스위칭하는 수단, 및 제 1 쉐이딩 동작들을 수행하도록 지정된 그래픽스 프로세싱 유닛의 하드웨어 쉐이딩 유닛으로, 렌더링 파이프라인의 제 2 의 상이한 쉐이더 스테이지와 연관된 제 2 쉐이딩 동작들을 수행하는 수단을 포함하는, 그래픽스를 렌더링하기 위한 장치에 관한 것이다.

[0017] 또 다른 예에서, 이 개시물의 양태들은, 실행될 경우, 하나 이상의 프로세서들로 하여금, 렌더링 파이프라인의 제 1 쉐이더 스테이지와 연관된 제 1 쉐이딩 동작들을 수행하도록 그래픽스 프로세싱 유닛의 하드웨어 쉐이딩 유닛을 지정하게 하고, 제 1 쉐이딩 동작들의 완료 시에 하드웨어 쉐이딩 유닛의 동작 모드들을 스위칭하게 하고, 제 1 쉐이딩 동작들을 수행하도록 지정된 그래픽스 프로세싱 유닛의 하드웨어 쉐이딩 유닛으로, 렌더링 파이프라인의 제 2 의 상이한 쉐이더 스테이지와 연관된 제 2 쉐이딩 동작들을 수행하게 하는 명령들을 저장한 비-일시적인 컴퓨터-판독가능한 매체에 관한 것이다.

[0018] 개시물의 하나 이상의 예들의 세부사항들은 첨부한 도면들 및 이하의 설명에서 기재되어 있다. 다른 특징들, 목적들 및 장점들은 설명 및 도면들로부터, 그리고 청구범위들로부터 명백할 것이다.

도면의 간단한 설명

[0019] 도 1 은 이 개시물에서 설명된 기술들을 구현할 수도 있는 컴퓨팅 디바이스를 예시하는 블록도이다.

도 2 는 예시적인 그래픽스 프로세싱 파이프라인 (80) 을 예시하는 블록도이다.

도 3a 및 도 3b 는 이 개시물의 양태들에 따라, 그래픽스 렌더링 파이프라인에서의 데이터 흐름들의 개념적인 도면들이다.

도 4 는 버텍스 쉐이딩 (vertex shading) 동작들 및 지오메트리 쉐이딩 (geometry shading) 동작들을 수행하기 위하여 이 개시물에서 설명된 기술들을 구현하는 하드웨어 쉐이딩 유닛의 일 예의 동작들을 예시하는 도면이다.

도 5a 는 버텍스 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행할 때, 병합된 버텍스 쉐이더/지오메트리 쉐이더 하드웨어 쉐이딩 유닛에 의해 수행되는 동작들의 흐름을 예시한다.

도 5b 는 병합된 버텍스 쉐이더/지오메트리 쉐이더 하드웨어 쉐이딩 유닛에 의해 실행될 수도 있는, 도 5a 에 도시된 동작들의 흐름에 대응하는 의사 코드를 예시한다.

도 6 은 이 개시물의 양태들에 따라, 병합된 버텍스 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행하기 위한 그래픽스 프로세싱 유닛의 일 예의 컴포넌트들을 예시하는 도면이다.

도 7 은 이 개시물의 양태들에 따라, 버텍스 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행하기 위한 일 예의 프로세스를 예시하는 플로우차트이다.

도 8 은 테셀레이션 스테이지 (tessellation stage) 들을 포함하는 일 예의 그래픽스 프로세싱 파이프라인을 예시하는 블록도이다.

도 9 는 테셀레이션을 예시하는 개념적인 도면이다.

도 10a 및 도 10b 는 이 개시물의 양태들에 따라, 그래픽스 렌더링 파이프라인에서의 데이터 흐름들의 개념적인 도면들이다.

도 11 은 버텍스 쉐이딩 및 헐 쉐이딩 (hull shading) 동작들을 수행하기 위하여 이 개시물에서 설명된 기술들을 구현하는 하드웨어 쉐이딩 유닛의 일 예의 동작들을 예시하는 도면이다.

도 12a 는 버텍스 쉐이딩 동작들 및 헐 쉐이딩 동작들을 수행할 때, 병합된 버텍스 쉐이더/헐 쉐이더 하드웨어 쉐이딩 유닛에 의해 수행되는 동작들의 흐름을 예시한다.

도 12b 는 병합된 버텍스 쉐이더/헐 쉐이더 하드웨어 쉐이딩 유닛에 의해 실행될 수도 있는, 도 12a 에 도시된 동작들의 흐름에 대응하는 의사 코드를 일반적으로 예시한다.

도 13a 는 도메인 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행할 때, 병합된 도메인 쉐이더/지오메트리 쉐이더 하드웨어 쉐이딩 유닛에 의해 수행되는 동작들의 흐름을 일반적으로 예시한다.

도 13b 는 병합된 도메인 쉐이더/지오메트리 쉐이더 하드웨어 쉐이딩 유닛에 의해 실행될 수도 있는, 도 13a 에 도시된 동작들의 흐름에 대응하는 의사 코드를 일반적으로 예시한다.

도 14 는 이 개시물의 양태들에 따라, 병합된 버텍스 쉐이딩, 헐 쉐이딩, 도메인 쉐이딩, 및 지오메트리 쉐이딩 동작들을 수행하기 위한 그래픽스 프로세싱 유닛의 일 예의 컴포넌트들을 예시하는 도면이다.

도 15 는 이 개시물의 양태들에 따라, 동일한 하드웨어 쉐이딩 유닛을 이용하여 2 개의 렌더링 패스들에서 그래픽스 렌더링을 수행하는 것을 예시하는 흐름도이다.

도 16 은 이 개시물의 양태들에 따라, 2 패스 그래픽스 렌더링 프로세스의 제 1 패스와 연관된 그래픽스 렌더링 동작들을 수행하는 것을 예시하는 흐름도이다.

도 17 은 이 개시물의 양태들에 따라, 2 패스 그래픽스 렌더링 프로세스의 제 2 패스와 연관된 그래픽스 렌더링 동작들을 수행하는 것을 예시하는 흐름도이다.

도 18 은 이 개시물의 양태들에 따라, 동일한 하드웨어 쉐이딩 유닛에 의한 실행을 위하여 하나를 초과하는 쉐이더 스테이지를 함께 패치하는 것을 예시하는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0020] 이 개시물의 기술들은 일반적으로, 그래픽스 렌더링 파이프라인의 쉐이더 스테이지들과 연관된 쉐이딩 동작들을 수행하는 것에 관한 것이다. 예를 들어, 그래픽스 프로세싱 유닛 (GPU) 은 그래픽스 렌더링 파이프라인의 쉐이더 스테이지와 연관된 쉐이딩 동작들을 수행하기 위하여 하나 이상의 쉐이딩 유닛들을 호출할 수도 있다.

이 개시물의 양태들에 따르면, 다음으로, GPU 는 제 1 쉐이딩 동작들을 수행하도록 지정된 쉐이딩 유닛들을 갖는 그래픽스 렌더링 파이프라인의 제 2 상이한 쉐이더 스테이지와 연관된 쉐이딩 동작들을 수행할 수도 있다.

예를 들어, GPU 는 제 1 쉐이더 스테이지와 연관된 입력/출력 인터페이스를 고수하면서 제 2 스테이지와 연관된 쉐이딩 동작들을 수행할 수도 있다. 이러한 방법으로, GPU 는 동일한 쉐이딩 유닛들로 다수의 쉐이딩 동작들을 수행함으로써 더 큰 쉐이딩 자원들을 갖는 GPU 를 에뮬레이팅할 수도 있다.

[0021] 도 1 은 이 개시물에서 설명된 기술들을 구현할 수도 있는 컴퓨팅 디바이스 (30) 를 예시하는 블록도이다. 컴퓨팅 디바이스 (30) 의 예들은 무선 디바이스들, 소위 스마트폰들을 포함하는 이동 또는 셀룰러 전화들, 개인 정보 단말 (PDA) 들, 비디오 디스플레이들을 포함하는 비디오 게임용 콘솔들, 이동 비디오 게임용 디바이스들, 화상 회의 유닛들, 랩톱 컴퓨터들, 데스크톱 컴퓨터들, 텔레비전 셋톱 박스들, 태블릿 컴퓨팅 디바이스들, 전자책 (e-book) 리더들, 고정 또는 이동 미디어 플레이어들, 등을 포함하지만, 이것으로 제한되지 않는다.

[0022] 도 1 의 예에서, 컴퓨팅 디바이스 (30) 는 CPU 메모리 (34) 를 갖는 중앙 프로세싱 유닛 (CPU; 32), GPU 메모리 (38) 및 하나 이상의 쉐이딩 유닛들 (40) 을 갖는 그래픽스 프로세싱 유닛 (GPU; 36), 디스플레이 유닛 (42), 디스플레이 버퍼 유닛 (44), 사용자 인터페이스 유닛 (46), 및 저장 유닛 (48) 을 포함한다. 추가적으로, 저장 유닛 (48) 은 컴파일러 (54) 를 갖는 GPU 드라이버 (50), GPU 프로그램 (52), 및 국소적으로 컴파일링된 GPU 프로그램 (56) 을 저장할 수도 있다.

[0023] CPU (32) 의 예들은 디지털 신호 프로세서 (DSP), 범용 마이크로프로세서, 주문형 반도체 회로 (ASIC), 필드 프로그래밍가능한 로직 어레이 (FPGA), 또는 다른 등가의 통합된 또는 별개의 로직 회로부를 포함하지만, 이것으로 제한되지 않는다. CPU (32) 및 GPU (36) 는 도 1 의 예에서 별도의 유닛들로서 예시되어 있지만, 일부의 예들에서, CPU (32) 및 GPU (36) 는 단일 유닛 내로 통합될 수도 있다. CPU (32) 는 하나 이상의 애플리케이션들을 실행할 수도 있다. 애플리케이션들의 예들은 웹 브라우저들, e-메일 애플리케이션들, 스프레드시트들, 비디오 게임들, 오디오 및/또는 비디오 캡처, 재생 또는 편집 애플리케이션들, 또는 디스플레이 유닛 (42) 을 통해 제시될 이미지 데이터에 대한 발생을 개시하는 다른 애플리케이션들을 포함할 수도 있다.

[0024] 도 1 에 도시된 예에서, CPU (32) 는 CPU 메모리 (34) 를 포함한다. CPU 메모리 (34) 는 머신 또는 오브젝트 코드를 실행함에 있어서 이용되는 온-칩 저장장치 또는 메모리를 나타낼 수도 있다. CPU 메모리 (34) 는 고정된 수의 디지털 비트들을 저장할 수 있는 하드웨어 메모리 레지스터를 각각 포함할 수도 있다. CPU (32) 는 예를 들어, 시스템 버스를 통해 액세스될 수도 있는 저장 유닛 (48) 으로부터 값들을 판독하거나 저장 유닛 (48) 에 값들을 기록하는 것보다 더 신속하게 로컬 CPU 메모리 (34) 로부터 값들을 판독할 수도 있거나 로

컬 CPU 메모리 (34) 에 값들을 기록할 수도 있다.

[0025] GPU (36) 는 그래픽 동작들을 수행하기 위한 하나 이상의 전용 프로세서들을 나타낸다. 즉, 예를 들어, GPU (36) 는 그래픽스를 렌더링하고 GPU 애플리케이션들을 실행하기 위한 프로그래밍가능한 컴포넌트들 및 고정된 기능을 갖는 전용 하드웨어 유닛일 수도 있다. GPU (36) 는 또한, DSP, 범용 마이크로프로세서, ASIC, FPGA, 또는 다른 등가의 통합된 또는 별개의 로직 회로부를 포함할 수도 있다.

[0026] GPU (36) 는 또한, 머신 또는 오브젝트 코드를 실행함에 있어서 이용되는 온-칩 저장장치 또는 메모리를 나타낼 수도 있는 GPU 메모리 (38) 를 포함한다. GPU 메모리 (38) 는 고정된 수의 디지털 비트들을 저장할 수 있는 하드웨어 메모리 레지스터를 각각 포함할 수도 있다. GPU (36) 는 예를 들어, 시스템 버스를 통해 액세스될 수도 있는 저장 유닛 (48) 으로부터 값들을 판독하거나 저장 유닛 (48) 에 값들을 기록하는 것보다 더 신속하게 로컬 GPU 메모리 (38) 로부터 값들을 값들을 판독할 수도 있거나 로컬 GPU 메모리 (38) 에 값들을 기록할 수도 있다.

[0027] GPU (36) 는 또한 쉐이딩 유닛들 (40) 을 포함한다. 이하에서 더욱 상세하게 설명된 바와 같이, 쉐이딩 유닛들 (40) 은 프로세싱 컴포넌트들의 프로그래밍가능한 파이프라인으로서 구성될 수도 있다. 일부의 예들에서, 쉐이딩 유닛들 (40) 은 "쉐이더 프로세서들" 또는 "통합된 쉐이더들" 이라고 지칭될 수도 있고, 그래픽스를 렌더링하기 위하여 지오메트리, 버텍스, 픽셀, 또는 다른 쉐이딩 동작들을 수행할 수도 있다. 쉐이딩 유닛들 (40) 은, 명령들을 페치 (fetch) 하고 디코딩하기 위한 컴포넌트들, 산술적 계산들을 수행하기 위한 하나 이상의 산술 논리 유닛 ("ALU") 들, 및 하나 이상의 메모리들, 캐쉬들, 또는 레지스터들과 같이, 명료함의 목적들을 위하여 도 1 에 구체적으로 도시되지 않은 하나 이상의 컴포넌트들을 포함할 수도 있다.

[0028] 디스플레이 유닛 (42) 은 비디오 데이터, 이미지들, 텍스트 또는 뷰어에 의한 소비를 위한 임의의 다른 타입의 데이터를 디스플레이할 수 있는 유닛을 나타낸다. 디스플레이 유닛 (42) 은 액정 디스플레이 (LCD), 발광 다이오드 (LED) 디스플레이, 유기 LED (OLED), 능동 매트릭스 OLED (AMOLED) 디스플레이, 등을 포함할 수도 있다.

[0029] 디스플레이 버퍼 유닛 (44) 은 디스플레이 유닛 (42) 을 위한, 사진들 또는 비디오 프레임들과 같은 이미지의 제시를 위한 데이터를 저장하는 전용으로 된 메모리 또는 저장 디바이스를 나타낸다. 디스플레이 버퍼 유닛 (44) 은 복수의 저장 로케이션들을 포함하는 2 차원 버퍼를 나타낼 수도 있다. 디스플레이 버퍼 유닛 (44) 내의 저장 로케이션들의 수는 디스플레이 유닛 (42) 상에 디스플레이될 픽셀들의 수와 실질적으로 유사할 수도 있다. 예를 들어, 디스플레이 유닛 (42) 이 640 x 480 픽셀들을 포함하도록 구성될 경우, 디스플레이 버퍼 유닛 (44) 은 640 x 480 저장 로케이션들을 포함할 수도 있다. 디스플레이 버퍼 유닛 (44) 은 GPU (36) 에 의해 프로세싱되는 픽셀들의 각각에 대한 최종 픽셀 값들을 저장할 수도 있다. 디스플레이 유닛 (42) 은 디스플레이 버퍼 유닛 (44) 으로부터 최종 픽셀 값들을 취출할 수도 있고, 디스플레이 버퍼 유닛 (44) 에 저장된 픽셀 값들에 기초하여 최종 이미지를 디스플레이할 수도 있다.

[0030] 사용자 인터페이스 유닛 (46) 은, 사용자가 CPU (32) 와 같이, 컴퓨팅 디바이스 (30) 의 다른 유닛들과 통신하기 위하여 상호작용할 수도 있거나 이와 다르게 인터페이스할 수도 있도록 하는 유닛을 나타낸다. 사용자 인터페이스 유닛 (46) 의 예들은 트랙볼, 마우스, 키보드, 및 다른 타입들의 입력 디바이스들을 포함하지만, 이들로 제한되지 않는다. 사용자 인터페이스 유닛 (46) 은 또한 터치 스크린일 수도 있고, 디스플레이 유닛 (42) 의 일부로서 통합될 수도 있다.

[0031] 저장 유닛 (48) 은 하나 이상의 컴퓨터-판독가능한 저장 매체들을 포함할 수도 있다. 저장 유닛 (48) 의 예들은 랜덤 액세스 메모리 (RAM), 판독전용 메모리 (ROM), 전기적 소거가능한 프로그래밍가능한 판독전용 메모리 (EEPROM), CD-ROM 또는 다른 광학 디스크 저장장치, 자기 디스크 저장장치, 또는 다른 자기 저장 디바이스들, 플래시 메모리, 또는 컴퓨터 또는 프로세서에 의해 액세스될 수 있는 명령들 또는 데이터 구조들의 형태로 희망하는 프로그램 코드를 저장하기 위하여 이용될 수 있는 임의의 다른 매체를 포함하지만, 이것으로 제한되지 않는다.

[0032] 일부의 예의 구현들에서, 저장 유닛 (48) 은 CPU (32) 및/또는 GPU (36) 로 하여금 이 개시물에서 CPU (32) 및 GPU (36) 에 부여된 기능들을 수행하게 하는 명령들을 포함할 수도 있다. 일부의 예들에서, 저장 유닛 (48) 은 비-일시적인 저장 매체로서 고려될 수도 있다. 용어 "비-일시적인 (non-transitory)" 은 저장 매체가 반송파 또는 전파된 신호에서 구현되지는 않음을 표시할 수도 있다. 그러나, 용어 "비-일시적인" 은 저장 유닛 (48) 이 비-이동가능한 (non-movable) 것임을 의미하도록 해독되지 않아야 한다. 하나의 예로서, 저장

유닛 (48) 은 컴퓨팅 디바이스 (34) 로부터 제거될 수도 있고, 또 다른 디바이스로 이동될 수도 있다. 또 다른 예에서, 저장 유닛 (48) 과 실질적으로 유사한 저장 유닛은 컴퓨팅 디바이스 (30) 내로 삽입될 수도 있다.

어떤 예들에서는, 비-일시적인 저장 매체가 (예를 들어, RAM 에서) 시간에 걸쳐 변화할 수 있는 데이터를 저장할 수도 있다.

[0033] 도 1 의 예에서 예시된 바와 같이, 저장 유닛 (48) 은 GPU 드라이버 (50) 및 컴파일러 (54), GPU 프로그램 (52), 및 국소적으로 컴파일링된 GPU 프로그램 (56) 을 저장한다. GPU 드라이버 (50) 는 GPU (36) 를 액세스하기 위한 인터페이스를 제공하는 컴퓨터 프로그램 또는 실행가능한 코드를 나타낸다. CPU (32) 는 GPU (36) 와 인터페이스하기 위하여 GPU 드라이버 (50) 또는 그 일부분들을 실행하고, 이 이유로, GPU 드라이버 (50) 는 CPU (32) 내에서 "GPU 드라이버 (50)" 로 라벨이 붙여진 점선 박스로서 도 1 의 예에 도시되어 있다.

GPU 드라이버 (50) 는 GPU 프로그램 (52) 을 포함하는, CPU (32) 에 의해 실행되는 프로그램들 또는 다른 실행가능물들에 의해 액세스가능하다.

[0034] GPU 프로그램 (52) 은 예를 들어, 애플리케이션 프로그래밍 인터페이스 (API) 를 이용하여 하이 레벨 (HL) 프로그래밍 언어로 기록된 코드를 포함할 수도 있다. API 들의 예들은 Microsoft, Inc 에 의해 개발된 바와 같이, 개방형 컴퓨팅 언어 ("Open-Computing Language; OpenCL"), 개방형 그래픽스 라이브러리 ("Open Graphics Library; OpenGL"), 및 DirectX 를 포함한다. 일반적으로, API 는 연관된 하드웨어에 의해 실행되는 커맨드들의 미리 결정되고 표준화된 세트를 포함한다. API 커맨드들은 사용자가 GPU 의 하드웨어 컴포넌트들에게 하드웨어 컴포넌트들의 특정 사항들에 대한 사용자 지식 없이 커맨드들을 실행할 것을 명령하게 한다.

[0035] GPU 프로그램 (52) 은 GPU 드라이버 (50) 에 의해 제공된 하나 이상의 기능들을 호출하거나 이와 다르게 이를 포함할 수도 있다. CPU (32) 는 일반적으로, GPU 프로그램 (52) 이 내장되고, GPU 프로그램과 조우할 시에, GPU 프로그램 (52) 을 (예를 들어, 커맨드 스트림의 형태로) GPU 드라이버 (50) 로 전달하는 프로그램을 실행한다. CPU (32) 는 GPU 프로그램 (52) 을 프로세싱하기 위하여 이 상황에서 GPU 드라이버 (50) 를 실행한다.

즉, 예를 들어, GPU 드라이버 (50) 는 GPU 프로그램 (52) 을 GPU (36) 에 의해 실행가능한 오브젝트 또는 머신 코드로 컴파일링함으로써 GPU 프로그램 (52) 을 프로세싱할 수도 있다. 이 오브젝트 코드는 국소적으로 컴파일링된 GPU 프로그램 (56) 으로서 도 1 의 예에서 도시되어 있다.

[0036] 일부의 예들에서, 컴파일러 (54) 는 GPU 프로그램 (52) 이 내장되어 있는 프로그램의 실행 동안에 실시간으로 또는 실시간에 근접하여 GPU 프로그램 (52) 을 컴파일링하도록 동작할 수도 있다. 예를 들어, 컴파일러 (54) 는 일반적으로, HL 프로그래밍 언어에 따라 정의된 HL 명령들을 LL 프로그래밍 언어의 로우-레벨 (LL) 명령들로 감소시키는 모듈을 나타낸다. 컴파일링 후에, 이 LL 명령들은 FPGA 들, ASIC 들, 등과 같이, 특정 타입들의 프로세서들 또는 다른 타입들의 하드웨어 (예를 들어, CPU (32) 및 GPU (36) 를 포함함) 에 의해 실행될 수 있다.

[0037] LL 프로그래밍 언어들은, 이들이 프로세서 또는 다른 타입들의 하드웨어의 명령 세트 아키텍처로부터 적은 추상화 (abstraction) 또는 더 낮은 레벨의 추상화를 제공한다는 의미에서 로우 레벨로 고려된다. LL 언어들은 일반적으로 어셈블리 및/또는 머신 언어들을 지칭한다. 어셈블리 언어들은 머신 언어들보다 약간 더 높은 LL 언어이지만, 일반적으로, 어셈블리 언어들은 컴파일러 또는 다른 변환 모듈의 이용 없이 머신 언어들로 변환될 수 있다. 머신 언어들은 x86 머신 코드 (여기서, x86 은 Intel 주식회사에 의해 개발된 x86 프로세서의 명령 세트 아키텍처를 지칭함) 와 같이, 기초적인 하드웨어, 예를 들어, 프로세서에 의해 선천적으로 실행되는 명령들과, 동일하지 않을 경우에는, 유사한 명령들을 정의하는 임의의 언어를 나타낸다.

[0038] 어떤 경우에도, 컴파일러 (54) 는 HL 프로그래밍 언어에 따라 정의된 HL 명령들을 기초적인 하드웨어에 의해 지원되는 LL 명령들로 변환할 수도 있다. 컴파일러 (54) 는, 이 HL 프로그래밍 언어들에 따라 정의된 소프트웨어가 실제의 기초적인 하드웨어에 의해 더욱 직접적으로 실행될 수 있도록 HL 프로그래밍 언어들 (및 API 들) 과 연관된 추상화를 제거한다.

[0039] 도 1 의 예에서, 컴파일러 (54) 는, GPU 프로그램 (52) 을 포함하는 HL 코드를 실행할 때, CPU (32) 로부터 GPU 프로그램 (52) 을 수신할 수도 있다. 컴파일러 (54) 는 LL 프로그래밍 언어를 준수하는 국소적으로 컴파일링된 GPU 프로그램 (56) 을 발생시키기 위하여 GPU 프로그램 (52) 을 컴파일링할 수도 있다. 다음으로, 컴파일러 (54) 는 LL 명령들을 포함하는 국소적으로 컴파일링된 GPU 프로그램 (56) 을 출력한다.

[0040] GPU (36) 는 일반적으로 (GPU (36) 내에서 "국소적으로 컴파일링된 GPU 프로그램 (56)" 으로 라벨이 붙여진 점선 박스에 의해 도시된 바와 같이) 국소적으로 컴파일링된 GPU 프로그램 (56) 을 수신하고, 그 때문에, 일부의

인스턴스들에서, GPU (36) 는 이미지를 렌더링하고, 이미지의 렌더링된 부분들을 디스플레이 버퍼 유닛 (44) 으로 출력한다. 예를 들어, GPU (36) 는 디스플레이 유닛 (42) 에서 디스플레이될 다수의 프리미티브 (primitive) 들을 발생시킬 수도 있다. 프리미티브들은 라인 (곡선들, 스플라인 (spline) 들, 등을 포함함), 포인트, 원, 타원, 다각형 (여기서, 전형적으로 다각형은 하나 이상의 삼각형들의 집합으로서 정의됨), 또는 임의의 다른 2 차원 (2D) 프리미티브 중의 하나 이상을 포함할 수도 있다. 용어 "프리미티브" 는 정육면체들, 원통형, 구, 원뿔, 피라미드, 토러스 (torus), 등과 같은 3 차원 (3D) 프리미티브들을 또한 지칭할 수도 있다. 일반적으로, 용어 "프리미티브" 는 디스플레이 유닛 (42) 을 통해 이미지 (또는 비디오 데이터의 상황에서는 프레임) 로서 디스플레이하기 위하여 GPU (36) 에 의해 렌더링될 수 있는 임의의 기본적인 기하학적 형상 또는 엘리먼트를 지칭한다.

[0041] GPU (36) 는 하나 이상의 모델 변환들 (상태 데이터에서 또한 특정될 수도 있음) 을 적용함으로써 프리미티브들 및 프리미티브들의 다른 상태 데이터 (예를 들어, 컬러, 텍스처, 조명, 카메라 구성, 또는 다른 양태를 정의함) 를 소위 "월드 공간 (world space)" 으로 변환할 수도 있다. 일단 변환되면, GPU (36) 는 프리미티브들 및 라이트 (light) 들의 좌표들을 카메라 또는 눈 공간으로 변환하기 위하여 활성 카메라에 대한 뷰 변환 (또한, 카메라를 정의하는 상태 데이터에서 다시 특정될 수도 있음) 을 적용할 수도 있다. GPU (36) 는 임의의 활성 라이트들을 고려하여 프리미티브들의 외관을 렌더링하기 위하여 버텍스 쉐이딩을 또한 수행할 수도 있다. (버텍스 쉐이딩은 월드 공간에서 통상적으로 수행되지만) GPU (36) 는 상기 모델, 월드 또는 뷰 공간 중의 하나 이상에서 버텍스 쉐이딩을 수행할 수도 있다.

[0042] 일단 프리미티브들이 쉐이딩되면, GPU (36) 는 이미지를, 하나의 예로서, (-1, -1, -1) 및 (1, 1, 1) 에서 극단적인 포인트들을 갖는 단위 정육면체로 투영하기 위하여 투영들을 수행할 수도 있다. 이 단위 정육면체는 통상적으로 정규 뷰 볼륨 (canonical view volume) 이라고 지칭된다. 모델을 눈 공간으로부터 정규 뷰 볼륨으로 변환한 후, GPU (36) 는 뷰 볼륨 내에 적어도 부분적으로 존재하지 않는 임의의 프리미티브들을 제거하기 위하여 클립핑을 수행할 수도 있다. 다시 말해서, GPU (36) 는 카메라의 프레임 내에 있지 않은 임의의 프리미티브들을 제거할 수도 있다. 다음으로, GPU (36) 는 프리미티브들의 좌표들을 뷰 볼륨으로부터 스크린 공간으로 맵핑할 수도 있고, 이것은 프리미티브들의 3D 좌표들을 스크린의 2D 좌표들로 효과적으로 감소시킬 수도 있다.

[0043] 그 연관된 쉐이딩 데이터로 프리미티브들을 정의하는 변환 및 투영된 버텍스들을 고려하면, 다음으로, GPU (36) 는 프리미티브들을 래스터라이징 (rasterizing) 할 수도 있다. 예를 들어, GPU (36) 는 프리미티브들에 의해 커버되는 스크린의 픽셀들에 대한 컬러들을 컴퓨팅 및 세팅할 수도 있다. 래스터라이제이션 (rasterization) 동안, GPU (36) 는 프리미티브들과 연관된 임의의 텍스처들을 적용할 수도 있다 (여기서, 텍스처들은 상태 데이터를 포함할 수도 있음). GPU (36) 는 또한, 프리미티브들 및/또는 오브젝트들 중의 임의의 것이 임의의 다른 오브젝트들에 의해 가려지는지 여부를 결정하기 위하여, 래스터라이제이션 동안에 심도 테스트 라고 또한 지칭되는 Z-버퍼 알고리즘을 수행할 수도 있다. Z-버퍼 알고리즘은 그 심도에 따라 프리미티브들을 정렬하여, GPU (36) 는 각각의 프리미티브를 스크린으로 드로우 (draw) 하기 위한 순서를 안다. GPU (36) 는 렌더링된 픽셀들을 디스플레이 버퍼 유닛 (44) 으로 출력한다.

[0044] 디스플레이 버퍼 유닛 (44) 은 전체 이미지가 렌더링될 때까지 렌더링된 이미지의 렌더링된 픽셀들을 임시로 저장할 수도 있다. 디스플레이 버퍼 유닛 (44) 은 이 상황에서 이미지 프레임 버퍼로서 고려될 수도 있다. 다음으로, 디스플레이 버퍼 유닛 (44) 은 디스플레이 유닛 (42) 상에서 디스플레이될 렌더링된 이미지를 송신할 수도 있다. 일부의 대안적인 예들에서, GPU (36) 는 이미지를 디스플레이 버퍼 유닛 (44) 에 임시로 저장하는 것이 아니라, 디스플레이를 위하여 이미지의 렌더링된 부분들을 디스플레이 유닛 (42) 으로 직접 출력할 수도 있다. 다음으로, 디스플레이 유닛 (42) 은 디스플레이 버퍼 유닛 (44) 에 저장된 이미지를 디스플레이할 수도 있다.

[0045] 위에서 설명된 방식으로 픽셀들을 렌더링하기 위하여, GPU (36) 는 (예를 들어, 도 2 및 도 8 에 대하여 더욱 상세하게 설명된 바와 같이) 다양한 쉐이딩 동작들 수행하도록 쉐이딩 유닛들 (40) 을 지정할 수도 있다. 그러나, 상대적으로 더 짧은 렌더링 파이프라인을 지원하도록 지정된 (GPU (36) 와 같은) 어떤 GPU 들은 확장된 렌더링 파이프라인들을 갖는 API 들을 지원하지 못할 수도 있다. 예를 들어, 일부의 GPU 들은 2 개를 초과하는 상이한 타입들의 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들 (40) 을 지정하는 것이 방지될 수도 있다.

[0046] 일 예에서, GPU (36) 는 버텍스 쉐이딩 및 픽셀 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들 (40) 을 지정할 수도 있다. 이 예에서, GPU (36) 는 헐 쉐이더, 도메인 쉐이더, 및/또는 지오메트리 쉐이더와 연관된 동작들

을 수행하도록 쉐이딩 유닛들 (40) 을 지정하기 위한 자원들을 결여할 수도 있다. 즉, 하드웨어 및/또는 소프트웨어 한정들은 GPU (36) 가 헐 쉐이딩, 도메인 쉐이딩, 및/또는 지오메트리 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들 (40) 을 지정하는 것을 방지할 수도 있다. 따라서, GPU (36) 는 이러한 기능성을 포함하는 API 들과 연관된 쉐이더 스테이지들을 지원하지 못할 수도 있다.

[0047] 예를 들어, 이전의 DirectX 9 API (Direct3D 9 API 를 포함할 수도 있으며, Microsoft 에 의해 개발됨) 를 지원하였던 선행 GPU 들은 DirectX 10 API (Direct3D 10 API 를 포함할 수도 있음) 를 지원하지 못할 수도 있다. 즉, (예를 들어, 어떤 쉐이더 스테이지들과 같은) DirectX 10 API 의 특징들 중의 적어도 일부는 선행 GPU 들을 이용하여 수행되지 못할 수도 있다. 또한, 이전의 DirectX 9 API 및 DirectX 10 API 를 지원하였던 GPU 들은 DirectX 11 API 의 모든 특징들을 지원하지 못할 수도 있다. 이러한 양립불가능성 (incompatibility) 들은 DirectX 10 또는 DirectX 11 에 의존하는 소프트웨어 또는 다른 애플리케이션들을 실행하기 위한 지원을 더 이상 제공할 수도 없는 많은 수의 현재 전개된 GPU 들로 귀착될 수도 있다. 상기 예는 API 들의 Microsoft 의 DirectX 패밀리에 대하여 설명되어 있지만, 유사한 양립가능성 쟁점들은 다른 API 들 및 레거시 GPU 들 (36) 과 함께 존재할 수도 있다.

[0048] 추가적으로, 상대적으로 더 긴 그래픽스 프로세싱 파이프라인 (예를 들어, 추가적인 쉐이더 스테이지들을 갖는 렌더링 파이프라인) 을 지원하는 것은 더욱 복잡한 하드웨어 구성을 요구할 수도 있다. 예를 들어, 쉐이딩 유닛들 (40) 중의 전용된 하나에 의해 구현될 때, 지오메트리 쉐이딩을 수행하기 위하여 지오메트리 쉐이더 스테이지를 렌더링 파이프라인으로 도입하는 것은 오프-칩 메모리에 대한 추가적인 판독들 및 기록들로 귀착될 수도 있다. 즉, GPU (36) 는 초기에 쉐이딩 유닛들 (40) 중의 하나로 버텍스 쉐이딩을 수행할 수도 있고, 버텍스들을 저장 유닛 (48) 에 저장할 수도 있다. GPU (36) 는 버텍스 쉐이더에 의해 출력된 버텍스들을 또한 판독할 수도 있고, 쉐이딩 유닛들 (40) 중의 하나에 의해 지오메트리 쉐이딩을 수행할 때에 발생된 새로운 버텍스들을 기록할 수도 있다. 렌더링 파이프라인으로의 테셀레이션 스테이지들 (예를 들어, 헐 쉐이더 스테이지 및 도메인 쉐이더 스테이지) 을 포함하는 것은 이하에서 설명된 바와 같이, 유사한 복잡성들을 도입할 수도 있다.

[0049] 판독들 및 기록들이 각각 메모리 버스 및 저장 유닛 (48) 에 전력을 공급할 것을 요구한다는 것을 고려하면, 오프-칩 메모리에 대한 추가적인 판독들 및 기록들은 소비되는 전력의 양을 잠재적으로 또한 증가시키면서, 메모리 버스 대역폭 (예를 들어, GPU (36) 를 저장 유닛 (48) 에 접속하는 통신 채널) 을 소비할 수도 있다. 이러한 의미에서, 각각의 쉐이더 스테이지에 대한 전용 쉐이딩 유닛들 (40) 을 이용하여 많은 스테이지들을 갖는 그래픽스 파이프라인을 구현하는 것은 더 적게 전력 효율적인 GPU 들로 귀착될 수도 있다. 추가적으로, 이러한 GPU 들 (36) 은 또한, 저장 유닛 (48) 으로부터 데이터를 취출함에 있어서의 지연으로 인해 렌더링된 이미지들을 출력하는 측면에서 더 느리게 수행할 수도 있다.

[0050] 이 개시물의 양태들은 일반적으로, 쉐이딩 유닛들 (40) 중의 하나가 하나를 초과하는 쉐이딩 기능을 수행할 수도 있도록 쉐이딩 유닛들 (40) 중의 하나 이상의 쉐이딩 유닛의 기능을 병합하는 것에 관한 것이다. 예를 들어, 전형적으로, GPU (36) 는 특정한 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들 (40) 을 지정함으로써 렌더링 프로세스 (쉐이더 스테이지들을 갖는 렌더링 파이프라인이라고 지칭될 수도 있음) 를 수행할 수도 있고, 여기서, 쉐이딩 유닛들 (40) 의 각각은 동일한 쉐이더의 다수의 인스턴스들을 동시에 구현할 수도 있다. 즉, GPU (36) 는 예를 들어, 버텍스 쉐이더의 256 개에 이르는 동시 인스턴스들을 지원하는 버텍스 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들 (40) 중의 하나 이상을 지정할 수도 있다. GPU (36) 는 또한, 예를 들어, 픽셀 쉐이더의 256 개에 이르는 동시 인스턴스들을 지원하는 픽셀 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들 (40) 중의 하나 이상을 지정할 수도 있다. 이 하드웨어 유닛들은, 그래픽스 프로세싱 파이프라인에서 이전의 하드웨어 유닛의 출력을 프로세싱하기 위하여 다음 지정된 하드웨어 유닛이 이용가능할 때까지, 3 개의 쉐이더들 중의 하나를 실행하는 것으로 인한 출력을 저장 유닛 (48) 과 같은 오프-칩 메모리에 저장할 수도 있다.

[0051] 이 개시물의 양태들은 특정한 하드웨어 쉐이딩 유닛들을 단수 (예를 들어, 하드웨어 쉐이딩 유닛) 로 지칭할 수도 있지만, 이러한 유닛들은 실제로, 하나 이상의 쉐이딩 유닛들 (40) (하나를 초과하는 쉐이더 프로세서) 뿐만 아니라, 쉐이딩 동작들을 수행하기 위한 GPU (36) 의 하나 이상의 다른 컴포넌트들도 포함할 수도 있다는 것을 이해해야 한다. 예를 들어, 위에서 언급된 바와 같이, GPU (36) 는 복수의 연관된 쉐이딩 유닛들 (40) 을 가질 수도 있다. GPU (36) 는 동일한 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들 (40) 중의 하나를 초과하는 것을 지정할 수도 있고, 쉐이딩 유닛들 (40) 의 각각은 쉐이딩 동작들을 병합하기 위한 이 개시물의 기술들을 수행하도록 구성된다. 일반적으로, 하드웨어 쉐이딩 유닛은 특별한 쉐이딩 동작을 수행하기 위하여, GPU

(36) 와 같은 GPU 에 의해 호출된 하드웨어 컴포넌트들의 세트를 지칭할 수도 있다.

[0052] 하나의 예에서, 이 개시물의 양태들은 단일 하드웨어 쉐이딩 유닛으로 버텍스 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행하는 것을 포함한다. 또 다른 예에서, 이 개시물의 양태들은 단일 하드웨어 쉐이딩 유닛으로 버텍스 쉐이딩 동작들 및 헐 쉐이딩 동작들을 수행하는 것을 포함한다. 또 다른 예에서, 이 개시물의 양태들은 단일 하드웨어 쉐이딩 유닛으로 도메인 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행하는 것을 포함한다. 이 개시물의 양태들은 또한, 하드웨어 쉐이딩 유닛이 쉐이딩 동작들 사이에서 천이 (transition) 하는 방식과 관련된다. 즉, 이 개시물의 양태들은 하드웨어 쉐이딩 유닛으로 제 1 쉐이딩 동작을 수행하는 것과, 동일한 하드웨어 쉐이딩 유닛으로 제 2 쉐이딩 동작을 수행하는 것 사이에서 천이하는 것과 관련된다.

[0053] 예를 들어, 이 개시물의 양태들에 따르면, GPU (36) 는 버텍스 쉐이딩 동작들을 수행하도록 지정된 쉐이딩 유닛 (40) 으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작들을 수행할 수도 있다. 이 예에서, 쉐이딩 유닛 (40) 은 단일 버텍스를 입력으로서 수신하며 단일 버텍스를 출력으로서 발생시키는 인터페이스로 구성될 수도 있다. 추가적으로, GPU (36) 는 동일한 쉐이딩 유닛 (40) 으로, 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 기초하여 하나 이상의 새로운 버텍스들을 발생시키기 위하여 지오메트리 쉐이딩 동작을 수행할 수도 있다. 지오메트리 쉐이딩 동작은 하나 이상의 새로운 버텍스들을 출력하기 위하여, 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대해 작용할 수도 있다. 다시, 단일 쉐이딩 유닛 (40) 에 대해 설명되었지만, 이 기술들은 GPU (36) 의 복수의 쉐이딩 유닛들 (40) 에 의해 동시에 구현될 수도 있다.

[0054] 어떤 API 들은, 버텍스 쉐이딩 동작들을 수행하도록 지정된 쉐이딩 유닛 (40) 이 1:1 인터페이스를 구현하거나 고수할 것을 요구할 수도 있고, 이 1:1 인터페이스는 단일 버텍스를 입력으로서 수신하며 단일 버텍스를 출력으로서 발생시킨다. 대조적으로, 지오메트리 쉐이딩 동작들을 수행하도록 지정된 쉐이딩 유닛 (40) 은 1:N 인터페이스를 구현하거나 고수할 수도 있고, 이 1:N 인터페이스는 하나 이상의 버텍스들을 입력으로서 수신하며 하나 이상의 (그리고 종종 다수, 이에 따라, 상기 "N" 의 이용) 버텍스들을 출력들로서 발생시킨다.

[0055] 이 개시물의 양태들에 따르면, GPU (36) 는 지오메트리 쉐이더 프로그램의 다수의 인스턴스들을 호출함으로써 이 1:N 지오메트리 쉐이더 인터페이스를 에뮬레이팅하기 위하여 버텍스 쉐이딩 동작들을 수행하도록 지정된 쉐이딩 유닛 (40) 의 1:1 인터페이스를 레버리지 (leverage) 로 활용할 수도 있다. GPU (36) 는 지오메트리 쉐이더 동작을 수행하는 것으로부터 기인하는 새로운 버텍스들 중의 하나를 발생시키기 위하여 이 지오메트리 쉐이더 프로그램들의 각각을 동시에 실행할 수도 있다. 즉, 쉐이딩 유닛들 (40) 이 "쉐이더 프로그램 (shader program)" 이라고 통상적으로 지칭되는 것의 다수의 인스턴스들을 동시에 실행할 수도 있도록, 쉐이딩 유닛들 (40) 은 HLSL 을 이용하여 (예를 들어, 그래픽스 렌더링 API 로) 프로그래밍가능할 수도 있다. 이 쉐이더 프로그램들은 "파이버들" 또는 "스레드들" 이라고 지칭될 수도 있다 (이 둘 모두는 프로그램 또는 실행의 스레드를 형성하는 명령들의 스트림을 지칭할 수도 있음). 이 개시물의 양태들에 따르면, 그리고 이하에서 더욱 상세하게 설명된 바와 같이, GPU (36) 는 버텍스 쉐이딩 동작들을 위해 지정된 하드웨어 쉐이딩 유닛을 이용하여 지오메트리 쉐이더 프로그램의 다수의 인스턴스들을 실행할 수도 있다. GPU (36) 는 지오메트리 쉐이더 명령들을 버텍스 쉐이더 명령들에 첨부할 수도 있어서, 동일한 쉐이딩 유닛 (40) 은 두 쉐이더들, 예를 들어, 버텍스 쉐이더 및 지오메트리 쉐이더를 순차적으로 실행한다.

[0056] 또 다른 예에서, 이 개시물의 양태들에 따르면, GPU (36) 는 버텍스 쉐이딩 동작들을 수행하도록 지정된 하드웨어 쉐이딩 유닛으로, 버텍스 쉐이딩된 버텍스들을 출력하도록 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작들을 수행할 수도 있다. 하드웨어 쉐이딩 유닛은, 단일 버텍스를 입력으로서 수신하며 단일 버텍스를 출력으로서 발생시키는 인터페이스를 고수할 수도 있다. 추가적으로, GPU 는 버텍스 쉐이딩 동작들을 수행하기 위해 지정된 동일한 하드웨어 쉐이딩 유닛으로, 버텍스 쉐이딩된 버텍스들 중의 하나 이상에 기초하여 하나 이상의 새로운 버텍스들을 발생시키기 위하여 하나 이상의 테셀레이션 동작들 (예를 들어, 헐 쉐이딩 동작들 및/또는 도메인 쉐이딩 동작들) 을 수행할 수도 있다. 하나 이상의 테셀레이션 동작들은 하나 이상의 새로운 버텍스들을 출력하기 위하여 하나 이상의 버텍스 쉐이딩된 버텍스들 중의 적어도 하나에 대해 작용할 수도 있다.

[0057] 예를 들어, 위에서 설명된 쉐이더 스테이지들에 추가하여, 일부의 그래픽스 렌더링 파이프라인들은 헐 쉐이더 스테이지, 테셀레이터 스테이지, 및 도메인 쉐이더 스테이지를 또한 포함할 수도 있다. 일반적으로, 헐 쉐이더 스테이지, 테셀레이터 스테이지, 및 도메인 쉐이더 스테이지는 하드웨어 테셀레이션을 수용하도록 포함된다. 즉, 헐 쉐이더 스테이지, 테셀레이터 스테이지, 및 도메인 쉐이더 스테이지는 예를 들어, CPU (32) 에

의해 실행되는 소프트웨어 애플리케이션에 의해 실행되는 것이 아니라, GPU (36) 에 의한 테셀레이션을 수용하도록 포함된다.

[0058] 이 개시물의 양태들에 따르면, GPU (36) 는 동일한 쉐이딩 유닛 (40) 으로 버텍스 쉐이딩 및 테셀레이션 동작들을 수행할 수도 있다. 예를 들어, GPU (36) 는 버텍스 쉐이딩 및 테셀레이션 동작들을 2 개의 패스들에서 수행할 수도 있다. 이하에서 더욱 상세하게 설명되는 이 개시물의 양태들에 따르면, GPU (36) 는 상이한 쉐이딩 동작들 사이의 천이들을 가능하게 하기 위하여 다양한 값들을 저장할 수도 있다.

[0059] 일 예에서, 제 1 패스에서는, GPU (36) 가 버텍스 쉐이딩 및 헐 쉐이딩 동작들을 수행하도록 하나 이상의 쉐이딩 유닛들 (40) 을 지정할 수도 있다. 이 예에서, GPU (36) 는 헐 쉐이더 명령들을 버텍스 쉐이더 명령들에 첨부할 수도 있다. 따라서, 동일한 쉐이딩 유닛 (40) 은 버텍스 쉐이딩 및 헐 쉐이더 명령들을 순차적으로 실행한다.

[0060] 제 2 패스에서, GPU (36) 는 도메인 쉐이딩 및 지오메트리 쉐이딩 동작들을 수행하도록 하나 이상의 쉐이딩 유닛들 (40) 을 지정할 수도 있다. 이 예에서, GPU (36) 는 도메인 쉐이더 명령들을 지오메트리 쉐이더 명령들에 첨부할 수도 있다. 따라서, 동일한 쉐이딩 유닛 (40) 은 도메인 쉐이딩 및 지오메트리 쉐이딩 동작들을 순차적으로 실행한다. 다수의 쉐이딩 동작들을 다수의 패스들에서 수행함으로써, GPU (36) 는 추가적인 쉐이딩 능력들을 갖는 GPU 를 에뮬레이팅하기 위하여 동일한 쉐이딩 하드웨어를 이용할 수도 있다.

[0061] 이 개시물의 양태들은 또한 GPU (36) 가 쉐이딩 동작들 사이에서 천이하는 방식에 관한 것이다. 예를 들어, 이 개시물의 양태들은, 쉐이딩 동작들이 함께 패치되어 동작들이 동일한 하드웨어 쉐이딩 유닛에 의해 순차적으로 실행되는 방식에 관한 것이다.

[0062] 일 예에서, 이 개시물의 양태들에 따르면, GPU (36) 는 렌더링 파이프라인의 제 1 쉐이더 스테이지와 연관된 제 1 쉐이딩 동작들을 수행하도록 하나 이상의 쉐이딩 유닛들 (40) 을 지정할 수도 있다. GPU (36) 는 제 1 쉐이딩 동작들의 완료 시에 쉐이딩 유닛 (40) 의 동작 모드들을 스위칭할 수도 있다. 다음으로, GPU (36) 는 제 1 쉐이딩 동작들을 수행하도록 지정된 동일한 쉐이딩 유닛 (40) 으로, 렌더링 파이프라인의 제 2 상이한 쉐이더 스테이지와 연관된 제 2 쉐이딩 동작들을 수행할 수도 있다.

[0063] 일부의 예들에 따르면, GPU (36) 는 복수의 모드들을 이용하여 쉐이딩 동작들을 함께 패치할 수도 있고, 각각의 모드는 연관된 쉐이딩 동작들의 특별한 세트를 가진다. 예를 들어, 제 1 모드는 드로우 콜 (draw call) 이 버텍스 쉐이딩 동작들만을 포함한다는 것을 표시할 수도 있다. 이 예에서, 드로우 콜을 실행 시에, GPU (36) 는 모드 정보에 따라 버텍스 쉐이딩 동작들을 수행하도록 하나 이상의 쉐이딩 유닛들 (40) 을 지정할 수도 있다. 추가적으로, 제 2 모드는 드로우 콜이 버텍스 쉐이딩 및 지오메트리 쉐이딩 동작들의 둘 모두를 포함한다는 것을 표시할 수도 있다. 이 예에서, 드로우 콜을 실행 시에, GPU (36) 는 버텍스 쉐이딩 동작들을 수행하도록 하나 이상의 쉐이딩 유닛들 (40) 을 지정할 수도 있다. 추가적으로, 이 개시물의 양태들에 따르면, GPU (36) 는 지오메트리 쉐이더 명령들을 버텍스 쉐이더 명령들에 첨부할 수도 있어서, 동일한 쉐이딩 유닛들은 버텍스 및 지오메트리 쉐이딩 동작들의 둘 모두를 실행한다. 추가적인 모드들은 이하에서 더욱 상세하게 설명되는 바와 같이, 쉐이더들의 다른 조합들을 표시하기 위하여 이용될 수도 있다.

[0064] 일부의 예들에서, GPU 드라이버 (50) 는 GPU (36) 에 의해 이용되는 모드 정보를 발생시킬 수도 있다. 이 개시물의 양태들에 따르면, 상이한 쉐이더들 (예를 들어, 버텍스 쉐이딩 동작들, 지오메트리 쉐이딩 동작들, 헐 쉐이딩 동작들, 도메인 쉐이딩 동작들, 등) 은 동일한 쉐이딩 유닛 (40) 에 의해 순차적으로 실행되도록 하기 위하여 특별한 방식으로 컴파일링되지 않아도 된다. 오히려, 각각의 쉐이더는 (임의의 다른 쉐이더를 참조하지 않고) 독립적으로 컴파일링될 수도 있고, GPU (36) 에 의해 드로우 시간에 함께 패치될 수도 있다. 즉, 드로우 콜을 실행 시에, GPU (36) 는 드로우 콜과 연관된 모드를 결정할 수도 있고, 컴파일링된 쉐이더들을 이에 따라 함께 패치할 수도 있다.

[0065] 이 개시물의 기술들은 쉐이딩 동작들을 수행하기 위한 제한된 수의 쉐이딩 유닛들 (40) 을 갖는 (GPU (36) 와 같은) GPU 가 더 큰 수의 쉐이딩 유닛들 (40) 을 갖는 GPU 를 에뮬레이팅하는 것을 가능하게 할 수도 있다. 예를 들어, GPU (36) 가 2 개를 초과하는 쉐이딩 동작들 (예를 들어, 버텍스 쉐이딩 동작들 및 픽셀 쉐이딩 동작들) 을 수행하도록 쉐이딩 유닛들 (40) 을 지정하는 것이 방지될 수도 있지만, 이 개시물의 기술들은 GPU (36) 가 쉐이딩 유닛들 (40) 을 재구성하지 않고도 추가적인 쉐이딩 동작들 (예를 들어, 지오메트리 쉐이딩 동작들, 헐 쉐이딩 동작들, 및/또는 도메인 쉐이딩 동작들) 을 수행하는 것을 가능하게 할 수도 있다. 즉, 기술들은 쉐이딩 유닛들 (40) 이 다른 쉐이딩 동작들을 수행하면서, 어떤 쉐이더 스테이지들의 입력/출력 제약들

도 고수하도록 할 수도 있다.

[0066] 또한, 동일한 쉐이딩 유닛들 (40) 로 다수의 쉐이딩 동작들을 수행함으로써, 기술들은 메모리 버스 대역폭 소비를 감소시킬 수도 있다. 예를 들어, 버텍스 쉐이딩이 다른 쉐이딩 동작들 (예를 들어, 지오메트리 쉐이딩) 과 함께 수행될 경우, 버텍스 쉐이딩을 위해 이용된 쉐이딩 유닛들 (40) 은 다른 쉐이더 동작들을 수행하기 전에, 버텍스 쉐이딩 결과들을 (저장 유닛 (48) 과 같은) 오프-칩 메모리에 저장할 필요가 없다. 오히려, 버텍스 쉐이딩 결과들은 GPU 메모리 (38) 에 저장될 수도 있고, 즉시, 지오메트리 쉐이딩 동작들을 위해 이용될 수도 있다.

[0067] 이러한 방식으로, 기술들은 추가적인 쉐이딩 유닛들 (40) 을 갖는 GPU 들에 비해 메모리 버스 대역폭 소비를 감소시킬 수도 있고, 이것은 전력 소비를 감소시킬 수도 있다. 그러므로, 기술들은 추가적인 하드웨어 쉐이더 유닛들을 갖는 GPU 들보다 더 작은 전력을 사용하는 더 많이 전력 효율적인 GPU 들을 촉진시킬 수도 있다. 따라서, 일부의 예들에서, 기술들은 이동 디바이스들, 랩톱 컴퓨터들, 및 전력의 일정한 전용 공급을 가지지 않는 임의의 다른 타입의 디바이스와 같은 전력-제한된 디바이스들에서 전개될 수도 있다.

[0068] 컴퓨팅 디바이스 (30) 는 명확함의 목적들을 위하여 도 1 에 도시되지 않은 추가적인 모듈들 또는 유닛들을 포함할 수도 있다. 예를 들어, 컴퓨팅 디바이스 (30) 는 데이터를 송신 및 수신하기 위한 트랜시버 모듈을 포함할 수도 있고, 컴퓨팅 디바이스 (30) 와 또 다른 디바이스 또는 네트워크 사이의 무선 또는 유선 통신을 허용하기 위한 회로부를 포함할 수도 있다. 컴퓨팅 디바이스 (30) 는 또한, 컴퓨팅 디바이스 (30) 가 이동 무선 전화인 예들에서는 전화 통신들, 또는 컴퓨팅 디바이스 (30) 가 미디어 플레이어인 경우에는 스피커를 달성하기 위하여, 어느 것도 도 1 에 도시되지 않은 스피커 및 마이크로폰을 포함할 수도 있다. 일부의 인스턴스들에서, 사용자 인터페이스 유닛 (46) 및 디스플레이 유닛 (42) 은, 컴퓨팅 디바이스 (30) 가 외부의 사용자 인터페이스 또는 디스플레이와 인터페이싱하도록 구비되는 데스크톱 컴퓨터 또는 다른 디바이스인 예들에서는 컴퓨팅 디바이스 (30) 의 외부에 있을 수도 있다.

[0069] 도 2 는 예시적인 그래픽스 프로세싱 파이프라인 (80) 을 예시하는 블록도이다. 일 예의 파이프라인 (80) 은 입력 어셈블러 스테이지 (82), 버텍스 쉐이더 스테이지 (84), 지오메트리 쉐이더 스테이지 (86), 래스터라이저 (rasterizer) 스테이지 (88), 픽셀 쉐이더 스테이지 (90), 및 출력 병합기 (output merger) 스테이지 (92) 를 포함한다. 일부의 예들에서, DirectX 10 (또는 Direct3D 10) API 와 같은 API 는 도 2 에 도시된 스테이지들의 각각을 이용하도록 구성될 수도 있다. 그래픽스 프로세싱 파이프라인 (80) 은 GPU (36) 에 의해 수행되는 것으로서 이하에서 설명되지만, 다양한 다른 그래픽스 프로세서들에 의해 수행될 수도 있다.

[0070] 그래픽스 프로세싱 파이프라인 (80) 은 일반적으로 프로그래밍가능한 스테이지들 (예를 들어, 곡선의 모서리들로 예시됨) 및 고정된 기능 스테이지들 (예를 들어, 직각의 모서리들로 예시됨) 을 포함한다. 예를 들어, 그래픽스 렌더링 파이프라인 (80) 의 어떤 스테이지들과 연관된 그래픽스 렌더링 동작들은 쉐이딩 유닛들 (40) 중의 하나와 같은 프로그래밍가능한 쉐이더 프로세서에 의해 일반적으로 수행되는 반면, 그래픽스 렌더링 파이프라인 (80) 의 다른 스테이지들과 연관된 다른 그래픽스 렌더링 동작들은 GPU (36) 와 연관된 비-프로그래밍가능한 고정된 기능 하드웨어 유닛들에 의해 일반적으로 수행된다. 쉐이딩 유닛들 (40) 에 의해 수행되는 그래픽스 렌더링 스테이지들은 "프로그래밍가능한" 스테이지들이라고 일반적으로 지칭될 수도 있는 반면, 고정된 기능 유닛들에 의해 수행되는 스테이지들은 고정된 기능 스테이지들이라고 일반적으로 지칭될 수도 있다.

[0071] 입력 어셈블러 스테이지 (82) 는 고정된 기능 스테이지로서 도 2 의 예에서 도시되어 있고, 그래픽스 데이터 (삼각형들, 라인들 및 포인트들) 를 그래픽스 프로세싱 파이프라인 (80) 에 공급하는 것을 일반적으로 담당한다. 예를 들어, 입력 어셈블러 스테이지 (82) 는 높은 차수의 표면들, 프리미티브들, 등에 대한 버텍스 데이터를 수집할 수도 있고, 버텍스 데이터 및 속성들을 버텍스 쉐이더 스테이지 스테이지 (84) 로 출력할 수도 있다. 따라서, 입력 어셈블러 스테이지 (82) 는 고정된 기능 동작들을 이용하여 저장 유닛 (48) 과 같은 오프-칩 메모리로부터 버텍스들을 판독할 수도 있다. 다음으로, 입력 어셈블러 스테이지 (82) 는 버텍스 식별자들 ("VertexIDs"), 인스턴스 식별자들 (버텍스 쉐이더에 의해 이용가능하게 되는 "InstanceIDs") 및 프리미티브 식별자들 (지오메트리 쉐이더 및 픽셀 쉐이더에 의해 이용가능한 "PrimativeIDs") 을 또한 발생시키면서, 이 버텍스들로부터 파이프라인 작업 항목들을 생성할 수도 있다. 입력 어셈블러 스테이지 (82) 는 버텍스들을 판독할 시에 VertexID 들, InstanceID 들, 및 PrimitiveID 들을 자동으로 발생시킬 수도 있다.

[0072] 버텍스 쉐이더 스테이지 (84) 는 수신된 버텍스 데이터 및 속성들을 프로세싱할 수도 있다. 예를 들어, 버텍스 쉐이더 스테이지 (84) 는 변환들, 스키닝 (skinning), 버텍스 변위, 및 버텍스 당 (per-vertex) 재료 속성

들의 계산과 같은 버텍스 당 프로세싱을 수행할 수도 있다. 일부의 예들에서, 버텍스 쉐이더 스테이지 (84) 는 텍스처 좌표들, 버텍스 컬러, 버텍스 조명, 안개 인자 (fog factor) 들, 등을 발생시킬 수도 있다. 버텍스 쉐이더 스테이지 (84) 는 일반적으로 단일 입력 버텍스를 취하고, 단일의 프로세싱된 출력 버텍스를 출력한다.

[0073] 지오메트리 쉐이더 스테이지 (86) 는 버텍스 데이터 (예를 들어, 삼각형에 대한 3 개의 버텍스들, 라인에 대한 2 개의 버텍스들, 또는 포인트에 대한 단일 버텍스) 에 의해 정의된 프리미티브를 수신할 수도 있고, 프리미티브를 더욱 프로세싱할 수도 있다. 예를 들어, 지오메트리 쉐이더 스테이지 (86) 는 다른 가능한 프로세싱 동작들 중에서, 실루엣-에지 (silhouette-edge) 검출 및 음영 볼륨 압출 (shadow volume extrusion) 과 같은 프리미티브 당 (per-primitive) 프로세싱을 수행할 수도 있다. 따라서, 지오메트리 쉐이더 스테이지 (86) 는 입력으로서 하나의 프리미티브 (하나 이상의 버텍스들을 포함할 수도 있음) 를 수신할 수도 있고, 제로, 하나, 또는 다수의 프리미티브들 (하나 이상의 버텍스들을 다시 포함할 수도 있음) 을 출력한다. 출력 프리미티브는 지오메트리 쉐이더 스테이지 (86) 없이 가능할 수도 있는 것보다 더 많은 데이터를 포함할 수도 있다.

출력 데이터의 총 양은 버텍스 카운트에 의해 곱해진 버텍스 사이즈와 동일할 수도 있고, 호출 마다 제한될 수도 있다. 지오메트리 쉐이더 스테이지 (86) 로부터의 스트림 출력은 이 스테이지에 도달하는 프리미티브들이 메모리 유닛 (48) 과 같은 오프-칩 메모리에 저장되도록 할 수도 있다. 스트림 출력은 전형적으로 지오메트리 쉐이더 스테이지 (86) 에 결부되고, 둘 모두는 (예를 들어, API 를 이용하여) 함께 프로그래밍될 수도 있다.

[0074] 래스터라이저 스테이지 (88) 는 전형적으로, 프리미티브들을 클립핑 (clipping) 하고 픽셀 쉐이더 스테이지 (90) 에 대한 프리미티브들을 준비하는 것을 담당하는 고정된 기능 스테이지이다. 예를 들어, 래스터라이저 스테이지 (88) 는 클립핑 (커스텀 클립 (custom clip) 경계들을 포함함), 원근 분할 (perspective divide), 뷰포트/가위 (viewport/scissor) 선택 및 구현, 렌더 타겟 선택 및 프리미티브 설정을 수행할 수도 있다. 이러한 방법으로, 래스터라이저 스테이지 (88) 는 픽셀 쉐이더 스테이지 (90) 에 의해 쉐이딩을 위한 다수의 프래그먼트들을 발생시킬 수도 있다.

[0075] 픽셀 쉐이더 스테이지 (90) 는 래스터라이저 스테이지 (88) 로부터 프래그먼트들을 수신하고, 컬러와 같은 픽셀 당 데이터를 발생시킨다. 픽셀 쉐이더 스테이지 (96) 는 또한 텍스처 배합 (texture blending) 및 조명 모델 연산과 같은 픽셀 당 프로세싱을 수행할 수도 있다. 따라서, 픽셀 쉐이더 스테이지 (90) 는 입력으로서 하나의 픽셀을 수신할 수도 있고, 동일한 상대 위치 (또는 픽셀에 대한 제로 값) 에서 하나의 픽셀을 출력할 수도 있다.

[0076] 출력 병합기 스테이지 (92) 는 일반적으로 최종 결과를 발생시키기 위하여 (픽셀 쉐이더 값들, 심도 및 스텐실 (stencil) 정보와 같은) 다양한 타입들의 출력 데이터를 조합하는 것을 담당한다. 예를 들어, 출력 병합기 스테이지 (92) 는 렌더 타겟 (픽셀 위치) 에 대한 고정된 기능 배합, 심도, 및/또는 스텐실 동작들을 수행할 수도 있다. 버텍스 쉐이더 스테이지 (84), 지오메트리 쉐이더 스테이지 (86), 및 픽셀 쉐이더 스테이지 (90) 에 대하여 일반적인 측면에서 위에서 설명되었지만, 상기한 설명의 각각은 각각의 쉐이딩 동작들을 수행하도록 GPU 에 의해 지정된 (쉐이딩 유닛들 (40) 과 같은) 하나 이상의 쉐이딩 유닛들을 지칭할 수도 있다.

[0077] 어떤 GPU 들은 도 2 에 도시된 쉐이더 스테이지들의 전부를 지원하지 못할 수도 있다. 예를 들어, 일부의 GPU 들은 하드웨어 및/또는 소프트웨어 한정들 (예를 들어, 제한된 수의 쉐이딩 유닛들 (40) 및 연관된 컴포넌트들) 로 인해, 2 개를 초과하는 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들을 지정하지 못할 수도 있다. 일 예에서, 어떤 GPU 들은 지오메트리 쉐이더 스테이지 (86) 와 연관된 동작들을 지원하지 않을 수도 있다. 오히려, GPU 들은 버텍스 쉐이더 스테이지 (84) 및 픽셀 쉐이더 스테이지 (90) 를 수행하도록 쉐이딩 유닛들을 지정하기 위한 지원을 포함하기만 할 수도 있다. 따라서, 쉐이딩 유닛들에 의해 수행된 동작들은 버텍스 쉐이더 스테이지 (84) 및 픽셀 쉐이더 스테이지 (90) 와 연관된 입력/출력 인터페이스를 고수해야 한다.

[0078] 추가적으로, 일부의 예들에서, 지오메트리 쉐이더 스테이지 (86) 를 파이프라인에 도입하는 것은, 지오메트리 쉐이더 스테이지 (86) 를 포함하지 않는 그래픽스 프로세싱 파이프라인과 관련하여, 저장 유닛 (48) 에 대한 추가적인 판독들 및 기록들로 귀착될 수도 있다. 예를 들어, 위에서 언급된 바와 같이, 버텍스 쉐이더 스테이지 (86) 는 저장 유닛 (48) 과 같은 오프-칩 메모리로 버텍스들을 기록할 수도 있다. 지오메트리 쉐이더 스테이지 (86) 는 이 버텍스들 (버텍스 쉐이더 스테이지 (84) 에 의해 출력된 버텍스들) 을 판독할 수도 있고, 이후에 픽셀 쉐이딩되는 새로운 버텍스들을 기록할 수도 있다. 저장 유닛 (48) 에 대한 이 추가적인 판독들 및 기록들은 소비되는 전력의 양을 또한 잠재적으로 증가시키면서 메모리 버스 대역폭을 소비할 수도 있다.

이러한 의미에서, 버텍스 쉐이더 스테이지 (84), 지오메트리 쉐이더 스테이지 (86), 및 픽셀 쉐이더 스테이지 (90) 의 각각을 포함하는 그래픽스 프로세싱 파이프라인을 구현하는 것은 저장 유닛 (48) 으로부터 데이터를 취출함에 있어서의 지연으로 인해 렌더링된 이미지들을 출력하는 측면에서 또한 더 느릴 수도 있는 더 적게 전력 효율적인 GPU 들로 귀착될 수도 있다.

[0079] 위에서 언급된 바와 같이, 이 개시물의 양태들은 일반적으로 쉐이딩 유닛들 (40) 중의 하나 이상의 쉐이딩 유닛의 기능을 병합하는 것에 관한 것이어서, 특별한 쉐이딩 동작을 위해 지정된 쉐이딩 유닛 (40) 은 하나를 초과하는 쉐이딩 동작을 수행할 수도 있다. 이하에서 더욱 상세하게 설명된 바와 같이, 일부의 예들에서, 하나의 쉐이딩 유닛 (40) 은 버텍스 쉐이더 스테이지 (84) 와 연관된 버텍스 쉐이딩 동작들을 수행하기 위해 지정될 수도 있다. 이 개시물의 양태들에 따르면, 동일한 쉐이딩 유닛 (40) 은 또한, 지오메트리 쉐이더 스테이지 (86) 와 연관된 지오메트리 쉐이딩 동작들을 수행하기 위하여 구현될 수도 있다. 즉, GPU (36) 는 버텍스 쉐이딩 동작들을 수행하기 위하여 쉐이딩 유닛 (40) 을 호출할 수도 있지만, 지오메트리 쉐이딩 태스크를 수행하도록 쉐이딩 유닛 (40) 을 재지정하지 않고도 지오메트리 쉐이딩 동작들을 수행하기 위한 쉐이딩 유닛 (40) 을 또한 구현할 수도 있다.

[0080] 도 3a 및 도 3b 는 이 개시물의 양태들에 따라, 그래픽스 렌더링 파이프라인에서의 데이터 흐름들의 개념적인 도면들이다. 예를 들어, 도 3a 는 버텍스 쉐이더 스테이지 (100), 지오메트리 쉐이더 스테이지 (102), 스트림 출력 (104), 및 픽셀 쉐이더 스테이지 (106) 를 예시한다. 일반적으로, 도 3a 에 도시된 버텍스 쉐이더 스테이지 (100), 지오메트리 쉐이더 스테이지 (102), 및 픽셀 쉐이더 스테이지 (106) 는 각각 쉐이딩 동작들을 수행하기 위한 연관된 하드웨어를 나타낸다. 즉, 예를 들어, 버텍스 쉐이더 스테이지 (100), 지오메트리 쉐이더 스테이지 (102), 및 픽셀 쉐이더 스테이지 (106) 의 각각은 각각의 태스크들을 수행하도록 지정된 쉐이딩 유닛들 (40) 과 같은, 별도로 지정된 프로세싱 유닛들과 연관될 수도 있다.

[0081] 예를 들어, 버텍스 쉐이더 스테이지 (100) 는 버텍스 쉐이딩 동작들을 수행하는 (쉐이딩 유닛들 (40) 과 같은) 하나 이상의 유닛들을 나타낸다. 즉, 버텍스 쉐이더 스테이지 (100) 는 버텍스 쉐이딩 동작들을 수행하기 위하여 GPU (36) 에 의해 호출되는 컴포넌트들을 포함할 수도 있다. 예를 들어, 버텍스 쉐이더 스테이지 (100) 는 입력으로서 버텍스를 수신할 수도 있고, 입력 버텍스를 3 차원 (3D) 모델 공간으로부터 스크린 공간 내의 2 차원 (2D) 좌표로 변환할 수도 있다. 다음으로, 버텍스 쉐이더 스테이지 (100) 는 버텍스의 변환된 버전 ("변환된 버텍스" 라고 지칭될 수도 있음) 을 출력할 수도 있다. 버텍스 쉐이더 스테이지 (100) 는 보통은 새로운 버텍스들을 생성하지 않지만, 한번에 하나의 버텍스에 대해 작용한다. 그 결과, 버텍스 쉐이더 스테이지 (100) 는, 버텍스 쉐이더 스테이지 (100) 가 단일 입력 버텍스를 수신하고 단일 출력 버텍스를 출력하는 일-대-일 (1:1) 스테이지라고 지칭될 수도 있다.

[0082] 지오메트리 쉐이더 스테이지 (102) 는 지오메트리 쉐이딩 동작들을 수행하는 (쉐이딩 유닛들 (40) 과 같은) 하나 이상의 유닛들을 나타낸다. 즉, 지오메트리 쉐이더 스테이지 (102) 는 지오메트리 쉐이딩 동작들을 수행하기 위하여 GPU (36) 에 의해 호출되는 컴포넌트들을 포함할 수도 있다. 예를 들어, 지오메트리 쉐이더 스테이지 (102) 는 정육면체 맵으로의 단일 패스 렌더링, 포인트 스프라이트 (point sprite) 발생, 등과 같은 폭넓게 다양한 동작들을 수행하기에 유용할 수도 있다. 전형적으로, 지오메트리 쉐이더 스테이지 (102) 는, 버텍스 쉐이더 스테이지 (100) 에 의해 버텍스 쉐이딩되었던 하나 이상의 변환된 버텍스들로 이루어진 프리미티브들을 수신한다. 지오메트리 쉐이더 스테이지 (102) 는 새로운 프리미티브들을 형성할 수도 있는 (또는 아마도 입력 프리미티브를 추가적인 새로운 버텍스들을 갖는 새로운 타입의 프리미티브로 변환함) 새로운 버텍스들을 생성하기 위하여 지오메트리 쉐이딩 동작들을 수행한다.

[0083] 예를 들어, 지오메트리 쉐이더 스테이지 (102) 는 전형적으로 하나 이상의 변환된 버텍스들에 의해 정의된 프리미티브를 수신하고, 수신된 프리미티브에 기초하여 하나 이상의 새로운 버텍스들을 발생시킨다. 다음으로, 지오메트리 쉐이더 스테이지 (102) 는 새로운 버텍스들 (하나 이상의 새로운 프리미티브들을 형성할 수도 있음) 을 출력한다. 그 결과, 지오메트리 쉐이더 스테이지 (102) 는, 지오메트리 쉐이더 스테이지 (102) 가 하나 이상의 변환된 버텍스들을 수신하고 다수의 새로운 버텍스들을 발생시킨다는 점에서, 일-대-다수 (1:N) 또는 심지어 다수-대-다수 (N:N) 스테이지라고 지칭될 수도 있다.

[0084] 일-대-다수 또는 심지어 다수-대-다수인 것으로 설명되었지만, 일부의 인스턴스들에서, 지오메트리 쉐이더 스테이지 (102) 는 또한, 임의의 새로운 버텍스들을 출력하지 않을 수도 있거나 단일의 새로운 버텍스를 출력하기만 할 수도 있다. 이 점에 있어서, 기술들은 모든 인스턴스에서 다수의 버텍스들을 출력하는 그러한 지오메트리 쉐이더들에만 제한되지 않아야 하지만, 이하에서 더욱 상세하게 설명되는 바와 같이, 제로, 하나 또는 다수

의 새로운 버텍스들을 출력할 수도 있는 임의의 지오메트리 쉐이더 스테이지 (102) 에 대하여 일반적으로 구현될 수도 있다.

[0085] 지오메트리 쉐이더 스테이지 (102) 의 출력은 (예를 들어, 스트림 출력 (104) 동안에) 추가적인 지오메트리 쉐이딩을 위해 저장될 수도 있다. 지오메트리 쉐이더 스테이지 (102) 의 출력은 또한, 픽셀들로 이루어진 래스터 이미지를 발생시키기 위하여 새로운 버텍스들 (및 변환된 버텍스들) 을 래스터라이징하는 래스터라이저로 출력될 수도 있다.

[0086] 지오메트리 쉐이더 스테이지 (102) 로부터의 픽셀들은 또한 픽셀 쉐이더 스테이지 (106) 로 전달될 수도 있다. (또한, 프래그먼트 쉐이더라고 지칭될 수도 있는) 픽셀 쉐이더 스테이지 (106) 는 각각의 픽셀의 컬러 및 다른 속성들을 컴퓨팅하여, 쉐이딩된 픽셀을 생성하기 위하여 폭넓게 다양한 동작들을 수행할 수도 있다. 쉐이딩된 픽셀들은 심도 맵과 병합될 수도 있고, 다른 포스트 쉐이딩 (post shading) 동작들은 컴퓨터 모니터, 텔레비전, 또는 다른 타입들의 디스플레이 디바이스들과 같은 디스플레이 디바이스를 통한 디스플레이를 위하여 출력 이미지를 발생시키도록 수행될 수도 있다.

[0087] 도 3a 에 도시된 쉐이더 스테이지들은 하나 이상의 그래픽스 API 들을 지원할 수도 있다. 예시의 목적들을 위한 예에서, 버텍스 쉐이더 스테이지 (100), 지오메트리 쉐이더 스테이지 (102), 및 픽셀 쉐이더 스테이지 (106) 는 DirectX 10 API 를 지원할 수도 있다. 즉, DirectX 10 API 를 이용하여 생성된 코드는 그래픽스 데이터를 렌더링하기 위하여 버텍스 쉐이더 스테이지 (100), 지오메트리 쉐이더 스테이지 (102), 및 픽셀 쉐이더 스테이지 (106) 에 의해 실행될 수도 있다. 그러나, 지오메트리 쉐이더 스테이지 (102) 는 모든 그래픽스 렌더링 파이프라인들 내에 포함되지 않을 수도 있고, 모든 GPU 들에 의해 실행가능하지 않을 수도 있다. 예를 들어, DirectX 10 API 는 지오메트리 쉐이더 스테이지 (102) 에 대한 지원을 포함하지만, 어떤 더 이전의 수정본들 (예를 들어, DirectX 9) 은 이러한 지원을 포함하지 않는다. 따라서, DirectX API 의 더 이전의 수정본들로 생성된 코드를 실행하도록 설계된 GPU 들 (또는 다른 API 들을 위해 설계된 GPU 들) 은 지오메트리 쉐이더 스테이지 (102) 를 수행하기 위한 쉐이딩 유닛들 (40) 을 지정할 수 없을 수도 있다.

[0088] 도 3b 는 이 개시물의 기술들에 따라 (도 3a 에 도시된 예와 관련하여) 그래픽스 렌더링 파이프라인에서의 데이터 흐름의 수정된 개념적인 도면을 예시한다. 도 3b 에 도시된 예는 병합된 버텍스 쉐이더/지오메트리 쉐이더 (VS/GS) 스테이지 (110), 스트림 출력 (112), 및 픽셀 쉐이더 스테이지 (114) 를 포함한다. 이 개시물의 양태들에 따르면, 병합된 VS/GS 스테이지 (110) 는 버텍스 쉐이더 스테이지 (100) 및 지오메트리 쉐이더 스테이지 (102) 에 대하여 위에서 설명된 기능들을 수행하기 위한 하나 이상의 프로세싱 유닛들을 포함할 수도 있다. 즉, 버텍스 쉐이더 스테이지 (100) 및 지오메트리 쉐이더 스테이지 (102) 는 버텍스 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 각각 수행하기 위한 (GPU (36) 와 같은) GPU 에 의해 호출된 별개의 유닛들을 나타내지만, 이 개시물의 양태들에 따르면, 이러한 기능들은 실질적으로 동일한 하드웨어 (예를 들어, 쉐이딩 유닛들 (40)) 에 의해 수행될 수도 있다.

[0089] 예를 들어, 버텍스 쉐이딩 동작들이 GPU (36) 에 의해 호출될 시에, VS/GS 스테이지 (110) 는 버텍스 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들의 둘 모두를 수행할 수도 있다. 즉, 병합된 VS/GS 스테이지 (110) 는, 버텍스 쉐이더 스테이지 (100) 에 대하여 위에서 설명된 동작들을 수행하고 지오메트리 쉐이더 스테이지 (102) 에 대하여 위에서 설명된 동작들을 수행하기 위한 쉐이딩 유닛들 (40) 의 동일한 세트를 포함할 수도 있다.

[0090] 그러나, GPU (36) 는 버텍스 쉐이딩 유닛으로서 각각의 쉐이딩 유닛 (40) 을 초기에 호출하므로, GPU (36) 의 컴포넌트들은 예를 들어, 1:1 입력/출력 인터페이스를 고수하는 특별한 포맷으로 버텍스 쉐이딩 유닛으로부터 데이터를 수신하도록 구성될 수도 있다. 예를 들어, GPU (36) 는 쉐이딩된 버텍스에 대한 쉐이딩 유닛 (40) 으로부터의 출력을 저장하기 위하여 단일 엔트리를 캐쉬 (예를 들어, 이하에서 더욱 상세하게 설명된 바와 같은 버텍스 파라미터 캐쉬) 내에 할당할 수도 있다. GPU (36) 는 또한 쉐이딩 유닛 (40) 이 호출되는 방식에 기초하여 일부의 래스터라이제이션 동작들을 수행할 수도 있다. 이하에서 더욱 상세하게 설명되는 바와 같이, 이 개시물의 양태들은 GPU (36) 가 적절한 인터페이스를 여전히 고수하면서, 버텍스 쉐이딩 동작들과 동일한 쉐이딩 유닛으로 지오메트리 쉐이딩 동작들을 수행하도록 한다.

[0091] 일부의 인스턴스들에서, 지오메트리 쉐이더 스테이지 (102) 는 데이터의 낮은 증폭 (예를 들어, 포인트-스프라이트 발생) 을 위하여 주로 이용될 수도 있다. 이러한 동작들은 지오메트리 쉐이더 호출 당 상대적으로 낮은 ALU 이용을 요구할 수도 있다. 따라서, 쉐이딩 유닛들 (40) 의 ALU 들은 지오메트리 쉐이더 스테이지 (102) 동안에 완전히 사용되지 않을 수도 있다. 이 개시물의 양태들에 따르면, 지오메트리 쉐이더 스테이지 (102) 는, GPU 아키텍처에서 버텍스 쉐이더 스테이지 (100) 로서 호출될 수도 있는 병합된 VS/GS 스테이지

(110) 를 형성하기 위하여 버텍스 쉐이더 스테이지 (100) 에 첨부될 수도 있다. 위에서 설명된 방식으로 병합된 VS/GS 스테이지 (110) 를 호출하는 것은, 버텍스 쉐이딩 및 지오메트리 쉐이딩 동작들의 둘 모두가 동일한 프로세싱 유닛들에 의해 수행되도록 함으로써 ALU 사용을 증가시킬 수도 있다.

[0092] 병합된 VS/GS 스테이지 (110) 를 가능하게 하기 위하여, GPU (36) 는 도 4 에 도시된 예에 대하여 더욱 상세하게 설명되는 바와 같이, 버텍스 쉐이딩 동작들 (1:1 스테이지) 과 지오메트리 쉐이딩 동작들 (1:N 스테이지) 사이에서 천이하기 위한 기능들을 수행할 수도 있다. 이러한 방법으로, 이 개시물의 기술들은 제한된 자원들 (예를 들어, GPU 가 2 개를 초과하는 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들 (40) 을 지정하는 것을 방지할 수도 있음) 을 갖는 GPU 가 추가적인 자원들을 갖는 GPU 를 에뮬레이팅하도록 한다.

[0093] 도 4 는 버텍스 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행하기 위하여 이 개시물에서 설명된 기술들을 구현하는 하드웨어 쉐이딩 유닛의 일 예의 동작들을 예시하는 도면이다. GPU (36; 도 1) 에 대하여 설명되었지만, 이 개시물의 양태들은 다양한 다른 컴포넌트들을 갖는 다양한 다른 GPU 들에 의해 수행될 수도 있다.

[0094] 도 4 의 예에서, GPU (36) 는 버텍스 쉐이딩 동작들을 수행하도록 쉐이딩 유닛 (40) 을 지정할 수도 있다. 따라서, GPU (36) 의 컴포넌트들은 버텍스에 대한 데이터를 쉐이딩 유닛 (40) 으로 전송하고 쉐이딩 유닛 (40) 으로부터 쉐이딩된 버텍스에 대한 데이터를 수신하도록 구성될 수도 있다 (예를 들어, 1:1 인터페이스). 쉐이딩 유닛 (40) 은 버텍스 쉐이딩 동작들을 수행하기 위하여 버텍스 쉐이더를 실행할 수도 있어서, 프리미티브들의 제 1 세트 (120) 를 발생시킬 수도 있다. 도 4 의 예에서, 프리미티브들의 제 1 세트 (120) 는 포인트들 (p0-p3) 로서 나타낸 바와 같이, 4 개의 버텍스들을 가지는 인접성 (adjacency) 을 갖는 삼각형을 포함한다.

[0095] 버텍스 쉐이딩 동작들을 실행한 후, GPU (36) 는 쉐이딩된 버텍스들을 로컬 메모리 자원들에 저장할 수도 있다. 예를 들어, GPU (36) 는 (만약 있다면) "컷 (cut)" 정보 및 streamid 와 함께, 버텍스 쉐이더 출력을 (예를 들어, GPU 메모리 (38) 의) 위치 캐쉬로 엑스포트 (export) 할 수도 있다. 버텍스 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들은 VS END 명령에 의해 분리될 수도 있다. 따라서, VS END 명령을 실행하고 버텍스 쉐이딩 동작들을 완료한 후, 버텍스 쉐이딩 동작들을 수행하도록 지정된 하나 이상의 쉐이딩 유닛들 (40) 은 각각 지오메트리 쉐이딩 동작들을 수행하기 시작한다.

[0096] 즉, 이 개시물의 양태들에 따르면, 버텍스 쉐이딩 동작들을 수행하도록 지정된 동일한 쉐이딩 유닛 (40) 은 또한 지오메트리 쉐이딩 동작들을 수행한다. 예를 들어, GPU (36) 는 하나 이상의 자원 포인터들을 변화시킴으로써 지오메트리 쉐이더 특정 자원들 (예를 들어, 지오메트리 쉐이더 상수들, 텍스처 오프셋들, 등) 에 대한 상태를 변화시킬 수도 있다. GPU (36) 는 쉐이딩 동작들에 배정된 모드 (드로우 모드 (draw mode)) 에 따라 이 상태 변화를 수행할 수도 있다.

[0097] 일부의 예들에서, GPU (36) 는 드로우 콜을 실행할 때에 드로우 모드를 세팅할 수도 있다. 드로우 모드는 어느 쉐이딩 동작들이 드로우 콜과 연관되는지를 표시할 수도 있다. 예시의 목적들을 위한 예에서, 0 의 드로우 모드는 드로우 콜이 버텍스 쉐이딩 동작들만을 포함하는 것을 표시할 수도 있다. 1 의 드로우 모드는 드로우 콜이 버텍스 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들의 둘 모두를 포함하는 것을 표시할 수도 있다.
이하에서 더욱 상세하게 설명된 바와 같이, 다른 드로우 모드들이 또한 가능하다. 표 1 은 2 개의 모드들을 갖는 일 예의 모드 표를 제공한다:

표 1

[0098] **모드 정보 병합된 VS/GS**

모드	**모드 0** **GS: 오프**	**모드 1** **GS: 온**
흐름	VS->PS	VS| GS->PS
인덱스 (32 비트들)	*버텍스 인덱스 (VS)*	*버텍스 인덱스 (VS)*
PrimitiveID (32 -비트들)	이용되지 않음	*PrimitiveID (GS)*
Misc (25 비트들)	이용되지 않음	misc-> rel_primID (4:0)
		misc-> rel_vertex (9:5)
		misc-> GsInstance (14:10)

		misc->Gsoutvertex (24:15)
Vs_valid (1 비트)		
Gshs_valid (1 비트)		
모드 (2:0)	모드 = mode_0	모드 = mode_1

[0099] 상기 표 1 의 예에서, "흐름" 은 각각의 모드들과 연관된 (GPU (36) 에 의해 실행되는 바와 같은) 동작들의 흐름을 표시한다. 예를 들어, 모드 0 은 버텍스 쉐이딩 (VS) 및 픽셀 쉐이딩 (PS) 동작들을 포함한다. 따라서, GPU (36) 는 모드 0 드로우 콜을 실행할 시에 버텍스 쉐이딩 동작들 및 픽셀 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들 (40) 을 지정할 수도 있다. 표 1 의 모드 1 은 버텍스 쉐이딩 및 픽셀 쉐이딩 동작들 뿐만 아니라, 지오메트리 쉐이딩 (GS) 동작들도 포함한다.

[0100] 따라서, GPU (36) 는 버텍스 쉐이딩 동작들 및 픽셀 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들 (40) 을 지정할 수도 있다. 그러나, GPU (36) 는 또한 지오메트리 쉐이더 명령들을 버텍스 쉐이더 명령들에 첨부할 수도 있어서, 지오메트리 쉐이더 동작들이 버텍스 쉐이더 동작들을 실행하는 것을 담당하는 동일한 쉐이딩 유닛들 (40) 에 의해 실행된다. "misc" 비트들은 동일한 쉐이딩 유닛 (40) 이 다수의 상이한 쉐이더들을 연속으로 실행하는 것을 가능하게 하기 위해 이용되는 변수들 (예를 들어, rel_primID, rel_vertex, GsInstance, Gsoutvertex) 을 위해 예약된다.

[0101] 도 4 의 예에서, 동일한 쉐이딩 유닛 (40) 은 또한, 프리미티브들의 제 1 세트 (120) 를 입력으로서 이용하여 버텍스들 (V0-V5) 을 가지는 (삼각형 스트립 (triangle strip) 이라고 지칭될 수도 있는) 프리미티브들의 제 2 세트 (124) 를 발생시킨다. 버텍스들 (V0-V5) 을 발생시키기 위하여, 버텍스 쉐이딩을 위해 지정된 쉐이딩 유닛 (40) 은 지오메트리 쉐이더 동작의 다수의 인스턴스들 (예를 들어, 그 출력 식별자들 (outID) 에 의해 나타내어지고 또한, 동일한 지오메트리 쉐이더 프로그램의 상이한 인스턴스들이라고 지칭될 수도 있음) 을 실행한다. 지오메트리 쉐이더 동작의 각각의 인스턴스는 동일한 지오메트리 쉐이딩 동작을 수행하기 위하여 동일한 알고리즘을 실행하고, 하나 이상의 새로운 버텍스들 (V0-V5) 의 각각의 인스턴스들을 발생시킨다.

[0102] 도 4 에 도시된 표의 8 개의 컬럼들은 지오메트리 쉐이더 동작 (또는 프로그램) 의 8 개의 별도의 인스턴스들에 대응하고, 여기서, 좌측으로부터 우측으로의 각각의 컬럼은 0-7 의 지오메트리 쉐이더 동작 outID 에 의해 식별될 수도 있다. 입력 프리미티브 당 병합된 VS/GS 출력들의 수는 dcl_maxoutputvertexcount*GSInstancecount 와 동일할 수도 있고, 여기서, 각각의 VS/GS 출력은 지오메트리 쉐이더 스테이지로부터 방출된 하나의 버텍스이다. 지오메트리 쉐이더 스테이지 출력 버텍스들의 수가 dcl_maxoutputvertexcount 보다 더 작은 인스턴스들에서는, 그 출력 버텍스가 이하에서 더욱 상세하게 설명되는 바와 같이, 조건부로 폐기 또는 생략 ("킬 (kill)" 되는 것이라고 지칭될 수도 있음) 될 수도 있다. 따라서, 각각의 파이버는 버텍스 쉐이더의 하나의 호출과, 그 다음으로, MaxVertexOutput 에 의해 특정된 지오메트리 쉐이더 출력 버텍스 당 지오메트리 쉐이더의 하나의 호출에 대응한다.

[0103] 도 4 에 도시된 예에서, 지오메트리 쉐이더 동작의 8 개의 인스턴스들의 각각은 하나 이상의 새로운 버텍스들의 별도의 인스턴스를 발생시키기 위하여 버텍스 쉐이딩 동작들을 위해 지정된 동일한 쉐이딩 유닛 (40) 에 의해, 종종 동시에 첨부되고 실행된다. 따라서, 지오메트리 쉐이더 동작들의 인스턴스들의 각각은 버텍스들 (V0-V5) 의 6개 모두를 발생시키지만, 6 개의 새로운 버텍스들 중의 대응하는 하나만을 출력한다. 지오메트리 쉐이더 동작의 각각의 인스턴스는 버텍스 쉐이딩 동작들을 수행하기 위하여 쉐이딩 유닛 (40) 을 호출하는 것과 연관된 1:1 인터페이스를 고수하도록 6 개의 새로운 버텍스들 중의 대응하는 하나를 출력하기만 한다.

[0104] 도 4 의 예에서 도시된 바와 같이, 지오메트리 쉐이더 동작들의 각각은 그 outID 와 일치하는 6 개의 새로운 버텍스들 중의 하나를 출력한다. 따라서, outID=0 을 갖는 지오메트리 쉐이더 동작의 제 1 인스턴스는 6 개의 새로운 버텍스들 중의 제 1 버텍스 (V0) 를 출력한다. outID=1 을 갖는 지오메트리 쉐이더 동작의 제 2 인스턴스는 6 개의 새로운 버텍스들 중의 제 2 버텍스 (V1) 를 출력한다. outID=2 를 갖는 지오메트리 쉐이더 동작의 제 3 인스턴스는 6 개의 새로운 버텍스들의 제 3 버텍스 (V2) 를 출력한다. outID=3 을 갖는 지오메트리 쉐이더 동작의 제 4 인스턴스는 6 개의 새로운 버텍스들의 제 4 버텍스 (V3) 를 출력한다. outID=4 를 갖는 지오메트리 쉐이더 동작의 제 5 인스턴스는 6 개의 새로운 버텍스들 중의 제 2 버텍스 (V4) 를 출력한다.

outID=5 를 갖는 지오메트리 쉐이더 동작의 제 6 인스턴스는 6 개의 새로운 버텍스들 중의 제 6 버텍스 (V5) 를 출력한다.

[0105] 지오메트리 쉐이더 동작은 6 개의 새로운 버텍스들을 발생시키기만 하고 지오메트리 쉐이더 동작의 제 7 및 제 8 인스턴스의 outID 들은 6 개의 새로운 버텍스들 중의 임의의 것에 대응하지 않으므로, 지오메트리 쉐이더 동작의 제 7 및 제 8 인스턴스들은 "킬" 또는 종결된다. 따라서, 지오메트리 쉐이더 동작의 이 인스턴스들과 연관된 대응하는 버텍스가 없는 것으로 결정할 시에, 쉐이딩 유닛 (40) 은 지오메트리 쉐이더 동작의 제 7 및 8 개의 인스턴스들의 실행을 종결시킨다.

[0106] 이하에서 도시된 표 2 는 버텍스 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행하기 위하여 GPU (36) 에 의해 유지될 수도 있는 몇몇 파라미터들을 예시한다.

표 2

[0107] **VS/GS 에 대한 파라미터들**

흐름	VS| GS->PS
인덱스 (32 비트들)	*버텍스 인덱스 (VS)*
uv_msb (2-비트들)	이용되지 않음
PrimitiveID (32 -비트들)	*PrimitiveID(GS)*
Rel_patchid (32-비트들)	이용되지 않음
Misc (25 비트들)	misc-> rel_primID (4:0)
	misc-> rel_vertex (9:5)
	misc-> GsInstance (14:10)
	misc-> Gsoutvertex (24:15)
Vs_valid (1 비트)	
Gshs_valid (1 비트)	
모드 (2:0)	모드 = mode_1
Instance_cmd (2-비트)	

[0108] 표 2 에 도시된 어떤 파라미터들 (예를 들어, uv_msb, Rel_patchid) 은 VS/GS 동작들에 대해 이용되지 않고, 이하에서 더욱 상세하게 설명된다. 표 2 의 예에서는, 인덱스가 버텍스들의 상대 인덱스를 표시한다. PrimitiveID 는 연관 버텍스들의 프리미티브를 식별하기 위하여 지오메트리 쉐이딩 동작들 동안에 이용되는 프리미티브 ID 를 표시하고, 시스템 발생된 값 (예를 들어, GPU (36) 의 하나 이상의 하드웨어 컴포넌트들에 의해 발생됨) 일 수도 있다. 위에서 언급된 바와 같이, Misc 는 VS 동작들 후에 GS 동작들을 수행하기 위한 예약된 캐쉬 값들을 표시한다. 예를 들어, 이하에서 도시되는 표 3 은 도 4 에 대하여 위에서 설명된 버텍스 쉐이딩 및 지오메트리 쉐이딩 동작들을 수행할 때의 파라미터 값들을 예시한다.

표 3

[0109] **VS/GS 동작들에 대한 파라미터 값들**

모드 1 **GS: 온**	파이버 0	파이버 1	파이버 2	파이버 3	파이버 4	파이버 5	파이버 6	파이버 7
Valid_as_input	1	1	1	0	0	0	0	0
버텍스 인덱스 (VS)	V0	V1	V2	0	0	0	0	0
primitiveID (GS)	5	5	5	5	5	5	5	5
Valid_as_output	1	1	1	1	1	1	1	1
misc-> rel_primID (4:0)	2	2	2	2	2	2	2	2
misc-> rel_vertex (9:5)	0	1	2	0	0	0	0	0
misc-> GsInstance (14:10)	0	0	0	0	0	0	0	0

misc-> Gsoutvertex (24:15)	0	1	2	3	4	5	6	7

[0110] 다수의 파이버들 (예를 들어, 명령들) 이 버텍스 쉐이딩 및 지오메트리 쉐이딩 동작들을 수행하기 위하여 할당되지만, 일부의 인스턴스들에서, GPU (36) 는 파이버들의 서브-세트를 실행하기만 할 수도 있다. 예를 들어, GPU (36) 는 쉐이딩 유닛들 (40) 로 명령들을 실행하기 전에 명령들이 유효한지 (상기 표 3 에 도시된 valid_as_input) 여부를 결정할 수도 있다. 할당된 파이버들 중의 3 개만이 쉐이딩된 버텍스들을 발생시키기 위하여 이용되므로, GPU (36) 는 버텍스 쉐이딩 동작들을 수행할 때에 나머지 파이버들 (상기 표 3 에서의 파이버들 3-7) 을 실행하지 않을 수도 있고, 이것은 전력을 절감할 수도 있다. 이하에서 더욱 상세하게 설명된 바와 같이, GPU (36) 는 마스크 (예를 들어, 이하의 도 5b 에서의 cov_mask_1) 에 기초하여 어느 파이버들을 실행할 것인지를 결정할 수도 있다.

[0111] 어떤 API 들 (예를 들어, DirectX 10 API) 은 지오메트리 쉐이더 스테이지로부터 소위 "스트림 출력" 을 제공하고, 여기서, 스트림 출력은 새로운 버텍스들을 지오메트리 쉐이더로부터 저장 유닛 (48) 과 같은 메모리로 출력하는 것을 지칭하여, 이 새로운 버텍스들은 지오메트리 쉐이더로 다시 입력될 수도 있다.

[0112] 기술들은 하드웨어 유닛이 지오메트리 쉐이더 동작을 수행하는 것으로부터 기인하는 새로운 버텍스들을 저장 유닛 (48) 으로 출력하는 것을 가능하게 함으로써 이 스트림 출력 기능성에 대한 지원을 제공할 수도 있다. 이 스트림 출력을 통해 출력된 새로운 버텍스들은 래스터라이저에 의해 예상된 포맷에서가 아니라, 예상된 지오메트리 쉐이더 포맷에서 특정된다. 하드웨어 유닛은 이 새로운 버텍스들을 취출할 수도 있고, 이 상황에서 "스트림 출력 버텍스들" 이라고 지칭될 수도 있는 이 버텍스들에 대하여 기존의 지오메트리 쉐이더 동작, 또는 새로운 지오메트리 쉐이더 동작을 계속 구현할 수도 있다. 이러한 방법으로, 기술들은 상대적으로 제한된 수의 쉐이딩 유닛들 (40) 을 갖는, GPU (36) 와 같은 GPU 가 더 많은 쉐이딩 유닛들을 갖는 GPU 를 에뮬레이팅하는 것을 가능하게 할 수도 있다.

[0113] 도 5a 및 도 5b 는 이 개시물의 기술들을 구현하는 하드웨어 쉐이딩 유닛에 의해 수행될 수도 있는 일 예의 동작들을 예시한다. 예를 들어, 도 5a 는 버텍스 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행할 때, 병합된 VS/GS 하드웨어 쉐이딩 유닛에 의해 수행되는 동작들의 흐름을 일반적으로 예시한다. 일부의 예들에서, 병합된 VS/GS 하드웨어 쉐이딩 유닛은, 버텍스 쉐이딩 동작들을 수행하도록 GPU (36) 에 의해 지정되지만, 이 개시물의 기술들에 따라 버텍스 쉐이딩 동작들 및 하드웨어 쉐이딩 동작들의 둘 모두를 수행하는 쉐이딩 유닛 (40) 을 포함할 수도 있다.

[0114] 도 5b 는 병합된 VS/GS 하드웨어 쉐이딩 유닛에 의해 실행될 수도 있는, 도 5a 에 도시된 동작들의 흐름에 대응하는 의사 코드를 일반적으로 예시한다. 도 5a 및 도 5b 의 어떤 양태들은 GPU (36; 도 1) 에 대하여 설명될 수도 있지만, 이 개시물의 양태들은 다양한 다른 컴포넌트들을 갖는 다양한 다른 GPU 들에 의해 수행될 수도 있다.

[0115] 도 5a 에 도시된 예에서, 병합된 VS/GS 하드웨어 쉐이딩 유닛은 버텍스 속성들, vertex_id, instance_id, primitive_id, misc 와 같은 시스템 값들을 일련의 레지스터들 (R0, R1, 및 R2) 에 기록한다 (140). 전형적으로, 시스템 값들은 GPU 의 임의의 이와 다르게 할당되지 않은 메모리에 저장될 수도 있다. 시스템 발생된 값들을 미리 결정된 로케이션에서의 일련의 레지스터들에 저장함으로써, GPU (36) 는 VS 및 GS 스테이지들의 각각에 대하여 시스템 발생된 값들을 액세스할 수도 있다. 따라서, GS 스테이지는 시스템 발생된 값들이 어디에 저장되었는지를 결정하기 위하여 VS 스테이지에 기초하여 컴파일링될 필요가 없다. 오히려, GPU (36) 는 요구되는 시스템 발생된 값들을 액세스하기 위하여 스테이지들의 각각을 수행할 때, 미리 결정된 메모리 로케이션들을 액세스할 수도 있다.

[0116] 다음으로, 병합된 VS/GS 하드웨어 유닛은 버텍스 쉐이딩 동작들을 수행한다 (142). 버텍스 쉐이딩 동작들에 후속하여, 병합된 VS/GS 하드웨어 쉐이딩 유닛은 범용 레지스터 (general purpose register; GPR) 들의 내용물 (예를 들어, 버텍스 쉐이딩 동작들로부터의 프리미티브 버텍스들) 을 GPU 메모리 (38) 와 같은 로컬 메모리에 기록할 수도 있다. 다음으로, 병합된 VS/GS 하드웨어 쉐이딩 유닛은 도 5b 에 대하여 이하에서 더욱 상세하게 설명된 바와 같이, GS 텍스처 및 상수 오프셋들로 스위칭하고 (146), 그리고 GS 프로그램 카운터로 스위칭할 수도 있다 (148).

[0117] 병합된 VS/GS 하드웨어 쉐이딩 유닛은 버텍스 쉐이딩 동작들로부터의 프리미티브 버텍스들과 같은, 로컬 메모리

의 내용물을 판독할 수도 있고, 지오메트리 쉐이딩 동작들을 수행할 수도 있다 (150). 병합된 VS/GS 하드웨어 쉐이딩 유닛은 하나의 버텍스 속성을 버텍스 파라미터 캐쉬 (vertex parameter cache; VPC) 에 출력할 수도 있을 뿐만 아니라, 지오메트리 쉐이딩된 버텍스들의 위치의 표시, stream_id, 임의의 컷 표시들, 및 임의의 해독된 값들을 위치 캐쉬에 출력할 수도 있다.

[0118] 도 5b 는 병합된 VS/GS 하드웨어 쉐이딩 유닛에 의해 실행될 수도 있는, 도 5a 에 도시된 동작들의 흐름에 대응하는 의사 코드를 일반적으로 예시한다. 각각의 쉐이더 스테이지는 (예를 들어, 특별한 스테이지가 어떻게 또 다른 스테이지와 연결될 것인지에 대한 지식 없이) 별도로 그리고 독립적으로 컴파일링될 수도 있다. 단일 하드웨어 쉐이딩 유닛이 다수의 쉐이딩 동작들을 수행하도록 하기 위하여, 하드웨어 쉐이딩 유닛은 로컬 메모리 내의 어떤 위치들을 예약할 수도 있다. 예를 들어, 하드웨어 쉐이딩 유닛은 쉐이더 스테이지들 (VS 또는 GS) 의 둘 모두에 의해 액세스될 수 있는 로컬 메모리 내의 위치들을 예약할 수도 있다. 어떤 변수들 (예를 들어, PrimitiveID, misc, 및 rel_patchid) 은 하나를 초과하는 쉐이더 스테이지에 의해 이용될 수도 있다. 따라서, 로컬 메모리 내의 예약된 위치들은 하나를 초과하는 쉐이더 스테이지에 의해 액세스될 수도 있는 통상적으로 이용되는 변수들에 대한 표준화된 위치를 제공한다.

[0119] 도 5b 에 도시된 예에서, 하드웨어 쉐이딩 유닛은 (도 5a 의 예에서 단계들 (140-142) 에 대응할 수도 있는, 상부로부터 하부로의 제 1 점선 박스에 포함된) 버텍스 쉐이딩 동작들 (VS) 을 초기에 수행할 수도 있다. 이 개시물의 양태들에 따르면, 다음으로, 하드웨어 쉐이딩 유닛 (또는 GPU 의 또 다른 컴포넌트) 은 (도 5a 의 예에서 단계들 (144-148) 에 대응할 수도 있는, 상부로부터 하부로의 제 2 점선 박스에 포함된) 버텍스 쉐이딩 동작들로부터 지오메트리 쉐이딩 동작들로의 전환을 개시하기 위하여 소위 "패치 코드" 를 실행할 수도 있다. 더욱 구체적으로, 커맨드들 CHMSK 및 CHSH 는 하드웨어 쉐이딩 유닛으로 하여금 (위에서 설명된 바와 같이) 드로우 콜이 실행되는 모드에 따라 동작 모드들을 스위칭하게 할 수도 있다.

[0120] 예를 들어, 하드웨어 쉐이딩 유닛은 버텍스 쉐이딩 동작들로부터의 버텍스 데이터를 로컬 GPU 메모리에 기록할 수도 있어서, 쉐이딩된 버텍스들은 지오메트리 쉐이딩 동작들을 수행할 때에 이용가능하다. 다음으로, 하드웨어 쉐이딩 유닛 (또는 GPU 의 또 다른 컴포넌트) 은 지오메트리 쉐이딩 동작들을 위한 하드웨어 쉐이딩 유닛의 자원들을 스위칭하는 마스크 변경 (CHMSK) 명령을 실행한다. 예를 들어, CHMSK 명령을 실행하는 것은 하드웨어 쉐이딩 유닛으로 하여금 어느 모드가 현재 실행되고 있는지를 결정하게 할 수도 있다.

[0121] 상기 표 2 에 대하여, CHMSK 를 실행하는 것은 또한, 하드웨어 쉐이딩 유닛으로 하여금 어느 쉐이더 스테이지들이 유효한지 (예를 들어, vs_valid, gs_valid, 등) 를 결정하게 할 수도 있다. 위에서 언급된 바와 같이, GPU (36) 는 버텍스 쉐이딩 및 지오메트리 쉐이딩 동작들을 수행하기 위한 다수의 파이버들을 할당할 수도 있다. 그러나, CHMSK 를 실행할 시에, GPU (36) 는 파이버들의 서브-세트를 실행하기만 할 수도 있다. 예를 들어, GPU (36) 는 명령들을 쉐이딩 유닛들 (40) 로 실행하기 전에 명령들이 유효한지 여부를 결정할 수도 있다. GPU (36) 는 유효하지 않은 (예를 들어, 쉐이딩된 버텍스를 발생시키지 않는) 파이버들을 실행하지 않을 수도 있고, 이것은 전력을 절감할 수도 있다.

[0122] 하드웨어 쉐이딩 유닛은 또한, 프로그램 카운터 (PC) 를 지오메트리 쉐이딩 동작들을 수행하기 위한 적절한 상태 오프셋들로 스위칭하기 위하여 쉐이더 변경 (change shader; CHSH) 명령을 실행한다. 이하에서 더욱 상세하게 설명되는 바와 같이, (도 5a 의 예에서 단계들 (144-148) 에 대응할 수도 있는, 상부로부터 하부로의 제 2 점선 박스에 포함된) 이 패치 코드는 어느 쉐이더 스테이지들이 병합되고 있는지에 관계없이 동일할 수도 있다.

[0123] 패치 코드를 실행한 후, 하드웨어 쉐이더 유닛은 버텍스 쉐이딩 동작들을 중단하고 (도 5a 의 예에서 단계 (150) 에 대응하는, 상부로부터 하부로의 제 3 점선 박스에 포함된) 지오메트리 쉐이딩 동작들을 수행한다. 전형적으로, 다수의 쉐이딩 동작들을 수행하는 하드웨어 쉐이딩 유닛에 의해 실행되는 쉐이더들 (쉐이딩 동작들을 수행하기 위한 코드) 은 리컴파일링 기반 쉐이더 종속성들을 요구할 수도 있다. 예를 들어, 프리미티브 ID (시스템 발생된 값) 가 GS 스테이지에 의해 이용될 경우, VS 스테이지는 GS 스테이지가 그로부터 값을 픽업 (pick up) 할 수 있는 로케이션 내에 primitiveID 값을 두기 위하여 (예를 들어, 컴파일러 (54) 에 의해) 컴파일링될 수도 있다. 따라서, VS 스테이지의 컴파일링은 GS 스테이지의 필요성들에 따라 종속적일 수도 있다.

[0124] 이 개시물의 양태들에 따르면, 쉐이더들의 각각은 다른 쉐이더들을 고려하지 않고 독립적으로 컴파일링될 수도 있다. 예를 들어, 쉐이더들은 다른 쉐이더들이 언제 실행될지에 대한 지식 없이 독립적으로 컴파일링될 수도 있다. 컴파일링 후에, GPU (36) 는 드로우 시간에 실행되는 드로우 콜과 연관된 모드 정보에 기초하여

도 5b 에 도시된 패치 코드를 이용하여 쉐이더들을 함께 패치할 수도 있다. 시스템 발생된 값들 vertexID 및 instanceID 는 버텍스 쉐이더에서 이용되기만 할 수도 있고, VS 스테이지를 컴파일링함으로써 컴퓨팅되는 바와 같이 특정된 범용 레지스터 슬롯들 (GPR 들) 에서 로딩될 수도 있다. 그러나, primitiveID 와, misc 및 rel_patchid 와 같은 (예를 들어, 도 6 에 도시된 바와 같은) 프리미티브 제어기 (primitive controller; PC) 로부터의 다른 병합 쉐이더 관련 값들은 쉐이더 스테이지들 중의 임의의 것에 의해 이용될 수도 있다.

[0125] 위에서 설명된 패치 코드는 GPU 드라이버 (50) 와 같은, GPU (36) 에 대한 드라이버에 의해 컴파일링된 쉐이더들에 추가될 수도 있다. 예를 들어, GPU 드라이버 (50) 는 각각의 드로우 콜에 대하여 어느 쉐이더들이 요구되는지를 결정한다. GPU 드라이버 (50) 는 소위 드라이버 시간 또는 링크 시간에 도 5b 에 도시된 패치 코드를 적절한 쉐이더들 (병합되고 있는 쉐이더들) 에 부착할 수도 있고, 이것에 의하여, 쉐이더들이 동일한 하드웨어 쉐이딩 유닛들에 의해 실행되도록 컴파일링된 쉐이더들을 링크할 수도 있다. GPU 드라이버 (50) 는 전체 쉐이더들을 리컴파일링할 필요가 없고, 이것에 의하여, 연산 자원들을 절감한다.

[0126] 이러한 방법으로, GPU (36) 는 복수의 모드들을 이용하여 쉐이딩 동작들을 함께 패치할 수도 있고, 각각의 모드는 연관된 쉐이딩 동작들의 특별한 세트를 가질 수도 있다. 이러한 기술들은 GPU (36) 가 쉐이딩 유닛들 (40) 을 재구성하지 않고도 추가적인 쉐이딩 동작들 (예를 들어, 지오메트리 쉐이딩 동작들, 헐 쉐이딩 동작들, 및/또는 도메인 쉐이딩 동작들) 을 수행하는 것을 가능하게 할 수도 있다. 즉, 기술들은 쉐이딩 유닛들 (40) 이 다른 쉐이딩 동작들을 수행하면서, 어떤 쉐이더 스테이지들의 입력/출력 제약들을 고수하도록 할 수도 있다.

[0127] 도 6 은 이 개시물의 양태들에 따라, 병합된 버텍스 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행하기 위한 그래픽스 프로세싱 유닛 (178) 의 일 예의 컴포넌트들을 예시하는 도면이다. 도 6 의 예는 병합된 VS/GS 유닛 (180), 버텍스 파라미터 캐쉬 (VPC; 182), 프리미티브 제어기 (PC; 184), 버텍스 페치 디코더 (VFD; 186), 그래픽스 래스터라이저 (GRAS; 188), 렌더 백엔드 (RB; 190), 커맨드 프로세서 (CP; 192), 및 픽셀 쉐이더 (PS; 194) 를 포함한다. 추가적으로, 도 6 은 PM4 패킷 버퍼들 (198), 버텍스 오브젝트들 (200), 인덱스 버퍼들 (202), 스트림 출력 버퍼 (204), 및 프레임 버퍼 (206) 를 갖는 메모리 (196) 를 포함한다.

[0128] 도 6 의 예에서, VS/GS 유닛 (180) 은 위에서 설명된 방식으로 버텍스 쉐이딩 동작들을 수행하도록 지정된 하나 이상의 쉐이딩 유닛들에 의해 구현된다. VPC (182) 는 스트림 출력 데이터를 스트림 출력 버퍼 (204) 에 저장하기 위한 스트림 출력 기능성을 구현할 수도 있다. PC (184) 는 변환될 필요가 있을 수도 있는 버텍스들을 관리할 수도 있다. 예를 들어, PC (184) 는 버텍스들을 삼각형 프리미티브들로 집합시킬 수도 있다. VFD (186) 는 버텍스 포맷 상태에 기초하여 버텍스 데이터를 페치할 수도 있다. GRAS (188) 는 삼각형 버텍스들을 입력으로서 수신할 수도 있고, 삼각형 경계들 내에 있는 픽셀들을 출력할 수도 있다. 프리-페치 파서 (pre-fetch parser; PFP) 는 커맨드 스트림 및 페치 데이터를 포인터들 (예를 들어, 자원 포인터들) 을 통해 프리-디코딩할 수도 있어서, 이 데이터는 주요 CP 엔진 (192) 이 이 데이터를 필요로 할 수도 있는 시간까지 준비되어 있다.

[0129] 예시의 목적들을 위한 예에서, DirectX 10 디스패치 메커니즘은 도 6 에 도시된 그래픽스 프로세싱 유닛 (178) 을 이용하여 구현될 수도 있다. 예를 들어, DirectX 드로우 콜은, VS 동작들 및 GS 동작들이 병합되는, 예를 들어, 동일한 쉐이딩 유닛들에 의해 수행되는 것을 표시하는 모드 비트들 (모드 정보) 을 갖는 드로우 개시자 (draw initiator) 들을 갖는 단일 패스 드로우 콜로서 다루어질 수도 있다. 이 모드는 PC (184) 내의 GSblock 이 GS 출력 vertexID 및 GS instanceID 로 VFD (186) 를 위한 데이터를 발생시키는 것을 가능하게 한다. GSblock 은 선언된 maxoutputvertexcount 및 GSinstancecount 에 기초하여 입력 프리미티브에 대한 VS 파이버들의 수를 생성한다. 웨이브 (wave) 내의 파이버들의 수 (예를 들어, 32 개의 파이버들과 같이, 쉐이딩 유닛에 의해 행해진 작업의 양) 가 maxoutputvertexcount * GSinstancecount 보다 더 클 경우, 웨이브는 다수의 완전한 입력 GS 프리미티브들을 가질 수도 있다. 이와 다를 경우, GS 입력 프리미티브 버텍스 인덱스들은 maxoutputvertexcount * GSinstancecount 파이버들이 생성될 때까지 다음 웨이브에 대해 반복될 수도 있다. 입력 프리미티브 버텍스들에 대하여 버텍스 재이용은 필요하지 않다.

[0130] VPC (182) 의 출력에서, PC (184) 는 GS 출력 프리미티브 타입에 기초하여 프리미티브 커넥티비티 (primitive connectivity) 를 발생시킬 것이다. 예를 들어, (VS/GS (180) 의) GS 로부터의 제 1 출력 버텍스는 전형적으로 위치 캐쉬에서의 "컷" 비트로 구성될 수도 있고, 이것은 이 버텍스 이전의 프리미티브 (스트립) 의 완료를 표시할 수도 있다. PC (184) 는 또한, GS 출력들을 주어진 스트림과 결부된 버퍼들 (204) 로 스트림 출력하기 위하여, VPC (182) 에 대한 streamid 와 함께, 완전한 프리미티브들에 대한 이 커넥티비티 정보를 VPC (182)

로 전송한다. GS (180) 내의 전체 프리미티브들 사이에 부분적인 프리미티브가 있을 경우, 이러한 부분적인 프리미티브는 프리미티브를 누락시키기 위하여 GRAS (188) 에 대한 PRIM_AMP_DEAD 로서 마킹된다. PC (184) 는 또한, 이러한 프리미티브에 대한 파라미터 캐쉬를 할당해제 (de-allocate) 하기 위하여 데드 (dead) 프리미티브 타입들을 VPC (182) 로 전송한다.

[0131] maxoutputvertexcount 에 기초하여, (도 1 에 도시된 GPU 드라이버 (50) 와 같은) GPU 드라이버는 얼마나 많은 입력 프리미티브 버텍스들이 로컬 메모리 내에 저장될 것인지를 컴퓨팅할 수도 있다. 이 입력 프리미티브 값은 다음의 수식에 따라 변수 GS_LM_SIZE 로서 컴퓨팅될 수도 있다:

[0132] $$\frac{fibers\_in\_a\_wave}{maxoutputvertexcount} * \text{프리미티브 당 버텍스들의 수} * \text{버텍스의 사이즈}$$

[0133] 이 타입의 드로우 콜을 수신하는 하이 레벨 시퀀서 (HLSQ) 는 어느 쉐이더 프로세서의 로컬 메모리 (LM) 가 (예를 들어, 아마도 라운드 로빈 (round robin) 접근법을 이용하여) GS_LM_SIZE 에 대한 충분한 저장을 가지는지를 검사할 수도 있다. HLSQ 는 이러한 할당의 시작 기본 어드레스뿐만 아니라, 할당된 웨이브에 의한 로컬 메모리에 대한 임의의 판독 또는 기록의 어드레스도 유지할 수도 있다. HLSQ 는 또한, 로컬 메모리에 기록할 때, 할당된 메모리 내의 컴퓨팅된 오프셋을 기본 어드레스에 추가할 수도 있다.

[0134] 따라서, 이 개시물의 양태들에 따르면, 입력 및 출력 사이의 관계는 VS/GS (180) 에 대하여 (버텍스 쉐이딩 동작들을 수행하도록 지정된 쉐이딩 유닛에 대해 전형적인 바와 같이) 1:1 이 아니다. 오히려, GS 는 각각의 입력 프리미티브로부터의 하나 이상의 버텍스들을 출력할 수도 있다. 추가적으로, GS 에 의해 출력되는 버텍스들의 수는 동적이고, 1 로부터 API 부과된 최대 GS 출력 (예를 들어, 1024 개의 버텍스들인 출력 최대치와 동등할 수도 있는 1024 더블 워드 (디워드; dword) 들) 까지 변동될 수도 있다.

[0135] 즉, GS 는 1 개의 버텍스인 최소치 및 1024 개의 버텍스들인 최대치를 생성할 수도 있고, GS 로부터의 총 출력은 1024 디워드들일 수도 있다. GS 는 변수 dcl_maxoutputvertexcount 를 이용하여 GS 로부터의 출력 버텍스들의 최대 수를 컴파일 시간에 선언할 수도 있다. 그러나, 출력 버텍스들의 실제적인 수는 GPU (36) 가 GS 를 실행할 때에 알려지지 않을 수도 있다. 오히려, 선언 dcl_maxoutputvertexcount 은 GS 에 대한 파리미터로서 요구되기만 할 수도 있다.

[0136] GS 는 또한, 입력 프리미티브 당 호출될 GS 인스턴스들 (동작들) 의 수에 대한 변수 instancecount 를 선언할 수도 있다. 이 선언은 GS 호출 (지오메트리 쉐이더 인스턴스들의 최대 수를 식별함) 에 대한 외부의 루프로서 작동할 수도 있다. 다른 값들이 또한 이용될 수도 있지만, 최대 instancecount 는 32 로 세팅될 수도 있다. 따라서, GS 는 주어진 GS 가 어느 인스턴스에 대해 작동하는지를 표시하는, 지오메트리 쉐이더 동작들에서의 변수 GSInstanceID 를 액세스한다. GS 인스턴스들의 각각은 1024 디워드들까지 출력할 수 있고, 각각은 최대 출력 버텍스들의 수로서 dcl_maxoutputvertexcount 를 가질 수도 있다. 추가적으로, 각각의 GS 인스턴스는 다른 GS 인스턴스들에 독립적일 수도 있다.

[0137] GPU (36) 가 GS 의 입력에서 선언할 수도 있는 입력 프리미티브 타입은 포인트, 라인, 삼각형, 인접성을 갖는 라인, 인접성을 갖는 삼각형, 및 patch1-32 일 수도 있다. 인접성을 갖는 삼각형은 DirectX 10 과 같은 어떤 API 들에 대한 새로운 특징일 수도 있다. 추가적으로, patch1-32 는 DirectX 11 API 에 대해 추가되도록 하기 위한 또 다른 개량일 수도 있다. GS 로부터의 출력 프리미티브 타입은 포인트, 라인 스트립, 또는 삼각형 스트립일 수 있다. GS 의 출력은 GS 에서 선언될 수도 있는 4 개의 스트림들 중의 하나로 갈 수도 있고, GS 는 얼마나 많은 스트림들이 이용되는지를 선언할 수도 있다. 일반적으로, "스트림" 은 (예를 들어, 메모리 버퍼로) 저장되거나, 래스터라이저와 같은 GPU 의 또 다른 유닛으로 전송되는 쉐이딩된 데이터를 지칭한다. 각각의 버텍스 "emit (방출)" 명령은 버텍스가 어느 스트림으로 가지는지를 표시할 수도 있는 "emit stream (스트림 방출)" 지정을 이용할 수도 있다.

[0138] GS 는 스트립 프리미티브 타입을 완료하기 위하여 "cut stream (스트림 컷)" 명령 또는 "emitthencut stream (스트림 방출후 컷)" 명령을 이용할 수도 있다. 이러한 예들에서, 다음 버텍스는 주어진 스트림에 대한 새로운 프리미티브를 시작할 것이다. 일부의 예들에서, 프로그래머는 (API 를 이용하여) 스트림들을 설정할 때에 래스터라이징된 스트림으로서 이용될 스트림들 중의, 기껏해야 하나를 선언할 수도 있다. 추가적으로, 4 개의 ID 버퍼들이 하나의 스트림에 묶일 수도 있지만, GS 스트림들의 전부에 결부된 버퍼들의 총 수는 4 개를 초과하지 않을 수도 있다. 오프-칩 버퍼들은 전형적으로 스트림들 사이에서 공유되지 않는다.

[0139] 버텍스가 주어진 스트림에 대해 방출될 때, 스트림에 결부된 각각의 버퍼에 대한 버텍스의 서브섹션 (subsection) 들은 완전한 프리미티브로서 (저장 유닛 (48) 과 같은) 오프-칩 버퍼에 기록된다. 즉, 부분적인 프리미티브들은 일반적으로 오프-칩 버퍼에 기록되지 않는다. 일부의 예들에서, 오프-칩 버퍼들에 기록된 데이터는 프리미티브 타입의 표시를 포함하도록 확장될 수도 있고, 하나를 초과하는 스트림이 주어진 GS 에 대해 가능하게 될 경우, GS 에 대한 출력 프리미티브 타입은 오직 "포인트" 일 수도 있다.

[0140] PrimitiveID 가 시스템 발생된 값이므로, GS 스테이지는 PrimitiveID 파라미터를 입력으로서 수신할 수도 있다. GS 는 또한, PrimitiveID 파라미터, ViewportIndex 파라미터, RenderTargetArrayIndex 파라미터를 하나 이상의 레지스터들에 출력할 수도 있다. GS 입력들에 대한 속성 보간 모드는 상수인 것으로 전형적으로 선언된다. 일부의 예들에서는, GS 가 NULL 이지만 여전히 출력을 가능하게 하는 것으로 선언하는 것이 가능하다. 이러한 예들에서, 스트림 제로 (zero) 만이 활성일 수도 있다. 그러므로, VS 출력은 프리미티브 타입을 리스트하도록 확장될 수도 있고, 값들을 스트림 제로에 결부된 버퍼들에 기록할 수도 있다. 입력 프리미티브 타입이 인접한 프리미티브 타입인 것으로 선언될 경우, 인접한 버텍스 정보는 누락될 수도 있다. 즉, 예를 들어, 인접한 프리미티브 (예를 들어, 짝수의 버텍스 번호) 의 내부 버텍스들은 비-인접 (non-adjacent) 프리미티브 타입을 형성하도록 프로세싱될 수도 있다.

[0141] NULL GS 를 갖는 패치 입력 프리미티브 타입의 경우, 패치는 스트림에 결부된 버퍼들에 대한 포인트들의 리스트로서 기록된다. 선언된 스트림이 또한 래스터라이징될 경우, GPU (36) 는 패치 제어 포인트들에 의해 특정된 바와 같이, 패치를 복수의 포인트들로서 렌더링할 수도 있다. 추가적으로, GS 가 NULL 일 때, viewportindex 파라미터 및 rendertargetarrayindex 파라미터는 제로인 것으로 가정될 수도 있다.

[0142] 쿼리 카운터 (query counter) 들은 얼마나 많은 VS 또는 GS 동작들이 GPU (36) 에 의해 프로세싱되고 있는지를 결정하도록 구현될 수도 있고, 이것에 의하여, 하드웨어 컴포넌트들이 프로그램 실행을 추적하도록 할 수도 있다. 쿼리 카운터들은 stat_start 이벤트 및 stat_end 이벤트에 기초하여 카운팅을 시작하고 정지할 수도 있다. 카운터들은 stat_sample 이벤트를 이용하여 샘플링될 수도 있다. stat_start 및/또는 stat_stop 이벤트를 수신하는 동작 블록은 증분 (increment) 신호들이 전송되는 다양한 포인트들에서, 이러한 이벤트들을 수신하는 카운팅을 시작하거나 정지할 것이다.

[0143] GPU (36) 의 드라이버가 이러한 카운터들을 판독할 필요가 있을 때, 드라이버는 도 5b 에 대하여 도시되고 설명된 바와 같이, 커맨드 프로세서 (CP) 를 통해 stat_sample 이벤트를 전송할 수도 있다. 레지스터 백본 관리 (register backbone management; RBBM) 유닛이 카운터들을 증분시키는 것을 담당하는 동작 블록들로부터 수신확인 (또는 "ack") 을 다시 얻을 때까지, CP 는 임의의 추가적인 드로우 콜들을 GPU (36) 로 전송하는 것을 억제할 수도 있다. 일단 "ack" 가 수신되면, RMMB 유닛은 카운터들을 판독할 수도 있고, 다음 드로우 콜 (들) 을 전송하는 것을 재개할 수도 있다.

[0144] GPU (36) 는 다양한 데이터를 로컬 GPU 메모리 (38) 에 저장할 수도 있다. 예를 들어, 다음의 쿼리 카운트은 하드웨어로 CP 에 의해 유지될 수도 있다. 일부의 예들에서, 다음의 쿼리 카운트들은 이하에서 표시된 바와 같이, 다양한 동작 블록들로부터의 1-3 비트 펄스들을 이용하여 증분될 수도 있는 64-비트 카운터들로서 형성될 수도 있다:

[0145] ● IAVertices 는 프리미티브들을 발생시킴에 있어서 이용되는 버텍스들의 수를 지칭할 수도 있다. 따라서, 입력 프리미티브 타입이 삼각형들을 생성하는 스트립일 경우, IAVertices 값은 6 일 수도 있다. 이 값은 윈도우 하드웨어 품질 랩 (Windows Hardware Quality Labs; WHQL) 번호와 일치할 수도 있다. 이 값은 프리미티브 제어기 (PC) 로부터의 2-비트 펄스를 이용하여 제어될 수도 있다. 패치 프리미티브에 대하여, 값은 제어 포인트 당 하나씩 증분될 수도 있다.

[0146] ● IAPrimitives 는 발생된 완전한 입력 프리미티브들의 수를 지칭할 수도 있다. 이 값은 리셋으로 귀착될 수도 있는 임의의 부분적인 프리미티브를 포함하지 않을 수도 있다. 이 값은 WQHL 번호와 일치할 수도 있다. 이 값은 프리미티브가 발생된 후 뿐만 아니라, 리셋 인덱스 및 부분적인 프리미티브 누락들에 대해 검사한 후에도, PC 로부터의 1 비트 펄스를 이용하여 제어될 수도 있다.

[0147] ● VSInvocations 는 VS 동작이 호출되는 횟수를 지칭할 수도 있다. 이 값은 버텍스 재이용 이후에 세팅될 수도 있고, VS 스테이지가 호출되는 고유한 버텍스들의 수를 결정할 수도 있다. 이 값은 GPU (36) 의 특별한 하드웨어에 종속될 수도 있다. PC 는 한번에 3 개에 이르는 버텍스들에 대한 버텍스 재이용에 대해 검사하므로, 이 값은 PC 로부터의 2-비트 펄스를 이용하여 제어될 수도 있다. 전형적으로, (예를 들어, 도 12a

- 도 13b 에 대하여, 예를 들어, 이하에서 설명되는 바와 같은) GS 및 헐 쉐이더 (HS) 경우들에 대한 버텍스 재이용은 없다. 따라서, PC 는 프리미티브들 내의 다수의 버텍스들을 드로우 콜에서 VSInvocations 로서 전송할 수도 있다.

[0148] ● HSInvocations 는 HS 를 통과한 패치들의 수를 지칭할 수도 있다. 이 값은 DirectX 11 과 같은 어떤 API 들에 대한 새로운 값일 수도 있다. 이 값은 임의의 부분적인 패치들을 포함하지 않을 수도 있다. 패치가 버텍스 페치 디코더 (VFD) 로 완전히 전송될 때, 이 값은 PC 로부터 그리고 HS 블록으로부터의 1 비트 펄스를 이용하여 제어될 수도 있다. 이 값은 또한 WHQL 번호와 일치해야 한다.

[0149] ● DSInvocations 는 도메인 쉐이더 (DS) 동작이 호출되는 횟수를 지칭할 수도 있다. 테셀레이션 출력 프리미티브 타입이 타입 포인트일 때, 이 값은 WHQL 과 일치해야 한다. 이 값은 발생되고 있는 각각의 도메인 포인트 (u, v) 에 대한 PC 에서 테셀레이션 엔진 (TE) 으로부터의 1 비트 펄스를 이용하여 제어된다.

[0150] ● GSInvocations 는 GS 동작이 호출되는 횟수를 지칭할 수도 있다. GSinstancecount 값이 이용될 경우, 각각의 인스턴스는 하나의 GS 호출로서 카운팅된다. 이 값은 WHQL 번호와 일치해야 한다. 이 값은 Gsinstance 마다 입력 프리미티브 당 한번 전송되는 GS 블록으로의 1 비트 펄스를 이용하여 제어될 수도 있다.
일부의 예들에서, GS 증폭이 웨이브 사이즈보다 더 클 때, GS 블록은 입력 GS 프리미티브를 몇 번 전송할 수도 있다. 이 값은 전형적으로 GS 입력 프리미티브 당 한번 카운팅된다.

[0151] ● GSPrimitives 는 발생되는 GS 출력 프리미티브들의 수를 지칭할 수도 있다. 이 값은 "컷" 동작으로부터 기인하는 임의의 부분적인 프리미티브들을 포함하지 않을 수도 있다. 이 값은 WHQL 번호와 일치할 수도 있다. 프리미티브들이 구성되는 위치 캐쉬에 대한 액세스 후에, 그리고 "컷" 동작 또는 버텍스 킬 이벤트로 인해 부분적인 프리미티브들을 누락한 후에, 이 값은 PC 로부터의 출력 프리미티브 당 1 비트 펄스를 이용하여 제어될 수도 있다.

[0152] ● CInvocations 는 소위 "클리퍼들" 이 실행되는 횟수를 지칭할 수도 있다. 이 값은 GPU (36) 의 특별한 하드웨어에 종속될 수도 있다.

[0153] ● CPrimitives 는 클리퍼가 발생시킨 프리미티브들의 수를 지칭할 수도 있다. 이 값은 GPU (36) 의 특별한 하드웨어에 종속될 수도 있다.

[0154] ● PSInvocations 는 픽셀 쉐이더 (PS) 스레드들 (또한, "파이버들" 이라고 지칭될 수도 있음) 이 호출되는 횟수를 지칭할 수도 있다.

[0155] ● CSInvocations 는 컴퓨트 파이버들이 호출되는 횟수를 지칭할 수도 있다.

[0156] 위에서 설명된 값들에 추가하여, 스트림 당 유지되는 2 개의 스트림 출력 관련 쿼리 카운트들이 있을 수도 있다. 이 스트림 출력 관련 값들은 다음의 값들을 포함할 수도 있다:

[0157] ● NumPrimitiveWritten 는 드로우 콜이 종료되기 전에 주어진 스트림에 대해 기록된 프리미티브들의 총 수를 지칭할 수도 있다. 버퍼가 완전한 프리미티브에 대한 저장공간을 다 써버릴 때, 이 값은 또한 스트림과 결부된 버퍼의 데이터를 포함할 수도 있다. 전체 프리미티브를 저장하기 위한 주어진 스트림의 버퍼들 중의 임의의 것 내의 공간이 있을 때마다, 이 값은 버텍스 파라미터 캐쉬 (VPC) 로부터 CP 로의 스트림 당 1 비트 펄스를 이용하여 제어될 수도 있다.

[0158] ● PrimitiveStorageNeeded 는, 스트림과 결부된 임의의 버퍼가 저장공간을 다 써버리지 않았을 경우에 기록되었을 수 있었던 프리미티브들의 총 수를 지칭할 수도 있다. 스트림에 대한 프리미티브가 GS 에 의해 발생될 때마다, 이 값은 VPC 로부터 CP 로의 스트림 당 1 비트 펄스를 이용하여 제어될 수도 있다.

[0159] 전형적으로, GPU (36) 는 VPC 로부터 직접적으로의 스트림 출력을 지원할 수도 있다. 위에서 언급된 바와 같이, GS 에 의해 지원되는 4 개에 이르는 스트림들이 있을 수도 있다. 이 스트림들의 각각은 4 개에 이르는 버퍼들에 의해 구속될 수도 있고, 버퍼들은 상이한 스트림들 사이에서 전형적으로 공유가능하지 않다. 각각의 버퍼로의 출력의 사이즈는 버텍스의 최대 사이즈와 동일한 128 디워드들에 이를 수도 있다. 그러나, 스트라이드 (stride) 는 512 디워드들에 이를 수도 있다. 스트림으로부터의 출력 데이터는 다수의 버퍼들에 저장될 수도 있지만, 데이터는 일반적으로 버퍼들 사이에서 복제되지 않을 수도 있다. 예시의 목적들을 위한 예에서, "color.x" 가 스트림에 결부된 버퍼들 중의 하나에 기록될 경우, 이 "color.x" 는 동일한 스트림에 결부된 또 다른 버퍼에 전송되지 않을 수도 있다.

[0160] 버퍼들로 스트림 출력하는 것은 완전한 프리미티브로서 수행될 수도 있다. 즉, 예를 들어, 오직 2 개의 버텍스들에 대한 주어진 스트림을 위한 임의의 버퍼에서 공간이 있고, 프리미티브 타입이 삼각형 (예를 들어, 3 개의 버텍스들을 가짐) 일 경우, 프리미티브 버텍스들은 그 스트림과 결부된 임의의 버퍼에 기록되지 않을 수도 있다.

[0161] GS 가 널 (null) 이고, 스트림 출력이 가능하게 될 경우, 스트림 출력은 디폴트 스트림 제로 (default stream zero) 로서 식별될 수도 있다. 스트림 출력이 수행되고 있을 때, 위치 정보는 VPC 뿐만 아니라, 여분의 슬롯을 소비할 수도 있는 PC 에도 기록될 수도 있다. 추가적으로, 비닝 (binning) 이 수행될 때 (예를 들어, 타일 기반 렌더링을 위하여 버텍스들을 빈 (bin) 들에 배정하는 프로세스), 스트림 출력이 비닝 패스 동안에 수행될 수도 있다.

[0162] DirectX 10 과 같은 일부의 API 들에서는, 스트림 출력 데이터를 소비하는 DrawAuto 기능 (이전에 생성된 스트림들을 패치하고 렌더링할 수도 있음) 이 특정될 수도 있다. 예를 들어, GPU 드라이버는 주어진 스트림에 대한 스트림 출력 플러쉬 (stream out flush) 를 위한 이벤트를 메모리 어드레스와 함께 전송할 수도 있다. 이러한 이벤트를 수신할 시에, VPC 는 수신확인 (ack) 비트를 RBBM 으로 전송할 수도 있다. ack 비트를 수신할 시에, RBBM 은 버퍼에서 이용가능한 버퍼 공간의 양 (버퍼링되는 충전된 사이즈) 을 드라이버 특정된 메모리 또는 메모리 로케이션에 기록한다.

[0163] 그 동안에, 커맨드 프로세서 (CP) 내에 포함될 수도 있는 프리-페치 파서 (PFP) 는 임의의 드로우 콜을 전송하기 위하여 대기한다. 일단 메모리 어드레스가 기록되면, PFP 는 다음 드로우 콜을 전송할 수도 있다. 다음 드로우 콜이 자동 드로우 콜 (auto draw call) 일 경우, GPU 드라이버는 드로우 콜들 및 상태 변화들을 표시하는 패킷 (예를 들어, 소위 "PM4" 패킷) 의 일부로서 버퍼 충전된 사이즈를 포함하는 메모리 어드레스를 전송할 수도 있다. PFP 는 그 메모리 로케이션으로부터 buffer_filled_size 를 판독하고, 드로우 콜을 PC 로 전송한다.

[0164] 도 7 은 이 개시물의 양태들에 따라, 버텍스 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행하기 위한 일 예의 프로세스를 예시하는 플로우차트이다. GPU (36; 도 1) 에 의해 수행되는 것으로 설명되었지만, 도 7 에 대하여 설명된 기술들은 다양한 GPU 들 또는 다른 프로세싱 유닛들에 의해 수행될 수도 있다.

[0165] GPU (36) 는 예를 들어, 버텍스 쉐이더 명령들을 수신할 시에, 버텍스 쉐이딩 동작들을 초기에 호출할 수도 있다 (210). 버텍스 쉐이딩 동작들을 호출하는 것은 GPU (36) 로 하여금 버텍스 쉐이딩 동작들을 위하여 하나 이상의 쉐이딩 유닛들 (40) 을 지정하게 할 수도 있다. 추가적으로, (버텍스 파라미터 캐쉬, 래스터라이저, 등과 같은) GPU (36) 의 다른 컴포넌트들은 지정된 쉐이딩 유닛들 (40) 의 각각으로부터 입력 당 단일 출력을 수신하도록 구성될 수도 있다.

[0166] GPU (36) 는 버텍스 쉐이딩 동작들을 위해 지정된 하드웨어 쉐이딩 유닛들로, 입력 버텍스들을 쉐이딩하기 위하여 버텍스 쉐이딩 동작들을 수행할 수도 있다 (212). 즉, 하드웨어 쉐이딩 유닛은 입력 버텍스들을 쉐이딩하고 버텍스 쉐이딩된 인덱스들을 출력하기 위하여 버텍스 쉐이딩 동작들을 수행할 수도 있다. 하드웨어 쉐이딩 유닛은 하나의 버텍스를 수신할 수도 있고 하나의 쉐이딩된 버텍스를 출력할 수도 있다 (예를 들어, 입력 및 출력 사이의 1:1 관계).

[0167] GPU (36) 는 지오메트리 쉐이딩 동작들을 수행할 것인지 여부를 결정할 수도 있다 (214). GPU (36) 는 예를 들어, 모드 정보에 기초하여 이러한 결정을 행할 수도 있다. 즉, GPU (36) 는 임의의 유효한 지오메트리 쉐이더 명령들이 실행된 버텍스 쉐이더 명령들에 첨부되는지 여부를 결정하기 위하여 패치 코드를 실행할 수도 있다.

[0168] GPU (36) 가 지오메트리 쉐이딩 동작들을 수행하지 않는 경우 (단계 (214) 의 아니오 브랜치), GPU 하드웨어 쉐이딩 유닛은 각각의 입력 버텍스에 대하여 하나의 쉐이딩된 버텍스를 출력할 수도 있다 (222). GPU (36) 가 지오메트리 쉐이딩 동작들을 수행할 경우 (단계 (214) 의 예 브랜치), 하드웨어 쉐이딩 유닛은 수신된 버텍스들에 기초하여 하나 이상의 새로운 버텍스들을 발생시키기 위하여 지오메트리 쉐이딩 동작들의 다수의 인스턴스들을 수행할 수도 있다 (216). 예를 들어, 하드웨어 쉐이딩 유닛은 미리 결정된 수의 지오메트리 쉐이딩 인스턴스들을 수행할 수도 있고, 각각의 인스턴스는 출력 식별자와 연관될 수도 있다. 하드웨어 쉐이딩 유닛은 지오메트리 쉐이딩 동작들의 각각의 인스턴스에 대한 출력 카운트를 유지할 수도 있다. 추가적으로, 출력 식별자는 각각의 출력 버텍스에 배정될 수도 있다.

[0169] 따라서, 지오메트리 쉐이딩된 버텍스를 언제 출력할 것인지를 결정하기 위하여, 하드웨어 쉐이딩 유닛은 출력

카운트가 출력 식별자와 언제 일치하는지를 결정할 수도 있다 (218). 예를 들어, 지오메트리 쉐이딩 동작에 대한 출력 카운트가 출력 식별자와 일치하지 않을 경우 (단계 (218) 의 아니오 브랜치), 지오메트리 쉐이딩 동작과 연관된 버텍스는 폐기된다. 지오메트리 쉐이딩 동작에 대한 출력 카운트가 출력 식별자와 일치할 경우 (단계 (218) 의 예 브랜치), 하드웨어 쉐이딩 유닛은 지오메트리 쉐이딩 동작과 연관된 버텍스를 출력할 수도 있다. 이러한 방법으로, 버텍스 쉐이딩을 위해 지정된 하드웨어 쉐이딩 유닛은 단일 쉐이딩된 버텍스를 출력하고 지오메트리 쉐이딩 프로그램의 각각의 인스턴스에 대한 임의의 이용되지 않은 버텍스들을 폐기하고, 이것에 의하여, 1:1 입력 대 출력 비율을 유지한다.

[0170] 도 8 은 테셀레이션 스테이지들을 포함하는 일 예의 그래픽스 프로세싱 파이프라인 (238) 을 예시하는 블록도이다. 예를 들어, 파이프라인 (238) 은 입력 어셈블러 스테이지 (240), 버텍스 쉐이더 스테이지 (242), 헐 쉐이더 스테이지 (244), 테셀레이터 스테이지 (246), 도메인 쉐이더 스테이지 (248), 지오메트리 쉐이더 스테이지 (250), 래스터라이저 스테이지 (252), 픽셀 쉐이더 스테이지 (254), 및 출력 병합기 스테이지 (256) 를 포함한다. 일부의 예들에서, DirectX 11 API 와 같은 API 는 도 8 에 도시된 스테이지들의 각각을 이용하도록 구성될 수도 있다. 그래픽스 프로세싱 파이프라인 (238) 은 GPU (36) 에 의해 수행되는 것으로 이하에서 설명되지만, 다양한 다른 그래픽스 프로세서들에 의해 수행될 수도 있다.

[0171] 도 8 에 도시된 어떤 스테이지들은 도 2 에 대하여 도시되고 설명된 스테이지들과 유사하거나 동일하게 구성될 수도 있다 (예를 들어, 어셈블러 스테이지 (240), 버텍스 쉐이더 스테이지 (242), 지오메트리 쉐이더 스테이지 (250), 래스터라이저 스테이지 (252), 픽셀 쉐이더 스테이지 (254), 및 출력 병합기 스테이지 (256)). 추가적으로, 파이프라인 (238) 은 하드웨어 테셀레이션을 위한 추가적인 스테이지들을 포함한다. 예를 들어, 그래픽스 프로세싱 파이프라인 (238) 은 도 2 에 대하여 위에서 설명된 스테이지들에 추가하여, 헐 쉐이더 스테이지 (244), 테셀레이터 스테이지 (246), 및 도메인 쉐이더 스테이지 (248) 를 포함한다. 즉, 헐 쉐이더 스테이지 (244), 테셀레이터 스테이지 (246), 및 도메인 쉐이더 스테이지 (248) 는 예를 들어, CPU (32) 에 의해 실행되는 소프트웨어 애플리케이션에 의해 수행되는 것이 아니라, GPU (36) 에 의한 테셀레이션을 수용하도록 포함된다.

[0172] 헐 쉐이더 스테이지 (244) 는 버텍스 쉐이더 스테이지 (242) 로부터 프리미티브들을 수신하고, 적어도 2 개의 작동들을 수행하는 것을 담당한다. 첫째, 헐 쉐이더 스테이지 (244) 는 전형적으로 테셀레이션 인자들의 세트를 결정하는 것을 담당한다. 헐 쉐이더 스테이지 (244) 는 프리미티브 당 한번 테셀레이션 인자들을 발생시킬 수도 있다. 주어진 프리미티브 (예를 들어, 더 작은 부분들로 분할된 프리미티브) 를 얼마나 세밀하게 테셀레이팅할 것인지를 결정하기 위하여, 테셀레이션 인자들은 테셀레이터 스테이지 (246) 에 의해 이용될 수도 있다. 헐 쉐이더 스테이지 (244) 는 또한, 도메인 쉐이더 스테이지 (248) 에 의해 더 이후에 이용될 제어 포인트들을 발생시키는 것을 담당한다. 즉, 예를 들어, 헐 쉐이더 스테이지 (244) 는, 궁극적으로 렌더링함에 있어서 이용되는 실제의 테셀레이팅된 버텍스들을 생성하기 위하여 도메인 쉐이더 스테이지 (248) 에 의해 이용될 제어 포인트들을 발생시키는 것을 담당한다.

[0173] 테셀레이터 스테이지 (246) 가 헐 쉐이더 스테이지 (244) 로부터 데이터를 수신할 때, 테셀레이터 스테이지 (246) 는 현재의 프리미티브 타입에 대한 적절한 샘플링 패턴을 결정하기 위하여 몇몇 알고리즘들 중의 하나를 이용한다. 예를 들어, 일반적으로, 테셀레이터 스테이지 (246) 는 (헐 쉐이더 스테이지 (244) 에 의해 결정된 바와 같은) 테셀레이션의 요청된 양을 현재의 "도메인" 내의 좌표 포인트들의 그룹으로 변환한다. 즉, 헐 쉐이더 스테이지 (244) 로부터의 테셀레이션 인자들뿐만 아니라, 테셀레이터 스테이지 (246) 의 특별한 구성에 따라, 테셀레이터 스테이지 (246) 는 입력 프리미티브를 더 작은 부분들로 테셀레이팅하기 위하여 현재의 프리미티브 내의 어느 포인트들이 샘플링될 필요가 있는지를 결정한다. 테셀레이터 스테이지의 출력은 무게중심 좌표 (barycentric coordinate) 들을 포함할 수도 있는 도메인 포인트들의 세트일 수도 있다.

[0174] 도메인 쉐이더 스테이지 (248) 는 헐 쉐이더 스테이지 (244) 에 의해 생성된 제어 포인트들에 추가하여, 도메인 포인트들을 취하고, 새로운 버텍스들을 생성하기 위하여 도메인 포인트들을 이용한다. 도메인 쉐이더 스테이지 (248) 는, 각각의 테셀레이팅된 포인트에 대한 무게중심 "로케이션" 을, 파이프라인에서 다음 스테이지로 전달되는 출력 지오메트리로 변환하기 위하여, 현재의 프리미티브, 텍스처들, 절차 알고리즘들, 또는 그 밖의 것에 대해 발생된 제어 포인트들의 완전한 리스트를 이용할 수 있다. 위에서 언급된 바와 같이, 어떤 GPU 들은 도 8 에 도시된 쉐이더 스테이지들의 모두를 지원하지 못할 수도 있다. 예를 들어, 일부의 GPU 들은 하드웨어 및/또는 소프트웨어 한정들 (예를 들어, 제한된 수의 쉐이딩 유닛들 (40) 및 연관된 컴포넌트들) 로 인해, 2 개를 초과하는 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들을 지정하지 못할 수도 있다. 일 예에서, 어떤 GPU 들은 지오메트리 쉐이더 스테이지 (250), 헐 쉐이더 스테이지 (244), 및 도메인 쉐이더 스테이지

(248) 와 연관된 동작들을 지원하지 않을 수도 있다. 오히려, GPU 들은 버텍스 쉐이더 스테이지 (242) 및 픽셀 쉐이더 스테이지 (252) 를 수행하도록 쉐이딩 유닛들을 지정하기 위한 지원을 포함하기만 할 수도 있다.

따라서, 쉐이딩 유닛들에 의해 수행되는 동작들은 버텍스 쉐이더 스테이지 (84) 및 픽셀 쉐이더 스테이지 (90) 와 연관된 입력/출력 인터페이스를 고수해야 한다.

[0175] 추가적으로, 상대적으로 더 긴 그래픽스 프로세싱 파이프라인을 지원하는 것은 상대적으로 더 복잡한 하드웨어 구성을 요구할 수도 있다. 예를 들어, 헐 쉐이더 스테이지 (244), 테셀레이터 스테이지 (246), 및 도메인 쉐이더 스테이지 (248) 로부터의 제어 포인트들, 도메인 포인트들, 및 테셀레이션 인자들은 오프-칩 메모리에 대한 판독들 및 기록들을 요구할 수도 있고, 이것은 메모리 버스 대역폭을 소비할 수도 있고, 소비되는 전력의 양을 증가시킬 수도 있다. 이러한 의미에서, 각각의 쉐이더 스테이지에 대한 전용 쉐이딩 유닛들 (40) 을 이용하여 다수의 스테이지들을 갖는 그래픽스 파이프라인을 구현하는 것은 덜 전력 효율적인 GPU 들로 귀착될 수도 있다. 추가적으로, 이러한 GPU 들은 또한, 제한된 메모리 버스 대역폭의 결과로서 오프-칩 메모리로부터 데이터를 취출함에 있어서의 지연으로 인해 렌더링된 이미지들을 출력하는 측면에서 더 느릴 수도 있다.

[0176] 이 개시물의 양태들에 따르면, 이하에서 더욱 상세하게 설명된 바와 같이, 특별한 쉐이딩 동작을 수행하도록 GPU (36) 에 의해 지정된 쉐이딩 유닛들 (40) 은 하나를 초과하는 동작을 수행할 수도 있다. 예를 들어, 버텍스 쉐이딩 (VS) 동작들을 수행하도록 지정된 쉐이딩 유닛 (40) 은 또한, 헐 쉐이더 스테이지 (244) 와 연관된 헐 쉐이딩 동작들을 수행할 수도 있다. 또 다른 예에서, 동일한 쉐이딩 유닛 (40) 은 또한, 도메인 쉐이더 스테이지 (248) 와 연관된 도메인 쉐이딩 동작들, 그 다음으로, 지오메트리 쉐이더 스테이지 (250) 와 연관된 지오메트리 쉐이더 동작들을 수행할 수도 있다.

[0177] 이하에서 더욱 상세하게 설명된 바와 같이, GPU (36) 는 드로우 콜을 2 개의 서브-드로우 콜들 (예를 들어, 패스 I 및 패스 II) 로 분해함으로써 상기 쉐이딩 동작들을 수행할 수도 있고, 각각의 서브-드로우 콜은 연관되는 병합된 쉐이더 스테이지들을 가질 수도 있다. 즉, GPU (36) 는 버텍스 쉐이딩 동작들을 수행하기 위하여 쉐이딩 유닛 (40) 을 호출할 수도 있지만, 제 1 패스 동안에 헐 쉐이딩 동작들을 수행하기 위하여 쉐이딩 유닛 (40) 을 또한 구현할 수도 있다. 다음으로, GPU (36) 는 헐 쉐이딩, 도메인 쉐이딩, 또는 지오메트리 쉐이딩 태스크들을 수행하도록 쉐이딩 유닛 (40) 을 결코 재지정하지 않으면서, 도메인 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행하기 위하여 (버텍스 쉐이딩 동작들을 수행하도록 지정된) 동일한 쉐이딩 유닛 (40) 을 이용할 수도 있다.

[0178] 도 9 는 테셀레이션을 더욱 상세하게 예시하는 개념적인 도면이다. 헐 쉐이더 (HS) 스테이지 (244) 및 도메인 쉐이더 (DS) (248) 는 완전히 자격을 갖춘 쉐이더 스테이지들일 수도 있고, 그 각각은 상수 버퍼들, 텍스처들, 및 다른 자원들의 그 자신의 세트를 가질 수도 있다. 일반적으로, 테셀레이션은 패치라고 지칭되는 프리미티브 타입을 이용하여 수행될 수도 있다. 따라서, 도 9 에 도시된 예에서, 헐 쉐이더 스테이지 (244) 는 초기에, 패치 제어 포인트들이라고 지칭될 수도 있는 하나 이상의 입력 제어 포인트들을 수신한다. 패치 제어 포인트들은 (예를 들어, API 를 이용하여) 개발자에 의해 제어될 수도 있다. 헐 쉐이더 스테이지 (244) 는, 이하에서 설명되는 바와 같이, 도메인 쉐이더 스테이지 (248) 에 의해 이용되는 제어 포인트들을 포함하는 소위 베지어 패치 (Bezier patch) 를 발생시키기 위하여 계산들을 수행할 수도 있다.

[0179] 헐 쉐이더 스테이지 (244) 는 또한, 패치의 테셀레이션의 양을 제어하기 위하여 이용될 수도 있는 테셀레이션 인자들을 발생시킨다. 예를 들어, 헐 쉐이더 스테이지 (244) 는 패치의 뷰포인트 및/또는 뷰 거리에 기초하여 얼마나 많이 테셀레이팅할 것인지를 결정할 수도 있다. 오브젝트가 장면에서 뷰어에게 상대적으로 근접할 경우, 일반적으로 평탄하게 보이는 패치를 생성하기 위하여 상대적으로 높은 양의 테셀레이션이 요구될 수도 있다. 오브젝트가 상대적으로 멀리 떨어져 있을 경우, 더 적은 테셀레이션이 요구될 수도 있다.

[0180] 테셀레이터 스테이지 (246) 는 테셀레이션 인자들을 수신하고 테셀레이션을 수행한다. 예를 들어, 테셀레이터 스테이지 (246) 는 다수의 {U, V} 좌표들을 발생시키기 위하여 균일한 등급을 갖는 주어진 패치 (예를 들어, 베지어 패치) 에 대해 작용한다. {U, V} 좌표들은 패치에 대한 텍스처를 제공할 수도 있다. 따라서, 도메인 쉐이더 스테이지 (248) 는 제어 포인트들 (변위 정보를 가짐) 및 {U, V} 좌표들 (텍스처 정보를 가짐) 을 수신할 수도 있고, 테셀레이팅된 버텍스들을 출력할 수도 있다. 다음으로, 이 테셀레이팅된 버텍스들은 위에서 설명된 바와 같이 지오메트리 쉐이딩될 수도 있다.

[0181] 이 개시물의 양태들에 따르면, 그리고 이하에서 더욱 상세하게 설명되는 바와 같이, 헐 쉐이더 스테이지 (244) 및 도메인 쉐이더 스테이지 (248) 와 연관된 쉐이딩 동작들은 (쉐이딩 유닛들 (40) 과 같은) GPU 의 동일한 쉐이딩 유닛들에 의해 수행될 수도 있다. 즉, 예를 들어, 하나 이상의 쉐이딩 유닛들 (40) 은 버텍스 쉐이딩

동작들을 수행하도록 지정될 수도 있다. 버텍스 쉐이딩 동작들에 추가하여, 쉐이더들이 순차적으로 그리고 테셀레이션 동작들을 수행하도록 재구성되지 않으면서 동일한 쉐이딩 유닛들에 의해 실행되도록, GPU 는 헐 쉐이더 스테이지 (244) 및 도메인 쉐이더 스테이지 (248) 와 연관된 쉐이더 명령들을 첨부할 수도 있다.

[0182] 도 10a 및 도 10b 는 이 개시물의 양태들에 따라, 그래픽스 렌더링 파이프라인에서의 데이터 흐름들의 개념적인 도면이다. 예를 들어, 도 10a 는 버텍스 쉐이더 스테이지 (260), 헐 쉐이더 스테이지 (262), 테셀레이터 스테이지 (264), 도메인 쉐이더 스테이지 (266), 지오메트리 쉐이더 스테이지 (268), 스트림 출력 (270), 및 픽셀 쉐이더 스테이지 (272) 를 예시한다. 일반적으로, 도 10a 에 도시된 쉐이더 스테이지들의 각각은 쉐이딩 동작들을 수행하기 위한 연관된 하드웨어를 나타낸다. 즉, 예를 들어, 버텍스 쉐이더 스테이지 (260), 헐 쉐이더 스테이지 (262), 도메인 쉐이더 스테이지 (266), 지오메트리 쉐이더 스테이지 (268), 및 픽셀 쉐이더 스테이지 (272) 의 각각은 쉐이딩 유닛들 (40) 과 같은, 별도로 지정된 프로세싱 유닛들과 연관될 수도 있다.

[0183] 도 10a 에 도시된 예에서, 버텍스 쉐이더 스테이지 (260) 는 소위 "패치 제어 포인트들" (또는 도 8 및 도 9 에 대하여 위에서 설명된 바와 같은 "제어 포인트들") 상에서 호출될 수도 있다. 주어진 패치에서의 포인트들은 헐 쉐이더 스테이지 (252) 에 가시적일 수도 있고, 이 헐 쉐이더 스테이지 (262) 는 테셀레이션 스테이지 (264) 에 의한 이용을 위한 테셀레이션 인자들을 컴퓨팅하기 위하여 포인트들을 이용한다. 헐 쉐이더 스테이지 (262) 는 또한, 도메인 쉐이더 스테이지 (266) 에 의한 이용을 위한 패치 제어 포인트들 및 상수 데이터를 출력할 수도 있다.

[0184] 일부의 예들에서, 테셀레이터 스테이지 (264) 는 테셀레이션을 수행하기 위한 고정된 기능 하드웨어 유닛들을 포함할 수도 있다. 테셀레이터 스테이지 (264) 는 헐 쉐이더 스테이지 (262) 로부터 테셀레이션 인자들 및 제어 포인트들을 수신할 수도 있고, 소위 도메인 포인트들 (예를 들어, 어디를 테셀레이팅할 것인지를 특정하는 {U, V} 포인트들) 을 출력할 수도 있다. 도메인 쉐이더 스테이지 (266) 는 헐 쉐이더 스테이지 (262) 로부터의 출력 패치 데이터를 이용하여 버텍스들을 컴퓨팅하기 위하여 이 도메인 포인트들을 이용한다. 도메인 쉐이더 스테이지 (266) 로부터의 가능한 출력 프리미티브들은 예를 들어, 래스터라이제이션을 위하여 스트림 출력 (270) 또는 지오메트리 쉐이더 스테이지 (268) 로 전송될 수도 있는 포인트, 라인, 또는 삼각형을 포함한다. 테셀레이션 인자들 중의 임의의 것이 제로 이하이거나, 어떤 수 (NaN) 가 아닐 경우, 패치는 컬링 (culling) 될 수도 있다 (더 이상 컴퓨팅되지 않고 폐기됨).

[0185] 도 10a 에 도시된 쉐이더 스테이지들은 하나 이상의 그래픽스 API 들을 지원할 수도 있다. 예시의 목적들을 위한 예에서, 버텍스 쉐이더 스테이지 (260), 헐 쉐이더 스테이지 (262), 도메인 쉐이더 스테이지 (266), 지오메트리 쉐이더 스테이지 (268) 및 픽셀 쉐이더 스테이지 (272) 는 DirectX 11 API 를 지원할 수도 있다. 즉, DirectX 11 API 를 이용하여 생성된 코드는 그래픽스 데이터를 렌더링하기 위하여 버텍스 쉐이더 스테이지 (260), 헐 쉐이더 스테이지 (262), 도메인 쉐이더 스테이지 (266), 지오메트리 쉐이더 스테이지 (268) 및 픽셀 쉐이더 스테이지 (272) 에 의해 실행될 수도 있다. 그러나, 헐 쉐이더 스테이지 (262), 도메인 쉐이더 스테이지 (266), 및/또는 지오메트리 쉐이더 스테이지 (268) 와 같은 어떤 스테이지들은 모든 그래픽스 렌더링 파이프라인들에 포함되지 않을 수도 있고, 모든 GPU 들에 의해 실행되지 않을 수도 있다. 예를 들어, DirectX 11 API 는 이러한 스테이지들에 대한 지원을 포함하지만, 더 이전의 수정본들 (예를 들어, DirectX 9 및 10) 은 이러한 지원을 포함하지 않는다. 따라서, DirectX API 의 더 이전의 수정본들로 생성된 코드를 실행하도록 설계된 GPU 들 (또는 다른 API 들을 위해 설명된 GPU 들) 은 헐 쉐이더 스테이지 (262), 도메인 쉐이더 스테이지 (266), 및/또는 지오메트리 쉐이더 스테이지 (268) 와 연관된 동작들을 수행하도록 쉐이딩 유닛들 (40) 을 지정할 수 없을 수도 있다.

[0186] 이 개시물의 양태들에 따르면, 쉐이더 스테이지들이 (예를 들어, 쉐이딩 유닛 (40) 과 같은) 단일 하드웨어 쉐이딩 유닛에 의해 수행된다는 점에서, 도 10a 에서의 쉐이더 스테이지들 중의 하나를 초과하는 것은 병합될 수도 있다. 예를 들어, 이 개시물의 양태들에 따르면, (GPU (36) 와 같은) GPU 는 도 10b 에 대하여 이하에서 설명되는 바와 같이, 도 10a 에 도시된 쉐이더 스테이지들을 수행하기 위하여 드로우 콜을 실행할 때에 다수의 패스들을 수행할 수도 있다.

[0187] 도 10b 는 병합된 버텍스 쉐이더 및 헐 쉐이더 (VS/HS) 스테이지 (280) 를 갖는 제 1 패스 (패스 I) 를 포함하는 그래픽스 렌더링 파이프라인에서의 데이터 흐름을 예시한다. 추가적으로, 데이터 흐름은 테셀레이션 스테이지 (282), 병합된 도메인 쉐이더 및 지오메트리 쉐이더 (DS/GS) 스테이지 (284), 스트림 출력 (286), 및 픽셀 쉐이더 스테이지 (288) 를 갖는 제 2 패스 (패스 II) 를 포함한다. 도 10b 에 도시된 패스들은 테셀레이션 동작들을 갖는 드로우 콜을 실행하도록 구현될 수도 있다.

[0188] 예를 들어, GPU (36) 는 도 10a 에 대하여 위에서 설명된 바와 같이, 테셀레이션 동작들을 포함하는 입력 드로우 콜을 실행할 수도 있다. GPU (36) 는 초기에 드로우 콜을 다수의 서브-드로우 콜들로 분해할 수도 있고, 각각의 서브-드로우 콜은 패스 I 동작들 및 패스 II 동작들을 포함할 수도 있다. GPU (36) 가 드로우 콜을 분할하는 방식은 이용가능한 메모리의 양 (예를 들어, 온-칩 GPU 메모리, L2, 글로벌 메모리 (global memory; GMEM), 또는 오프칩 메모리) 에 적어도 부분적으로 종속될 수도 있다. 예를 들어, GPU (36) 가 패스 I 동작들에 의해 발생된 데이터의 전부를 패스 II 동작들과 함께 이용하기 위하여 로컬 메모리에 저장할 수 있도록, GPU (36) 는 서브-드로우 콜들을 구성할 수도 있다. 드로우 콜의 분할은 입력 드로우 콜 타입에 기초할 수도 있는 CP 코드의 제어 하에서 커맨드 프로세서 (CP) 에서 행해질 수도 있다.

[0189] 예시의 목적들을 위한 예에서, 드로우 콜이 렌더링을 위한 1000 개의 연관된 패치들을 포함하는 것으로 가정한다. 추가적으로, 로컬 메모리는 100 개의 패치들과 연관된 데이터를 저장하기 위한 용량을 가지는 것으로 가정한다. 이 예에서, GPU (36) (또는, GPU 드라이버 (50) 와 같이, GPU 를 위한 드라이버) 는 드로우 콜을 10 개의 서브-드로우 콜들로 분할할 수도 있다. 다음으로, GPU (36) 는 10 개의 서브-드로우 콜들의 각각에 대하여 패스 I 동작들 및 패스 II 동작들을 순차적으로 수행한다.

[0190] 패스 I 동작들에 대하여, 버텍스 쉐이딩 동작들이 GPU (36) 에 의해 호출될 시에, VS/HS 스테이지 (280) 는 버텍스 쉐이딩 동작들 및 헐 쉐이딩 동작들의 둘 모두를 수행할 수도 있다. 즉, 병합된 VS/HS 스테이지 (280) 는 하나 이상의 쉐이딩 유닛들의 단일 세트를 포함할 수도 있고, 버텍스 쉐이더 스테이지 (260) 및 헐 쉐이더 스테이지 (262) 에 대하여 위에서 설명된 동작들을 순차적으로 수행할 수도 있다. 이하에서 더욱 상세하게 설명된 바와 같이, 이 개시물의 양태들은 GPU (36) 가 적절한 인터페이스를 여전히 고수하면서, 버텍스 쉐이딩 동작들과 동일한 쉐이딩 유닛으로 헐 쉐이딩 동작들을 수행하도록 한다. 일부의 예들에서, 헐 쉐이더 명령들은 패치 코드를 이용하여 버텍스 쉐이더 명령들에 첨부될 수도 있고, 이것에 의하여, 동일한 쉐이딩 유닛이 명령들의 두 세트들을 실행하도록 할 수도 있다.

[0191] 다음으로, GPU (36) 는 패스 II 동작들을 수행할 수도 있다. 예를 들어, 테셀레이션 스테이지 (282) 는 상기 테셀레이션 스테이지 (264) 에 대하여 설명된 바와 같이, 테셀레이션을 수행할 수도 있다. 병합된 DS/GS 스테이지 (284) 는 위에서 설명된 병합된 VS/HS 스테이지 (280) 와 동일한 하나 이상의 쉐이딩 유닛들 (40) 의 세트를 포함할 수도 있다. 병합된 DS/GS 스테이지 (284) 는 도메인 쉐이더 스테이지 (266) 및 지오메트리 쉐이더 스테이지 (268) 에 대하여 위에서 설명된 도메인 쉐이딩 및 지오메트리 쉐이딩 동작들을 순차적으로 수행할 수도 있다. 일부의 예들에서, 지오메트리 쉐이더 명령들은 패치 코드를 이용하여 도메인 쉐이더 명령들에 첨부될 수도 있고, 이것에 의하여, 동일한 쉐이딩 유닛이 명령들의 두 세트들을 실행하도록 할 수도 있다. 또한, 이 도메인 쉐이더 명령들 및 지오메트리 쉐이더 명령은 (패스 I 의) 헐 쉐이더 명령들에 첨부될 수도 있어서, 동일한 쉐이딩 유닛이 재구성되지 않고도, 버텍스 쉐이딩, 헐 쉐이딩, 도메인 쉐이딩, 및 지오메트리 쉐이딩을 수행할 수도 있다.

[0192] 패스 II 지오메트리 쉐이딩 동작들은 위에서 설명된 것들과 동일한 지오메트리 쉐이딩 동작들을 본질적으로 포함할 수도 있다. 그러나, 패스 II 동작들을 시작할 때, (이전에는 VS 스테이지에 대한 것이고, 지금은 DS 스테이지에 대한) GPR 초기화된 입력은 버텍스 페치 디코더 (VFD) 로부터의 페치된 데이터가 아니라, 테셀레이션 스테이지 (282) 에 의해 생성된 (u, v, patch_id) 를 포함할 수도 있다. PC 는 또한 패스 II 에 대하여 rel_patch_id 를 컴퓨팅할 수도 있고, 테셀레이션 스테이지 (282) 에 의해 컴퓨팅된 (u, v) 와 함께, 패치 ID 정보를 DS 로 전달할 수도 있다. 테셀레이션 스테이지 (282) 는 테셀레이팅된 버텍스들에 대한 (u, v) 좌표들을 생성하기 위하여 테셀레이션 인자들을 이용할 수도 있다. 테셀레이션 스테이지 (282) 의 출력은 추가의 증폭 (지오메트리 쉐이딩) 또는 스트림 출력 (286) 에 대해 테셀레이팅되도록 준비하기 위하여 병합된 DS/GS 스테이지 (284) 로 공급될 수 있다. DS 는 오프-칩 스크래치 메모리로부터의 헐 쉐이더 (HS) 출력 제어 포인트 데이터 및 HS 패치 상수 데이터를 이용한다.

[0193] 일부의 예들에서, 도 10b 에 도시된 2 개의 패스들은 연속적으로 수행될 수도 있지만, 2 개의 패스들 사이의 유휴상태 대기 (wait for idle) 에 의해 분리될 수도 있다. 예를 들어, GPU 의 CP 는 패스 I 동작들에 대한 드로우 콜을 전송할 수도 있다. 데이터에 대하여 패스 II 를 시작하기 전에, GPU 는 로컬 메모리에 완전히 기록될 제어 포인트 값들을 대기할 수도 있다. 정확한 값들이 로컬 메모리에서 이용가능한 것을 보장하기 위하여, GPU 는 패스 II 동작들을 시작하기 전에 GPU 의 컴포넌트들이 유휴상태인 것을 확인할 수도 있다.

[0194] 다음으로, 커맨드 프로세서 (CP) 는 패스 II 에 대한 드로우 콜을 전송할 수도 있다. 일 예에서, 제 1 유용한 버텍스를 시작하기 위한 지연시간 (latency) 의 양 대 패스 II 에서 행해진 작업의 양의 비율은 대략 2 % 미

만일 수도 있다. 따라서, 일부의 예들에서는, 패스 I 및 패스 II 사이에 중첩이 전혀 없을 수도 있다. 다른 예들에서는, 이하에서 설명된 바와 같이, GPU 는 패스 I 및 패스 II 동작들 사이에서 중첩을 포함할 수도 있다. 즉, 픽셀 쉐이더 프로세싱은 버텍스 쉐이더 프로세싱보다 더 오래 걸릴 수도 있으므로, GPU 는 이전 드로우 콜의 패스 II 의 픽셀 쉐이더 스테이지 (288) 의 픽셀 쉐이딩 동작들을, 현재의 드로우 콜의 패스 I 의 VS/HS 스테이지 (280) 의 버텍스 쉐이딩 동작들과 중첩시킬 수도 있다.

[0195] 이 개시물의 양태들에 따르면, 프리미티브 제어기 (PC) 는 패스 I 후에 PASS_done 이벤트를 전송할 수도 있고, 이것은 하드웨어 유닛이 패스 II 로 스위칭하는 것을 도울 수도 있다. 패스 I 및 패스 II 사이에서 중첩이 있을 수도 있는 예에서, 패스 I 동작들 및 패스 II 동작들의 존재는 명령들을 실행하는 쉐이더 프로세서에서 상호 배타적일 수도 있다. 그러나, 패스 II 에 대한 테셀레이션 인자들은 패스 I 이 여전히 실행하고 있을 동안에 페치될 수도 있다.

[0196] 도 11 에 대하여 이하에서 설명되는 바와 같이, PC 는 얼마나 많은 패스 I 웨이브들이 완료되는지를 레코딩하기 위하여 쉐이딩된 패치 당 카운터를 유지할 수도 있다. 이 카운터들은 얼마나 많은 패치들이 패스 I 에 대한 프로세싱을 완료하였는지를 표시할 수도 있다. 모든 카운터 값들이 제로보다 더 크자마자, 테셀레이션 인자들은 패스 II 에 대하여 페치될 수도 있다. 따라서, 패스 II 는 패스 I 이 완료되기 전에 시작할 수도 있다. 그러나, 패스 I 드로우 콜에 대한 인덱스들의 전부가 프로세싱될 때까지, 패스 II 에 대한 드로우 콜은 프로세싱을 시작하지 않을 수도 있다. 이러한 방법으로, 패스들 사이의 파이프라인 플러싱 (pipeline flushing) (로컬 GPU 메모리로부 외부 메모리로의 전송) 이 회피될 수도 있다.

[0197] 도 11 은 버텍스 쉐이딩 및 헐 쉐이딩 동작들을 수행하기 위하여 이 개시물에서 설명된 기술들을 구현하는 하드웨어 쉐이딩 유닛의 일 예의 동작들을 예시하는 도면이다. 예를 들어, 도 11 은 이 개시물의 기술들에 따라, 도 10b 에 대하여 위에서 설명된 바와 같이, 드로우 콜의 제 1 패스 (패스 I) 동안에 버텍스 쉐이딩 동작들 및 헐 쉐이딩 동작들을 수행하는 것을 일반적으로 예시한다. GPU (36; 도 1) 에 대하여 설명되었지만, 이 개시물의 양태들은 다양한 다른 컴포넌트들을 갖는 다양한 다른 GPU 들에 의해 수행될 수도 있다.

[0198] 도 11 의 예에서, GPU (36) 는 이러한 쉐이딩 동작들을 수행하도록 재구성되지 않고도, 이하에서 더욱 상세하게 설명된 바와 같이, 헐 쉐이딩, 도메인 쉐이딩, 및 지오메트리 쉐이딩을 또한 궁극적으로 수행할 수도 있는 버텍스 쉐이딩 동작들을 수행하도록 쉐이딩 유닛 (40) 을 지정할 수도 있다. 예를 들어, 쉐이딩 유닛 (40) 은 초기에 포인트들 (po-p2) 로서 나타낸 3 개의 버텍스들을 갖는 입력 프리미티브 (삼각형 스트립) 를 발생시키기 위하여 버텍스 쉐이딩 동작들을 수행할 수도 있다.

[0199] 버텍스 쉐이딩 동작들을 실행한 후, GPU (36) 는 쉐이딩된 버텍스들을 로컬 메모리 자원들에 저장할 수도 있다. 예를 들어, GPU (36) 는 버텍스 쉐이더 출력을 (GPU 메모리 (38) 의) 위치 캐쉬에 엑스포트할 수도 있다. 버텍스 쉐이딩 동작들 및 헐 쉐이딩 동작들은 VS END 명령에 의해 분리될 수도 있다. 따라서, VS END 명령을 실행하고 버텍스 쉐이딩 동작들을 완료한 후, 버텍스 쉐이딩 동작들을 수행하도록 지정된 하나 이상의 쉐이딩 유닛들 (40) 은 각각 헐 쉐이딩 동작들을 수행하는 것을 시작한다.

[0200] 다음으로, 동일한 쉐이딩 유닛 (40) 은 제어 포인트들 (V0-V3) 을 갖는 출력 패치를 발생시키기 위하여 헐 쉐이딩 동작들을 수행할 수도 있다. 이 예에서, 쉐이딩 유닛 (40) 은 도 4 에 대하여 위에서 설명된 지오메트리 쉐이더 동작들과 유사한 방식으로, 그 출력 식별자들 (Outvert) 에 의해 나타내어지는 헐 쉐이더 동작의 다수의 인스턴스들을 실행한다. 헐 쉐이더 동작의 각각의 인스턴스는 동일한 헐 쉐이딩 동작을 수행하도록 동일한 알고리즘을 수행하고, 하나 이상의 새로운 제어 포인트들 (V0-V3) 의 각각의 인스턴스들을 발생시킨다.

[0201] 즉, 도 11 에 도시된 표의 4 개의 컬럼들은 헐 쉐이더 동작 (또는 프로그램) 의 4 개의 별도의 인스턴스들에 대응하고, 여기서, 좌측으로부터 우측으로의 각각의 컬럼은 0-3 의 헐 쉐이더 동작 Outvert 에 의해 식별될 수도 있다. 하나 이상의 새로운 제어 포인트들의 별도의 인스턴스를 발생시키기 위하여, 헐 쉐이더 동작의 이 4 개의 인스턴스들의 각각은 쉐이딩 유닛 (40) 에 의해 종종 동시에 실행된다. 따라서, 헐 쉐이더 동작들의 인스턴스들의 각각은 제어 포인트들 (V0-V3) 의 4 개 모두를 발생시키지만, 4 개의 새로운 제어 포인트들 중의 대응하는 하나를 출력하기만 한다. 헐 쉐이더 동작의 각각의 인스턴스는 버텍스 쉐이딩 동작들에 대해 호출되었던 쉐이딩 유닛 (40) 의 1:1 인터페이스를 고수하기 위하여 4 개의 새로운 제어 포인트들 중의 대응하는 하나를 출력하기만 한다.

[0202] 도 11 의 예에서, 헐 쉐이더 동작들의 각각은 그 Outvert 와 일치하는 4 개의 새로운 제어 포인트들 중의 하나를 출력한다. 따라서, Outvert = 0 을 갖는 헐 쉐이더 동작의 제 1 인스턴스는 4 개의 새로운 제어 포인트

들 중의 제 1 제어 포인트 (V0) 를 출력한다. Outvert = 1 을 갖는 헐 쉐이더 동작의 제 2 인스턴스는 4 개의 새로운 제어 포인트들 중의 제 2 제어 포인트 (V1) 를 출력한다. Outvert = 2 를 갖는 헐 쉐이더 동작의 제 3 인스턴스는 4 개의 새로운 제어 포인트들 중의 제 3 제어 포인트 (V2) 를 출력한다. Outvert = 3 을 갖는 헐 쉐이더 동작의 제 4 인스턴스는 4 개의 새로운 제어 포인트들 중의 제 4 제어 포인트 (V3) 를 출력한다. 헐 쉐이더 값들이 로컬 메모리에 기록된 후, 도메인 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들은 위에서 설명된 바와 같이, 제 2 패스 (패스 II) 동안에 수행될 수도 있다.

[0203] 이 개시물의 양태들에 따르면, 버텍스 쉐이딩 동작들을 수행하도록 지정된 동일한 쉐이딩 유닛 (40) 은 또한 위에서 설명된 헐 쉐이딩 동작들을 수행한다. 또한, 동일한 쉐이딩 유닛 (40) 은 드로우 콜의 제 2 패스 (패스 II) 동안에 도메인 쉐이딩 및 지오메트리 쉐이딩 동작들을 또한 수행할 수도 있다. 예를 들어, GPU (36) 는 상태를 쉐이더 특정 자원들 (예를 들어, 헐, 도메인, 및/또는 지오메트리 쉐이더 상수들, 텍스처 오프셋들, 등) 로 변화시킬 수도 있다. GPU (36) 는 쉐이딩 동작들에 배정된 모드 (드로우 모드) 에 따라 이 상태 변화를 수행할 수도 있다.

[0204] 이하에 도시된 표 4 는 동일한 쉐이딩 유닛 (40) 으로 버텍스 쉐이딩, 헐 쉐이딩, 도메인 쉐이딩, 및 지오메트리 쉐이딩을 수행하기 위하여 GPU (36) 에 의해 유지될 수도 있는 동작 모드들 및 파라미터들을 예시한다.

표 4

[0205] **쉐이딩 동작들을 수행하기 위한 모드들**

모드	**모드 0** **GS:오프,** **HS: 오프**	**모드 1** **GS: 온** **HS: 오프**	**모드 4** **GS: 온,** **HS: 온** **(패스II)**	**모드 3** **GS: 오프,** **HS: 온** **(패스II)**	**모드 2** **GS: 오프,** **HS: 온** **(패스 I)**
흐름	VS->PS	VS| GS->PS	DS| GS ->PS	DS->PS	VS|HS
인덱스 (32 비트들)	*버텍스 인덱스 (VS)*	*버텍스 인덱스 (VS)*	u(15:0) v (31:16)	u(15:0) v (31:16)	버텍스 인덱스
uv_msb (2-비트들)	이용되지 않음	이용되지 않음	u, v 의 상위 비트	u, v 의 상위 비트	*이용되지 않음*
PrimitiveID (32-비트들)	이용되지 않음	*PrimitiveID (GS)*	*PrimtiiveID (DS, GS)*	*PrimtiiveID (DS)*	*PrimtiiveID (HS)*
Rel_patchid (32-비트들)	이용되지 않음	이용되지 않음	Rel_patchid (DS)	Rel_patchid (DS)	Rel_patchid (HS)
Misc (25 비트들)	이용되지 않음	misc-> rel_primID (4:0)	misc-> rel_primID (4:0)	이용되지 않음	misc-> rel_primID (4:0)
		misc-> rel_vertex (9:5)	misc-> rel_vertex (9:5)		misc-> rel_vertex (9:5)
		misc-> GsInstance (14:10)	misc-> GsInstance (14:10)		misc-> outvertID (14:10)
		misc-> Gsoutvertex (24:15)	misc-> Gsoutvertex (24:15)		
Vs_valid (1 비트)					
Gshs_valid (1 비트)					
모드 (2:0)	모드 = mode_0	모드 = mode_1	모드 = mode_4	모드 = mode_3	모드 = mode_2
Instance_cmd (2-비트)					

[0206] 일부의 인스턴스들에서, 상기 표 4 에서 표시된 바와 같이, 어떤 쉐이딩 동작들은 특별한 드로우 콜에 대해 수행되지 않을 수도 있다. 예를 들어, 드로우 콜은 버텍스 쉐이딩, 헐 쉐이딩, 도메인 쉐이딩, 및 픽셀 쉐이딩 동작들을 포함할 수도 있지만, (모드 3 에 대해 도시된 바와 같은) 지오메트리 쉐이딩 동작들을 포함하지 않을 수도 있다. GPU (36) 는 드로우 콜을 실행할 때에 어느 쉐이딩 동작들을 수행할 것인지를 결정하기 위하여 모드 정보를 이용할 수도 있다.

[0207] 이하에 도시된 표 5 는 지오메트리 쉐이딩 동작들을 수행하지 않으면서 패스 II 동작들을 수행할 때의 파라미터 값들을 예시한다.

표 5

[0208] **지오메트리 쉐이딩을 갖지 않는 파라미터 값들**

모드 3 **GS: 오프,** **HS: 온**	파이버 0	파이버 1	파이버 2	파이버 3	파이버 4	파이버 5	파이버 6	파이버 7
Valid_as_input	1	1	1	1	1	1	1	1
버텍스 인덱스 (VS)	U|V	U|V	U|V	U|V	U|V	U|V	U|V	U|V
Uv_msb	u|v	u|v	u|v	u|v	u|v	u|v	u|v	u|v
primitiveID (HS)	105	105	105	105	105	105	105	105
Rel_patchID	5	5	5	5	5	5	5	5

[0209] 이하에 도시된 표 6 은 지오메트리 쉐이딩 동작들을 수행하는 것을 포함하는 패스 II 동작들을 수행할 때의 파라미터 값들을 예시한다.

표 6

[0210] **지오메트리 쉐이딩을 갖는 파라미터 값들**

모드 4 **GS: 온, HS: 온**	파이버 0	파이버 1	파이버 2	파이버 3	파이버 4	파이버 5	파이버 6	파이버 7
Valid_as_input	1	1	1	0	0	0	0	0
버텍스 인덱스 (VS)	U|V	U|V	U|V	U|V	0	0	0	0
Uv_msb	u|v	u|v	u|v	u|v	0	0	0	0
primitiveID(HS & GS)	105	105	105	105	105	105	105	105
Rel_patchID	5	5	5	5	5	5	5	5
Valid_as_output	1	1	1	1	1	1	1	1
misc-> rel_primID (4:0)	0	0	0	0	0	0	0	0
misc-> rel_vertex (9:5)	0	1	2	0	0	0	0	0
misc-> GSInstancel (4:10)	0	0	2	0	0	0	0	0
misc-> GsOutvertex (24:15)	0	1	2	3	4	5	6	7

[0211] 도 11 에 도시된 바와 같이 제 1 패스 (패스 I) 와 연관된 동작들을 완료한 후, GPU (36) 는 유휴상태를 대기할

수도 있다. 다음으로, GPU (36) 는 드로우 콜을 완료하기 위하여 드로우 콜의 제 1 패스 (패스 II) 를 수행할 수도 있다.

[0212] 도 12a 및 도 12b 는 이 개시물의 기술들을 구현하는 하드웨어 쉐이딩 유닛에 의해 수행될 수도 있는 일 예의 동작들을 예시한다. 도 12a 및 도 12b 는 일반적으로 패스 I 에 대하여 위에서 설명된 쉐이딩 동작들에 대응한다.

[0213] 예를 들어, 도 12a 는 버텍스 쉐이딩 동작들 및 헐 쉐이딩 동작들을 수행할 때, 병합된 VS/HS 하드웨어 쉐이딩 유닛에 의해 수행되는 동작들의 흐름을 일반적으로 예시한다. 일부의 예들에서, 병합된 VS/HS 하드웨어 쉐이딩 유닛은, 버텍스 쉐이딩 동작들을 수행하도록 GPU (36) 에 의해 지정되지만, 이 개시물의 기술들에 따라 버텍스 쉐이딩 동작들 및 헐 쉐이딩 동작들의 둘 모두를 수행하는 쉐이딩 유닛 (40) 을 포함할 수도 있다. 도 12b 는 병합된 VS/HS 하드웨어 쉐이딩 유닛에 의해 실행될 수도 있는 도 12a 에 도시된 동작들의 흐름에 대응하는 의사 코드를 일반적으로 예시한다.

[0214] 도 12a 에 도시된 바와 같이, 하드웨어 쉐이딩 유닛은 VS 동작들, 그 다음으로 HS 동작들을 수행할 수도 있다. 예를 들어, (GPU (36) 와 같은) GPU 는 (위에서 설명된 바와 같이) 버텍스 속성들, vertex_id, instance_id, primitive_id, 및 misc 를 포함하는 시스템 발생된 값들을 레지스터들에 기록할 수도 있다. 위에서 언급된 바와 같이, 시스템 발생된 값들을 미리 결정된 로케이션에서의 일련의 레지스터들에 저장함으로써, GPU (36) 는 VS 및 HS 스테이지들의 각각에 대한 시스템 발생된 값들을 액세스할 수도 있다. 따라서, HS 스테이지는 시스템 발생된 값들이 어디에 저장되었는지를 결정하기 위하여 VS 스테이지에 기초하여 컴파일링될 필요가 없다. 오히려, GPU (36) 는 요구되는 시스템 발생된 값들을 액세스하기 위하여 스테이지들의 각각을 수행할 때에 미리 결정된 메모리 로케이션들을 액세스할 수도 있다.

[0215] 다음으로, 하드웨어 쉐이딩 유닛은 하나 이상의 쉐이딩된 버텍스들을 발생시키기 위하여 버텍스 쉐이딩 동작들을 수행할 수도 있다. 하드웨어 쉐이딩 유닛은 쉐이딩된 버텍스들을 로컬 메모리에 기록할 수도 있어서, 쉐이딩된 버텍스들이 헐 쉐이딩 동작들을 위해 이용가능하다.

[0216] 다음으로, GPU 는 헐 쉐이딩 동작들을 수행하기 전에 메모리 오프셋들 및 프로그램 카운터를 스위칭할 수도 있다. GPU 는 예를 들어, 위에서 설명된 패치 코드를 실행할 때, 이러한 태스크들을 수행할 수도 있다. 다음으로, 하드웨어 쉐이딩 유닛은 로컬 메모리로부터 쉐이딩된 버텍스들을 판독할 수도 있고, 하나 이상의 제어 포인트들 및 테셀레이션 인자들을 발생시키기 위하여 헐 쉐이딩 동작들을 수행할 수도 있다.

[0217] 제 1 패스 동안에 발생된 제어 포인트들 및 테셀레이션 인자들은 예를 들어, 로컬 GPU 메모리에 저장될 수도 있다. 일부의 예들에서, 제어 포인트들 및 테셀레이션 인자들은 로컬 GPU 메모리 내의 별도의 버퍼들에 저장될 수도 있다.

[0218] 도 12b 는 위에서 설명된 패스 I 동작들을 수행하는 하드웨어 쉐이딩 유닛에 의해 실행될 수도 있는 코드의 일 예의 부분이다. 도 12b 에 도시된 예에서, 대문자로 된 단어들은 상태 또는 상수 레지스터들이다. 이탤릭체로 된 단어들은 쉐이더 입력을 표시한다. VS/HS 동작들에 대해 할당된 GPR 들의 수는 (gprs_needed_for_vs, gprs_needed_for_hs) 의 최대치이다. 따라서, VS 동작들에서 이용된 후, GPR 들은 해제되고 HS 동작들을 위해 이용된다.

[0219] 일부의 인스턴스들에서는, 쉐이딩 동작들의 VS 부분에서, (도 5b 에 대하여 위에서 언급된 바와 같이) 유효한 VS 파이버들만이 실행된다. 'SWITCH_ACTIVE' 명령을 조우할 시에, 커버리지 마스크 비트들은 HS 쉐이더와 연관되도록 변화되고, 활성 HS 파이버들만이 실행된다. 이러한 방법으로, 예약된 레지스터들은 VS 및 HS 의 둘 모두에 의해 이용될 수도 있고, VS 및 HS 는 HS 동작들을 수행하도록 쉐이딩 유닛을 재지정하지 않고도, 단일 하드웨어 쉐이딩 유닛에 의해 구현될 수도 있다.

[0220] 도 13a 및 도 13b 는 또한, 이 개시물의 기술들을 구현하는 하드웨어 쉐이딩 유닛에 의해 수행될 수도 있는 일 예의 동작들을 예시한다. 도 13a 및 도 13b 는 일반적으로 위에서 설명된 패스 II 쉐이딩 동작들에 대응할 수도 있다.

[0221] 예를 들어, 도 13a 는 일반적으로, 도메인 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행할 때, 병합된 DS/GS 하드웨어 쉐이딩 유닛에 의해 수행되는 동작들의 흐름을 예시한다. 일부의 예들에서, 병합된 DS/GS 하드웨어 쉐이딩 유닛은, 도 12a 및 도 12b 에 대하여 위에서 설명된 것과 동일한 쉐이딩 유닛 (40) 으로서, 버텍스 쉐이딩 동작들을 수행하도록 GPU (36) 에 의해 최초에 지정되는 상기 동일한 쉐이딩 유닛 (40) 을 포함할 수도 있다. 도 13b 는 병합된 DS/GS 하드웨어 쉐이딩 유닛에 의해 실행될 수도 있는, 도 13a 에 도시된 동

작들의 흐름에 대응하는 의사 코드를 일반적으로 예시한다.

[0222] 이 개시물의 양태들에 따르면, (도 12a 및 도 12b 에 대하여 설명된) 제 1 패스 이후에 "유휴상태 대기" 가 후속될 수도 있다. 즉, 제 1 패스 동안에 데이터가 완전히 메모리에 기록되기 전에, 제 2 패스 동안에 데이터가 로컬 메모리로부터 판독되는 것을 방지하기 위하여, GPU 는 도 13a 및 도 13b 에 도시된 제 2 패스 동작들을 개시하기 전에, GPU 의 하나 이상의 컴포넌트들이 유휴상태 (예를 들어, 컴퓨팅하거나 데이터를 전송하지 않음) 인 것으로 등록하는 것을 대기할 수도 있다.

[0223] 어떤 경우에도, 도 13a 에 도시된 바와 같이, 하드웨어 쉐이딩 유닛은 도메인 쉐이딩 및 지오메트리 쉐이딩을 포함하는 패스 II 동작들을 수행할 수도 있다 (테셀레이션은 또한 고정된-기능 테셀레이션 유닛에 의해 수행될 수도 있음). 예를 들어, GPU 는 (위에서 설명된 바와 같이) {U, V} 좌표들, primitive_id, 및 misc 를 포함하는 시스템 발생된 값들을 레지스터들에 기록할 수도 있다. 위에서 언급된 바와 같이, 시스템 발생된 값들을 미리 결정된 로케이션에서의 일련의 레지스터들에 저장함으로써, GPU (36) 는 DS 및 GS 스테이지들의 각각에 대한 시스템 발생된 값들을 액세스할 수도 있다. 따라서, GS 스테이지는 시스템 발생된 값들이 어디에 저장되었는지를 결정하기 위하여 DS 스테이지에 기초하여 컴파일링될 필요가 없다. 오히려, GPU (36) 는 요구되는 시스템 발생된 값들을 액세스하기 위하여 스테이지들의 각각을 수행할 때에 미리 결정된 메모리 로케이션들을 액세스할 수도 있다.

[0224] 다음으로, 하드웨어 쉐이딩 유닛은 하나 이상의 테셀레이팅된 버텍스들을 발생시키기 위하여 도메인 쉐이딩 동작들을 수행할 수도 있다. 하드웨어 쉐이딩 유닛은 테셀레이팅된 버텍스들을 로컬 메모리에 기록할 수도 있어서, 테셀레이팅된 버텍스들은 지오메트리 쉐이딩 동작들을 위해 이용가능하다.

[0225] 다음으로, GPU 는 지오메트리 쉐이딩 동작들을 수행하기 전에, 메모리 오프셋들 및 프로그램 카운터를 스위칭할 수도 있다. GPU 는 예를 들어, 위에서 설명된 패치 코드를 실행할 때, 이러한 태스크들을 수행할 수도 있다. 다음으로, 하드웨어 쉐이딩 유닛은 로컬 메모리로부터 테셀레이팅된 버텍스들을 판독할 수도 있고, 버텍스 파라미터 캐쉬에 저장될 수도 있는 하나 이상의 지오메트리 쉐이딩된 버텍스들을 발생시키기 위하여 지오메트리 쉐이딩 동작들을 수행할 수도 있다.

[0226] 도 13b 에 도시된 예에서, 대문자로 된 단어들은 상태 또는 상수 레지스터들이다. 이탤칙체 단어는 쉐이더 입력을 표시한다. 이 쉐이더에 대해 할당된 GPU 들의 수는 (gprs_needed_for_vs, gprs_needed_for_gs) 의 최대치이다. 따라서, DS 동작들에서 이용된 GPR 들은 해제되고 GS 동작들을 위해 이용된다. 'SWITCH_ACTIVE' 명령을 조우할 시에, 커버리지 마스크 비트들은 GS 동작들과 연관되도록 변화되고, 활성 GS 파이버들만이 실행된다. 'END_1st' 명령을 조우할 시에, 하드웨어 쉐이더 유닛은 상수 파일 및 텍스처 포인터들 (예를 들어, 자원 포인터) 에 대한 자원 오프셋들을 GS 프로그래밍된 오프셋들로 스위칭할 수도 있고, GS 의 제 1 명령으로 점프할 수도 있다. 이러한 방법으로, 예약된 레지스터들은 DS 및 GS 둘 모두의 쉐이더 스테이지들에 의해 이용될 수도 있고, DS 및 GS 쉐이더 스테이지들은 패스 I 동작들을 수행한 동일한 하드웨어 쉐이딩 유닛에 의해 실행될 수도 있다.

[0227] 도 12a 내지 도 13b 의 예들에 도시된 바와 같이, 단일 하드웨어 쉐이딩 유닛은 4 개의 상이한 쉐이더 스테이지들의 동작들을 수행할 수도 있다. 일부의 예들에 따르면, 쉐이더 스테이지들을 병합하기 위한 패치 코드는 어느 쉐이더 스테이지들이 병합되고 있는지에 관계없이 동일할 수도 있다. 예를 들어, DS 동작들은 (도 12b 의 상부로부터의 제 2 점선 박스에 도시된) VS 및 HS 동작들을 병합하기 위해 이용되는 것과 동일한 (도 13b 의 상부로부터 제 2 점선 박스에 도시된) 패치 코드를 이용하여 GS 동작들과 병합될 수도 있다. 하드웨어 쉐이딩 유닛은 드로우 시간에 GPU 에 의해 결정될 수도 있는 (상기 표들에 대하여 도시되고 설명된 바와 같은) 동작 모드에 기초하여 적절한 쉐이딩 동작들로 스위칭할 수도 있다.

[0228] 이 개시물의 양태들에 따르면, 각각의 쉐이더 스테이지 (VS/GS/HS/DS) 는 별도로, 그리고 스테이지들이 실행 동안에 어떻게 링크될 것인지에 대해 알지 못하면서 컴파일링될 수도 있다. 따라서, primitveID, rel_patch_ID 및 misc 와 같은 파라미터들을 저장하기 위하여 3 개의 GPR 들이 예약될 수도 있다. 컴파일러는 입력 속성들 또는 내부 변수들로 하여금 DX10/DX11 애플리케이션들에 대해 2 를 초과하는 GPR 들의 ID 들에 저장되게 할 수도 있다.

[0229] 도 14 는 이 개시물의 양태들에 따르면, 병합된 버텍스 쉐이딩, 헐 쉐이딩, 도메인 쉐이딩, 및 지오메트리 쉐이딩 동작들을 수행하기 위한 그래픽스 프로세싱 유닛 (330) 의 일 예의 컴포넌트들을 예시하는 도면이다. 도 14 의 예는 병합된 VS/HS 유닛 (패스 I) 및 병합된 DS/GS 유닛 (패스 II) (332), 버텍스 파라미터 캐쉬 (VPC;

334), 테셀레이터 (337) 를 갖는 프리미티브 제어기 (PC; 336), 버텍스 페치 디코더 (VFD; 338), 그래픽스 래스터라이저 (GRAS; 340), 렌더 백엔드 (RB; 342), 커맨드 프로세서 (CP; 344), 및 픽셀 쉐이더 (PS; 346) 를 포함한다. 추가적으로, 도 14 는 PM4 패킷 버퍼들 (350), 버텍스 오브젝트들 (352), 인덱스 버퍼들 (354), 시스템 스크래치 (356) 및 프레임 버퍼 (358) 를 갖는 메모리 (348) 를 포함한다.

[0230] 도 14 의 예에서, VS/GS 유닛 (332) 은 위에서 설명된 방식으로 하나 이상의 쉐이딩 유닛들에 의해 구현된다. VPC (334) 는 스트림 출력 데이터를 메모리 (348) 에 저장하기 위한 스트림 출력 기능성을 구현할 수도 있다. PC (336) 는 변환될 필요가 있을 수도 있는 버텍스들을 관리할 수도 있고, 버텍스들을 삼각형 프리미티브들로 집합시킨다. VFD (338) 는 버텍스 포맷 상태에 기초하여 버텍스 데이터를 페치할 수도 있다. GRAS (340) 는 삼각형 버텍스들을 입력으로서 수신할 수도 있고, 삼각형 경계들 내에 있는 픽셀들을 출력할 수도 있다. 프리-페치 파서 (PFP) 는 커맨드 스트림을 프리-디코딩할 수도 있고 포인터들 (예를 들어, 자원 포인터들) 을 통해 데이터를 페치할 수도 있어서, 이 데이터는 주요 CP 엔진 (344) 이 이 데이터를 필요로 하는 시간까지 준비된다.

[0231] DirectX 11 에 대한 디스패치 메커니즘에 대하여, 드로우 콜은 CP (344) 에 의해 2 패스 드로우로 분할된다. 패스 I 의 출력을 저장하기 위한 이용가능한 저장공간에 기초하여, 드로우 콜은 다수의 서브-드로우 콜들로 분할될 수도 있고, 각각의 서브-드로우 콜은 패스 I 및 패스 II 를 가진다. 각각의 서브-드로우 콜은, 패스 I 이 서브-드로우 콜에 대해 수행되고, 그 다음으로, 패스 II 가 서브-드로우 콜에 대해 수행되도록 패스들의 순서화를 고수할 수도 있다.

[0232] 패스 I 을 갖는 서브-드로우 콜을 수신할 시에, PC (336) 는 인덱스들을 페치할 수도 있고, VS/HS (332) 를 이용하여 패치 프리미티브 타입을 프로세싱할 수도 있다. VS/HS (332) 는 패치 당 $\mathrm{HS\_FIBERS\_PER\_PATCH} = 2^{\mathrm{ceil}(\log_2(\max(\mathrm{input}_{\mathrm{patch}}, \mathrm{output}_{\mathrm{patch}})))}$ VS 파이버들을 생성하고, 웨이브 당 패치들의 정수 개수에 맞춘다 (여기서, 웨이브는 작업의 주어진 양이다). 입력에서 버텍스 재이용은 전혀 없다. VS/HS (332) 의 출력은 시스템 스크래치 (356) 에 오프-칩 (off-chip) 으로 전송되므로, 위치 및 파라미터 캐쉬의 할당이 전혀 없을 수도 있다.

[0233] HS_FIBERS_PER_PATCH 에 기초하여, (도 1 에 도시된 GPU 드라이버 (50) 와 같은) GPU 드라이버는 얼마나 많은 입력 프리미티브 버텍스들이 (VS/HS (332) 에 로컬인) 로컬 메모리에 저장될 것인지를 컴퓨팅할 수도 있다. 이것은 다음과 같이 컴퓨팅될 수도 있다:

$$\mathrm{HS\_LM\_SIZE}\left\lceil \frac{fibers\_in\_a\_wave}{HS\_FIBERS\_PER\_PATCH} \right\rceil * control\_points\_in\_input\_patch * size\_of\_vertex$$

[0234]

[0235] 드라이버가 최종 데이터를 메모리 (348) 에 기록하기 전에 중간 데이터를 로컬 메모리에 기록해야 할 경우, 드라이버는 추가적인 사이즈를 HS_LM_SIZE 에 또한 추가할 수도 있다. HS 가 HS 의 다수의 페이즈들에서 (예를 들어, HS 의 상수 페이즈에서) 컴퓨팅된 제어 포인트를 이용하고 있을 경우, 이러한 추가적인 공간은 유용할 수도 있다. 이 타입의 드로우 콜을 수신하는 하이 레벨 시퀀서 (HLSQ) 는 어느 쉐이딩 유닛의 로컬 메모리 (LM) 가 GS_LM_SIZE 에 대해 충분한 저장공간을 가지는지를 검사할 수도 있다. HLSQ 는 이러한 할당의 시작 기본 어드레스뿐만 아니라, 할당된 웨이브에 의한 로컬 메모리에 대한 임의의 판독 또는 기록의 어드레스를 유지할 수도 있다. HLSQ 는 또한, 로컬 메모리에 기록할 때에 할당된 메모리 내의 컴퓨팅된 오프셋을 기본 어드레스에 추가할 수도 있다.

[0236] 시스템 해독된 값 (SIV) 들 (예를 들어, 클립/컬 거리들, 렌더타겟 (rendertarget), 뷰포트) 은 PS (346) 로의 로딩을 위하여 VPC (334) 에 또한 제공될 수도 있다. 쉐이더 스테이지 (예를 들어, VS 또는 GS) 는 조건부로 값들을 출력할 수도 있다. 따라서, PS (346) 가 값들을 필요로 할 경우, PS (346) 는 이러한 조건을 상태의 일부로서 세팅할 수도 있다. PS(346) 가 값들을 필요로 하지 않고, 이러한 결정이 픽셀 쉐이딩 동작들의 컴파일링 후에 행해질 경우, 이 SIV 들을 출력하는 상태는 리셋될 수 있어서, VS 또는 GS 는 드로우 시간에 값들을 VPC (334) 에 기록하지 않을 것이다.

[0237] 널 (null) GS (지오메트리 쉐이더 스테이지가 전혀 실행되고 있지 않을 경우) 에 대하여, 컴파일러는 또한 템플릿 GS 를 생성할 수도 있어서, 널 또는 비-널 (non-null) GS 에 대한 별도의 경로가 전혀 없다. 이 템플릿 GS 는 VS 또는 도메인 쉐이더 (DS) 출력을 로컬 메모리에 복사할 수도 있고, VPC (334) 로 출력하기 위하여 로

컬 메모리로부터 더욱 복사할 수도 있다. 이것은 스트림 출력이 수행되는 경우에 대해서 행해지기만 할 수도 있다.

[0238] 가시성 스트림 (visibility stream) 들을 비닝하고 소비하는 프로세스는 어느 쉐이더들이 구현되고 있는지에 따라 상이할 수도 있다. 예를 들어, 어떤 GPU 들은 타일 (tile) 들 또는 "빈 (bin) 들" 로 렌더링될 이미지 데이터를 분할할 수도 있고, 이것은 전체 이미지가 렌더링될 때까지 각각의 빈을 연속으로 (또는 때때로 동시에 또는 병렬로) 렌더링할 수도 있다. 이미지를 빈들로 분할함으로써, GPU 들은 (온-칩 메모리가 타일을 렌더링하기 위한 충분한 이미지 데이터를 저장할 정도로 충분히 클 수도 있음을 고려하여) 오프-칩 메모리로부터의 더 적은 데이터 취출을 또한 촉진하면서, 온-칩 메모리 요건들을 감소시킬 수도 있다.

[0239] 가시성 스트림에 대하여, Z-버퍼 알고리즘은 다른 프리미티브들에 의해 가려지는 (그러므로, 렌더링될 필요가 없는) 프리미티브들을 결정하기 위하여 이용될 수도 있다. 예를 들어, GPU 는 최후방 (깊이-방향) 프리미티브로부터 최전방 (다시, 깊이-방향) 프리미티브로 작업하여, 각각의 프리미티브를 드로우할 수도 있다. 이 예에서, 일부의 프리미티브들은 다른 프리미티브들에 의해 드로우 오버 (draw over) 되기만 하도록 렌더링될 수도 있다.

[0240] 이 소위 "오버드로우 (overdraw)" 의 결과로서, GPU 들은 조기 Z-버퍼 알고리즘 테스팅을 수행하도록 구비될 수도 있고, 이 조기 Z-버퍼 알고리즘 테스팅은, GPU 들이 렌더링을 수행할 때에 무시되거나 우회되도록 완전히 가려지거나 아이 뷰 (eye view) 내에 있지 않은 프리미티브들을 GPU 들이 식별하도록 한다. 이 점에 있어서, GPU 들은 각각의 프리미티브 및/또는 오브젝트에 대하여 무엇이 가시성 정보라고 지칭될 수도 있는지를 결정하도록 구비될 수도 있다.

[0241] DX10 에 대하여, 비닝 패스 동안에, PC (336) 는 GS 로부터의 모든 출력 프리미티브들의 종료 시에 "프리미티브의 종료 (end of primitive)" 를 GRAS (340) 로 전송한다. 그러므로, 가시성 정보는 입력 프리미티브 당 레코딩된다. 스트림 출력은 비닝 패스 동안에 수행될 수도 있다. CP (344) 는 비닝 패스의 종료 시에 모든 스트림 출력 버퍼 관련된 정보를 판독할 수 있다. 지오메트리 관련된 쿼리 카운터들은 비닝 패스 동안에 업데이트될 수도 있다.

[0242] 가시성 패스는 가시성 스트림을 판독할 수도 있고, 프리미티브 당 가시성 정보가 판독됨에 따라 스트림을 전진시킬 수도 있다. 스트림이 전혀 래스터라이징되지 않을 경우, 가시성 패스는 건너띄어질 수도 있다. 이와 다를 경우, PC (336) 는 가시성 입력 GS 프리미티브와, 임의의 스트림 출력들 없이 렌더링하기 위한 프로세스에 대해 검사한다.

[0243] DX 11 에 대하여, 비닝 패스 동안에, PC (336) 는 패스 II 에서의 GS 로부터의 모든 출력 프리미티브들의 종료 시에 "프리미티브의 종료" 를 GRAS (340) 로 전송한다 (예를 들어, 입력 패치 당 1 비트). 스트림 출력은 위에서 설명된 바와 같이 수행될 수도 있다. 가시성 패스 동안, 가시성 스트림은 패치들과 함께 패스 I 에서 프로세싱된다 (가시성을 갖는 패치들만이 프로세싱될 수도 있음). 패스 II 는 가시적 패치들을 프로세싱하기만 하고, 가시적 패치들만을 위한 테셀레이션 인자들을 페치한다.

[0244] 이하에 도시된 표 7 은 5 개의 상이한 동작의 모드들의 각각에 대한 비닝 패스 및 렌더링 패스에 관한 정보를 제공한다. 각각의 모드는 위에서 설명된 바와 같이, 단일 하드웨어 쉐이딩 유닛에 의해 수행되고 있는 어떤 동작들에 대응한다.

표 7

[0245] **상이한 모드들에 대한 비닝**

모드들	VS 스테이지	PS 스테이지	비닝 패스	렌더링 패스
Mode_0	VS	PS	프리미티브 당 비즈 (Viz) 정보	비즈 스트림 소비
Mode_1	VS+GS	PS	입력 프리미티브 당 비즈 정보: 증폭된 프리미티브에 대하여, 입력 프리미티브에 대한 비즈 정보를 발생시키기 위하여 빈 커버리지가 논리합 (or) 됨	비즈 스트림 소비
Mode_2	VS+HS		비즈 발생 없음	비즈 스트림 소비

Mode_3	DS	PS	비즈 정보가 입력 패치 당 발생되고, 입력 프리미티브에 대한 비즈 정보를 발생시키기 위하여 모든 테셀레이팅된 프리미티브들 빈-커버리지가 논리합 됨	비즈 스트림 소비하지 않음
Mode_4	(DS+GS)	PS	비즈 정보가 입력 패치 당 발생되고, 입력 프리미티브에 대한 비즈 정보를 발생시키기 위하여 모든 테셀레이팅된 GS 프리미티브들 빈-커버리지가 논리합 됨	비즈 스트림 소비하지 않음

[0246] 도 15 는 이 개시물의 양태들에 따라, 동일한 하드웨어 쉐이딩 유닛을 이용하여 2 개의 렌더링 패스들에서 그래픽스 렌더링을 수행하는 것을 예시하는 흐름도이다. GPU (36, 도 1) 에 대하여 설명되었지만, 이 개시물의 양태들은 다양한 다른 컴포넌트들을 갖는 다양한 다른 GPU 들에 의해 수행될 수도 있다.

[0247] 도 15 의 예에서, GPU (36) 는 그래픽스를 렌더링하기 위하여 현재 실행되고 있는 드로우 콜이 테셀레이션 동작들을 포함하는지 여부를 결정한다 (380). 테셀레이션 동작들은 예를 들어, 위에서 설명된 바와 같이, 헐 쉐이더 스테이지, 테셀레이션 스테이지, 및 도메인 쉐이더 스테이지와 연관된 동작들을 포함할 수도 있다. 드로우 콜이 테셀레이션 동작들을 포함하지 않을 경우, GPU (36) 는 단일 패스로 렌더링을 수행할 수도 있다 (382). 예를 들어, GPU (36) 는 위에서 설명된 방식으로 버텍스 쉐이딩, 지오메트리 쉐이딩, 및 픽셀 쉐이딩을 수행할 수도 있다.

[0248] 드로우 콜이 테셀레이션 동작들을 포함할 경우, GPU (36) 는 GPU 메모리 (38) 와 같은 로컬 GPU 메모리 자원들의 사이즈를 결정할 수도 있다 (384). 다음으로, GPU (36) 는 드로우 콜을 복수의 서브-드로우 콜들로 분할할 수도 있다 (386). 일부의 예들에서, 각각의 서브-드로우 콜은 위에서 설명된 패스 I 동작들 및 패스 II 동작들을 포함할 수도 있다. 예를 들어, 패스 I 동작들은 버텍스 쉐이딩 동작들 및 헐 쉐이딩 동작들을 포함할 수도 있지만, 패스 II 동작들은 도메인 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 포함할 수도 있다.

[0249] 각각의 서브-드로우 콜에 의해 렌더링되는 데이터의 양은 GPU 메모리 (38) 의 사이즈에 기초하여 결정될 수도 있다. 예를 들어, GPU (36) 가 패스 I 동작들에 의해 발생된 데이터의 전부를 패스 II 동작들과 함께 이용하기 위하여 로컬 메모리에 저장할 수 있도록, GPU (36) 는 서브-드로우 콜들을 구성할 수도 있다. 이러한 방법으로, GPU (36) 는 로컬 GPU 메모리와, GPU 의 외부에 있는 메모리와의 사이에서 전송되고 있는 데이터의 양을 감소시킬 수도 있고, 이것은 위에서 설명된 바와 같이, 렌더링과 연관된 지연시간을 감소시킬 수도 있다.

[0250] 서브-드로우 콜들을 결정한 후, GPU (36) 는 제 1 서브-드로우 콜에 대하여 패스 I 동작들을 수행할 수도 있다 (388). 위에서 언급된 바와 같이, 패스 I 동작들은 동일한 하드웨어 쉐이딩 유닛, 예를 들어, 하나 이상의 쉐이딩 유닛들 (40) 의 각각을 이용하여 버텍스 쉐이딩 동작들 및 헐 쉐이딩 동작들을 수행하는 것을 포함할 수도 있다. 즉, GPU (36) 는 버텍스 쉐이딩을 수행하도록 다수의 쉐이딩 유닛들 (40) 을 지정할 수도 있지만, 쉐이딩 유닛들 (40) 의 각각은 버텍스 쉐이딩 및 헐 쉐이딩 동작들의 둘 모두를 수행할 수도 있다.

[0251] GPU (36) 는 또한 제 1 서브-드로우 콜에 대하여 패스 II 동작들을 수행할 수도 있다 (390). 위에서 언급된 바와 같이, 패스 II 동작들은 동일한 하나 이상의 쉐이딩 유닛들 (40) 을 이용하여 도메인 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 수행하는 것을 포함할 수도 있다. 다시, GPU (36) 는 버텍스 쉐이딩을 수행하도록 다수의 쉐이딩 유닛들 (40) 을 지정할 수도 있지만, 쉐이딩 유닛들 (40) 의 각각이 버텍스 쉐이딩 동작들, 헐 쉐이딩 동작들, 도메인 쉐이딩 동작들, 및 지오메트리 쉐이딩 동작들을 수행하도록, 쉐이딩 유닛들 (40) 의 각각은 패스 II 동작들을 수행할 수도 있다.

[0252] GPU (36) 는 또한 서브-드로우 콜에 대하여 픽셀 쉐이딩 동작들을 수행할 수도 있다 (392). GPU (36) 는 하나 이상의 다른 쉐이딩 유닛들 (40) 을 이용하여 픽셀 쉐이딩 동작들을 수행할 수도 있다. 다른 예들에서, GPU (36) 는 서브-드로우 콜들의 모두가 완료된 후에 전체 드로우 콜에 대하여 픽셀 쉐이딩을 수행할 수도 있다.

[0253] 다음으로, GPU (36) 는 완료된 서브-드로우 콜이 드로우 콜의 최종 서브-드로우 콜인지 여부를 결정할 수도 있다 (394). 서브-드로우 콜이 드로우 콜의 최종 서브-드로우 콜일 경우, GPU (36) 는 드로우 콜과 연관된 렌더링된 그래픽스 데이터를 출력할 수도 있다. 서브-드로우 콜이 드로우 콜의 최종 서브-드로우 콜이 아닐 경우, GPU (36) 는 단계 (388) 로 복귀할 수도 있고, 다음 서브-드로우 콜에 대한 패스 I 동작들을 수행할 수도 있다.

[0254] 도 15 에 도시된 단계들은 하나의 예에 불과한 것으로서 제공된다는 것을 이해해야 한다. 즉, 도 15 에 도시된 단계들은 도시된 순서로 반드시 수행될 필요가 없고, 더 적은, 추가적인, 또는 대안적인 단계들이 수행될 수도 있다.

[0255] 도 16 은 이 개시물의 양태들에 따라, 2 패스 그래픽스 렌더링 프로세스의 제 1 패스와 연관된 그래픽스 렌더링 동작들을 예시하는 흐름도이다. 도 16 에 도시된 프로세스는 도 15 의 단계 (388) 에 대해 위에서 설명된 패스 I 동작들에 대응할 수도 있다. GPU (36; 도 1) 에 대하여 설명되었지만, 이 개시물의 양태들은 다양한 다른 컴포넌트들을 갖는 다양한 다른 GPU 들에 의해 수행될 수도 있다.

[0256] 도 16 의 예에서, GPU (36) 는 초기에 위에서 설명된 바와 같이, 그래픽스 렌더링 파이프라인의 버텍스 쉐이더 스테이지와 연관된 버텍스 쉐이딩 동작들을 수행하도록 하나 이상의 쉐이딩 유닛들 (40) 을 지정할 수도 있다 (400). 버텍스 쉐이딩 동작들을 수행한 후, 지정된 쉐이딩 유닛들 (40) 의 각각은 헐 쉐이딩 동작들을 위하여 쉐이딩된 버텍스들을 로컬 메모리에 저장할 수도 있다 (402). GPU (36) 는 또한, 헐 쉐이딩 동작들을 추적하기 위한 프로그램 카운터를 변화시킬 수도 있을 뿐만 아니라, 헐 쉐이더 자원들 오프셋에 대한 하나 이상의 자원 포인터들을 변화시킬 수도 있다. 예를 들어, 자원 포인터들은 헐 쉐이딩 동작들에 대하여 할당된 데이터 로케이션들을 지시할 수도 있다.

[0257] 이러한 의미에서, 쉐이딩 유닛들 (40) 의 각각은 헐 쉐이딩 동작들을 수행하도록 동작 모드들을 변화시킨다. 그러나, 모드 변화는 헐 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들 (40) 을 재지정하는 것을 포함하지 않는다. 즉, GPU (36) 의 컴포넌트들은 여전히, 버텍스 쉐이딩 동작들을 위해 지정된 쉐이딩 유닛의 1:1 인터페이스 포맷으로 데이터를 전송하고 데이터를 수신하도록 구성될 수도 있다.

[0258] 다음으로, GPU (36) 는 위에서 설명된 바와 같이, 버텍스 쉐이딩 동작들을 수행하였던 동일한 쉐이딩 유닛들 (40) 을 이용하여 그래픽스 렌더링 파이프라인의 헐 쉐이더 스테이지와 연관된 헐 쉐이딩 동작들을 수행할 수도 있다 (404). 예를 들어, 각각의 쉐이딩 유닛 (40) 은 테셀레이션을 위해 이용될 수도 있는 하나 이상의 제어 포인트들을 발생시키기 위하여 쉐이딩된 버텍스들에 대해 작용할 수도 있다.

[0259] 도 16 에 도시된 단계들은 하나의 예에 불과한 것으로서 제공된다는 것을 이해해야 한다. 즉, 도 16 에 도시된 단계들은 도시된 순서로 반드시 수행될 필요가 없고, 더 적은, 추가적인, 또는 대안적인 단계들이 수행될 수도 있다.

[0260] 도 17 은 이 개시물의 양태들에 따르면, 2 패스 그래픽스 렌더링 프로세스의 제 2 패스와 연관된 그래픽스 렌더링 동작들을 수행하는 것을 예시하는 흐름도이다. 도 17 에 도시된 프로세스는 도 15 의 단계 (390) 에 대하여 위에서 설명된 패스 II 동작들에 대응할 수도 있다. GPU (36; 도 1) 에 대하여 설명되었지만, 이 개시물의 양태들은 다양한 다른 컴포넌트들을 갖는 다양한 다른 GPU 들에 의해 수행될 수도 있다.

[0261] 도 17 의 예에서, GPU (36) 는 도 17 의 동작들을 수행하기 위하여 도 16 에 대하여 위에서 설명된 동일한 쉐이딩 유닛들 (40) 을 이용할 수도 있다. 예를 들어, 패스 II 동작들을 수행하기 위하여, 동일한 쉐이딩 유닛들 (40) 은 먼저, 위에서 설명된 바와 같이, 그래픽스 렌더링 파이프라인의 도메인 쉐이더 스테이지와 연관된 도메인 쉐이딩 동작들을 수행할 수도 있다 (420). 즉, 쉐이딩 유닛들 (40) 은 도메인 쉐이딩된 버텍스들을 발생시키기 위하여 (헐 쉐이더 스테이지로부터의) 제어 포인트들에 대해 작용할 수도 있다.

[0262] 도메인 쉐이딩 동작들을 수행한 후, 지정된 쉐이딩 유닛들 (40) 의 각각은 지오메트리 쉐이딩 동작들을 위하여 도메인 쉐이딩된 버텍스들을 로컬 메모리에 저장할 수도 있다 (422). GPU (36) 는 또한, 헐 쉐이딩 동작들을 추적하기 위한 프로그램 카운터를 변화시킬 수도 있을 뿐만 아니라, 헐 쉐이더 자원들 오프셋에 대한 하나 이상의 자원 포인터들을 변화시킬 수도 있다. 도 17 의 동작들이 도 16 에 대하여 설명된 것들에 후속하는 예들에서는, 이 기능들 (예를 들어, 값들을 로컬 메모리에 저장하는 것, 프로그램 카운터를 변화시키는 것, 자원 오프셋들을 변화시키는 것) 은 또한 단계 (420) 이전에 수행될 수도 있다.

[0263] 이러한 의미에서, 쉐이딩 유닛들 (40) 의 각각은 도메인 쉐이딩 및 지오메트리 쉐이딩 동작들을 수행하도록 동작 모드들을 변화시킨다. 그러나, 모드 변화는 도메인 쉐이딩 및 지오메트리 쉐이딩 동작들을 수행하도록 쉐이딩 유닛들 (40) 을 재지정하는 것을 포함하지 않는다. 즉, GPU (36) 의 컴포넌트들은 여전히, 버텍스 쉐이딩 동작들을 위해 지정된 하드웨어 쉐이딩 유닛의 1:1 인터페이스 포맷으로 데이터를 전송하고 데이터를 수신하도록 구성될 수도 있다.

[0264] 다음으로, GPU (36) 는 위에서 설명된 바와 같이, 도메인 쉐이딩 동작들을 수행하였던 동일한 쉐이딩 유닛들

(40) 을 이용하여 그래픽스 렌더링 파이프라인의 지오메트리 쉐이더 스테이지와 연관된 지오메트리 쉐이딩 동작들을 수행할 수도 있다 (424). 예를 들어, 각각의 쉐이딩 유닛 (40) 은 하나 이상의 지오메트리 쉐이딩된 버텍스들을 발생시키기 위하여 도메인 쉐이딩된 버텍스들에 대해 작용할 수도 있다.

[0265] 도 17 에 도시된 단계들은 하나의 예에 불과한 것으로서 제공된다는 것을 이해해야 한다. 즉, 도 17 에 도시된 단계들은 도시된 순서로 반드시 수행될 필요가 없고, 더 적은, 추가적인, 또는 대안적인 단계들이 수행될 수도 있다.

[0266] 도 18 은 이 개시물의 양태들에 따라, 동일한 하드웨어 쉐이딩 유닛에 의한 실행을 위하여 하나를 초과하는 쉐이더 스테이지를 함께 패치하는 것을 예시하는 흐름도이다. GPU (36, 도 1) 에 대하여 설명되었지만, 이 개시물의 양태들은 다양한 다른 컴포넌트들을 갖는 다양한 다른 GPU 들에 의해 수행될 수도 있다.

[0267] 도 18 의 예에서, GPU (36) 는 제 1 쉐이더 스테이지와 연관된 쉐이딩 동작들을 수행하도록 하나 이상의 하드웨어 쉐이딩 유닛들, 예를 들어, 하나 이상의 쉐이딩 유닛들 (40) 을 지정할 수도 있다 (440). 일부의 예들에서, 제 1 쉐이더 스테이지는 버텍스들을 발생시키기 위한 버텍스 쉐이더 스테이지일 수도 있어서, GPU (36) 는 버텍스 쉐이딩 동작들을 수행하도록 하나 이상의 쉐이딩 유닛들을 지정한다.

[0268] 제 1 쉐이더 스테이지와 연관된 동작들을 완료할 시에, GPU (36) 는 동작 모드들을 스위칭할 수도 있어서, 동일한 쉐이딩 유닛들 (40) 이 다양한 다른 쉐이딩 동작들을 수행하도록 할 수도 있다 (442). 예를 들어, 위에서 설명된 바와 같이, GPU (36) 는 제 2 쉐이딩 동작들을 수행하기 위한 프로그램 카운터 및 하나 이상의 자원 포인터들을 변화시킬 수도 있다.

[0269] 일부의 예들에서, GPU (36) 는 실행되고 있는 드로우 콜과 연관된 모드 정보에 기초하여 쉐이딩 유닛들 (40) 의 동작 모드를 스위칭할 수도 있다. 예를 들어, (GPU 드라이버 (50) 와 같은) GPU (36) 의 드라이버는 드로우 콜에서 어느 쉐이더 스테이지들이 실행되어야 하는지를 표시하는 드로우 콜을 위한 모드 번호를 발생시킬 수도 있다. GPU (36) 는 위에서 설명된 바와 같이, 패치 코드를 실행할 시에 쉐이딩 유닛들의 동작 모드들을 변화시키기 위하여 이 모드 번호를 이용할 수도 있다.

[0270] 이하에서 도시된 표 8 은 쉐이더 스테이지들의 다양한 조합들에 대한 모드 번호들을 포함하는 모드 정보를 일반적으로 예시한다.

표 8

[0271] **쉐이더 파이프라인 구성들**

VS	(HS, *TE*,DS)	GS	SO	PS	**드로우 모드**
온	오프	오프	오프	온	모드 0
온	오프	오프	온	온/오프	모드 0
온	오프	온	오프	온	모드 1
온	오프	온	온	온/오프	모드 1
온	온	오프	오프	온	패스 1: 모드 2 패스 2 : 모드 3
온	온	오프	온	온/오프	패스 1: 모드 2 패스 2 : 모드 3
온	온	온	오프	온	패스 1: 모드 2 패스 2 : 모드 4
온	온	온	온	온	패스 1: 모드 2 패스 2 : 모드 4

[0272] 표 8 에 도시된 바와 같이, 각각의 모드는 어느 쉐이더 스테이지들이 쉐이딩 유닛들에 의해 수행되는지를 기술한다. 따라서, GPU (36) 는 쉐이더 명령들을 함께 스트링 (string) 할 수도 있어서, 동일한 쉐이딩 유닛들 (40) 이 다수의 쉐이딩 동작들을 수행하도록 할 수도 있다. 즉, GPU (36) 는 실행되고 있는 드로우 콜의 모드 번호에 기초하여 적절한 쉐이더 명령들을 함께 패치할 수 있다.

[0273] 이러한 방법으로, 다음으로, GPU (36) 는 제 1 쉐이딩 동작들을 수행하도록 지정된 동일한 쉐이딩 유닛들 (40) 로 제 2 쉐이딩 동작들을 수행할 수도 있다 (444). 예를 들어, GPU (36) 는 상기 표 8 에 도시된 바와 같이, 버텍스 쉐이딩 동작들, 헐 쉐이딩 동작들, 도메인 쉐이딩 동작들, 및 지오메트리 쉐이딩 동작들의 조합을

수행할 수도 있다.

[0274] 도 18 예 도시된 단계들은 하나의 예에 불과한 것으로 제공된다는 것을 이해해야 한다. 즉, 도 18 에 도시된 단계들은 도시된 순서로 반드시 수행될 필요가 없고, 더 적은, 추가적인, 또는 대안적인 단계들이 수행될 수도 있다.

[0275] 위에서 설명된 어떤 예들은 버텍스 쉐이딩 동작들을 수행하도록 하드웨어 쉐이딩 유닛들을 초기에 지정하는 것과, 동일한 하드웨어 쉐이딩 유닛들로 다른 쉐이딩 동작들을 수행하는 것으로 천이하는 것을 포함하지만, 이 개시물의 기술들은 이러한 방법으로 제한되지 않는다는 것에 주목해야 한다. 예를 들어, GPU 는 초기에 다양한 다른 쉐이딩 동작들을 수행하도록 하드웨어 쉐이딩 유닛들의 세트를 지정할 수도 있다. 즉, GPU 가 3 개의 상이한 쉐이딩 동작들을 수행하도록 하드웨어 쉐이딩 유닛들을 지정하하는 시스템에서, GPU 는 버텍스 쉐이딩 동작들, 헐 쉐이딩 동작들, 및 픽셀 쉐이딩 동작들을 수행하도록 하드웨어 쉐이딩 유닛들을 지정할 수도 있다. 이 예에서, GPU 는 초기에 헐 쉐이딩 동작들을 수행하도록 하나 이상의 하드웨어 쉐이딩 유닛들을 지정할 수도 있지만, 위에서 설명된 바와 같이, 동일한 하드웨어 쉐이딩 유닛들로 도메인 쉐이딩 동작들 및 지오메트리 쉐이딩 동작들을 또한 수행할 수도 있다. 다양한 다른 동작 조합들이 또한 가능하다.

[0276] 하나 이상의 예들에서는, 설명된 기능들이 하드웨어, 소프트웨어, 펌웨어, 또는 그 임의의 조합으로 구현될 수도 있다. 소프트웨어로 구현될 경우, 기능들은 하나 이상의 명령들 또는 코드로서, 비-일시적인 컴퓨터-판독가능한 매체를 포함하는 제조 물품 상에 저장될 수도 있다. 컴퓨터-판독가능한 매체들은 컴퓨터 데이터 저장 매체들을 포함할 수도 있다. 데이터 저장 매체들은 이 개시물에서 설명된 기술들의 구현을 위한 명령들, 코드 및/또는 데이터 구조들을 취출하기 위해 하나 이상의 컴퓨터들 또는 하나 이상의 프로세서들에 의해 액세스될 수 있는 임의의 이용가능한 매체들일 수도 있다. 제한이 아닌 예로서, 이러한 컴퓨터-판독가능한 매체들은 RAM, ROM, EEPROM, CD-ROM 또는 다른 광학 디스크 저장, 자기 디스크 저장, 또는 다른 자기 저장 디바이스들, 플래쉬 메모리, 또는 명령들 또는 데이터 구조들의 형태로 희망하는 프로그램 코드를 운반하거나 저장하기 위해 이용될 수 있으며 컴퓨터에 의해 액세스될 수 있는 임의의 다른 매체를 포함할 수 있다. 본원에서 이용된 바와 같이, 디스크 (disk) 및 디스크 (disc) 는 컴팩트 디스크 (CD), 레이저 디스크, 광학 디스크, 디지털 다기능 디스크 (DVD), 플로피 디스크 및 블루레이 디스크 (Blu-ray disc) 를 포함하고, 여기서 디스크 (disk) 들은 통상 데이터를 자기적으로 재생하는 반면, 디스크 (disc) 들은 데이터를 레이저들로 광학적으로 재생한다. 상기의 조합들은 컴퓨터-판독가능한 매체들의 범위 내에 또한 포함되어야 한다.

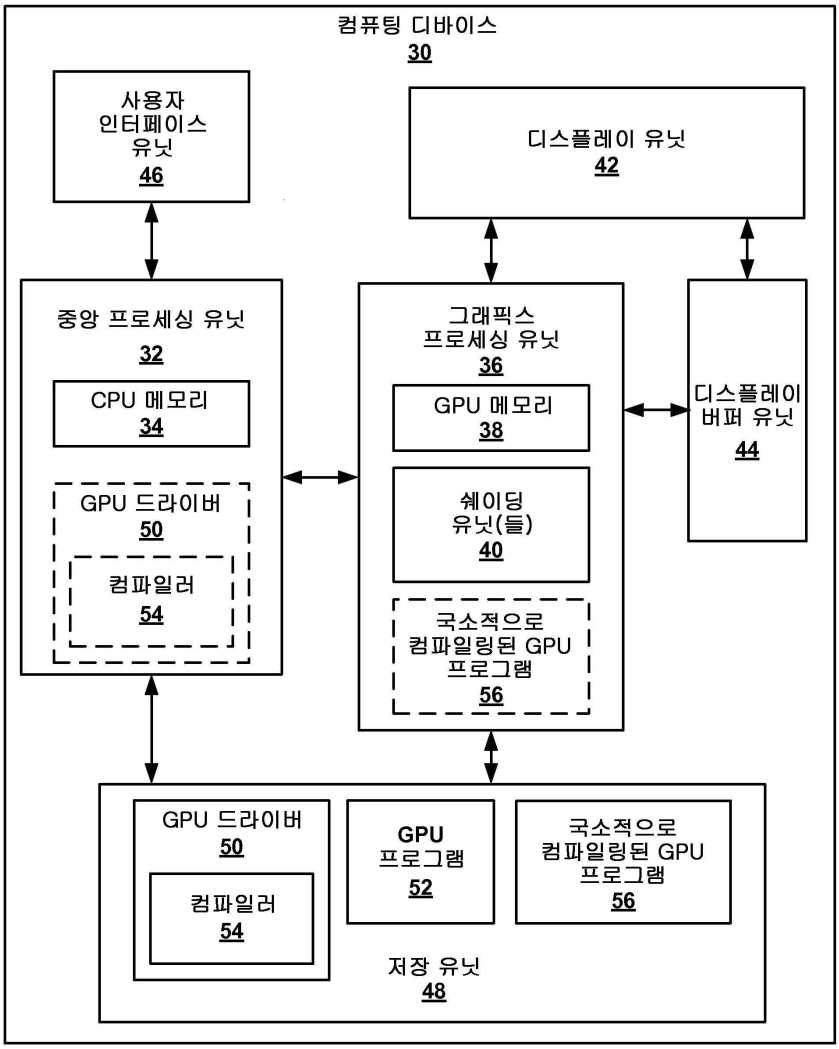
[0277] 코드는 하나 이상의 DSP 들, 범용 마이크로프로세서들, ASIC 들, FPGA 들, 또는 다른 등가의 통합된 또는 별개의 로직 회로부와 같은 하나 이상의 프로세서들에 의해 실행될 수도 있다. 추가적으로, 일부의 양태들에서는, 본원에서 설명된 기능성이 전용 하드웨어 및/또는 소프트웨어 모듈들 내에 제공될 수도 있다. 또한, 기술들은 하나 이상의 회로들 또는 로직 엘리먼트들에서 완전히 구현될 수 있다.

[0278] 이 개시물의 기술들은 무선 핸드셋, 집적 회로 (IC) 또는 IC 의 세트 (예를 들어, 칩셋) 를 포함하는 광범위한 디바이스들 또는 장치들에서 구현될 수도 있다. 다양한 컴포넌트들, 모듈들, 또는 유닛들은 개시된 기술들을 수행하도록 구성된 디바이스들의 기능적 양태들을 강조하기 위하여 이 개시물에서 설명되지만, 상이한 하드웨어 유닛들에 의한 실현을 반드시 요구하지는 않는다. 오히려, 위에서 설명된 바와 같이, 다양한 유닛들은 코덱 하드웨어 유닛에서 조합될 수도 있거나, 적당한 소프트웨어 및/또는 펌웨어와 함께, 위에서 설명된 바와 같은 하나 이상의 프로세서들을 포함하는 상호작용 하드웨어 유닛들의 집합에 의해 제공될 수도 있다.

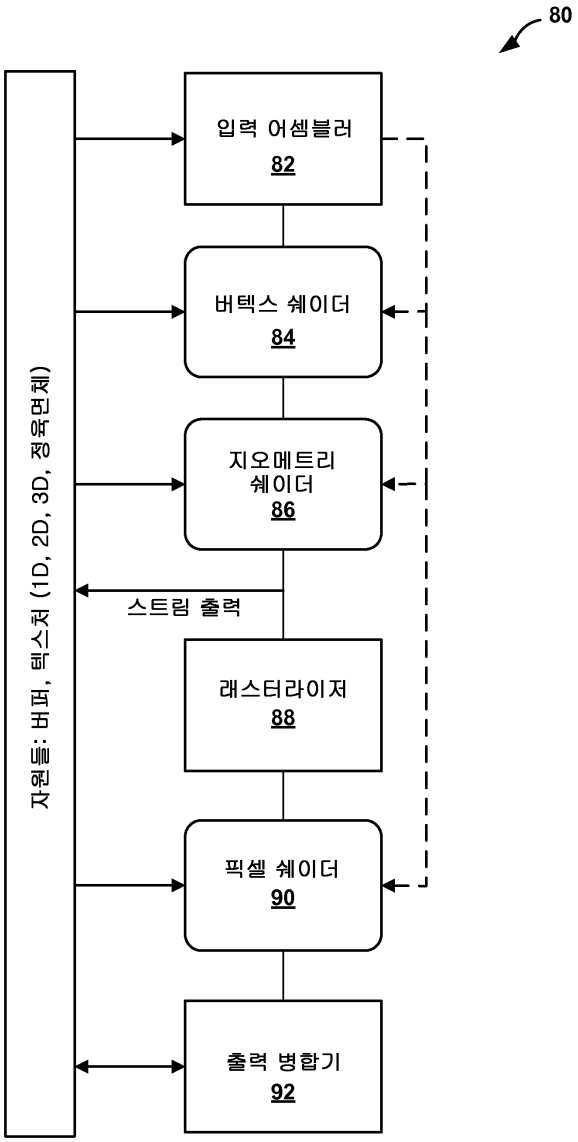
[0279] 다양한 예들이 설명되었다. 이러한 그리고 다른 예들은 다음의 청구항들의 범위 내에 있다.

도면

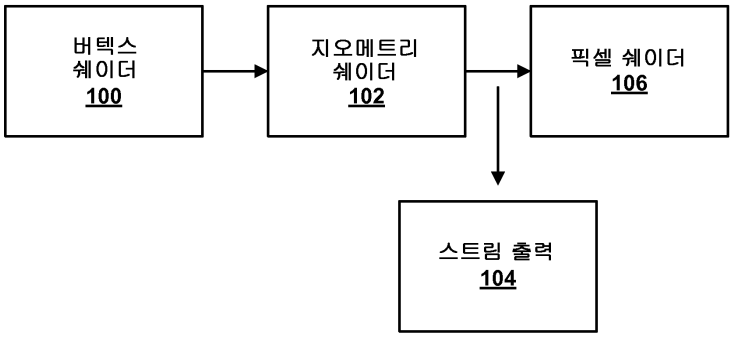
도면1


컴퓨팅 디바이스
30
사용자
인터페이스
유닛
46
디스플레이 유닛
42
중앙 프로세싱 유닛
32
CPU 메모리
34
GPU 드라이버
50
컴파일러
54
그래픽스
프로세싱 유닛
36
GPU 메모리
38
쉐이딩
유닛(들)
40
국소적으로
컴파일링된 GPU
프로그램
56
디스플레이
버퍼 유닛
44
GPU 드라이버
50
컴파일러
54
GPU
프로그램
52
국소적으로
컴파일링된 GPU
프로그램
56
저장 유닛
48

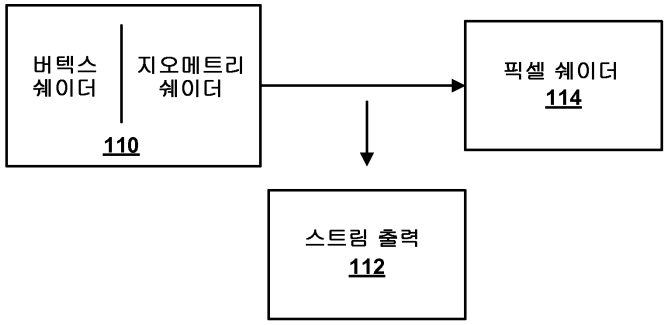
도면2


80
입력 어셈블러
82
버텍스 쉐이더
84
지오메트리
쉐이더
86
스트림 출력
래스터라이저
88
픽셀 쉐이더
90
출력 병합기
92
자원들: 버퍼, 텍스쳐 (1D, 2D, 3D, 정육면체)

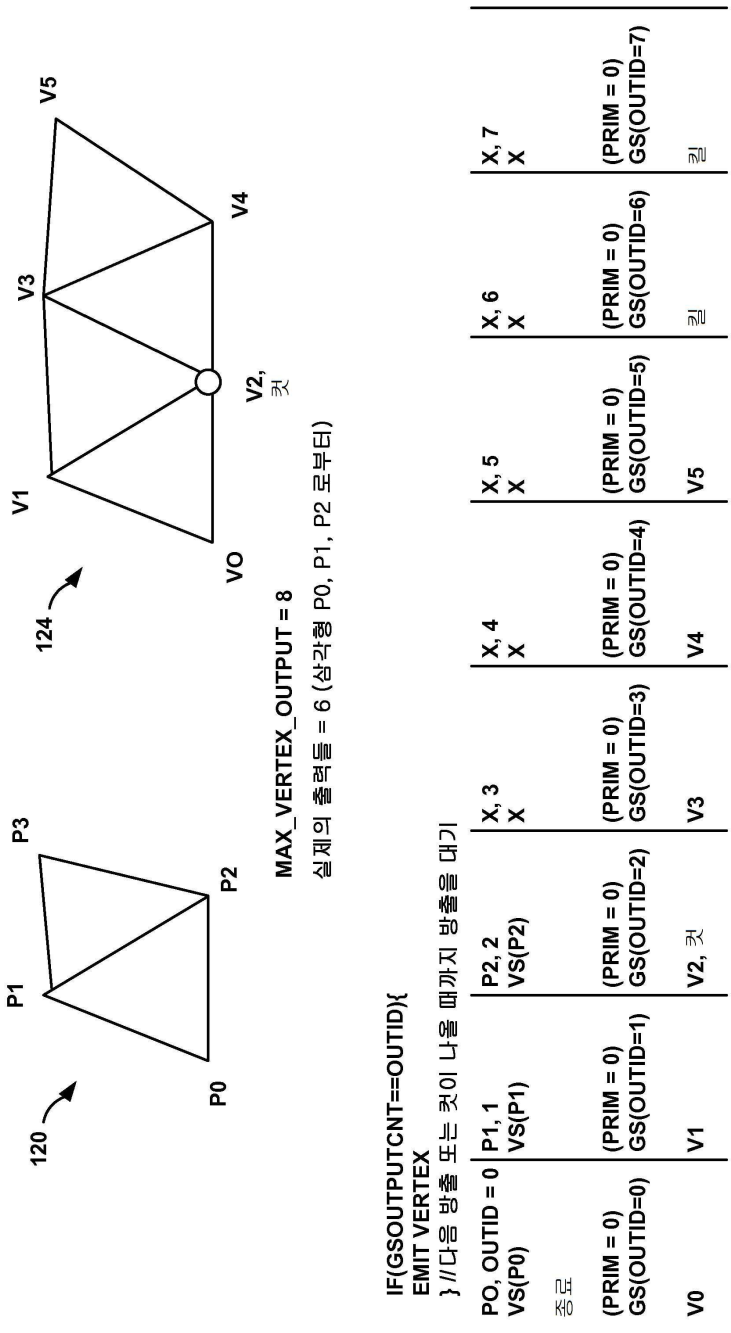
도면3a


버텍스
쉐이더
100
지오메트리
쉐이더
102
픽셀 쉐이더
106
스트림 출력
104

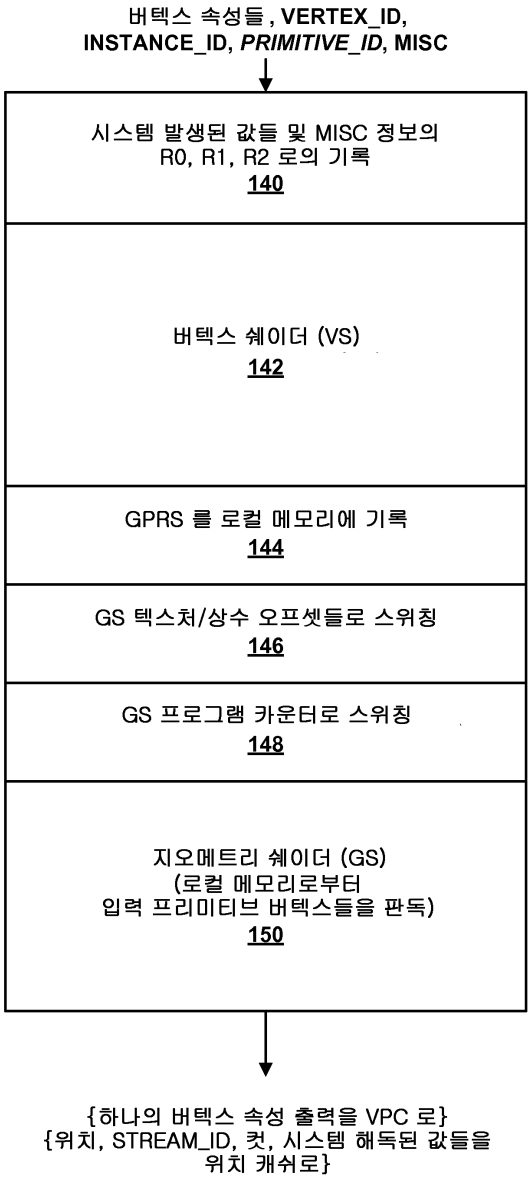
도면3b


버텍스 쉐이더
지오메트리 쉐이더
110
픽셀 쉐이더
114
스트림 출력
112

도면4


P0, OUTID = 0
VS(P0)
종료
(PRIM = 0)
GS(OUTID=0)
V0
P1, 1
VS(P1)
(PRIM = 0)
GS(OUTID=1)
V1
P2, 2
VS(P2)
(PRIM = 0)
GS(OUTID=2)
V2, 컷
X, 3
X
(PRIM = 0)
GS(OUTID=3)
V3
X, 4
X
(PRIM = 0)
GS(OUTID=4)
V4
X, 5
X
(PRIM = 0)
GS(OUTID=5)
V5
X, 6
X
(PRIM = 0)
GS(OUTID=6)
킬
X, 7
X
(PRIM = 0)
GS(OUTID=7)
킬
IF(GSOUTPUTCNT==OUTID){
EMIT VERTEX
} //다음 방출 또는 컷이 나올 때까지 방출을 대기
MAX_VERTEX_OUTPUT = 8
실제의 출력들 = 6 (삼각형 P0, P1, P2 로부터)
120
P0
P1
P2
P3
124
V0
V1
V2, 컷
V3
V4
V5

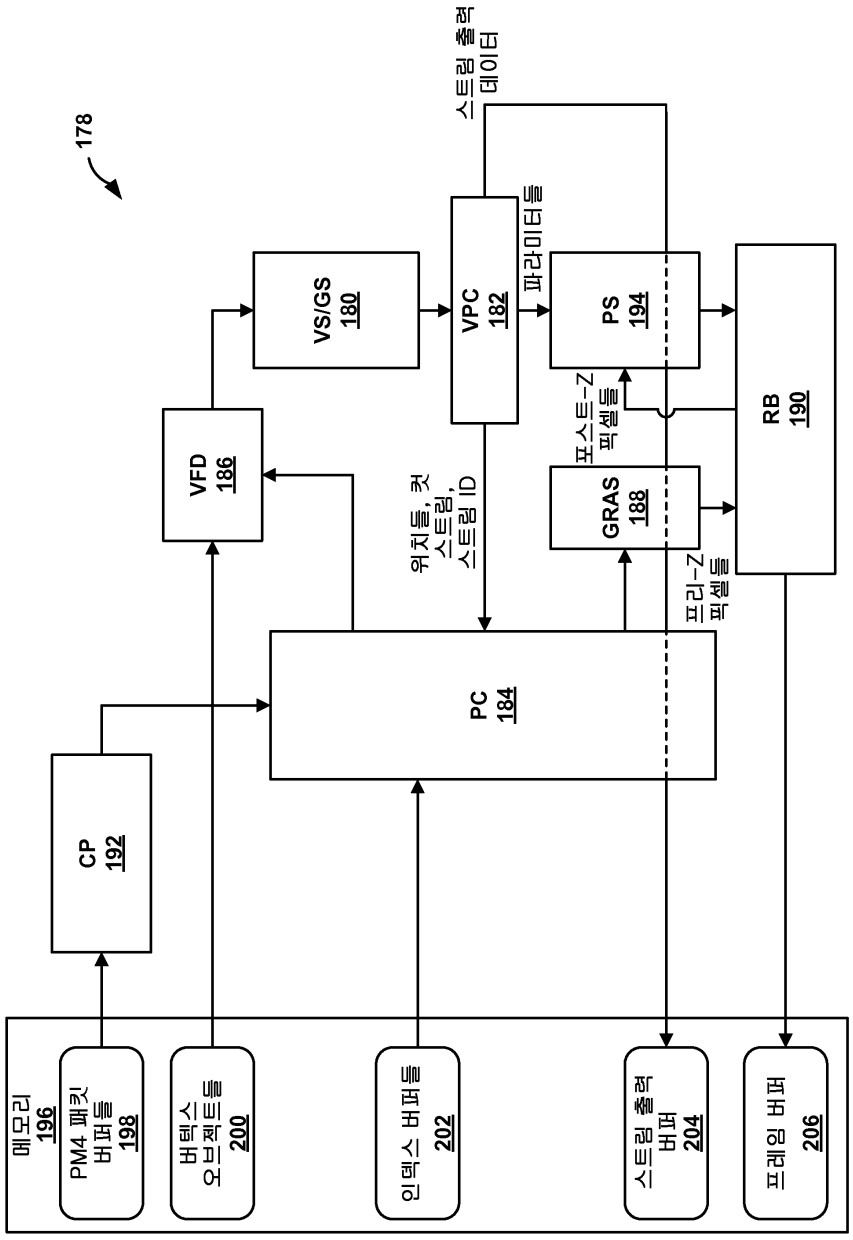
도면5a


버텍스 속성들, VERTEX_ID, INSTANCE_ID, PRIMITIVE_ID, MISC
시스템 발생된 값들 및 MISC 정보의 R0, R1, R2 로의 기록
140
버텍스 쉐이더 (VS)
142
GPRS 를 로컬 메모리에 기록
144
GS 텍스처/상수 오프셋들로 스위칭
146
GS 프로그램 카운터로 스위칭
148
지오메트리 쉐이더 (GS)
(로컬 메모리로부터
입력 프리미티브 버텍스들을 판독)
150
{하나의 버텍스 속성 출력을 VPC 로}
{위치, STREAM_ID, 컷, 시스템 해독된 값들을
위치 캐쉬로}

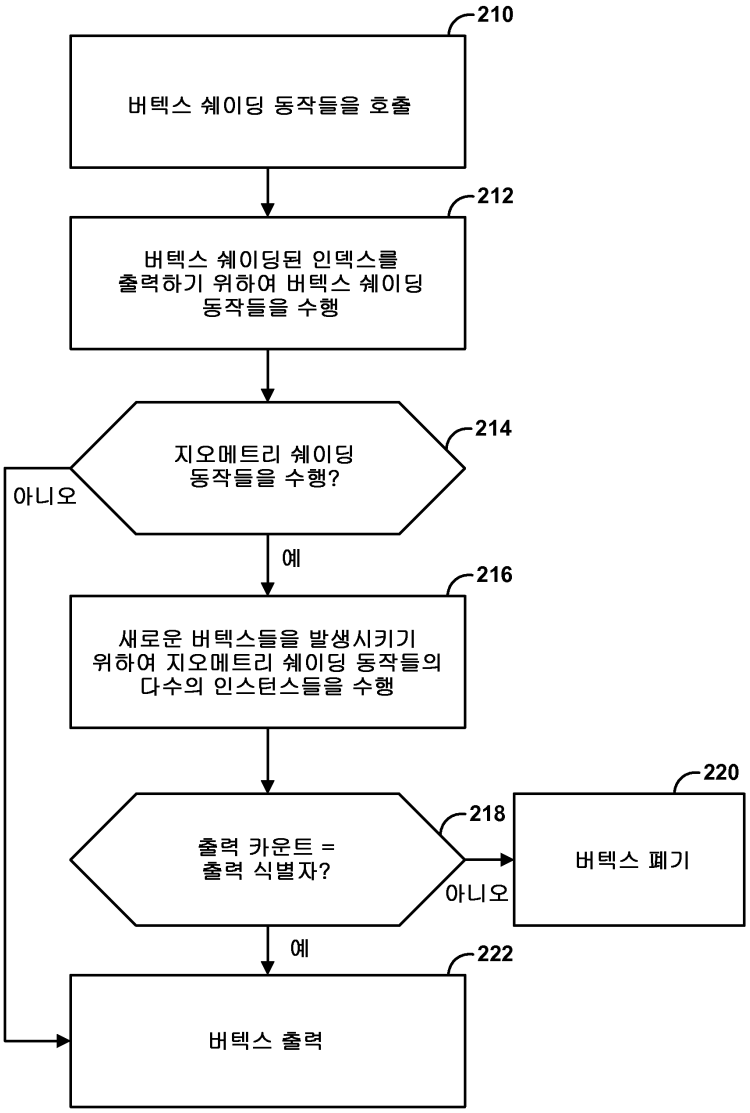
도면5b

```
VS:   'Normal' VS code   //instanceID 를 이용할 수 있고, 초기 레지스터들은 primitiveiD,
                         //misc, rel_prim 을 저장
                         //GPR 들에 남아 있는 데이터
      END_FINAL;         //DX9 스타일 쉐이더일 경우에 종료 – 이와 다를 경우에는 무시하고,
                         //쉐이더가 종료될 경우에 gprs 를 VPC 에 기록하는 것은 SP 인 것에 주목

      LM_Vs_offset = rel_primID*PRIM_SIZE + rel_vertex*VERT_SIZE;

      Write vertex data  in GPR to LM_Vs_offset

      CHMSK              //제 2 스테이지에 대하여 cov_mask_1 을 이용하고, 제 2 쉐이더 자원들로
                         //스위칭 (자원 ptr 을 GS 자원들 오프셋으로
                         //스위칭)

      CHSH               //GS 프로그램 카운터 및 상태 오프셋들로 스위칭

GS:   For  (vertex_id= 0; vertex_id++; vertex_id <= max_input)
        {    Load from LM using  offset = rel_primID*PRIM_SIZE + vertex_id *
             VERT_SIZE
             // HLSQ 에 의해 컴퓨팅된 기본 어드레스를 이용하여 LM 으로부터 페치
        }
      for(streamid =0;streamid++;streamid<4);
      cut[streamid] = true;
      gsoutount = 0;

// 'normal' GS code
      {..  using primitiveID, GSInstanceID
      CUT(streamid):     { Cut[streamid] = true;}
      EMIT(streamid): {
            If(Gsoutcount==GsoutvertID){
                  oPos = vertex.pos; oColor =vertex.clr;
                  oMisc = (cut[streamid],streamid)
                  gsoutcount = maxoutputvertexcount;
                  go to END_FINAL;   // 선택적
            }
            cut[streamid] = false;
            gsoutcount++;
      }

      If (gsoutcount < maxoutputvertexcount)
            KILL PRIM;   //임의의 "남겨진" 파이버들을 제거

END_FINAL;
```

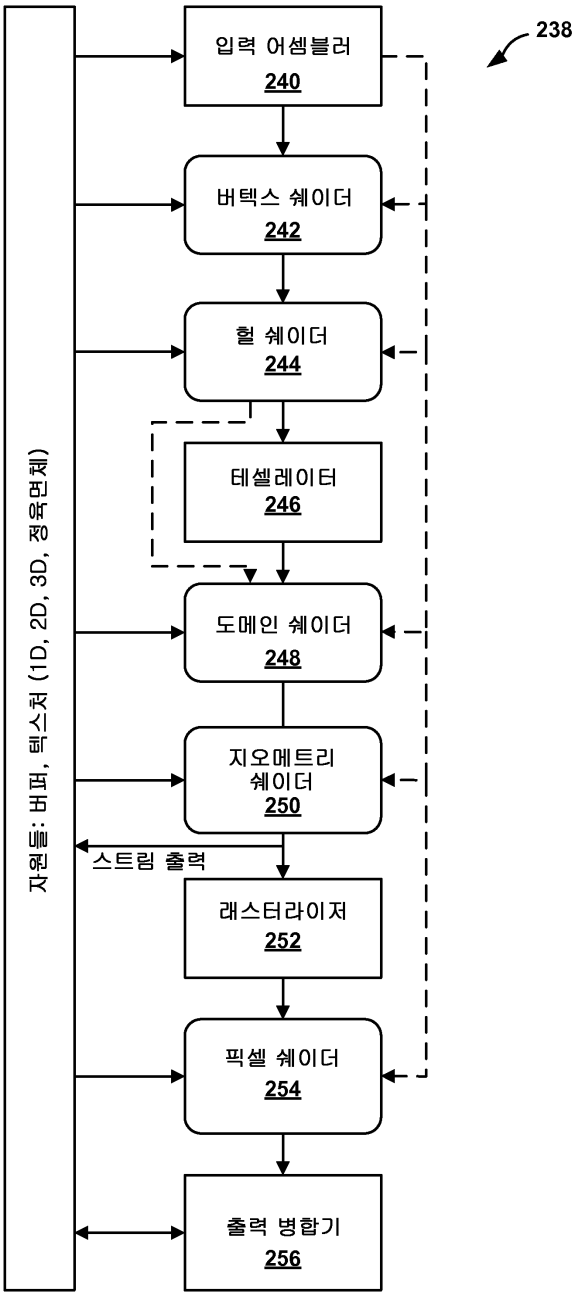
도면6


178
스트림 출력 데이터
VS/GS 180
VPC 182
파라미터들
PS 194
RB 190
VFD 186
포스트-Z 픽셀들
GRAS 188
위치들, 컷 스트림, 스트림 ID
프리-Z 픽셀들
PC 184
CP 192
메모리 196
PM4 패킷 버퍼들 198
버텍스 오브젝트들 200
인덱스 버퍼들 202
스트림 출력 버퍼 204
프레임 버퍼 206

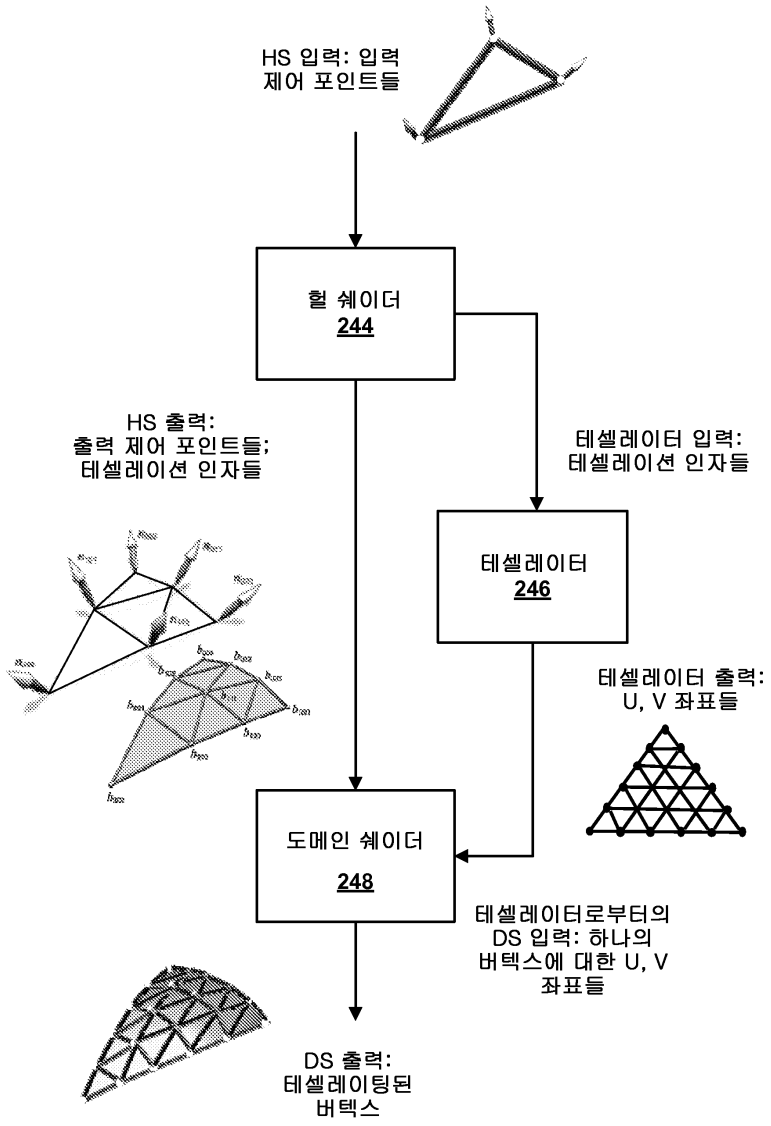
도면7


210
버텍스 쉐이딩 동작들을 호출
212
버텍스 쉐이딩된 인덱스를 출력하기 위하여 버텍스 쉐이딩 동작들을 수행
214
지오메트리 쉐이딩 동작들을 수행?
아니오
예
216
새로운 버텍스들을 발생시키기 위하여 지오메트리 쉐이딩 동작들의 다수의 인스턴스들을 수행
218
출력 카운트 = 출력 식별자?
아니오
220
버텍스 폐기
예
222
버텍스 출력

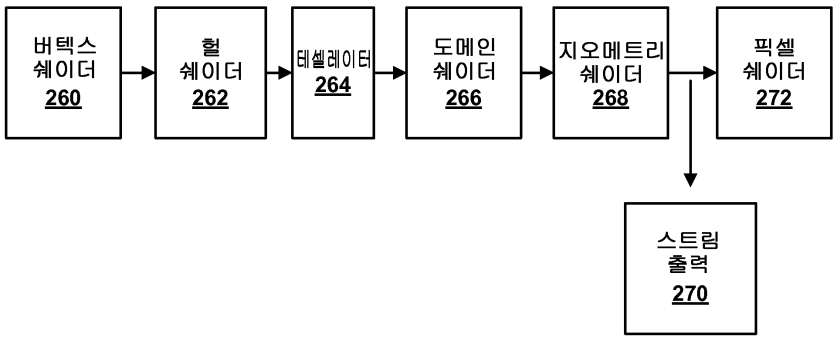
도면8


238
입력 어셈블러
240
버텍스 쉐이더
242
헐 쉐이더
244
테셀레이터
246
도메인 쉐이더
248
지오메트리 쉐이더
250
스트림 출력
래스터라이저
252
픽셀 쉐이더
254
출력 병합기
256
자원들: 버퍼, 텍스처 (1D, 2D, 3D, 정육면체)

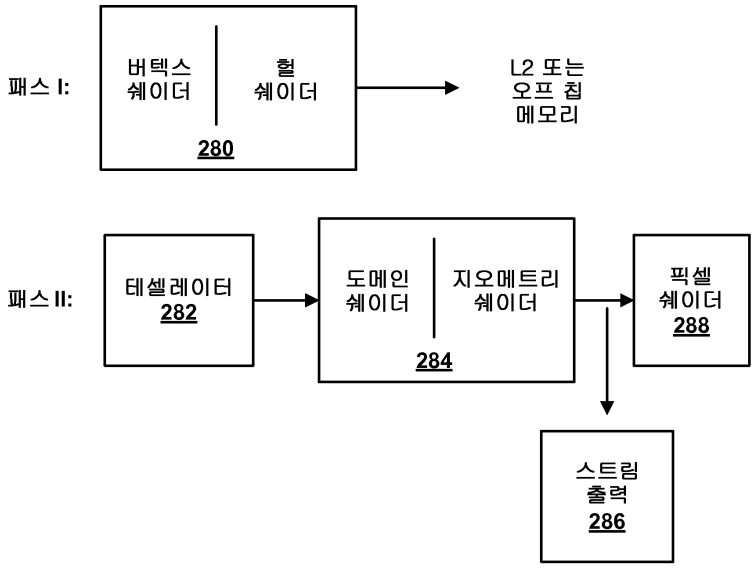
도면9


HS 입력: 입력 제어 포인트들
헐 쉐이더
244
HS 출력:
출력 제어 포인트들;
테셀레이션 인자들
테셀레이터 입력:
테셀레이션 인자들
테셀레이터
246
테셀레이터 출력:
U, V 좌표들
도메인 쉐이더
248
테셀레이터로부터의
DS 입력: 하나의
버텍스에 대한 U, V
좌표들
DS 출력:
테셀레이팅된
버텍스

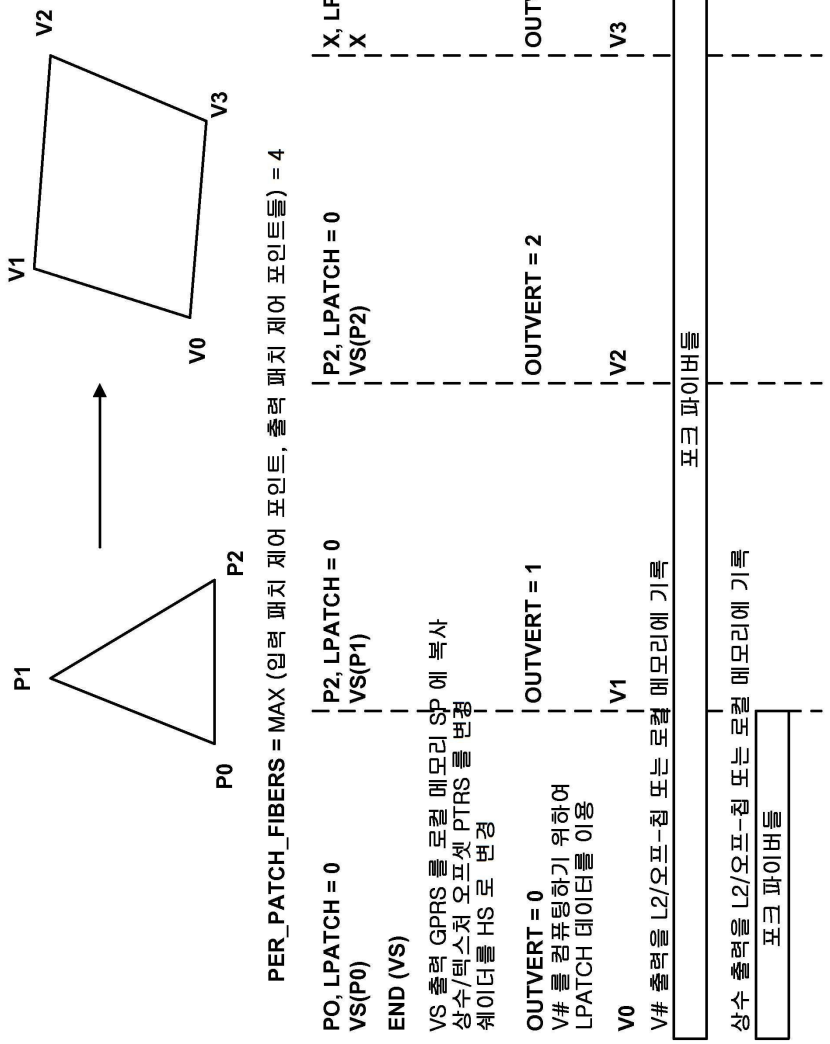
도면10a


버텍스
쉐이더
260
헐
쉐이더
262
테셀레이터
264
도메인
쉐이더
266
지오메트리
쉐이더
268
픽셀
쉐이더
272
스트림
출력
270

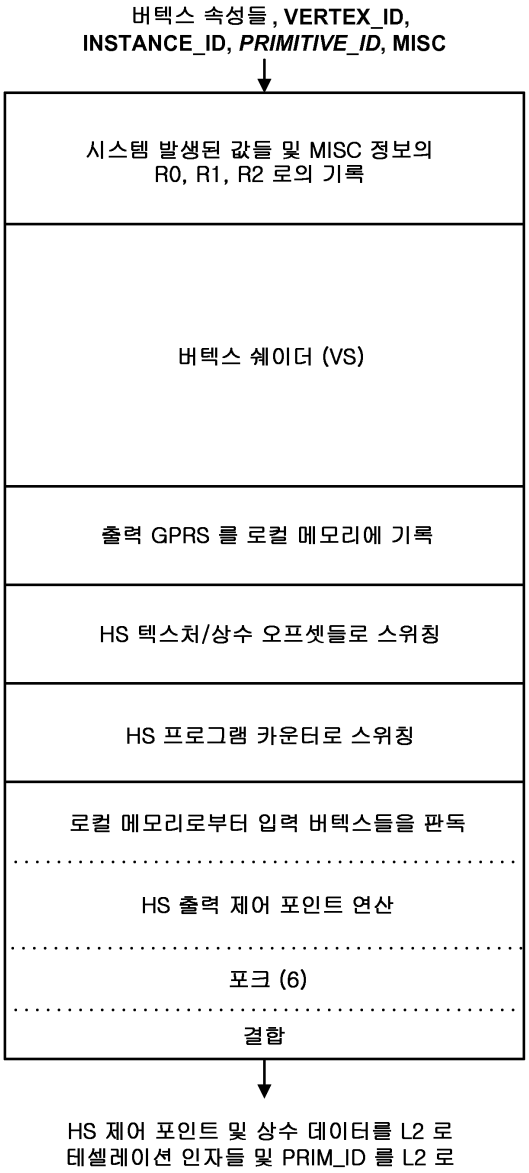
도면10b


패스 I:
버텍스 쉐이더
헐 쉐이더
280
L2 또는 오프 칩 메모리
패스 II:
테셀레이터 282
도메인 쉐이더
지오메트리 쉐이더
284
픽셀 쉐이더 288
스트림 출력 286

도면11


PER_PATCH_FIBERS = MAX (입력 패치 제어 포인트, 출력 패치 제어 포인트들) = 4
P0
P1
P2
V0
V1
V2
V3
PO, LPATCH = 0
VS(P0)
END (VS)
P2, LPATCH = 0
VS(P1)
VS 출력 GPRS 를 로컬 메모리 SP 에 복사
상수/텍스처 오프셋 PTRS 를 변경
쉐이더를 HS 로 변경
P2, LPATCH = 0
VS(P2)
X, LPATCH = 0
X
OUTVERT = 0
V# 를 컴퓨팅하기 위하여
LPATCH 데이터를 이용
OUTVERT = 1
V1
OUTVERT = 2
V2
OUTVERT = 3
V3
V0
V# 출력을 L2/오프-칩 또는 로컬 메모리에 기록
포크 파이버들
상수 출력을 L2/오프-칩 또는 로컬 메모리에 기록
포크 파이버들

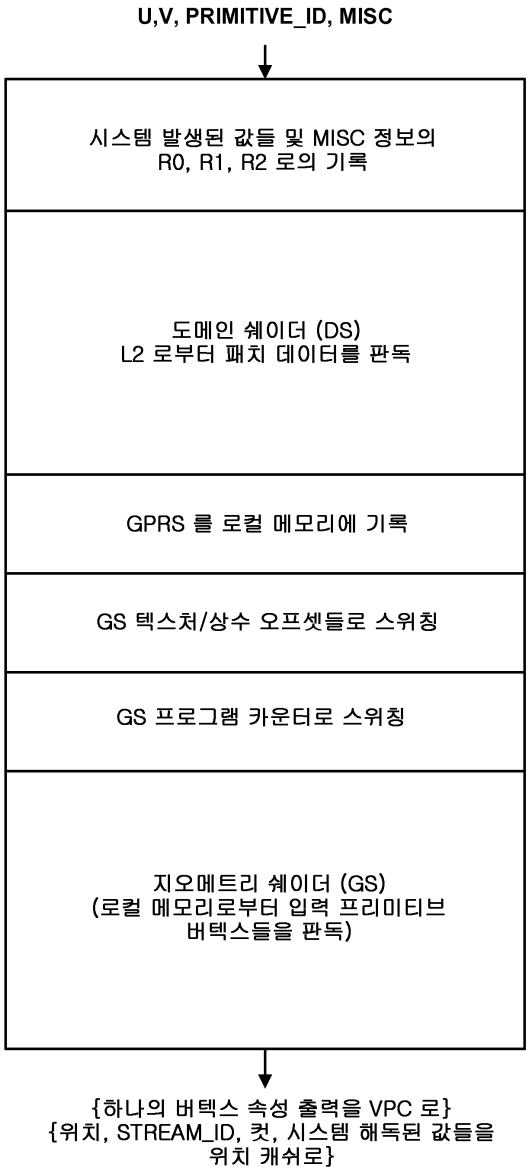
도면12a


버텍스 속성들, VERTEX_ID,
INSTANCE_ID, PRIMITIVE_ID, MISC
시스템 발생된 값들 및 MISC 정보의
R0, R1, R2 로의 기록
버텍스 쉐이더 (VS)
출력 GPRS 를 로컬 메모리에 기록
HS 텍스처/상수 오프셋들로 스위칭
HS 프로그램 카운터로 스위칭
로컬 메모리로부터 입력 버텍스들을 판독
HS 출력 제어 포인트 연산
포크 (6)
결합
HS 제어 포인트 및 상수 데이터를 L2 로
테셀레이션 인자들 및 PRIM_ID 를 L2 로

도면12b

```
//VFD 는 버텍스 인덱스 및 유효한 버텍스에 기초하여 버텍스 데이터를
//페치. 그것은 버텍스 데이터를 유효한 버텍스들로 로딩해야 함.
VS:   'Normal' VS  code     //instanceID 를 이용할 수 있음
                            // GPR 들에 남아 있는 데이터
      END_FINAL;            // DX9 스타일 쉐이더일 경우에 종료 - 그 외에는 무시

      LM_Vs_offset = rel_primID*PRIM_SIZE + rel_vertex*VERT_SIZE;

      Write vertex data  in GPR to LM_Vs_offset

      CHMSK;                //제 2 스테이지에 대하여 cov_mask_1 을 이용하고, 제 2 쉐이더 자원들로
                            //스위칭 (자원 ptr 을 HS 자원들 오프셋으로
                            //스위칭)

      CHSH                  //쉐이더 프로그램 카운터 및 상태 오프셋들로 스위칭
                            //(HS PC/오프셋들)

HS:   For  (vertex_id= 0; vertex_id++; vertex_id <= max_input)
      {     Load from LM using  offset = rel_primID*PRIM_SIZE + vertex_id *
            VERT_SIZE
            //HLSQ 에 의해 컴퓨팅된 기본 어드레스를 이용하여 LM 으로부터 페치
      }

      For (hsoutcount =0;hscountcount++;hsoutcount<=maxout)    //이것은 포크(#_output_pnts)를
                                          //이용하여 행해질 수 있음
      {     //HS 코드는 primitiveID 를 이용하여 … HS 출력 버텍스를 컴퓨팅

          Mem_offset = rel_patchid*OUTPUT_PRIM_SIZE +  outVertID *
          OUTPUT_VERTEX_SIZE + SCRATCH_MEMORY_CONTROL_BASE

          If (hscount == outVertID)         //패치 출력 포인트들이 직렬로
                                            //프로세싱될 경우에 행해짐
              (컴퓨팅된 버텍스를 mem_offset 으로 엑스포트)
      }
      For (fork_id = outVertID;fork_id += WAVE_OUT_SIZE;fork_id <n)
      {
            Fork(fork_id)           //병렬 연산을 위하여 firbers_per_prim 사이에
                                    //fork(n) 명령을 확산시킴
      }
          if (outVertID == 0)
      {     Join                    // outVertID=0 인 하나의 파이버만을 이용
            Tf_offset = rel_patchid* TF_OUT _SIZE+  tess_attr +
            SCRATCH_MEMORY_TESS_BASE;     //테셀레이션 인자들을 테셀레이션 인자
                                          //메모리에 기록
            Write primitiveID
            Write tessellation factors
      }
       END_FINAL;
```

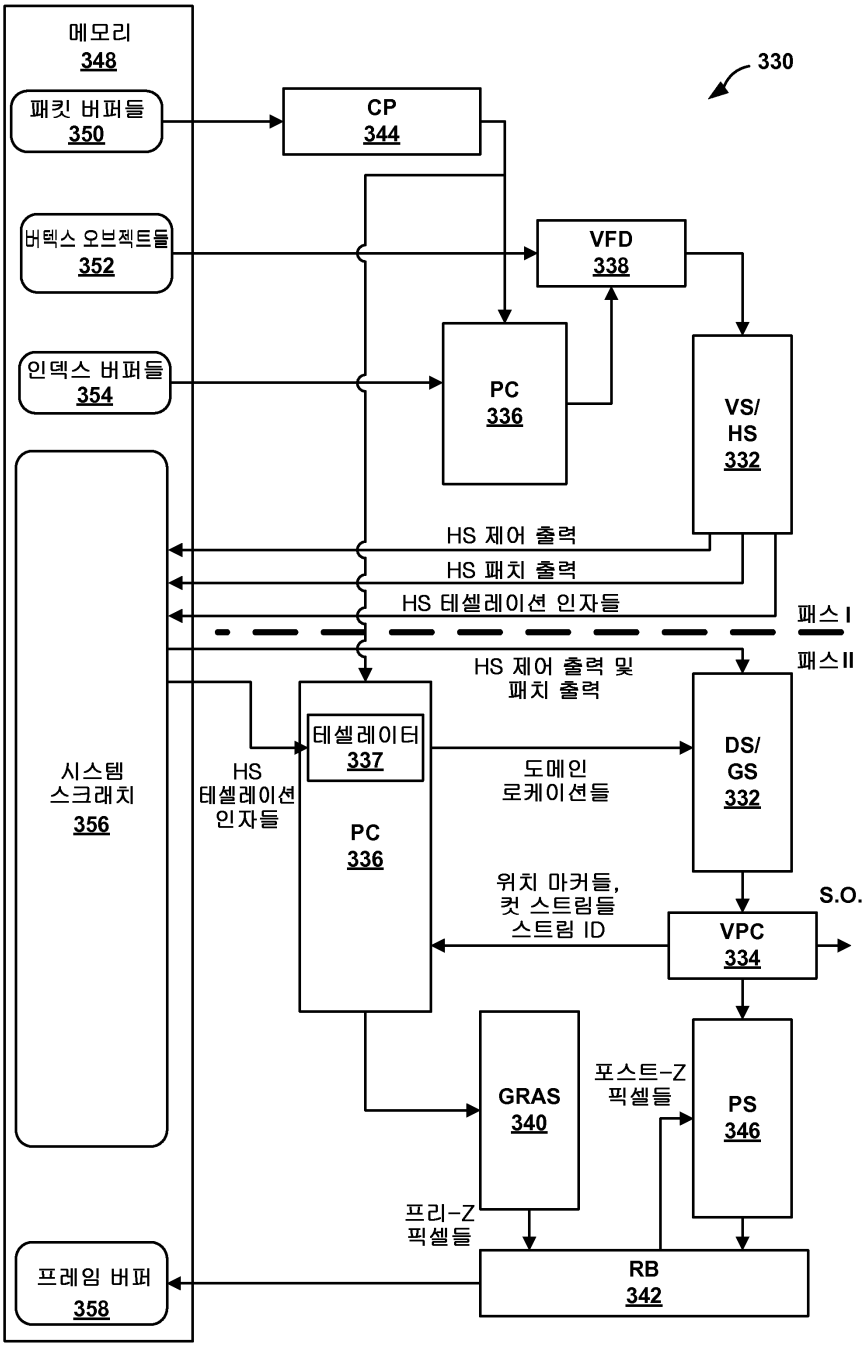
도면13a


U,V, PRIMITIVE_ID, MISC
시스템 발생된 값들 및 MISC 정보의
R0, R1, R2 로의 기록
도메인 쉐이더 (DS)
L2 로부터 패치 데이터를 판독
GPRS 를 로컬 메모리에 기록
GS 텍스쳐/상수 오프셋들로 스위칭
GS 프로그램 카운터로 스위칭
지오메트리 쉐이더 (GS)
(로컬 메모리로부터 입력 프리미티브
버텍스들을 판독)
{하나의 버텍스 속성 출력을 VPC 로}
{위치, STREAM_ID, 컷, 시스템 해독된 값들을
위치 캐쉬로}

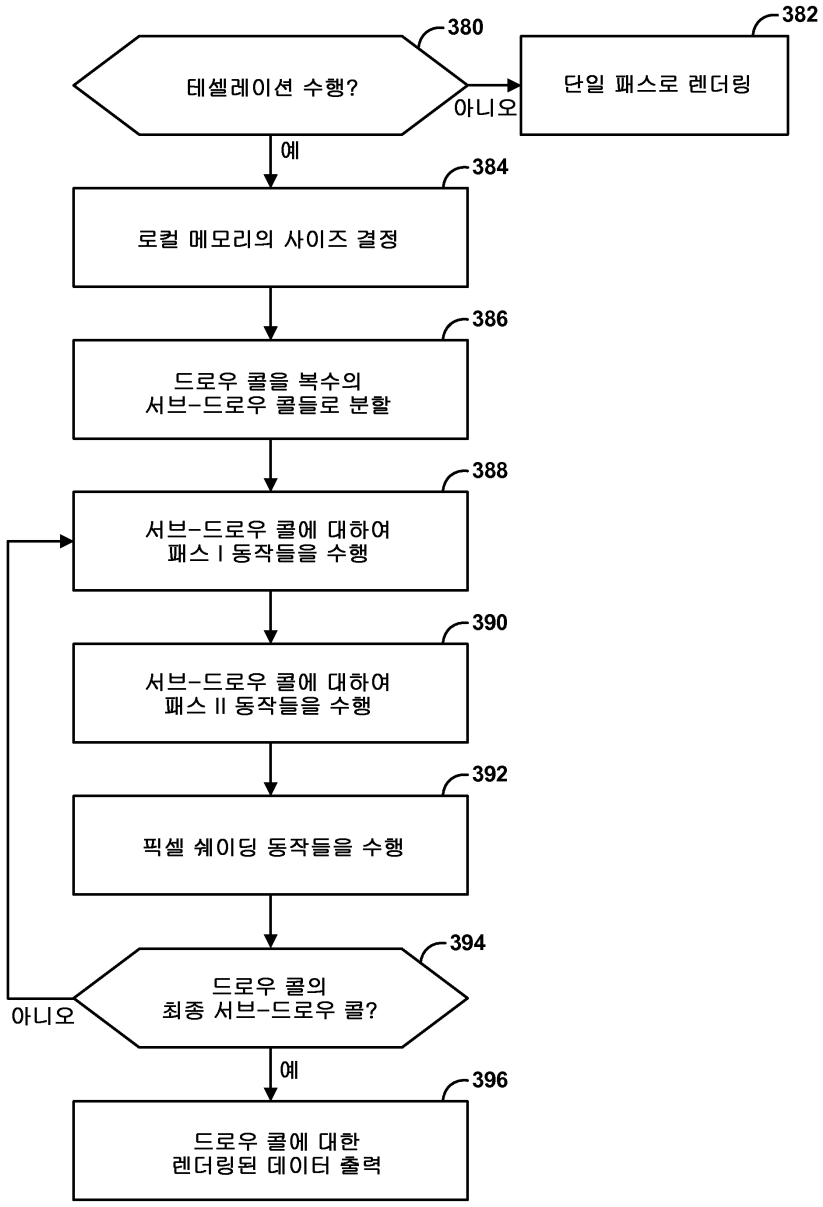
도면13b

```
// VFD 는 임의의 데이터를 폐치하지 않음
// InstanceID = rel_patchID
// (u,v) 는 상이한 gprs 에서 나옴
// 컴파일러는 gpr 에서의 patchID 에 대한 예상과 함께 ds 코드를 컴파일링하였고, 그래서,
   SP 는 rel_patchID 를 실제로 나타내는 instanceID 를 이 gpr 에 로딩할 수 있음

DS:   Compute patch_ptr = rel_patchid * OUTPUT_PRIM_SIZE +
      SCRATCH_MEMORY_CONTROL_BASE.
      'Normal' DS code
      // HS 출력 제어 포인트들을 액세스하기 위하여 patch_ptr +
        OUTPUT_VERTEX_SIZE * control_pointID 를 이용
      // (u,v) 를 이용하여 HS 상수 데이터를 액세스해서 테셀레이팅된 포인트를 컴퓨팅하기
        위하여 patch_ptr + OUTPUT_VERTEX_SIZE *num_output_control_pts 를 이용

                  // GPR 들에 남아 있는 데이터
      END_FINAL; // GS 없는 DX11 DS 쉐이더일 경우에 종료 – 그 외에는 무시
                  // 쉐이더가 종료될 경우에 gprs 를 VPC 에 기록하는 것은 SP 인 것에 주목
=================================================================
      LM_Vs_offset = rel_primID*PRIM_SIZE + rel_vertex*VERT_SIZE;

      Write vertex data  in GPR to LM_Vs_offset

      CHMSK;      // 자원 ptr 을 GS 자원들 오프셋으로 스위칭
                  // 제 2 스테이지에 대하여 cov_mask_1 을 이용

      CHSH        // GS 프로그램 카운터 및 상태 오프셋들로 스위칭
=================================================================
GS:   For  (vertex_id= 0; vertex_id++; vertex_id <= max_input)
        {   Load from LM using  offset = rel_primID*PRIM_SIZE + vertex_id *
            VERT_SIZE
            // HLSQ 에 의해 컴퓨팅된 기본 어드레스를 이용하여 LM 으로부터 페치
        }
      for(streamid =0;streamid++;streamid<4);
      cut[streamid] = true;
      gsoutount = 0;

// 'normal' GS code
      {.. using primitiveID, GSInstanceID
      CUT(streamid):     { Cut[streamid] = true;}
      EMIT(streamid): {
            If(Gsoutcount==GsoutvertID){
                  oPos = vertex.pos; oColor =vertex.clr;
                  oMisc = (cut[streamid],streamid)
                  gsoutcount = maxoutputvertexcount;
                  go to END_FINAL;   // 선택적
            }
            cut[streamid] = false;
            gsoutcount++;
      }

      If (gsoutcount < maxoutputvertexcount)
            KILL PRIM;  // 임의의 "남겨진" 파이버들을 제거

END_FINAL;
```

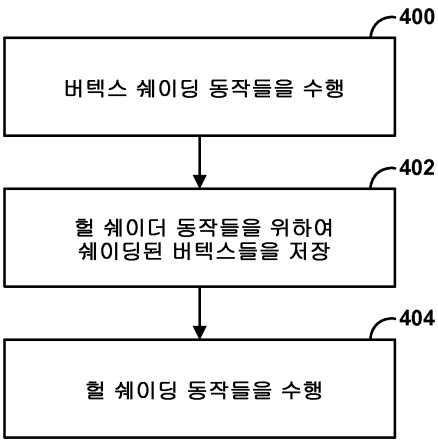
도면14


메모리
348
330
패킷 버퍼들
350
CP
344
버텍스 오브젝트들
352
VFD
338
인덱스 버퍼들
354
PC
336
VS/
HS
332
HS 제어 출력
HS 패치 출력
HS 테셀레이션 인자들
패스 I
패스 II
HS 제어 출력 및
패치 출력
시스템
스크래치
356
테셀레이터
337
HS
테셀레이션
인자들
PC
336
도메인
로케이션들
DS/
GS
332
위치 마커들,
컷 스트림들
스트림 ID
S.O.
VPC
334
GRAS
340
포스트-Z
픽셀들
PS
346
프리-Z
픽셀들
프레임 버퍼
358
RB
342

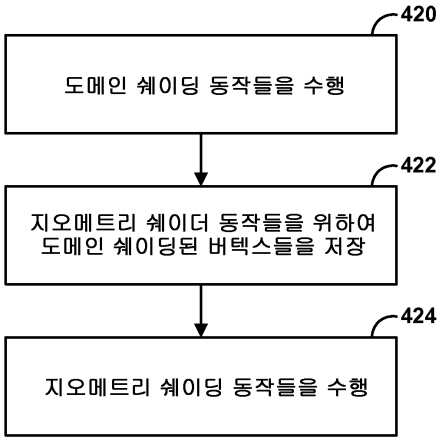
도면15


380
테셀레이션 수행?
아니오
382
단일 패스로 렌더링
예
384
로컬 메모리의 사이즈 결정
386
드로우 콜을 복수의
서브-드로우 콜들로 분할
388
서브-드로우 콜에 대하여
패스 I 동작들을 수행
390
서브-드로우 콜에 대하여
패스 II 동작들을 수행
392
픽셀 쉐이딩 동작들을 수행
394
드로우 콜의
최종 서브-드로우 콜?
아니오
예
396
드로우 콜에 대한
렌더링된 데이터 출력

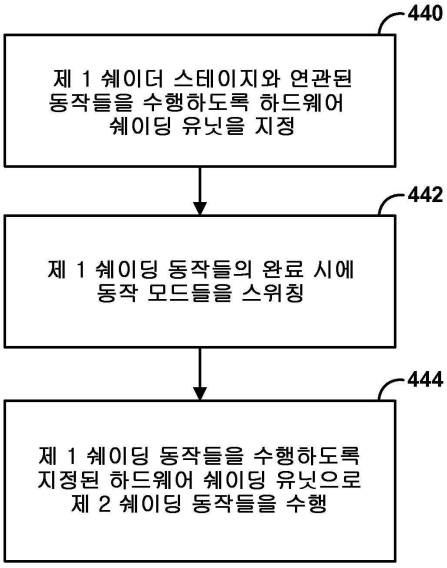
도면16


400
버텍스 쉐이딩 동작들을 수행
402
헐 쉐이더 동작들을 위하여
쉐이딩된 버텍스들을 저장
404
헐 쉐이딩 동작들을 수행

도면17


420
도메인 쉐이딩 동작들을 수행
422
지오메트리 쉐이더 동작들을 위하여 도메인 쉐이딩된 버텍스들을 저장
424
지오메트리 쉐이딩 동작들을 수행

도면18


440
제 1 쉐이더 스테이지와 연관된 동작들을 수행하도록 하드웨어 쉐이딩 유닛을 지정
442
제 1 쉐이딩 동작들의 완료 시에 동작 모드들을 스위칭
444
제 1 쉐이딩 동작들을 수행하도록 지정된 하드웨어 쉐이딩 유닛으로 제 2 쉐이딩 동작들을 수행