(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2007-0121728
(43) 공개일자 2007년12월27일

(51) Int. Cl.
G06F 17/00 (2006.01)
(21) 출원번호 10-2007-7023193
(22) 출원일자 2007년10월10일
심사청구일자 없음
번역문제출일자 2007년10월10일
(86) 국제출원번호 PCT/US2006/009463
국제출원일자 2006년03월16일
(87) 국제공개번호 WO 2006/115604
국제공개일자 2006년11월02일
(30) 우선권주장
11/110,295 2005년04월20일 미국(US)

(71) 출원인
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
라만, 샤피크 우르
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
그리고로비치, 알렉산드레, 브이.
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
지, 시키앙, 다니엘
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
(74) 대리인
양영준, 백만기

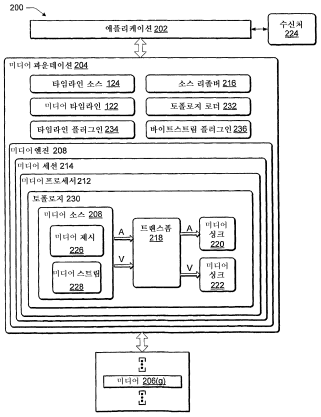
전체 청구항 수 : 총 20 항

(54) 미디어 타임라인 정렬 방법

(57) 요 약

미디어 타임라인 정렬이 설명된다. 일 구현에서, 방법은 애플리케이션 프로그래밍 인터페이스에서 미디어 타임라인을 수신하는 단계를 포함하는데, 미디어 타임라인은 복수의 타임라인 객체를 포함한다. 타임라인 객체들 중 하나 이상은 복수의 미디어의 각각을 참조한다. 순차적 렌더링을 위해 미디어 타임라인으로부터 복수의 세그먼트가 생성되어, 각 세그먼트는 세그먼트의 지속 기간 동안 렌더링되는 타임라인 객체들의 특정 세트를 참조한다.

대표도 - 도2



특허청구의 범위

청구항 1

애플리케이션 프로그래밍 인터페이스에서 미디어 타임라인을 수신하는 단계-상기 미디어 타임라인은 복수의 타임라인 객체를 포함하고, 하나 이상의 상기 타임라인 객체는 복수의 미디어의 각각을 참조함-; 및

렌더링을 위해 상기 미디어 타임라인으로부터 복수의 세그먼트를 하나씩 생성하는 단계

를 포함하고,

각각의 상기 세그먼트는 상기 세그먼트의 지속 기간 동안 렌더링하는 특정 세트의 상기 타임라인 객체들을 참조하는 방법.

청구항 2

제1항에 있어서, 적어도 하나의 상기 타임라인 객체는 또 하나의 상기 타임라인 객체에 의해 참조되는 미디어에 적용할 효과를 지정하는 방법.

청구항 3

제1항에 있어서, 적어도 하나의 상기 타임라인 객체는 상기 미디어 타임라인이 렌더링되는 동안에 상기 미디어 타임라인에 변경이 이루어질 수 있도록 또 하나의 상기 노드에 대한 이벤트의 통신을 위해 구성되는 방법.

청구항 4

제1항에 있어서, 적어도 하나의 상기 타임라인 객체는 또 하나의 상기 타임라인 객체에 의해 참조되는 상기 미디어와 다른 포맷을 가진 상기 미디어를 참조하는 방법.

청구항 5

제1항에 있어서,

상기 생성 단계는

각각의 상기 노드에 포함된 메타데이터를 상기 노드의 시작 시간 및 중지 시간에 대해 검사하는 단계;

각각의 상기 시작 시간에 대한 입력 및 각각의 상기 중지 시간에 대한 입력을 어레이에 추가하는 단계;

각각의 상기 입력을 서로에 대해 시간순으로 정렬하는 단계; 및

정렬된 어레이를 워킹(walking)하여 각각의 상기 세그먼트를 결정하는 단계

를 포함하는 방법.

청구항 6

제5항에 있어서,

상기 생성 단계는

적어도 하나의 상기 노드에 포함된 메타데이터가 중지 시간을 지정하지 않는 것으로 결정하는 단계; 및

적어도 하나의 상기 노드가 완료시까지 렌더링되어야 함을 나타내는 입력을 어레이에 추가하는 단계

를 포함하는 방법.

청구항 7

제6항에 있어서,

상기 생성 단계는

특정 상기 노드에 대한 시작 또는 중지 시간이 적어도 하나의 상기 노드의 시작 또는 중지 시간에 종속하는 것으로 결정하는 단계; 및

적어도 하나의 상기 노드에 대응하는 미디어 소스를 취득하여 적어도 하나의 상기 노드의 시작 또는 중지 시간을 확인하는 단계

를 포함하는 방법.

청구항 8

제7항에 있어서,

상기 생성 단계는

확인된 시작 또는 중지 시간을 상기 어레이에 추가하는 단계; 및

상기 어레이를 재정렬하는 단계

를 포함하는 방법.

청구항 9

제1항에 있어서, 상기 워킹은 시작 시간을 가진 입력에 의해 참조되는 각각의 타임라인 객체를 수집하고, 중지 시간을 가진 입력에 의해 참조되는 각각의 타임라인 객체를 폐기함으로써 특정 상기 세그먼트에 의해 정의되는 특정 시간까지 정렬된 어레이의 시간 순서를 따름으로써, 특정 상기 세그먼트 내에 상기 복수의 노드 중 어느 노드가 포함되는지를 결정하는 단계를 포함하는 방법.

청구항 10

제1항에 있어서,

제1 상기 세그먼트에 의해 참조되는 각각의 상기 타임라인 객체를 로딩하는 단계; 및

제1 상기 세그먼트가 렌더링되는 동안, 제2 상기 세그먼트에 의해 참조되는 각각의 상기 타임라인 객체를 로딩하는 단계

를 더 포함하는 방법.

청구항 11

미디어 타임라인에 대응하는 복수의 입력을 어레이 내에 시간순으로 정렬하는 단계 - 상기 미디어 타임라인에 포함된 각각의 타임라인 객체는 시작 시간에 대응하는 상기 입력 및 중지 시간에 대응하는 상기 입력을 구비하고, 하나 이상의 상기 타임라인 객체는 렌더링을 위한 미디어를 참조함-; 및

정렬된 어레이를 워킹하여 렌더링을 위한 복수의 세그먼트를 형성하는 단계

를 포함하고,

각각의 상기 세그먼트는 상기 세그먼트 동안 렌더링될 미디어들 중 하나 이상을 참조하는 방법.

청구항 12

제11항에 있어서, 적어도 하나의 상기 타임라인 객체는 또 하나의 상기 타임라인 객체에 의해 참조되는 미디어에 적용할 효과를 지정하는 방법.

청구항 13

제11항에 있어서, 적어도 하나의 상기 타임라인 객체는 또 하나의 상기 타임라인 객체에 의해 참조되는 상기 미디어와 다른 포맷을 가진 상기 미디어를 참조하는 방법.

청구항 14

제11항에 있어서, 상기 워킹은 시작 시간을 가진 입력에 의해 참조되는 각각의 타임라인 객체를 수집하고, 중지 시간을 가진 입력에 의해 참조되는 각각의 타임라인 객체를 폐기함으로써 특정 상기 세그먼트에 의해 정의되는 특정 시간까지 정렬된 어레이의 시간 순서를 따름으로써, 특정 상기 세그먼트 내에 상기 복수의 타임라인 객체 중 어느 객체가 포함되는지를 결정하는 단계를 포함하는 방법.

청구항 15

애플리케이션 프로그래밍 인터페이스를 통해, 특정 시점에 복수의 타임라인 객체를 가진 미디어 타임라인을 렌더링하기 위한 요청을 수신하는 단계;

시작 시간을 가진 정렬된 어레이 내의 입력에 의해 참조되는 각각의 상기 타임라인 객체를 수집하고, 중지 시간을 가진 상기 정렬된 어레이 내의 입력에 의해 참조되는 각각의 상기 타임라인 객체를 폐기함으로써 특정 시점에 대응하는 세그먼트에 도달할 때까지 상기 정렬된 어레이를 워킹하는 단계; 및

수집된 상기 타임라인 객체를 렌더링하는 단계

를 포함하는 방법.

청구항 16

제15항에 있어서, 상기 미디어 타임라인의 대응 타임라인 객체에 포함된 메타데이터를 검사함으로써, 상기 중지 시간을 가진 상기 정렬된 어레이 내의 적어도 하나의 상기 입력이 취득되는 방법.

청구항 17

제15항에 있어서, 대응하는 상기 타임라인 객체에 의해 참조되는 미디어 소스를 취득함으로써 상기 중지 시간이 적어도 하나의 상기 입력으로부터 취득되는 방법.

청구항 18

제15항에 있어서, 적어도 하나의 상기 타임라인 객체는 또 하나의 상기 타임라인 객체에 의해 참조되는 미디어에 적용할 효과를 지정하는 방법.

청구항 19

제15항에 있어서, 적어도 하나의 상기 타임라인 객체는 또 하나의 상기 타임라인 객체에 의해 참조되는 미디어와 다른 포맷을 가진 미디어를 참조하는 방법.

청구항 20

제15항에 있어서,

제1 상기 세그먼트에 의해 참조되는 각각의 상기 타임라인 객체를 로딩하는 단계; 및

제1 상기 세그먼트의 렌더링이 진행되는 동안, 제2 상기 세그먼트에 의해 참조되는 각각의 상기 타임라인 객체를 로딩하는 단계

를 더 포함하는 방법.

명 세 서

기 술 분 야

<1> 본 발명은 일반적으로 미디어에 관한 것으로서, 구체적으로는 미디어 타임라인 정렬에 관한 것이다.

배 경 기 술

<2> 데스크탑 PC, 셋톱 박스, 개인 휴대 단말기(PDA) 등과 같은 컴퓨터의 사용자는 계속 증가하는 다양한 소스로부터 계속 증가하는 양의 미디어에 액세스한다. 예를 들어, 사용자는 홈 비디오, 노래, 슬라이드쇼 제시(presentation) 등과 같은 미디어의 출력을 제공하기 위해 복수의 애플리케이션을 실행하는 데스크탑 PC와 상호작용할 수 있다. 사용자는 또한 셋톱 박스를 이용하여, 방송 네트워크를 통해 셋톱 박스로 방송되는 통상의 텔레비젼 프로그래밍을 수신할 수 있다. 또한, 셋톱 박스는 개인용 비디오 레코더(PVR)로서 구성될 수 있으며, 따라서 사용자는 후일의 재생을 위해 방송 콘텐츠를 셋톱 박스의 메모리에 저장할 수 있다. 또한, 사용자는 복수의 애플리케이션을 실행하는 무선 전화와 상호작용하여 이메일을 판독 및 전송하고, 비디오 게임을 행하고, 스프레드시트를 보는 것 등을 행할 수 있다.

<3> 미디어를 제공하고 미디어와 상호작용하는 데 사용될 수 있는 다양한 미디어 소스 및 다양한 컴퓨터로 인하여, 통상의 애플리케이션 및 컴퓨터는 종종 각각의 특정 타입의 미디어에 구체적으로 어드레스하도록 구성된다. 예를 들어, 비디오 게임 콘솔 상에서 비디오 게임을 출력하도록 실행되는 애플리케이션은 일반적으로 애플리케이션의 출력을 텔레비젼에 제공하도록 구성되며, 다른 컴퓨터 및 다른 장치에 의해 이용될 수 있는 출력을 제공하도록 구성되지는 않는다. 따라서, 컴퓨터 및/또는 애플리케이션과 같은 상이한 미디어 소스에 의해 제공되는 콘텐츠의 제시는 시간 및 장치 집약적일 수 있는 다수의 애플리케이션 및 장치를 필요로 할 수 있다. 또한, 동일 컴퓨터 상에서 실행되는 다수의 애플리케이션은 각각의 애플리케이션에 의해 제공되는 특정 타입의 미디어에 구체적으로 어드레스하도록 구성될 수 있다. 예를 들어, 제1 오디오 재생 애플리케이션은 노래로서 구성된 미디어를 출력하도록 구성될 수 있다. 그러나, 제2 오디오 재생 애플리케이션은 오디오-구술 포맷과 같이 제1 오디오 재생 애플리케이션과 호환성이 없는 오디오 포맷의 레코딩을 기록하고 재생하도록 구성될 수 있다. 따라서, 동일 컴퓨터 및 동일 타입의 미디어, 예를 들어 오디오 상에서의 실행을 위해 구성된 애플리케이션들 조차도 서로 호환되지 않는 미디어를 제공할 수 있다.

<4> 타임라인은 사용자가 미디어의 제시를 정의하는 방법을 제공한다. 예를 들어, 미디어 재생기는 일반적으로 "재생 리스트"로서 지칭되는 노래들의 리스트를 재생할 수 있다. 그러나, 통상의 타임라인은 미디어를 제공하고 미디어와 상호작용하는 데 사용될 수 있는 다양한 미디어 소스 및 다양한 컴퓨터 구성에 의해 제한된다. 예를 들어, 상이한 애플리케이션들로부터 미디어의 출력을 원할 때, 각 타입의 미디어는 상이한 애플리케이션들의 사용을 필요로 하는 상이한 타임라인을 요구할 수 있다. 이것은 컴퓨터의 하드웨어 및 소프트웨어 자원들의 비효율적인 이용으로 귀착될 수 있다. 또한, 상이한 타임라인은 개별 타임라인들로부터의 미디어를 동시에 출력하는 것과 같이 각각의 타임라인으로부터의 출력들을 조정하는 것을 어렵게 할 수 있다.

<5> 또한, 대형 타임라인들의 실행은 컴퓨터의 소프트웨어 및/또는 하드웨어 자원의 비효율적인 이용으로 귀착될 수 있다. 예를 들어, 대형 노래 재생 리스트를 로딩할 때, 재생 리스트 내의 각 노래가 로딩된다. 따라서, 재생 리스트의 초기 로딩은 상당한 양의 하드웨어 및/또는 소프트웨어 자원을 소비하여, 재생 리스트 내의 노래들의 로딩 및 재생에 있어서의 지연을 초래할 수 있다.

<6> 따라서, 타임라인을 렌더링하는 방법을 결정하기 위해 타임라인들을 정렬하기 위한 향상된 타임라인 및 기술이 계속 요구되고 있다.

<7> <발명의 요약>

<8> 미디어 타임라인 정렬이 설명된다. 미디어 타임라인은 사용자가 미디어에 기초하여 제시를 정의하기 위한 기술을 제공한다. 미디어 타임라인은 미디어들의 그룹핑 및/또는 조합을 표현하고, 미디어 타임라인에 의해 기술되는 미디어의 제시를 제공하기 위해 타임라인 소스에 의해 사용되는 구성 메타데이터를 제공하는 데 이용될 수 있다. 미디어 타임라인은 다양한 문제를 해결하기 위해 다양한 방식으로 구성될 수 있다.

<9> 또한, 미디어 타임라인을 다수의 세그먼트로 분할하도록 실행될 수 있는 타임라인 정렬기가 설명된다. 일 구현에서, 방법은 애플리케이션 프로그래밍 인터페이스에서 미디어 타임라인을 수신하는 단계를 포함하는데, 미디어 타임라인은 복수의 타임라인 객체를 포함한다. 타임라인 객체들 중 하나 이상은 복수의 미디어의 각각을 참조한다. 순차적 렌더링을 위해 미디어 타임라인으로부터 복수의 세그먼트가 생성되어, 각 세그먼트는 세그먼트의 지속 기간 동안 렌더링되는 타임라인 객체들의 특정 세트를 참조한다.

발명의 상세한 설명

<39> **개요**

<40> 미디어 타임라인 정렬이 설명된다. 미디어 타임라인은 사용자가 기존의 미디어(예를 들어, 비디오, 노래, 문서 등과 같은 저장된 미디어) 및/또는 스트리밍 오디오 및/또는 비디오와 같이 미디어 소스로부터 실시간으로 출력되는 미디어와 같은 미디어에 기초하여 제시를 정의하는 기술을 제공한다. 미디어 타임라인은 미디어들의 그룹핑 및/또는 조합을 표현하고, 미디어 타임라인에 의해 기술되는 미디어를 포함하는 최종 제시를 제공하기 위해 미디어 타임라인을 실행, 예를 들어 렌더링하는 타임라인 소스에 의해 이용되는 구성 메타데이터를 제공하는 데 이용될 수 있다.

<41> 일 구현에서, 미디어 타임라인은 미디어 타임라인의 동적 생성 및/또는 로딩을 위해 구성된다. 전술한 바와 같이, 대형 미디어 타임라인, 예를 들어 상당한 수의 노드 및/또는 상당한 양의 데이터를 갖는 미디어 타임라인은 미디어 타임라인이 로딩될 때 비효율적일 수 있다. 예를 들어, 미디어 타임라인을 로딩하는 컴퓨터는 미디어

타임라인의 모든 노드가 그 시간에 출력되지 않는 경우에도 상당한 처리 및 메모리 자원을 사용할 수 있다. 따라서, 미디어 타임라인은 동적 생성 및/또는 로딩을 위해 구성되어, 미디어 타임라인이 렌더링될 때, 미디어 타임라인의 노드들의 로딩 및/또는 생성을 지능적으로 지연시킬 수 있다. 미디어 타임라인을 동적 로딩 및 생성을 위해 구성함으로써, 컴퓨터의 시동 동안, 그리고 일반적으로 미디어 타임라인을 로딩하기 위해 컴퓨터의 하드웨어 및/또는 소프트웨어 자원들이 효율적으로 사용될 수 있다.

<42> 예를 들어, 타임라인 객체 모델에 의해 기술되는 미디어 타임라인을 미디어 처리 파이프라인을 이용하여 독립적으로 렌더링될 수 있는 세그먼트들의 시퀀스로 변환하기 위해 타임라인 정렬기가 실행될 수 있다. 일 구현에서, 타임라인 정렬기는 "온더플라이(on the fly)" 방식으로 세그먼트들을 결정할 수 있으며, 따라서 타임라인 정렬기는 "업프론트(up front)" 방식으로 전체 제시를 정의할 필요가 없게 된다. 예를 들어, 각각의 세그먼트는 미디어 처리 파이프라인에서 독립 개체로서 렌더링될 수 있어, 전체 제시는 단일 엔티티로서 업프론트 방식으로 정의되지 않는다. 따라서, 미디어 타임라인의 소정 부분이 갱신 등과 같이 변경되어야 하는 경우에 미디어 타임라인을 렌더링하는 데 사용되는 미디어 파이프라인을 "분해(tear down)"하지 않고도 미디어 타임라인에 대한 변경을 행할 수 있다. 미디어 타임라인의 노드들의 동적 생성 및/또는 로딩을 제공하는 미디어 타임라인들에 대한 추가 설명은 도 12-27과 관련하여 발견될 수 있다.

<43> 아래의 설명에서, 미디어 타임라인 정렬 기술을 이용할 수 있는 예시적인 환경이 먼저 설명된다. 이어서, 예시적인 환경은 물론, 다른 환경에서도 이용할 수 있는 예시적인 프로시저들이 설명된다.

<44> **예시적인 환경**

<45> 도 1은 컴퓨터(102)가 복수의 미디어에 대한 액세스를 제공하는 예시적인 구현의 환경을 나타낸다. 도시된 바와 같이, 컴퓨터(102)는 개인용 컴퓨터(PC)로 구성된다. 컴퓨터(102)는 또한 이동국, 오락 설비, 표시 장치에 통신 결합되는 셋톱 박스, 무선 전화, 비디오 게임 콘솔, 개인 휴대 단말기(PDA) 등과 같은 다양한 다른 구성을 가질 수 있다. 따라서, 컴퓨터(102)는 많은 메모리 및 프로세서 자원을 가진 충분한 자원의 장치(예를 들어, PC, 하드 디스크를 구비한 텔레비젼 레코더)에서 제한된 메모리 및/또는 처리 자원을 가진 적은 자원의 장치(예를 들어, 통상의 셋톱 박스)까지의 범위일 수 있다. 컴퓨터(102)의 추가적인 구현이 도 28과 관련하여 설명된다.

<46> 컴퓨터(102)는 다양한 미디어 소스로부터 다양한 미디어를 취득할 수 있다. 예를 들어, 컴퓨터(102)는 복수의 미디어 104(1), ..., 104(k),..., 104(K)를 국지적으로 저장할 수 있다. 복수의 미디어 104(1)-104(K)는 WMV, WMA, MPEG1, MPEG2, MP3 등과 같은 다양한 포맷을 가진 다양한 오디오 및 비디오 콘텐츠를 포함할 수 있다. 또한, 미디어 104(1)-104(K)는 입력 장치, 애플리케이션의 실행 등의 다양한 소스로부터 취득될 수 있다.

<47> 컴퓨터(102)는 예를 들어 복수의 애플리케이션 106(1),..., 106(n), ..., 106(N)을 포함할 수 있다. 복수의 애플리케이션 106(1)-106(N) 중 하나 이상은 문서, 스프레드시트, 비디오, 오디오 등과 같은 미디어를 제공하도록 실행될 수 있다. 또한, 복수의 애플리케이션 106(1)-106(N) 중 하나 이상은 미디어 104(1)-104(K)의 인코딩, 편집 및/또는 재생과 같은 미디어 상호작용을 제공하도록 구성될 수 있다.

<48> 컴퓨터(102)는 또한 복수의 입력 장치 108(1),..., 108(m),...108(M)를 포함할 수 있다. 복수의 입력 장치 108(1)-108(M) 중 하나 이상은 컴퓨터(102)에 입력하기 위한 미디어를 제공하도록 구성될 수 있다. 입력 장치 108(1)는 예를 들어 사용자의 음성, 콘서트에서의 노래 등과 같은 오디오 데이터의 입력을 제공하도록 구성되는 마이크로폰으로 예시된다. 복수의 입력 장치 108(1)-108(M)는 또한 복수의 애플리케이션 106(1)-106(N)의 실행을 제어하는 입력을 제공하기 위한 사용자에 의한 상호작용을 위해 구성될 수 있다. 예를 들어, 입력 장치 108(1)는 복수의 애플리케이션 106(1)--106(N) 중 특정 애플리케이션의 실행을 개시하고, 복수의 애플리케이션 106(1)-106(N)의 실행을 제어하는 등과 같은 사용자로부터의 음성 커맨드를 입력하는 데 사용될 수 있다. 다른 예에서, 입력 장치 108(m)는 컴퓨터(102)의 셋팅을 조정하는 것과 같은 컴퓨터(102)를 제어하기 위한 입력을 제공하도록 구성되는 키보드로 예시된다.

<49> 또한, 컴퓨터(102)는 복수의 출력 장치 110(1), ..., 110(j),..., 110(J)를 포함할 수 있다. 출력 장치들 110(1)-110(J)은 사용자에게 출력하기 위한 미디어 104(1)-104(K)를 렌더링하도록 구성될 수 있다. 예를 들어, 출력 장치 110(1)는 오디오 데이터를 렌더링하기 위한 스피커로 예시된다. 출력 장치 110(j)는 오디오 및/또는 비디오 데이터를 렌더링하도록 구성되는 텔레비젼과 같은 표시 장치로 예시된다. 따라서, 복수의 미디어 104(1)-104(K) 중 하나 이상은 입력 장치들 108(1)-108(M)에 의해 제공되고, 컴퓨터(102)에 의해 국지적으로 저장될 수 있다. 복수의 입력 및 출력 장치들 108(1)-108(M), 110(1)-110(J)이 별개로 도시되어 있지만, 입력 및

출력 장치들 108(1)-108(M), 110(1)-110(J) 중 하나 이상은 입력용 버튼, 표시 장치 및 스피커를 구비한 텔레비전과 같은 단일 장치로 결합될 수 있다.

<50> 컴퓨터(102)는 또한 네트워크(112)를 통해 통신을 행하여 네트워크(112)를 통해 원격으로 이용 가능한 미디어를 취득하도록 구성될 수 있다. 네트워크(112)는 인터넷으로 예시되며, 인트라넷, 유선 또는 무선 전화 네트워크, 방송 네트워크 및 다른 광역 네트워크와 같은 다양한 다른 네트워크를 포함할 수 있다. 원격 컴퓨터(114)가 네트워크(112)에 통신 결합되어, 컴퓨터(102)에 미디어를 제공할 수 있다. 예를 들어, 원격 컴퓨터(114)는 하나 이상의 애플리케이션, 및 홈 무비와 같은 미디어를 제공하는 비디오 카메라(116)를 포함할 수 있다. 원격 컴퓨터(114)는 또한 도시된 바의 표시 장치(118)와 같이 미디어를 출력하기 위한 출력 장치를 포함할 수 있다. 네트워크(112)를 통해 컴퓨터(102)에 의해 원격 컴퓨터(114)로부터 취득되는 미디어는 미디어 104(1)-104(K)와 함께 국지적으로 저장될 수 있다. 즉, 미디어 104(1)-104(K)는 네트워크(112)를 통해 원격 컴퓨터(114)로부터 취득된 미디어의 국지 저장 사본을 포함할 수 있다.

<51> 따라서, 컴퓨터(102)는 국지적으로(예를 들어, 복수의 애플리케이션 106(1)-106(N)의 실행 및/또는 복수의 입력 장치 108(1)-108(M)의 사용을 통해), 그리고 원격 컴퓨터(114)로부터 원격적으로(예를 들어, 애플리케이션의 실행 및/또는 입력 장치의 사용을 통해) 제공될 수 있는 복수의 미디어 104(1)-104(K)를 취득하고 저장할 수 있다. 복수의 미디어 104(1)-104(K)가 컴퓨터(102)에 저장되는 것으로 설명되었지만, 미디어 104(1)-104(K)는 실시간으로 제공될 수도 있다. 예를 들어, 오디오 데이터를 저장하지 않고, 마이크로폰으로 예시되는 입력 장치 108(1)로부터 오디오 데이터가 스트리밍될 수 있다.

<52> 컴퓨터(102)는, 컴퓨터(102) 상에서 실행될 때 미디어 타임라인(122)을 생성하는 타임라인 생성기(120)를 포함한다. 예를 들어, 타임라인 생성기(120)는 사용자에 의해 사용자 인터페이스를 통해서와 같이 미디어 타임라인(122)을 생성하는 데 사용될 수 있는 하나 이상의 소프트웨어 컴포넌트를 노출하는 애플리케이션으로서 구성될 수 있다. 전술한 바와 같이, 미디어 타임라인(122)은 사용자가 복수의 미디어 소스로부터 저장 및/또는 실시간 미디어의 제시를 정의하는 기술을 제공한다. 예를 들어, 미디어 타임라인(122)은 입력 장치들 108(1)-108(M), 애플리케이션들 106(1)-106(N) 및/또는 원격 컴퓨터(114)로부터 취득한 미디어들의 집합을 기술할 수 있다. 사용자는 입력 장치들 108(1)-108(M) 중 하나 이상을 이용하여 타임라인 생성기(120)와 상호작용함으로써 미디어들 104(1)-104(K)의 그룹핑 및/또는 조합을 정의할 수 있다. 사용자는 또한 미디어들 104(1)-104(K)의 제시를 위한 순서 및 효과를 정의할 수 있다. 이어서, 타임라인 소스(124)가 컴퓨터(102) 상에 실행되어 미디어 타임라인(122)을 렌더링할 수 있다. 미디어 타임라인(122)은 렌더링 시에 복수의 출력 장치 110(1)-110(J) 중 하나 이상에 의한 렌더링을 위해 미디어들 104(1)-104(K)의 표현된 그룹핑 및/또는 조합을 제공한다. 또한, 타임라인 생성기(120)는 아래의 구현에서 보다 상세히 설명되는 바와 같이 미디어 타임라인(122)을 프로그램 방식으로 생성할 수도 있다.

<53> 표현된 그룹핑을 결정하기 위하여, 컴퓨터(102)는 또한 타임라인 정렬기(126)를 포함한다. 타임라인 정렬기(126)는 미디어 타임라인을 복수의 세그먼트로 분할하도록 실행될 수 있는데, 각각의 세그먼트는 변경되지 않는 당해 세그먼트 동안 렌더링되는 한 세트의 타임라인 객체들을 포함한다. 타임라인 정렬기(126)의 실행에 대한 추가 설명은 도 12-27과 관련하여 발견될 수 있다.

<54> 도 2는 소프트웨어로 구현되는 시스템(200)이 복수의 미디어 206(g)("g"는 1에서 "G"까지의 임의의 수일 수 있다)의 제시를 제어하기 위해 미디어 파운데이션(204)과 상호작용하는 애플리케이션(202)을 포함하는 예시적인 구현에서의 시스템(200)의 하이 레벨 블록도이다. 미디어 파운데이션(204)은 미디어 206(g)의 재생을 제공하기 위해 운영 체제의 일부로서 포함될 수 있으며, 따라서 운영 체제와 상호작용하는 애플리케이션들은 미디어 포맷의 특정 상세를 알지 않고도 미디어 206(g)의 재생을 제어할 수 있다. 미디어 206(g)는 애플리케이션 106(1)-106(N)의 실행, 입력 장치 108(1)-108(M), 출력 장치 110(1)-110(J)의 사용 등을 통해 도 1의 미디어 104(1)-104(K)와 같은 다양한 소스로부터 제공될 수 있다.

<55> 도 1의 애플리케이션 106(1)-106(N)과 같거나 다를 수 있는 애플리케이션(202)은 미디어 104(1)-104(K)를 제어하기 위해 미디어 엔진(208)과 상호작용한다. 적어도 일부 실시예에서, 미디어 엔진(208)은 제시에 관여하기를 원하는 애플리케이션(202)의 중심 초점으로 기능한다. 본 명세서에서 사용되는 제시는 미디어의 처리를 지칭하거나 기술한다. 도시되고 설명되는 실시예에서, 제시는 미디어 엔진(208)이 조작을 수행하는 데이터의 포맷을 기술하는 데 사용된다. 따라서, 제시는 오디오 및 이에 수반되는 비디오가 데스크탑 PC와 연관될 수 있는 표시 장치로서 예시된 도 1의 출력 장치 110(j)와 같은 표시 장치 상에 렌더링된 윈도우 내에서 사용자에게 제시되는 멀티미디어 제시와 같이 시각적으로 그리고/또는 청각적으로 미디어를 제시할 수 있다. 제시는 또한 미디어 콘

텐츠를 디스크 파일과 같은 컴퓨터 판독 가능 미디어에 기록할 수 있다. 따라서, 제시는 멀티미디어 콘텐츠가 컴퓨터 상에서 렌더링되는 시나리오로 한정되지 않는다. 몇몇 실시예에서, 디코딩, 인코딩 및 다양한 트랜스폼(전이, 효과 등)과 같은 조작들은 제시의 결과로서 발생할 수 있다.

<56> 일 실시예에서, 미디어 파운데이션(204)은 미디어 206(g)와 상호작용하기 위해 애플리케이션(202)에 의해 호출될 수 있는 하나 이상의 애플리케이션 프로그램 인터페이스를 노출한다. 예를 들어, 미디어 파운데이션(204)은 도 1의 컴퓨터(102) 상에서 실행되는 소프트웨어의 "기반구조" 레벨에 존재하는 것으로 간주할 수 있다. 즉, 미디어 파운데이션(204)은 미디어 206(g)와 상호작용하기 위해 애플리케이션(202)에 의해 사용되는 소프트웨어 계층이다. 미디어 파운데이션(204)은 출력, 렌더링, 저장 등과 같은 미디어 206(g)의 다양한 양태를 제어하는데 사용될 수 있다. 따라서, 미디어 파운데이션(204)은 각각의 애플리케이션(202)이 시스템(200)에서 사용될 수 있는 각 타입의 미디어 206(g)에 대한 개별 코드를 구현할 필요가 없도록 사용될 수 있다. 이러한 방식으로, 미디어 파운데이션(204)은 미디어 고유 태스크를 수행하기 위해 한 세트의 재사용 가능 소프트웨어 컴포넌트를 제공한다.

<57> 미디어 파운데이션(204)은 미디어 타임라인(122), 타임라인 소스(124), 미디어 소스(210), 미디어 프로세서(212), 미디어 세션(214), 미디어 엔진(208), 소스 리졸버(216), 하나 이상의 트랜스폼(218), 하나 이상의 미디어 싱크(220,222) 등을 포함하는 여러 컴포넌트를 이용할 수 있다. 도시되고 설명되는 다양한 실시예의 하나의 이점은 시스템(200)은 다양한 상이한 종류의 컴포넌트가 본 명세서에서 설명되는 시스템들과 함께 이용될 수 있다는 의미에서 플러그 가능한 모델이라는 점이다. 또한, 후술하는 수신처(224)가 시스템(200)의 일부로서 포함된다. 그러나, 적어도 하나의 실시예에서, 수신처(224)는 어디서 제시가 제시되는지(예를 들어, 윈도우, 디스크 파일 등) 그리고 제시에 무엇이 발생하는지를 정의하는 객체이다. 즉, 수신처는 데이터가 흘러 들어가는 미디어 싱크들(220, 222) 중 하나 이상에 대응할 수 있다.

<58> 미디어 타임라인(122)은 사용자가 타임라인 소스(124)에 의해 렌더링되는 미디어에 기초하여 제시를 정의하는 방법을 제공하는 타임라인 객체 모델을 이용한다. 미디어 타임라인(122)은 미디어 파일들의 순차적 리스트에서 보다 복잡한 형식까지의 범위일 수 있다. 예를 들어, 미디어 타임라인(122)은 미디어, 효과 등 간의 전이를 포함하는 미디어 재생 경험을 표현하기 위해 SMIL 및 AAF와 같은 파일 구조를 이용할 수 있다. 예를 들어, 애플리케이션(202)은 일반적으로 재생 리스트로서 지칭되는 노래 리스트를 재생할 수 있는 미디어 재생기로서 구성될 수 있다. 다른 예로서, 편집 시스템에서, 사용자는 하나의 비디오를 다른 비디오 상에 오버레이하고, 미디어를 클립하고, 미디어에 효과를 추가하는 등을 행할 수 있다. 이러한 미디어들의 그룹핑 또는 조합은 미디어 타임라인(122)을 이용하여 표현될 수 있다. 미디어 타임라인(122)에 대한 추가 설명은 도 3과 관련하여 발견된다.

<59> 미디어 소스(210)는 미디어의 제공자를 추상화하는 데 사용된다. 예를 들어, 미디어 소스(210)는 특정 소스로부터 특정 타입의 미디어를 판독하도록 구성될 수 있다. 예를 들어, 일 타입의 미디어 소스는 외부 세계로부터 비디오를 캡쳐할 수 있으며(예를 들어, 카메라), 다른 타입의 미디어 소스는 오디오를 캡쳐할 수 있다(예를 들어, 마이크로폰). 대안으로 또는 부가적으로, 미디어 소스(210)는 디스크로부터 압축된 데이터 스트림을 판독하고, 데이터 스트림을 그의 압축된 비디오 및 압축된 오디오 컴포넌트로 분리할 수 있다. 또 다른 미디어 소스(210)는 도 1의 네트워크(112)로부터 데이터를 취득할 수 있다. 따라서, 미디어 소스(210)는 미디어를 취득하기 위한 안정된 인터페이스를 제공하는 데 사용될 수 있다.

<60> 미디어 소스(210)는 하나 이상의 미디어 제시(226) 객체(미디어 제시)를 제공한다. 미디어 제시(226)는 관련된 미디어 스트림들의 세트의 기술을 추상화한다. 예를 들어, 미디어 제시(226)는 영화를 위한 한 쌍의 오디오 및 비디오 스트림을 제공할 수 있다. 또한, 미디어 제시(226)는 주어진 시점에서 미디어 소스(210)의 구성을 기술할 수 있다. 예를 들어, 미디어 제시(226)는 미디어 소스(210)의 이용 가능한 스트림들 및 이들의 미디어 타입, 예를 들어 오디오, 비디오, MPEG 등의 기술을 포함하는 미디어 소스(210)에 대한 정보를 포함할 수 있다.

<61> 미디어 소스(210)는 또한 애플리케이션(202)에 의해 액세스될 수 있는, 즉 애플리케이션(202)에 노출될 수 있는 미디어 소스(210)로부터의 단일 스트림을 표현할 수 있는 미디어 스트림(228) 객체(미디어 스트림)를 제공할 수 있다. 따라서, 미디어 스트림(228)은 애플리케이션(202)이 미디어 206(g)의 샘플을 검색하는 것을 가능하게 한다. 일 구현에서, 미디어 스트림(228)은 단일 미디어 타입을 제공하도록 구성된다. 미디어 소스는 둘 이상의 미디어 스트림을 제공할 수 있다. 예를 들어, wmv 파일은 동일 파일 내에 오디오 비디오 양자를 가질 수 있다. 따라서, 이 파일에 대한 미디어 소스는 두개의 스트림, 즉 오디오에 대한 스트림 및 비디오에 대한 스트림

을 제공할 것이다.

<62> 따라서, 미디어 파운데이션(204)에서, 미디어 소스(210)는 제시를 위한 샘플을 출력하는 소프트웨어 컴포넌트로서 정의된다. 타임라인 소스(124)는 미디어 타임라인(122)을 해석하지만, 동시에 미디어 소스(210)와 유사한 방식으로 동작할 수도 있다. 예를 들어, 타임라인 소스(210)는 미디어 파운데이션(204)의 다른 컴포넌트로부터 미디어 타임라인(122)에 의해 기술되는 미디어를 제공하기 위해 미디어 타임라인(122)을 렌더링하는 복잡함을 숨기는 데 사용될 수 있다.

<63> 미디어 프로세서(212)는 토폴로지(230)에서 데이터 흐름을 관리한다. 토폴로지(230)는 데이터가 주어진 제시를 위한 다양한 컴포넌트를 통해 어떻게 흐르는지를 정의한다. "풀" 토폴로지는 데이터가 상이한 컴포넌트들 간에 올바르게 포맷 변환되어 흐르도록 데이터를 조작하는 데 사용되는 컴포넌트들, 예를 들어 소프트웨어 모듈들 각각을 포함한다. 토폴로지가 생성될 때, 사용자는 토폴로지를 부분적으로 생성하는 것을 선택할 수 있다. 이러한 부분적 토폴로지는 자체적으로 최종 제시를 제공하기에 충분하지 않다. 따라서, 토폴로지 로더(232)라고 하는 컴포넌트가 부분적 토폴로지를 취하고, 부분적 토폴로지 내의 컴포넌트들 간에 적절한 데이터 변환 트랜스폼을 추가함으로써 부분적 토폴로지를 풀 토폴로지로 변환할 수 있다.

<64> 토폴로지(230)에서, 예를 들어, 데이터는 일반적으로 미디어 소스(210)에서 발생하고, 하나 이상의 트랜스폼(218)을 통과하여, 하나 이상의 미디어 싱크(220, 222)로 진행한다. 트랜스폼(218)은 일반적으로 제시에서 사용되는 임의의 적절한 데이터 처리 컴포넌트를 포함할 수 있다. 이러한 컴포넌트는 압축된 데이터를 풀고 그리고/또는 전문가가 이해하듯이 데이터에 효과를 부여하는 것과 같은 소정의 방식으로 데이터를 조작하는 컴포넌트를 포함할 수 있다. 예를 들어, 비디오 데이터에 대해, 트랜스폼은 휘도, 칼라 변환 및 크기 조정에 영향을 미치는 컴포넌트를 포함할 수 있다. 오디오 데이터에 대해, 트랜스폼은 반향 및 리샘플링에 영향을 미치는 컴포넌트를 포함할 수 있다. 또한, 디코딩 및 인코딩이 트랜스폼으로 간주될 수 있다.

<65> 미디어 싱크들(220, 222)은 일반적으로 특정 타입의 미디어 콘텐츠와 연관된다. 따라서, 오디오 콘텐츠는 오디오 렌더러와 같은 관련 오디오 싱크를 가질 수 있다. 마찬가지로, 비디오 콘텐츠는 비디오 렌더러와 같은 관련 비디오 싱크를 가질 수 있다. 추가적인 미디어 싱크들이 컴퓨터 판독 가능 미디어, 예를 들어 디스크 파일 등과 같은 것들에 데이터를 전송하고, 라디오 프로그램을 방송하는 것과 같이 네트워크를 통해 데이터를 스트리밍하는 등을 행할 수 있다.

<66> 미디어 세션(214)은 다수의 제시를 스케쥴링할 수 있는 컴포넌트이다. 따라서, 미디어 프로세서(212)는 주어진 제시를 드라이브하는 데 사용될 수 있고, 미디어 세션(214)은 다수의 제시를 스케쥴링하는 데 사용될 수 있다. 예를 들어, 미디어 세션(214)은 미디어 프로세서(212)에 의해 렌더링되는 토폴로지를 변경할 수 있다. 예를 들어, 미디어 세션(214)은 미디어 프로세서(212)에 의해 렌더링되는 제1 토폴로지에서 제2 토폴로지로 변경할 수 있으며, 따라서 각각의 토폴로지에 의해 기술되는 연속적인 제시들로부터의 샘플들의 렌더링들 사이에 갭이 존재하지 않게 된다. 따라서, 미디어 세션(214)은 미디어의 재생이 하나의 제시에서 다른 제시로 이동함에 따라 결함 없는 사용자 경험을 제공할 수 있다.

<67> 소스 리졸버(216) 컴포넌트는 URL 및/또는 바이트 스트림 객체로부터 미디어 소스(210)를 생성하는 데 사용될 수 있다. 소스 리졸버(216)는 지정된 자원에 의해 생성되는 데이터의 형식에 대한 사전 지식을 요구하지 않고 미디어 소스(210)를 생성하는 동기식 방법 및 비동기식 방법 양자를 제공할 수 있다.

<68> 적어도 일 실시예에서, 미디어 파운데이션(204)은 미디어 파운데이션(204)의 다양한 컴포넌트의 존재 및 그들 간의 상호작용의 특정 상세를 추상화하는 데 사용된다. 즉, 몇몇 실시예에서, 미디어 파운데이션(204) 내에 위치하는 것으로 보여지는 컴포넌트들은 프로그램적인 의미에서 애플리케이션(202)에 보이지 않는다. 이것은 미디어 파운데이션(204)이 소위 "블랙 박스" 세션을 실행하는 것을 허가한다. 예를 들어, 미디어 엔진(208)은 미디어(예를 들어, URL) 및 수신처(224)와 연관된 정보와 같은 소정의 데이터를 미디어 세션에 제공함으로써 미디어 세션(214)과 상호작용할 수 있으며, 애플리케이션(202)의 커맨드(예를 들어, 개방, 시작, 중지 등)를 미디어 세션(214)에 전송할 수 있다. 이어서, 미디어 세션(214)은 제공된 정보를 취하고, 적절한 수신처를 이용하여 적절한 제시를 생성한다.

<69> 미디어 파운데이션(204)은 또한 타임라인 플러그인(234)을 포함할 수 있다. 타임라인 플러그인(234)은 상이한 미디어 타임라인 파일 포맷들이 미디어 파운데이션(204)에 플러그인될 수 있도록 사용될 수 있다. 예를 들어, 바이트 스트림 플러그인(236)이 당해 포맷에 대해 작성되고 미디어 파운데이션(204)에 등록될 수 있다. 이어서, 소스 리졸버(216)는 그 타입의 파일이 열릴 때 바이트 스트림 플러그인(236)을 호출할 수 있다. 이어

서, 바이트 스트림 플러그인(236)은 파일을 분석하고, 파일 내에 기술된 제시를 표현하는 미디어 타임라인(122)을 생성하고, 그에 대한 타임라인 소스(124)를 생성할 수 있다. 일반적으로, 바이트 스트림 플러그인(236)은 로우 바이트 스트림(raw bytestream)을 판독하고 그에 대한 미디어 소스(210)의 생성을 담당할 수 있다. 일 구현에서, 미디어 파운데이션(204)의 잔여 컴포넌트들은 이 예에서 생성되는 미디어 소스가 타임라인 소스(124)라는 것을 알지 못한다. 따라서, 타임라인 소스(124)는 임의의 다른 미디어 소스(208)와 같이 처리된다. 일 구현에서, 미디어 타임라인(122)을 분석하고 타임라인 소스(124)를 생성할 수 있는 바이트 스트림 플러그인(236)은 도 23과 관련하여 보다 상세히 설명되는 타임라인 플러그인으로 지칭된다.

<70> 타임라인 플러그인(234)은 또한 인터페이스를 제공할 수 있으며, 따라서 애플리케이션(202)은 파일로부터 미디어 타임라인(122)을 로딩하고 파일에 미디어 타임라인(122)을 저장하는 것과 같이 타임라인 플러그인과 직접 상호작용할 수 있다. 예를 들어, 타임라인 플러그인(234)은 생성된 후, 바이트 스트림을 제공하기 위한 로드 기능을 개시하도록 호출될 수 있다. 이어서, 타임라인 플러그인(234)은 파일을 분석하고 루트 노드 및 임의의 추가 노드를 생성하여, 도 3과 관련하여 보다 상세히 설명되는 미디어 타임라인(122)을 생성할 수 있다. 타임라인 플러그인(234)은 또한 미디어 타임라인(122)을 상이한 포맷으로 유지하는 데 사용될 수 있다. 예를 들어, 애플리케이션(202)은 미디어 타임라인(122)을 프로그램 방식으로 생성할 수 있다. 즉, 애플리케이션은 도 1의 타임라인 생성기(120)로서 동작할 수 있다. 이어서, 애플리케이션(202)은 ASX 파일들에 대한 타임라인 플러그인을 생성하고, 미디어 타임라인(122)을 ASX 포맷으로 저장하도록 타임라인 플러그인에 요청할 수 있다. 다른 예에서, 사용자는 m3u 파일, 즉 다수의 MP3 파일을 지정하기 위한 재생 리스트 파일 포맷을 열고, 그로부터 미디어 타임라인(122)을 취득한 후, 미디어 타임라인(122)을 ASX 포맷으로 저장하도록 타임라인 플러그인에 요청할 수 있다. 이 예에서, 타임라인 플러그인은 타임라인 생성기(120)로서 동작한다. 따라서, 타임라인 파운데이션(204)은 애플리케이션(202)에 의해 사용하기 위한 애플리케이션 프로그래밍 인터페이스를 통해 미디어 기능을 제공하는 다수의 소프트웨어 컴포넌트를 노출시킬 수 있다.

<71> 도 3은 미디어 타임라인(300)이 제시를 위한 미디어의 출력을 기술하는 복수의 노드를 포함하는 트리로서 도시되는 예시적인 구현의 도면이다. 도 1 및 2의 미디어 타임라인(122)에 대응하거나 대응하지 않을 수 있는 미디어 타임라인(300)은 복수의 노드(302-312)를 포함하는 트리로서 구성된다. 복수의 노드(302-312) 각각은 노드 및/또는 이 특정 노드의 "자식"에 대한 다양한 속성 및 거동을 기술하는 각각의 메타데이터(314-322)를 포함한다. 예를 들어, 노드 304 및 노드 306은 각각 "부모" 및 "자식"으로서 배열된다. 노드 304는 이 노드(304)의 거동 및 속성을 기술하는 메타데이터(316)를 포함한다. 메타데이터(316)는 또한 노드들(306, 308)의 렌더링 순서와 같이 "자식" 노드들(306, 308) 각각을 기술할 수 있다.

<72> 일 구현에서, 미디어 타임라인(300)은 단독으로는 사용자 인터페이스(UI), 재생 또는 편집에 대한 결정을 행하도록 실행될 수 없다. 대신에, 미디어 타임라인(300) 상의 메타데이터(314-324)가 도 2의 타임라인 소스(124)와 같이 미디어 타임라인(300)을 렌더링하는 소프트웨어 및/또는 하드웨어 컴포넌트에 의해 해석된다. 또한, 미디어 타임라인(122)의 렌더링 동안 이용되는 애플리케이션들은 당해 특정 애플리케이션에 대한 관련 메타데이터를 취득할 수 있다. 예를 들어, 도 2의 애플리케이션(202)은 미디어 타임라인에서 참조되는 각 미디어가 개시되는 시간에만 관여하는 재생 엔진으로서 구성될 수 있다. 반면, 미디어 재생기와 같은 다른 애플리케이션은 각 노드 상에 메타데이터로서 저장되는 노래 제목을 표시하는 것에만 관심을 가질 수 있다. 이러한 방식으로, 메타데이터는 미디어의 출력을 이용하는 하나 이상의 애플리케이션에 의해 동시에 이용될 수 있다.

<73> 미디어 타임라인(300) 상에 위치하는 노드들(302-312)은 미디어 타임라인(300)의 기본 레이아웃을 기술한다. 이 레이아웃은 사용자 인터페이스에 타임라인 구조를 표시하는 데 이용되거나, 노드들의 렌더링을 순위화하기 위해 도 2의 타임라인 소스(124)에 의해 사용되거나, 기타 등등일 수 있다. 예를 들어, 원하는 레이아웃이 달성되도록 다양한 타입의 노드들(302-312)이 제공될 수 있다. 노드 타입은 루트 노드(302) 및 리프 노드들(308-312)과 같은 당해 노드의 자식들이 어떻게 해석되는지를 지시한다. 루트 노드(302)는 메타데이터 타임라인(300)을 렌더링하기 위한 시작점을 지정하며, 렌더링이 어떻게 개시되는지를 기술하는 메타데이터(314)를 포함한다.

<74> 도 3의 예시적인 구현에서, 미디어 타임라인(122)의 리프 노드들(308, 310, 312)은 미디어로 직접 맵핑된다. 예를 들어, 리프 노드들(308, 310, 312)은 리프 노드들(308-312) 각각이 나타내는 미디어를 어떻게 검색할지를 기술하는 각각의 메타데이터(320, 322, 324)를 가질 수 있다. 리프 노드는 오디오 및/또는 비디오 파일에 대한 경로를 지정하고, 미디어 타임라인(300)의 렌더링 동안 비디오 프레임을 프로그램 방식으로 생성하는 컴포넌트를 지시하는 등등을 행할 수 있다. 예를 들어, 리프 노드(308)는 마이크로폰으로서 구성되는 입력 장치 108(1)로 맵핑되는 포인터(326)를 가진 메타데이터(320)를 포함한다. 리프 노드(310)는 도 1의 컴퓨터(102) 상에

국지적으로 포함되는 저장 장치(332) 내의 미디어(330)의 어드레스로 맵핑되는 포인터(328)를 가진 메타데이터(322)를 포함한다. 리프 노드(312)는 네트워크(112) 상의 원격 컴퓨터(114)의 네트워크 어드레스로 맵핑되는 포인터(334)를 가진 메타데이터(324)를 포함한다. 원격 컴퓨터(114)는 네트워크(112)를 통해 도 1의 컴퓨터(102)에 미디어를 제공하기 위한 비디오 카메라(116)를 포함한다. 따라서, 이 구현에서, 타임라인(300)은 실제 미디어를 포함하지는 않지만, 참조되는 미디어를 어디서 그리고/또는 어떻게 찾는지를 기술하는 포인터들(326, 328, 334)을 이용하여 미디어를 참조한다.

<75> 노드들(304, 306)은 또한 미디어 타임라인(300)의 추가 노드들을 기술할 수 있다. 예를 들어, 노드(304)는 노드들(306, 308)에 대한 실행 순서를 기술하는 데 사용될 수 있다. 즉, 노드(304)는 그의 "자식들"의 순위화 및 추가 기술을 제공하기 위한 "접합형" 노드로서 동작한다. 시퀀스 노드 및 병렬 노드와 같이, 미디어 타임라인(300)에서 이용될 수 있는 다양한 접합형 노드가 존재한다. 도 4-6은 시퀀스 및 병렬 노드들 배후의 예시적인 시맨틱을 기술한다.

<76> 도 4는 시퀀스 노드(402), 및 시퀀스 노드(402)의 자식들인 복수의 리프 노드(404, 406, 408)가 도시되는 예시적인 구현(400)의 도면이다. 시퀀스 노드(402)의 자식들은 하나씩 렌더링된다. 또한, 시퀀스 노드(402)는 복수의 리프 노드(404-408)의 렌더링 순서를 기술하는 메타데이터(410)를 포함할 수 있다. 도시된 바와 같이, 리프 노드 404가 먼저 렌더링되고, 이어서 리프 노드 406이, 이어서 리프 노드 408이 렌더링된다. 각각의 리프 노드(404-408)는 각각의 미디어(424, 426, 428)에 대한 각각의 포인터(418, 420, 422)를 갖는 각각의 메타데이터(412, 414, 416)를 포함한다. 따라서, 시퀀스 노드(402)는 파일들의 선형 재생 리스트의 기능을 나타낼 수 있다.

<77> 이 구현에서, 시퀀스 노드(402)의 자식 노드들이 리프 노드로서 구성되지만, 시퀀스 노드(402)의 자식 노드들은 임의의 다른 타입의 노드를 나타낼 수 있다. 예를 들어, 자식 노드들은 도 3에 도시된 바와 같은 복잡한 트리 구조를 제공하는 데 이용될 수 있다. 예를 들어, 도 3의 노드 306은 다른 접합형 노드, 즉 노드 304의 자식이다.

<78> 도 5는 시퀀스 노드(502), 및 시퀀스 노드(502)의 자식들인 복수의 노드(504, 506)가 복수의 노드(504, 506) 각각의 실행을 위한 타이밍 정보를 지정하는 메타데이터를 포함하는 예시적인 구현(500)의 도면이다. 복수의 리프 노드(504, 506, 508) 각각은 전술한 바와 같이 각각의 메타데이터(510, 512, 514)를 포함한다. 노드(502)에 의해 지정되는 시퀀스의 제1 자식 노드인 리프 노드(504)의 메타데이터(510)는 부모 노드, 즉 노드(502) 상의 시작 시간에 상대적인 시작 시간을 지정하는 시간(516) 데이터를 포함한다. 다른 자식 노드들, 즉 리프 노드들(506, 508)은 시퀀스 내의 이전 노드에 상대적으로 지정되는 그들의 시작 시간을 갖는다. 예를 들어, 리프 노드(504)에 대응하는 미디어의 출력이 노드(502)가 처음 실행될 때 요구되는 것으로 가정한다. 또한, 리프 노드(504)에 의해 참조되는 미디어의 출력이 종료될 때, 리프 노드(506)에 대응하는 미디어는 20초의 갭 후에 출력된다. 따라서, 리프 노드(504)의 메타데이터(510)에 의해 지정되는 시간(516)은 "0"이고, 리프 노드(506)의 메타데이터(5120에 의해 지정되는 시간(518)은"20초"이다.

<79> 이전 노드에 상대적인 시간들(516, 518)을 지정하는 것은 시퀀스 내의 각 자식 노드에 의해 참조되는 미디어의 출력 지속 기간이 알려지지 않는 시퀀스를 정의할 수 있게 한다. 리프 노드(508)의 메타데이터(514)에 의해 도시되는 바와 같이, 노드에 대한 시작 시간이 지정되지 않을 때, 이는 이전 노드, 즉 리프 노드(506)가 출력을 종료한 후에 즉시 노드, 즉 리프 노드(508)가 출력을 시작해야 한다는 것을 의미한다.

<80> 도 6은 병렬 노드(602)가 병렬 노드(602)의 자식들인 복수의 리프 노드(606, 608)를 지정하는 메타데이터를 포함하는 예시적인 구현(600)의 도면이다. 도 4 및 5와 관련하여 설명된 이전 구현에서는, 시퀀스 노드의 자식들인 노드들이 하나씩 렌더링되는 시퀀스 노드를 설명하였다. 노드들의 렌더링을 동시에 제공하기 위하여, 병렬 노드(602)가 사용될 수 있다.

<81> 병렬 노드(602)의 자식들은 동시에 렌더링될 수 있다. 예를 들어, 리프 노드 606 및 리프 노드 608은 병렬 노드(602)의 자식들이다. 리프 노드들(606, 608) 각각은 각각의 미디어(618, 620)에 대한 각각의 포인터(614, 616)를 갖는 각각의 메타데이터(610, 612)를 포함한다. 리프 노드들(606, 608) 각각은 각각의 리프 노드(606, 608)가 렌더링될 때를 지정하는 각각의 메타데이터(610, 612)에 포함되는 각각의 시간(622, 624)을 포함한다. 리프 노드들(606, 608) 상의 시간들(622, 624)은 병렬 노드(620), 즉 부모 노드에 상대적이다. 자식 노드들 각각은 조합된 기능을 갖는 복잡한 트리 구조를 제공하는 임의의 다른 타입의 노드 및 노드들의 조합을 나타낼 수 있다. 예를 들어, "접합형" 노드도 미디어를 참조하는 등등일 수 있다. 시간 데이터를 포함하는 메타데이터가 설명되었지만, 다양한 메타데이터가 미디어 타임라인의 노드들 상에 포함될 수 있으며, 그 일례는 아래의 구현

에서 설명된다.

<82> **메타데이터를 저장하도록 구성되는 미디어 타임라인**

<83> 도 7은 노드에 포함될 수 있는 메타데이터의 예를 나타내는 예시적인 구현(700)의 도면이다. 이 구현(700)에서, 미디어 타임라인 내의 각 노드는 메타데이터를 저장할 수 있다. 메타데이터는 노드에 의해 참조되는 미디어, 노드의 자식들에 대한 렌더링 순서와 같은 노드의 특성들을 기술하고, 노드 타입(예를 들어, 순차, 병렬, 루트 및 리프)을 지정하는 등등을 위해 사용될 수 있다. 미디어 타임라인은 노드 상에 메타데이터로서 설정되는 임의의 특성을 처리할 수 있다. 예를 들어, 노드의 시작 및 중지 시간, 노드의 미디어에 대한 URL 등과 같은 특성들이 메타데이터로서 저장된다.

<84> 또한, 미디어 타임라인의 작성자는 커스텀 메타데이터를 노드들에 추가할 수 있다. 예를 들어, 도 2의 애플리케이션(202)은 CD 트랙에 대한 앨범 아트를 그 특정 트랙에 대응하는 리프 노드 상에 저장하는 미디어 재생기로서 구성될 수 있다. 표준 특성들 및 커스텀 특성들은 동일 방식으로 처리될 수 있으며, 따라서 메타데이터를 취득할 때 불명료함이 존재하지 않게 된다. 따라서, 메타데이터에 의해 기술되는 각 특성이 상이한 각각의 인터페이스 또는 소스에 의해 제공되는 경우에도, 미디어 타임라인은 다양한 특성을 추적하기 위한 메카니즘을 제공한다.

<85> 또한, 미디어 타임라인에 의해 일관된 방식으로 메타데이터를 처리함으로써 상이한 소스들로부터의 특성들이 수집될 수 있다. 예를 들어, 재생 리스트는 상이한 작자를 각각 가진 복수의 트랙을 포함할 수 있다. 재생 리스트의 각 트랙은 시퀀스 노드의 자식인 리프 노드로 표현될 수 있다. 미디어 타임라인은 메타데이터를 수집할 수 있으며, 따라서 시퀀스 노드, 즉 부모 노드에 대한 조회는 각 리프 노드, 즉 자식 노드로부터 재생 리스트 내의 모든 미디어의 작자들을 리턴한다. 메타데이터의 일관된 사용은 또한 노드들 각각에 대한 정렬을 제공한다. 예를 들어, 노드 상의 모든 특성이 메타데이터로서 처리되는 경우, 애플리케이션은 일관된 방식으로 메타데이터에 정의된 임의의 특성에 기초하여 노드들을 정렬할 수 있다.

<86> 노드(702)는 노드들에 대한 재생 거동 및 속성을 정의하는 특성들과 같은 다양한 메타데이터(704)를 포함할 수 있다. 메타데이터(704)에 의해 정의되는 특성들의 예는 아래에 설명된다.

<87> URL(706)

<88> 이 특성은 미디어에 대한 URL을 유지한다. 파일의 경우, URL(706) 특성은 파일에 대한 경로를 제공할 수 있다. 예를 들어, URL(706) 특성은 특정 미디어를 찾기 위한 저장 장치에 대한 경로를 제공할 수 있다.

<89> 소스 객체(708), 소스 객체 ID(710)

<90> 몇몇 예에서, 미디어의 소스는 URL에 의해 지정될 수 없다. 예를 들어, 블랙 칼라 프레임을 출력하기 위한 미디어 소스는 URL에 의해 찾지 못할 수 있다. 소스 객체(708) 및 소스 객체 ID(710) 특성은 사용자가 미디어 소스 자체 또는 소정의 다른 객체와 같이 미디어 소스로 변할 수 있는 객체를 지정함으로써 미디어 소스를 지정하는 것을 가능하게 한다. 미디어 소스가 소스 객체로서 지정될 때, 소스 객체(708) 특성은 미디어 소스에 대한 포인터를 제공하며, 소스 객체 ID(710) 특성은 전역적으로 고유한 소스 객체의 식별자를 지정한다. 일 구현에서, 소스 객체(708) 특성 및 URL(706) 특성 양자가 정의되는 경우, 소스 객체(708) 특성은 URL(706) 특성보다 우선 순위를 갖는다.

<91> 시작 시간(712), 중지 시간(714)

<92> 시작 및 중지 시간들(712, 714)은 노드(702)가 다른 노드들과 관련하여 어떤 시간에 시작되고 중지되는지를 정의한다. 예를 들어, 병렬 노드의 자식들인 노드들에 대해, 시작 및 중지 시간들(712, 714)은 병렬 노드, 즉 자식들의 부모에 상대적으로 정의된다. 시퀀스 노드의 자식들인 노드들에 대해, 제1 자식 노드는 시퀀스 노드에 대해 상대적으로 정의되는 시작 및 중지 시간들(712, 714)을 포함한다. 잔여 노드들 각각은 이전 자식에 대해 상대적으로 정의되는 시작 및 중지 시간들을 포함한다. 일 구현에서, 노드(702)에 대한 시작 및 중지 시간들(712, 714)을 정의할 필요가 없다. 예를 들어, 시작 및 중지 시간들(712, 714)이 지정되지 않을 때, 시작 시간(712)은 0인 것으로 가정되며, 노드(702)는 노드(702)에 의해 참조되는 미디어의 렌더링이 완료될 때 중지된다.

<93> 미디어 시작(716), 미디어 중지(718)

<94> 미디어 타임라인 내의 각 노드는 미디어를 참조할 수 있다. 미디어 시작(716) 및 미디어 중지(718) 특성은 출

력될 미디어의 일부를 정의한다. 예를 들어, 노드(702)는 총 50초의 길이를 갖는 파일로부터의 미디어를 나타낼 수 있다. 그러나, 사용자는 파일에서 20초에서 30초까지의 미디어의 일부만을 출력하기를 원할 수 있다. 이를 위해, 미디어 시작(716)은 20초로 지정될 수 있고, 미디어 중지(718)는 30초로 지정될 수 있다.

<95> 노드의 시작 시간(712) 및 중지 시간(714), 즉 "노드 시간"에 의해 정의되는 지속 기간은 미디어 시작(716) 및 미디어 중지(718), 즉 "미디어 시간"에 의해 정의되는 지속 기간과 동일할 필요가 없다. 예를 들어, 지정된 노드 시간이 미디어 시간보다 클 때, 노드(702)에 의해 참조되는 미디어의 출력은 느려질 수 있다. 따라서, 미디어 시작(716) 및 미디어 중지(718)에 의해 정의되는 미디어의 일부는 노드의 시작 및 중지 시간들(712, 714), 즉 "노드 시간"에 의해 정의되는 지속 기간 동안 출력될 수 있다. 즉, 일부의 출력은 노드 시간과 미디어 시간이 동일하도록 연장될 수 있다. 다른 예에서, 미디어의 최종 프레임이 노드 시간이 경과할 때까지 동결될 수 있으며, 비디오 프레임이 공백(예를 들어, 블랙)화될 수 있는 등등일 수 있다. 마찬가지로, 노드 시간이 미디어 시간보다 작은 경우, 미디어는 지정된 노드 시간 내에 출력이 완료되도록 보다 빠른 속도로 출력될 수 있다. 또 다른 예에서, 미디어의 출력은 절단될 수 있다. 예를 들어, 노드 시간보다 큰 미디어 시간에 의해 정의되는 세그먼트의 임의 부분이 출력되지 않는다. 일 구현에서, 미디어 타임라인 자체는 이들 거동을 강제하지 않고, 오히려 이들 거동은 도 1과 관련하여 설명되는 바와 같이 미디어 타임라인(122)을 렌더링할 때 타임라인 소스(124)에 의해 판독된다.

<96> 노드(702)에 대한 미디어 중지(718)가 지정되지 않을 때, 노드(702)에 의해 참조되는 미디어는 완료시까지 출력된다. 예를 들어, 재생기 시나리오에서, 사용자는 참조되는 각 미디어 항목의 지속 기간을 갖지 않은 미디어의 재생 리스트의 출력을 원할 수 있다. 또한, 재생 리스트에 포함된 미디어의 "백투백(back to back)" 출력을 원할 수도 있다. 미디어 타임라인 상에 이러한 경우를 나타내기 위하여, 지정된 미디어 중지(718) 특성을 갖지 않는 그의 자식들인 리프 노드들을 갖는 시퀀스 노드가 생성될 수 있다.

<97> 시간 포맷(720)

<98> 전술한 시간 기반 특성들은 그에 수반되는 시간 포맷(720) 특성(시간 포맷)을 가질 수 있다. 시간 포맷의 예는 100 나노초 단위, 프레임 수, 시간 코드 등을 포함한다. 따라서, 시간 포맷(720)은 시작 시간(712), 중지 시간(714), 미디어 시작(716) 및 미디어 중지(718)에 대한 시간 포맷을 지정할 수 있다. 또한, 시간 포맷(720)은 시간 기반 특성들 각각에 대해 상이한 포맷을 지정할 수 있다. 예를 들어, 시작 및 중지 시간들(712, 714)은 100 나노초 단위의 시간 포맷을 사용할 수 있는 반면, 미디어 시작(716) 및 미디어 중지(718)는 프레임 카운트를 사용할 수 있다.

<99> 스트림 선택(722)

<100> 스트림 선택(722) 특성은 노드(702) 상에서 다양한 방식으로 이용될 수 있다. 예를 들어, 스트림 선택(722) 특성은 원하는 특성을 가진 미디어가 제공될 수 있도록 필터로서 동작할 수 있다. 예를 들어, 노드(702)는 텔레비젼 프로그램과 같은 미디어의 오디오 및 비디오 스트림 양자를 참조할 수 있다. 그러나, 사용자는 노드(702) 상에서 지정된 URL(706)이 오디오 및 비디오 스트림 양자를 지시하는 경우에도 비디오 스트림에만 관심을 가질 수 있다. 이 경우, 미디어로부터의 오디오 스트림은 노출되지 않으며, 따라서 사용자에게는 노드(702)가 비디오 미디어만을 제공하는 것으로 보인다. 스트림 선택의 몇몇 다른 예는 스트림에 대한 언어 선택, 스트림에 대한 비트 레이트 선택 등을 포함한다. 또한, 파일들은 동일한 주요 타입의 다수의 스트림을 포함할 수 있다. 예를 들어, 몇몇 파일들은 스트림들 각각에 대한 언어 및 비트 레이트의 선택을 제공하는 많은 오디오 스트림을 포함한다.

<101> 포맷 기반(724)

<102> 포맷 기반(724) 특성들은 노드(702)로부터 요구되는 프레임 레이트, 픽셀 종횡비, 오디오 샘플링 레이트 등과 같은 다른 특성들을 지정하는 데 사용될 수 있다. 이어서, 이들 포맷으로부터/이들 포맷으로의 변환을 위한 적절한 트랜스폼들이 재생 동안 렌더링된 미디어 타임라인에 삽입된다.

<103> 루프 카운트(726)

<104> 루프 카운트(726) 특성은 노드(702)의 렌더링이 몇 회 반복되는지를 지정하는 데 사용될 수 있다. 예를 들어, 루프 카운트(726) 특성이 음인 경우, 노드(702)에 의해 참조되는 미디어의 출력은 무한히 반복될 수 있다.

<105> 디스에이블드(728)

<106> 노드(702)는 디스에이블드(728) 특성을 설정함으로써 디스에이블될 수 있다. 예를 들어, 디스에이블드(728) 특

성이 "참"으로 설정되는 경우, 노드(702)는 미디어 타임라인의 렌더링 동안 무시된다. 예를 들어, 3개의 리프 노드들의 시퀀스가 미디어 타임라인에 제공될 수 있다. 미디어 타임라인 내의 제2 노드가 디스에이블되면, 즉 디스에이블드(728) 특성이 "참"으로 설정되면, 미디어 타임라인에 의해 참조되는 미디어의 출력은 미디어 타임라인이 제1 및 제3 노드만을 가진 것처럼 나타날 것이다.

<107> 노스킵(730)

<108> 노스킵(730) 특성은 미디어 타임라인의 렌더링 동안 스킵될 수 없는 미디어를 지정하기 위해 타임라인 작성자에 의해 사용될 수 있는 특징이다. 노드(702)가 노스킵 노드로서 지정될 때, 즉 노스킵 특성이 "참"으로 설정될 때, 사용자는 지정된 노드(702) 후에 다른 노드로 스킵할 수 없으며, 그 노드(702)의 일부로서 출력되는 미디어를 빠르게 전송할 수 없다. 그러나, 사용자는 노드(702) "전의" 임의의 노드를 스킵할 수 있다. 다른 구현에서, 노스킵(730) 특성이 부모 노드 상에서 지정되는 경우, 사용자는 그 노드의 서브 트리 내의 어떠한 자식도 스킵할 수 없을 것이다. 또 다른 구현에서, 노스킵(730) 특성은 시퀀스 노드 및 그의 직계 자식들, 예를 들어 시퀀스 노드의 자식인 다른 시퀀스 노드에 포함되는 대신에 시퀀스 노드를 직접 따르는 시퀀스 노드의 자식들에만 적용되며, 병렬 노드 및 그의 직계 자식들에게는 지정되지 않는다. 예를 들어, 노스킵(730) 특성은 제1 시퀀스 노드의 자식들인 리프 노드들에 의해 참조되는 광고들의 스킵을 방지하는 데 사용될 수 있다. 제2 시퀀스 노드는 또한 제1 시퀀스 노드의 자식일 수 있으며, 텔레비젼 프로그램과 같이 스킵될 수 있는 미디어를 참조하는 리프 노드들을 포함할 수 있다.

<109> 노스킵(730) 특성은 또한 사용자가 네비게이트할 수 있는 노드들의 집합을 정의하는 데 사용될 수 있다. 예를 들어, 미디어 타임라인은 10개 리프 노드의 시퀀스를 포함할 수 있는데, 제3 및 제7 노드는 노스킵 노드, 즉 노스킵 특성이 "참"으로 설정되는 노드이다. 따라서, 사용자는 제1 및 제2 리프 노드의 렌더링을 스킵할 수 있지만, 제4, 제5, 제6, 제7, 제8, 제9 또는 제10 노드로 스킵할 수 없다. 유사하게, 제4 노드에서 제7 노드까지의 미디어 타임라인의 렌더링 동안, 사용자는 제7 노드 이하의 임의의 노드로 스킵할 수 있지만, 제7 노드 "위의" 노드, 즉 제8, 제9 및 제10 노드로는 스킵할 수 없다.

<110> 노스킵 자식(732)

<111> 미디어 타임라인은 성긴 자식들을 지원할 수 있는데, 즉 미디어 타임라인이 최초 로딩될 때 미디어 타임라인 상에 모든 노드가 로딩 및/또는 생성되지는 않는다. 따라서, 자식들은 필요에 따라 로딩 및/또는 생성될 수 있다. 노드들의 동적 로딩 및 생성에 대한 추가 설명은 도 12 및 13과 관련하여 발견될 수 있다. 이 예에서 미디어 타임라인 내의 노드들이 로딩될 때, "노스킵"으로 지정되는 자식 노드들을 갖는 부모 노드들이 로딩될 수 있다. 자식 노드에 대해 노스킵(730) 특성이 존재한다는 것을 지시하기 위하여, 부모 노드에 대해 노스킵 자식(732) 특성이 사용될 수 있다.

<112> 노스킵 자식(732) 특성은 부모 노드가 "참"으로 설정된 노스킵(730) 특성을 갖는 자식 노드를 포함하는지를 지시하기 위해 부모 노드에서 설정될 수 있다. 미디어 타임라인의 렌더링 동안, 노스킵 자식(732)은 한 노드의 모든 이전 자식들이 그 노드에 대한 네비게이션이 유효한지를 결정하기 위해 검사되어야 한다는 것을 지시하기 위해 사용된다. 노스킵 자식(732)은 또한 병렬 노드 상에서 설정될 수 있다. 예를 들어, 병렬 노드의 서브 트리 내의 임의 노드는 "참"으로 설정된 노스킵(730) 특성을 갖는다. 이러한 방식으로, 노스킵(730) 특성의 사용을 보호하는 노드들 간의 네비게이션이 제공될 수 있다.

<113> "참"으로 설정된 노스킵(730) 특성을 갖는 노드가 미디어 타임라인에 추가될 때, 미디어 타임라인은 추가된 노드의 모든 부모 상에서 노스킵 자식(732) 특성을 "참"으로 자동 설정할 수 있다. 이러한 방식으로, 렌더링 엔진, 예를 들어 도 1의 타임라인 소스(124)는 노스킵(730) 특성이 "참"으로 설정되어 있는지를 결정하기 위해 미디어 타임라인의 어느 노드들을 로딩하여 검사할지를 최적화할 수 있다.

<114> **타임라인 효과**

<115> 타임라인 효과는 미디어 타임라인의 작성자가 미디어의 외형을 분석 및/또는 변경하는 컴포넌트를 지정하는 것을 가능하게 한다. 예를 들어, 작성자는 비디오를 흑백으로 보고, 오디오 파일에 에코를 추가하고, 하나의 비디오를 다른 비디오의 상부에서 보는(예를 들어 픽처 내의 픽처) 것 등을 원할 수 있다. 일 구현에서, 효과는 단독적으로는 개별 노드가 아니다. 미디어에 대한 효과를 제공하기 위하여, 작성자는 노드 내의 메타데이터에서 효과를 지정할 수 있다. 예를 들어, 메타데이터는 노드 상에서 정의되는 효과들의 어레이를 포함할 수 있다. 어레이는, 즉 노드에 의해 참조되는 미디어가 렌더링될 때 그 노드의 출력에 적용될 일련의 효과를 지정할 수 있다. 이 구현에서, 효과는 효과를 실제로 구현하는 객체는 아니며, 오히려 효과를 생성하고 적용하는

방법을 기술하는 특성 및 속성을 지정한다. 이것은 이전 구현에서 노드가 미디어를 참조하는 방법과 유사하다. 예를 들어, 도 3과 관련하여 설명한 바와 같이, 리프 노드들(308-312 자체는 미디어를 포함하지 않고, 오히려 미디어를 취득하는 방법을 지정하는 각각의 포인터(326, 328, 332)를 갖는 각각의 메타데이터(320-324)를 포함한다. 효과를 실제로 구현하는 컴포넌트는 미디어 타임라인을 실행하는 타임라인 소스에 의해 실행 시간에 로딩된다. 효과를 포함하는 메타데이터가 설명되었지만, 효과는 또한 개별적으로 지정될 수 있다. 또한, 다른 구현에서, 효과는 효과를 구현하는 미디어 타임라인 내의 객체에 의해 제공된다.

<116> 미디어 타임라인의 노드들 상에서 지정되는 효과들은 당해 노드의 시작 시간에 상대적으로 지정되는 시간들을 가질 수 있다. 예를 들어, 효과는 10초의 시작 시간을 갖는 리프 노드 상에서 지정될 수 있다. 따라서, 효과는 렌더링 시에 노드가 출력을 시작하고 10초가 경과한 후에 그 노드에 적용될 것이다.

<117> 다수의 효과가 하나의 노드 상에 지정될 수 있다. 또한, 미디어 타임라인의 작성자는 또한 이들 효과가 적용되는 순서를 제어할 수 있다. 예를 들어, 작성자는 효과 상에 우선 순위를 설정할 수 있다. 노드에 의해 지정될 수 있는 다양한 효과가 존재한다. 미디어 타임라인 상에서 지정될 수 있는 효과들의 예는 (1) 단순 효과, (2) 복합 효과 및 (3) 전이 효과를 포함한다. 이들 예시적인 효과에 대한 추가 설명은 도 8-10과 관련하여 발견될 수 있다.

<118> 단순 효과

<119> 단순 효과는 오디오/비디오의 단일 스트림을 수신하고 다른 하나의 스트림을 출력하는 컴포넌트를 나타낸다. 즉, 이것은 1 입력/1 출력 컴포넌트이다. 예를 들어, 에코 효과는 오디오 스트림을 수신하고 에코화된 변경 데이터 스트림을 출력하고, 비디오가 흑백으로 보이는 "흑백" 효과, 비디오가 수십년 전에 캡쳐된 것처럼 보이게 하는 숙성(age) 효과 등을 제공할 수 있다.

<120> 도 8은 노드(802)가 복수의 단순 효과(806, 808, 810)를 지정하는 메타데이터(804)를 포함하는 예시적인 구현(800)의 도면이다. 도 8에 도시된 바와 같이, 복수의 단순 효과(806-810)는 포인터(816)에 의해 참조되는 저장 장치(814)에 포함된 미디어(812)에 적용될 수 있다. 이 예에서, 복수의 효과(806-810)는 연결되어, 즉 제1 효과(806)의 출력은 제2 효과(808)의 입력을 가는 것 등과 같이 되어, 미디어(812)에 다수의 효과를 제공한다. 복수의 효과(806-810) 각각은 효과들을 처리하기 위한 순서를 제공하는 우선 순위를 부여받을 수 있다. 예를 들어, 효과들의 우선 순위는 효과들이 연결되는 순서를 결정할 수 있다. 우선 순위의 충돌이 존재하는 경우, 효과들은 효과 어레이에서 각각의 효과가 지정된 순서로 추가될 수 있다.

<121> 일 구현에서, 복수의 효과(806-810)의 지속 기간은 미디어(812)의 지속 기간을 변경하지 않는다. 예를 들어, 복수의 효과(806-810)의 처리는 노드(802)의 시간 경계들에서 중단될 수 있다. 예를 들어, 미디어(812)의 렌더링은 10초의 지속 기간을 가질 수 있다. 그러나, 복수의 효과(806-810)의 처리는 20초의 지속 기간을 가질 수 있다. 이 경우, 도 1의 타임라인 소스(124)는 노드(802)에 대한 복수의 효과(806-810)의 처리를 10초에 종료할 수 있다.

<122> 효과들(806-810)을 정의할 때, 타임라인의 작성자는 효과들(806-810) 각각의 입력 및 출력을 명시적으로 지정할 수 있다. 예를 들어, 효과들(806-810) 각각은 어느 스트림이 어느 효과 입력에 연결되는지를 기술하는 데이터를 포함할 수 있다. 효과들(806-810) 각각은 또한 각 효과(806-810)의 출력의 주요 타입, 예를 들어 오디오, 비디오 등을 기술하는 각각의 데이터를 가질 수 있다. 또한, 효과들(806-810) 각각은 노드 내의 효과의 시작 시간 및/또는 중지 시간을 기술하는 메타데이터를 포함할 수 있다.

<123> 복합 효과

<124> 복합 효과는 병렬 노드의 자식들의 미디어를 처리하여 결과 출력을 제공하는 데 사용될 수 있다. 예를 들어, 도 9는 둘 이상의 자식 노드의 출력에 복합 효과를 제공하는 병렬 노드(902)를 나타내는 예시적인 구현(900)의 도면이다. 이 구현에서 병렬 노드(902)는 도 6과 관련하여 기술된 병렬 노드(602)와 유사하다.

<125> 병렬 노드(902)는 복합 효과들(906, 908, 910)의 어레이를 포함한다. 복합 효과를 지정할 때, 미디어 타임라인의 작성자는 효과들(906-910)의 입력들을 연결하는 방법, 및 효과들(906-910)로부터의 출력들에 대한 주요 타입을 지정한다. 예를 들어, 리프 노드(912) 및 리프 노드(914)는 병렬 노드(902)의 자식들로서 구성될 수 있다. 전술한 바와 같이, 각각의 리프 노드(912, 914)는 각각의 미디어(924, 926)를 참조하는 각각의 포인터(920, 922)를 가진 각각의 메타데이터(916, 918)를 포함한다. 리프 노드들(912, 914)은 렌더링시 출력을 위한 미디어(924, 926)를 제공한다.

<126> 효과들(906, 908, 910)은 병렬 노드(902)에 의해 지정되는 미디어(924, 926)의 출력에 적용된다. 예를 들어, 병렬 노드(902)는 각 면에 상이한 미디어(예를 들어, 비디오)를 갖는 회전 큐브, 각 프레임에서 상이한 미디어가 재생되는 필름의 스크롤링 롤 등을 제공할 수 있다.

<127> 병렬 노드(902)가 리프 노드들(912, 914) 각각에 복수의 효과(906-910)를 적용하는 것으로 설명되었지만, 추가적인 구현에서 병렬 노드(902)는 병렬 노드의 소수의 자식들에게만 효과들(906-910)을 적용할 수도 있다. 즉, 효과들(906-910)은 병렬 노드(902)의 자식들인 모드들 모두에 적용될 필요는 없다. 예를 들어, 메타데이터(904) 및/또는 효과들(906-910)은 복수의 효과(906-910) 중 하나 이상을 적용할 하나 이상의 특정 노드를 지정할 수 있다.

<128> 전이 효과

<129> 도 10은 전이 효과가 이전 노드에 의해 참조되는 미디어의 출력과 후속 노드에 의해 참조되는 미디어의 출력 간에 효과를 제공하도록 지정되는 예시적인 구현(1000)의 도면이다. 도 10에는, 도 4와 관련하여 설명된 시퀀스 노드(402)와 유사한 시퀀스 노드(1002)가 도시되어 있다. 시퀀스 노드(1002)는 시퀀스 노드(1002)의 자식들인 복수의 리프 노드(1004, 1006, 1008)를 갖는다. 복수의 리프 노드(1004-1008) 각각은 각각의 미디어(1026-1030)를 참조하는 각각의 포인터(1020-1024)를 갖는 각각의 메타데이터(1014-1018)를 포함한다.

<130> 시퀀스 노드(1002)는 각각의 리프 노드(1004-1008)에 의해 참조되는 미디어(1026-1030)의 출력 사이에 사용될 전이 효과를 기술하는 메타데이터(1010)를 포함한다. 따라서, 전이 효과(1012)는 시퀀스 노드(1002)의 자식들로부터 유래하는 미디어(1026-1030)에 적용된다. 전이 효과(1012)는 둘 이상의 미디어(1026-1030)를 단일 출력으로 결합하는 데 사용된다. 또한, 전이 효과(1012)는 전이 효과가 적용되는 리프 노드들(1004-1008) 중 하나 이상을 지정하는 데이터를 포함할 수 있다. 예를 들어, 데이터는 전이 효과(1012)가 미디어(1026, 1028)의 출력 사이에 적용됨을 지정할 수 있다. 전이 효과(1012)에 대한 제1 입력은 그가 정의되는 노드, 즉 리프 노드(1004)에 의해 제공된다. 전이 효과(1012)에 대한 다음 입력은 시퀀스 내의 다음 노드, 즉 리프 노드(1006)이다. 전이 효과들의 예는 시퀀스 내의 출력인 2개 노드 간의 오디오 크로스 페이드, 제1 비디오의 제2 비디오와의 "스와이프(swipe)" 등을 포함한다.

<131> 전이 효과(1012)는 지속 기간(1032)을 갖는다. 지속 기간(1032)은 시퀀스 내의 둘 이상의 노드 간에 요구되는 중복의 양을 지정하는 데 사용될 수 있다. 예를 들어, 시퀀스 내의 제2 입력, 즉 미디어(1026)는 전이 효과(1012)의 지속 기간(1032) 동안 중복되도록 출력될 수 있다. 따라서, 시퀀스 노드(1002)의 출력 지속 기간은 리프 노드들(1004-1008) 상에서 지정되는 시간들 및 전이 효과(1012)의 지속 기간에 의해 지정되는 중복의 함수가 된다.

<132> 글로벌 효과도 지정될 수 있다. 예를 들어, 전이 효과(1012)는 노드의 자식들, 예를 들어 리프 노드들(1004-1008) 각각에 대한 글로벌 전이를 지정할 수 있다. 따라서, 미디어 타임라인의 작성자가 모든 리프 노드(1004-1008)에 대해 동일한 전이의 사용을 원하는 경우, 작성자는 전이 효과(1012)를 글로벌 전이로서 지정함으로써 그렇게 할 수 있다. 따라서, 글로벌 전이를 지정함으로써 작성자는 각각의 노드에 대해 개별 전이를 지정할 필요가 없다.

<133> 효과 메타데이터

<134> 도 11은 복수의 효과 메타데이터(1106)를 포함하는 메타데이터(1104)를 갖는 노드(1102)를 나타내는 예시적인 구현(1100)의 도면이다. 도 7과 관련하여 기술된 바의 노드(702) 상에서 지정되는 메타데이터(704)와 같이, 효과 메타데이터(1106)는 또한 효과에 대한 다양한 특성을 지정할 수 있다. 예를 들어, 효과 메타데이터(1106)는 미디어 타임라인에 대해 특정 의미를 갖는 표준 특성들을 정의할 수 있다. 효과 메타데이터(1106)는 효과를 제공하는 데 사용되는 트랜스폼 객체들을 구성하는 데 사용될 수 있다. 예를 들어, 트랜스폼 객체들은 효과를 제공하도록 트랜스폼 객체를 구성하는 데 사용되는 그들 자신의 특성 세트를 발행할 수 있다. 예를 들어, 칼라 변환기 트랜스폼 객체에 대해, 색조, 채도 및 명도를 제어하기 위한 특정 특성들이 존재한다. 이들 특성 외에, 다른 중요한 커스텀 특성들이 효과에 대해 지정될 수 있다. 다음은 미디어 타임라인에 의해 지원되는 효과 메타데이터(1106)의 예들의 리스트이다.

<135> 효과 객체 GUID(1108)

<136> 노드들이 미디어를 참조할 수 있는 방법과 유사하게, 효과는 효과를 제공하는 트랜스폼 객체를 참조할 수 있다. 효과 객체 GUID(1108) 특성은 효과를 제공하는 트랜스폼 객체를 생성하는 데 사용될 GUID를 지정한다. 예를 들

어, 미디어의 출력 동안, 효과 객체 GUID(1108)에 의해 참조되는 트랜스폼 객체는 효과를 제공할 필요가 있을 때 생성될 수 있다.

<137> 효과 객체(1110)

<138> 노드(1102)는 효과를 제공하는 효과 객체를 참조하기 위한 포인터로서 효과 객체(1110) 특성을 이용할 수 있다. 참조되는 효과 객체는 노드(1102)의 미디어의 출력 동안 직접 사용될 수 있다. 효과 객체(1110) 특성 및 효과 GUID 양자가 지정되는 경우, 효과 객체(1110) 특성은 효과 GUID보다 우선 순위를 갖는다.

<139> 우선 순위(1112)

<140> 전술한 바와 같이, 효과들이 함께 연결될 때, 우선 순위(1112) 특성은 효과들의 렌더링을 지정하는 데 사용될 수 있다. 동일 우선 순위를 가진 둘 이상의 효과가 존재하는 경우, 효과들은 이 효과들이 노드(1102)에 추가된 순서로 적용된다.

<141> 시작 시간(1114), 중지 시간(1116)

<142> 시작 및 중지 시간들(1114, 1116)은 효과가 지정되는 노드(1102)에 상대적으로 지정된다. 시작 및 중지 시간들(1114, 1116)은 효과가 활성화되는 시간을 정의한다. 이들 특성이 지정되지 않으면, 효과는 노드(1102)에 의해 참조되는 미디어의 출력의 전체 지속 기간 동안 적용될 것이다. 이들 특성은 도 8과 관련하여 설명된 단순 효과들 및 도 9와 관련하여 설명된 복합 효과들 양자에 대해 적용될 수 있다.

<143> 시간 포맷(1118)

<144> 시작 및 중지 시간들(1114, 1116)은 다양한 포맷으로 지정될 수 있다. 시간 포맷(1118) 특성은 이들 시간 값의 포맷을 지정하는 데 사용될 수 있다. 100 나노초 단위, 프레임 수, 시간 코드 등과 같은 다양한 포맷이 사용될 수 있다.

<145> 지속 기간(1120)

<146> 도 10과 관련하여 전술한 바와 같이, 지속 기간(1120) 특성은 각각의 미디어의 출력 간의 전이의 지속 기간을 지정하는 데 사용될 수 있다. 예를 들어, 지속 기간(1120)은 2개의 연속 노드에 의해 참조되는 미디어의 출력 간의 중복의 양을 지정하는 데 사용될 수 있다.

<147> 입력 수(1122), 출력 수(1124)

<148> 단순 효과들은 하나의 입력 및 하나의 출력을 이용하며, 따라서 단순 효과들에 대한 입력 및 출력의 수(1122, 1124)는 미디어 타임라인에서 자동으로 설정될 수 있다. 전이 효과는 2개의 입력 및 하나의 출력을 이용할 수 있다. 따라서, 전이 효과들에 대한 입력 및 출력의 수(1122, 1124)도 미디어 타임라인에서 자동으로 설정될 수 있다. 복합 효과들에 대해, 작성자는 원하는 만큼 많은 입력 및/또는 출력을 정의할 수 있다. 따라서, 입력 및 출력의 수(1122, 1124)는 효과를 제공하는 트랜스폼 객체에 대한 입력 및 출력의 수를 반영하도록 작성자에 의해 설정될 수 있다.

<149> 출력 주요 타입(1126)

<150> 출력 주요 타입(1126)은 효과의 각 출력에 대해 지정된다. 출력 주요 타입(1126) 특성의 지정은 효과를 다른 효과들 또는 수신처에 연결하는 것을 용이하게 한다. 예를 들어, 미디어 타임라인의 작성자는 출력의 주요 타입, 즉 오디오, 비디오 등을 쉽게 결정할 수 있으며, 따라서 관련 효과들 간의 연결, 예를 들어 오디오 효과 대 오디오 효과의 연결을 효율적으로 지정할 수 있다.

<151> 입력 연결(1128)

<152> 효과가 정의되면, 작성자는 효과에 의해 처리될 미디어를 지정할 수 있다. 입력 연결(1128) 특성은 효과 입력들 각각에 연결되는 미디어를 식별하는 데 사용될 수 있다.

<153> 미디어 **타임라인의 노드들의 동적 생성 및 로딩**

<154> 미디어 타임라인의 노드들의 동적 생성 및 로딩은 미디어 타임라인의 효율적인 렌더링을 위해 사용될 수 있다. 렌더링을 효율적으로 개선하기 위하여, 미디어 타임라인은 제한된 하드웨어 및/또는 소프트웨어 자원을 가진 장치들과 같은 저 자원 장치들 상에서 이용될 수 있다. 예를 들어, 미디어 타임라인의 동적 생성은 미디어 타임라인의 노드들의 지연 생성을 포함할 수 있다. 예를 들어, 부모 노드의 자식들은 필요할 때까지 생성될 필요가

없다. 노드들의 지연 생성은 상당한 수의 노드들 및/또는 각 노드에 대한 많은 양의 데이터를 가진 미디어 타임라인에 대한 시동 및 응답 시간을 개선하는 데 이용될 수 있다. 예를 들어, 미디어 재생기는 상당한 수의 선택들을 포함하는 미디어 라이브러리로부터 재생 리스트를 생성하고 재생하는 데 사용될 수 있다. 이러한 재생 리스트의 생성은 미디어 라이브러리에 대한 다수의 조회를 필요로 할 수 있으며, 이는 상당한 양의 시간, 프로세서 및 메모리 자원을 취할 수 있다. 노드들의 지연 생성을 이용함으로써, 재생 리스트는 필요에 따라 생성될 수 있으며, 따라서 임의의 하나의 특정 시간에 필요한 노드들에 의해 요구되는 만큼의 처리 및 메모리 자원만을 이용할 수 있다. 미디어 타임라인의 노드들의 동적 생성 및/또는 로딩에 이용될 수 있는 다양한 구현이 존재한다.

<155> 도 12는 미디어 타임라인의 노드들이 노드들에 포함된 메타데이터에 기초하여 동적으로 로딩되는 예시적인 구현에서의 미디어 타임라인(1200)의 일례의 도면이다. 미디어 타임라인(1200)은 루트 노드(1202) 및 루트 노드(1202)의 자식들인 복수의 노드(1204-1216)를 포함하는 트리 구조로서 도시되어 있다. 미디어 타임라인(1200)을 렌더링하기 위하여, 루트 노드(1202)가 먼저 구현되고, 그 안에 포함된 메타데이터(1218)가 조사된다. 메타데이터(1218)는 노드들(1204, 1206)을 포함하는 제1 그룹(1220)을 지정한다. 따라서, 루트 노드(1202)가 렌더링될 때, 노드(1204) 및 노드(1206)도 렌더링을 위해 로딩된다.

<156> 노드(1206)에 의해 참조되는 미디어의 렌더링 동안 또는 그 후에, 노드(1208, 1210)를 포함하는 제2 그룹(1224)을 지정하는 노드(1204)의 메타데이터(1222)가 조사된다. 따라서, 노드(1208, 1210)가 로딩되고, 노드(1210)에 의해 참조되는 미디어가 출력된다. 마찬가지로, 노드(1208)의 메타데이터(1226)는 노드(1212, 1214, 1216)를 포함하는 제3 그룹(1228)을 지정한다. 따라서, 노드(1210)에 의해 참조되는 데이터의 출력이 완료된 후 노드들(1214, 1216)에 의해 참조되는 데이터를 출력하도록 노드들(1212, 1214, 1216)이 로딩된다.

<157> 도 13은 미디어 타임라인의 노드들이 노드 소스에 의해 필요에 따라 정의되고 구현되는 예시적인 구현에서의 미디어 타임라인(1300)의 도면이다. 도 12와 관련하여 설명된 이전 구현에서, 미디어 타임라인(1200)은 사전에 생성되고 노드들은 필요에 따라 로딩되었다. 이 구현에서, 작성자는 미디어 타임라인(1300)의 한 노드(1302)의 자식들인 다수의 노드를 정의한다. 이어서, 노드들은 미디어 타임라인(1300)의 렌더링 동안 필요에 따라 생성된다. 이것은 노드(1302)에 노드 소스(1304) 모듈(노드 소스)을 첨부함으로써 수행된다. 노드 소스(1304)는 충분한 데이터를 포함하며, 따라서 실행시 노드 소스(1304)는 노드(1302)의 자식들, 즉 노드들(1306, 1308)을 생성하고 도 7과 관련하여 전술한 바와 같은 특성 및 상호 관계를 정의하는 메타데이터(1310, 1312)를 제공하는 것과 같이 노드들(1306, 1308)의 특성들을 채울 수 있다(fill-out). 따라서, 노드들(1306, 1308) 중 특정 노드가 필요할 때, 노드 소스(1304)가 구현, 예를 들어 호출되어 노드들(1306, 1308) 중 특정 노드를 생성한다. 예를 들어, 노드 소스(1304)는 도 1의 타임라인 소스(124)로부터의 요청에 응답하여 노드들(1306, 1308)을 생성하도록 실행될 수 있다.

<158> 미디어 **타임라인의 노드들의 동적 변경**

<159> 하나 이상의 구현에서, 미디어 타임라인은 동적으로 변경되도록 구성된다. 예를 들어, 미디어 타임라인의 노드들은 타임라인 소스에 의해 미디어 타임라인의 렌더링 동안 제거, 추가 또는 변경될 수 있다. 노드들에 대한 동적 변경을 제공하기 위하여, 각 노드는 이벤트를 생성할 수 있다.

<160> 도 14는 미디어 타임라인(1400)에 발생하는 변경이 변경에 의해 영향을 받을 수 있는 노드들에 통신될 수 있도록 노드에 의해 이벤트가 제공되는 예시적인 구현에서의 미디어 타임라인(1400)의 도면이다. 미디어 타임라인(1400)은 루트 노드(1402) 및 루트 노드(1402)의 자식들인 복수의 노드(1404-1412)를 포함한다. 노드들(1408-1410)은 전술한 바와 같이 미디어를 참조하는 데 사용된다.

<161> 노드들(1402-1412) 각각은 노드에 대한 변경 및/또는 그 노드의 자식들에 대한 변경에 의해 영향을 받을 수 있는 미디어 타임라인(1400)의 다른 노드들에게 통지하는 데 사용될 수 있는 이벤트를 생성할 수 있다. 예를 들어, 노드(1406) 및 노드(1406)의 임의의 자식들, 즉 노드들(1410-1412)에 대한 모든 이벤트가 미디어 타임라인(1400)의 루트 노드(1402) 및/또는 작성자에게 통신될 수 있다. 즉, 미디어 타임라인(1400)의 이벤트들은 트리의 루트까지 트리를 위로 올라갈 수 있다. 이러한 방식으로, "이벤팅"은 타임라인 구조에 대한 동적 변경에 대해 미디어 타임라인(1400)의 다양한 노드에 알리는 데 사용될 수 있다. 또한, 미디어 타임라인(1400)의 노드들은 미디어 타임라인의 다른 노드들에 의해 개시되는 이벤트들에 가입할 수 있다. 예를 들어, 노드(1408)는 노드(1406)의 "부모"는 아니지만 노드(1406)로부터 이벤트를 수신하기 위해 가입할 수 있다. 더욱이, 타임라인을 이용하는 컴포넌트, 예를 들어 도 2의 미디어 파운데이션(204) 컴포넌트는 임의 노드에 의해 개시되는 이벤트를 수신하도록 등록할 수 있다. 다양한 이벤트(1414)가 하나 이상의 노드(1402-1412)에 의해 지원될 수 있으며,

그 예가 아래에 설명된다.

<162> 노드 추가(1416)

<163> 이 이벤트는 노드가 미디어 타임라인(1400)에 추가될 때 발생한다. 예를 들어, 노드(1412)는 노드(1412)에 의해 참조되는 추가 미디어의 출력을 제공하기 위해 미디어 타임라인(1400)에 추가될 수 있다. 노드(1406)는 노드(1412)의 추가에 대해 통지 받을 때 노드 추가(1416) 이벤트를 발행하여 이것이 노드(1404)를 통해 루트 노드(1402)로 통신되게 할 수 있다. 따라서, 이 예에서, 새로 추가된 노드(1412)의 부모인 각 노드(1402-1406)는 그 노드의 부모에 의해 개시되는 이벤트를 통지 받는다.

<164> 노드 제거(1418)

<165> 노드 제거(1418) 이벤트는 노드가 미디어 타임라인(1400)으로부터 제거될 때 발행된다. 이전 예에 계속하여, 노드(1412)는 노드(1412)에 의해 참조되는 미디어의 출력을 제거하기 위하여 미디어 타임라인(1400)으로부터 제거될 수 있다. 노드(1406)는 노드(1412)의 제거를 통지 받을 때 노드 제거(1418) 이벤트를 발행하여 이것이 노드(1404)를 통해 루트 노드(1402)로 통신되게 할 수 있다. 따라서, 이 예에서, 제거된 노드(1412)의 부모인 각 노드(1402-1406)도 통지 받는다.

<166> 노드 변경중(1420)

<167> 노드 변경중(1420) 이벤트는 미디어 타임라인(1400)의 노드 상의 메타데이터가 변경되고 있을 때 발행된다. 예를 들어, 노드(1406)는 도 7과 관련하여 설명된 메타데이터(704)와 같은 메타데이터를 포함할 수 있다. 메타데이터에 대한 변경은 노드(1406)가 노드 변경중(1420) 이벤트를 발행하게 할 수 있는데, 이 이벤트는 도 2의 애플리케이션 및/또는 노드(1406)의 부모, 즉 노드들(1402, 1404)에 통신될 수 있다. 따라서, 노드 변경중(1420) 이벤트는 노드를 이용하는 다른 노드들 및/또는 애플리케이션들에게 노드(1406)에 대해 변경이 이루어지고 있음을 통지하고, 따라서 노드 변경 완료(1422) 이벤트가 수신될 때까지 노드의 렌더링을 대기하는 것과 같이 적절히 응답하기 위해 이용될 수 있다.

<168> 노드 변경 완료(1422)

<169> 노드 변경 완료(1422) 이벤트는 미디어 타임라인(1400)의 노드 상의 메타데이터가 변경된 때 발행된다. 이전 예에 계속하여, 노드(1406)는 노드 변경중(1420) 이벤트를 발행하여 다른 노드들 및/또는 애플리케이션들이 노드(1406)에 대해 변경이 이루어지고 있음을 통지받게 할 수 있다. 변경이 완료된 때, 노드(1406)는 노드 변경 완료(1422) 이벤트를 발행하여 애플리케이션 및/또는 노드들에게 변경이 완료되었음을 통지할 수 있다. 이러한 방식으로, 노드(1406)는 노드 변경 완료(1422) 이벤트를 이용하여 렌더링을 위한 준비가 되어 있음을 통지할 수 있다.

<170> 자식 제거(1424)

<171> 자식 제거(1424) 이벤트는 노드의 모든 자식이 제거될 때 발행된다. 예를 들어, 노드들(1410, 1412)은 미디어 타임라인(1400)으로부터 제거될 수 있다. 노드(1406)는 자식 제거(1424) 이벤트를 발행하여 루트 노드(1402)에게 노드(1406)의 자식들, 즉 노드들(1410, 1412)이 제거되었음을 통지한다. 따라서, 자식 제거(1424) 이벤트는 노드들(1410, 1412) 각각에 대해 노드 제거(1418)를 발행하는 대신에 사용될 수 있다.

<172> 노드 소스 추가(1426), 노드 소스 제거(1428)

<173> 노드 소스 추가(1426) 이벤트는 도 13과 관련하여 설명된 노드 소스(1304)와 같은 노드 소스가 노드에 추가될 때 발행된다. 마찬가지로, 노드 소스 제거(1428) 이벤트는 노드 소스가 노드로부터 제거될 때 발행된다.

<174> 노드 정렬(1430)

<175> 노드 정렬(1430) 이벤트는 하나 이상의 노드가 정렬될 때 발행된다. 예를 들어, 미디어 타임라인(1400)은 노드들(1402-1412)이 시간순으로, 종속성에 기초하는 등등과 같이 하나 이상의 기준에 따라 정렬되는 기능을 지원할 수 있다. 따라서, 노드 정렬(1430) 이벤트는 노드 및/또는 그 노드(1406)의 자식들(예를 들어, 노드들 1410, 1412)이 정렬될 때 노드(1406)에 의해 개시될 수 있다.

<176> 노드 이동(1432)

<177> 노드 이동(1432) 이벤트는 노드가 이동할 때 발행된다. 예를 들어, 노드(1406)는 미디어 타임라인(1400)에서 이동하여 다른 노드, 예를 들어 노드(1402)의 자식이 될 수 있다. 따라서, 노드 이동(1432) 이벤트는 노드

(1406)가 이동될 때 노드(1406) 및/또는 노드의 부모(예를 들어, 이전 부모 및/또는 새 부모 노드)에 의해 개시될 수 있다.

<178> **판독 전용 미디어 타임라인**

<179> 미디어 타임라인의 작성자는 미디어 타임라인의 모두 또는 일부를 판독 전용으로 표시할 수 있다. 이것은 미디어 타임라인의 기능을 보호하는 데 사용될 수 있다. 제1 시나리오에서, 타임라인의 작성자는 사용자가 광고를 스킵 및/또는 삭제하는 것과 같이 미디어 경험을 변경하는 것을 원하지 않는다. 다른 시나리오에서, 작성자는 미디어 타임라인을 동적으로 변경하기를 원할 수 있지만, 다른 컴포넌트들이 그를 수정하는 것은 원하지 않는다. 또 다른 시나리오에서, 작성자는 다른 컴포넌트들이 타임라인 노드들 상의 커스텀 메타데이터를 설정하는 것을 허가할 수 있지만, 타임라인에 새로운 자식들을 추가하는 것은 허가하지 않을 수 있다.

<180> 미디어 타임라인은 이러한 판독 전용 시나리오들 중 하나 또는 모두에 적합하도록 주문화될 수 있다. 판독 전용 미디어 타임라인은 미디어 타임라인의 판독 전용 랩퍼(read-only wrapper)를 생성함으로써 구현될 수 있다. 판독 전용 랩퍼는 오리지날 타임라인의 구조를 미러링하는, 즉 오리지날 타임라인의 노드들로부터 "복제"되는 노드들을 포함한다. 판독 전용 미디어 타임라인의 복제 노드들은 오리지날 타임라인의 노드들로 후진하는 포인터들을 포함할 수 있다. 또한, 복제 노드들 각각은 오리지날 타임라인의 노드들 상에서 생성되는 이벤트들에 가입하도록 구성될 수 있다. 이것은 오리지날 미디어 타임라인의 트리의 구조에 대한 변경과 같이 오리지날 미디어 타임라인이 변경될 때 복제 타임라인의 구조가 갱신을 유지할 수 있게 한다.

<181> 판독 전용 미디어 타임라인의 복제 노드들은 사용자가 판독 전용 미디어 타임라인에 대해 노드들을 추가/제거하는 것을 허가하는 기능들이 없도록 구성될 수 있다. 판독 전용 타임라인을 생성할 때, 작성자는 또한 복제 노드들에 대한 메타데이터가 수정 가능해야 하는지를 지정할 수 있다. 이러한 설계는, 판독 전용 미디어 타임라인을 실행하는 다른 컴포넌트, 예를 들어 애플리케이션들이 미디어 타임라인 구조에 대해 판독 전용 또는 제한된 액세스를 갖는 반면, 미디어 타임라인의 작성자가 원하는 만큼 미디어 타임라인을 수정하는 것을 가능하게 한다.

<182> 일 구현에서, 도 3의 미디어 타임라인(300)의 루트 노드(302)의 메타데이터(314)는 미디어 타임라인(300)이 사용자에 의해 편집되지 않을 수 있도록 표시될 수 있다. 다른 구현에서, 미디어 타임라인의 특정 노드 및/또는 노드들의 그룹이 판독 전용으로 표시될 수 있다. 예를 들어, 도 3을 다시 참조하면, 리프 노드(308)의 메타데이터(320)는 판독 전용으로 표시될 수 있다. 다른 예에서, 노드(306)의 메타데이터(318)는 판독 전용으로 표시되어, 노드(306), 리프 노드(310) 및 리프 노드(312)가 편집되지 않을 수 있다.

<183> **예시적인 미디어 타임라인 구현**

<184> 전술한 미디어 타임라인들은 하나 이상의 윈도우 미디어 재생기 재생 리스트 파일, 실행 가능 시간 언어(XTL) 파일 등과 같은 타임라인 데이터를 저장하고 복원하는 다양한 방법을 이용할 수 있다.

<185> 예를 들어, 미디어 타임라인은 ASX 파일 확장자에 의해 식별되는 아래의 윈도우 미디어 재생기 재생 리스트 파일에서 기술될 수 있다.

```
<Asx Version = "3.0" >
<Entry>
<Ref href = "file://\\wmp\content\mpeg\Boom.mpe"/>
</Entry>
<Entry>
<Ref href = "\\wmp\content\Formats\MovieFile\chimp.mpg"/>
</Entry>
<Entry>
<Ref href = "file://\\wmp\content\mpeg\Boom.mpe"/>
</Entry>
</Asx>
```

<186>

<187> 이 ASX 파일은 출력을 위한 3개의 파일을 백투백(back to back) 방식으로 지정한다. 이 파일들에 대한 시작 및 중지 시간들은 지정되어 있지 않다. ASX 파일은 시퀀스 노드(1502) 및 3개의 리프 노드(1504, 1506, 1508)를 포함하는 도 15에 도시된 미디어 타임라인(1500)에 의해 표현될 수 있다. 리프 노드들(1504-1508) 각각은 미디어 타임라인(1500)에 의해 출력될 미디어에 대한 각각의 소스(1516, 1518, 1520)를 기술하는 각각의 메타데이터(1510, 1512, 1514)를 포함한다.

<188> 미디어 타임라인의 다른 예가 아래의 XTL 파일에 도시된다.

```
<timeline>
<group type="video">
        <track>
<clip src="V1.wmv" start="0" stop="30" mstart="50" mstop=”80”
/>
<clip src=”V2.wmv” start=”30” stop=”40” mstart=”0” />
        </track>
</group>
<group type="audio">
        <track>
<clip src="A1.asf" start="20" stop="40" mstart="0" />
<clip src=”A2.asf” start=”40” stop=”60” mstart=”0” />
</track>
</group>
</timeline>
```

<189>

<190> 이 XTL 파일은 출력을 위한 미디어의 2개의 트랙, 예를 들어 스트림들을 기술한다. 트랙들 중 하나는 오디오 트랙이고 다른 하나는 비디오 트랙이다.

<191> XTL 파일은 2개의 자식 시퀀스 노드(1604, 1606)를 갖는 병렬 노드(1602)를 포함하는 도 16에 도시된 미디어 타임라인(1600)에 의해 표현될 수 있다. 이 예에서, 시퀀스 노드(1604)는 비디오로 설정된 주요 타입(1608) 필터를 가지며, 시퀀스 노드(1606)는 오디오로서 설정된 주요 타입(1610) 필터를 갖는다. 시퀀스 노드(1604)는 2개의 자식 리프 노드(1612, 1614)를 갖는다. 리프 노드(1612)는 "0"의 시작 시간(1616), "30"의 중지 시간(1618), "50"의 미디어 시작(1620) 및 "80"의 미디어 중지(1622)를 지정하는 메타데이터를 포함한다. 리프 노드(1614)는 "30"의 시작 시간(1624), "40"의 중지 시간(1626) 및 "0"의 미디어 시작(1628)을 지정하는 메타데이터를 포함한다. 리프 노드(1614)는 미디어 중지 시간을 포함하지 않으며, 따라서 리프 노드(1614)에 의해 참조되는 미디어의 전체 길이가 출력될 것이라는 점에 유의해야 한다.

<192> 시퀀스 노드(1606)는 또한 2개의 자식 리프 노드(1630, 1632)를 갖는다. 리프 노드(1630)는 "20"의 시작 시간(1634), "40"의 중지 시간(1636) 및 "0"의 미디어 시작(1638)을 지정하는 메타데이터를 포함한다. 리프 노드(1632)는 "40"의 시작 시간(1640), "60"의 중지 시간(1642) 및 "0"의 미디어 시작(1644)을 지정하는 메타데이터를 포함한다.

<193> 도 17은 제1 및 제2 미디어들 간의 전이 효과를 이용하는 지정된 기간 동안의 제1 및 제2 미디어의 출력(1700)을 나타내는 예시적인 구현의 도면이다. 도시된 예에서, A1.asf(1702) 및 A2.asf(1704)는 2개의 상이한 오디오 파일이다. A1.asf(1702)는 출력 길이 20초를 갖고, A2.asf(1704)도 출력 길이 20초를 갖는다. A1.asf(1720) 및 A2.asf(1704)의 출력들 간에 크로스 페이드(1706) 효과가 정의된다. 즉, A1.asf(1702)의 출력에서 A2.asf(1704)의 출력으로 전이하도록 크로스 페이드(1706)가 정의된다. 크로스 페이드(1706) 효과는 A1.asf(1702)d의 출력의 10초 후에 개시되어 A1.asf(1702)의 출력 종료시에 종료된다. 따라서, A2.asf(1704)의 출력도 10초 후에 개시된다. 크로스 페이드(1706)는 2개의 상이한 미디어, 즉 A1.asf(1702) 및

A2.asf(1704)를 입력하고 원하는 효과를 갖는 단일 출력을 제공하는 것으로 도시되어 있다.

<194> 도 18은 도 17의 크로스 페이드(1706) 효과를 구현하기에 적합한 예시적인 구현에서의 미디어 타임라인(1800)의 도면이다. 미디어 타임라인(1800)은 2개의 자식, 즉 리프 노드(1804, 1806)를 가진 병렬 노드(1802)를 포함한다. 병렬 노드(1802)는 0초의 시작 시간(1808) 및 20초의 중지 시간(1810)을 지정하는 메타데이터를 포함한다. 병렬 노드(1802)는 또한 크로스 페이드를 기술하는 복합 효과(1812)를 포함한다. 리프 노드(1804)는 0초의 시작 시간(1814) 및 20초의 중지 시간(1816)을 지시하는 메타데이터를 포함한다. 리프 노드(1806)는 10초의 시작 시간(1818) 및 30초의 중지 시간(1820)을 갖는 메타데이터를 포함한다.

<195> 리프 노드(1804)는 또한 도 17과 관련하여 설명된 A1.asf(1702)를 참조하는 포인터(1822)를 포함한다. 마찬가지로, 리프 노드(1806)는 도 17과 관련하여 설명된 A2.asf(1704)를 참조하는 포인터(1824)를 포함한다. 따라서, 미디어 타임라인(1800)이 실행될 때, A1.asf 파일(1702) 및 A2.asf 파일(1704)은 도 17에 도시된 바와 같은 효과(1812)를 이용하는 방식으로 출력된다.

<196> **타임라인 정렬**

<197> 타임라인 정렬기(126)는 미디어 타임라인에 의해 표현되는 세그먼트들을 식별하기 위해 하나 이상의 타임라인 정렬 알고리즘을 이용할 수 있다. 예를 들어 복합 타임라인이 주어진 경우, 타임라인 정렬기(126)는 미디어 타임라인을 독립적으로 렌더링될 수 있는 개별 세그먼트들로 "분해"한다. 예를 들어, 타임라인 정렬기(126)는 타임라인 객체(예를 들어, 노드 또는 효과)가 시작하고 중지하는 모든 시점을 결정할 수 있으며, 이것으로부터 렌더링을 위한 새로운 세그먼트를 결정할 수 있다. 따라서, 타임라인 정렬기(126)는 미디어 타임라인(122)을 세그먼트화하는 미디어 파운데이션(204)의 기능을 나타낸다. 이러한 정렬 프로세스의 도해적인 예들이 아래의 도면들에 도시된다.

<198> 도 19는 정렬을 위한 예시적인 타임라인(1900)의 사용자 인터페이스 뷰의 도면이다. 도 19의 타임라인은 편집을 위해 사용자 인터페이스에 나타날 수 있는 것으로 도시되어 있다. 타임라인(1900)은 비디오를 위한 트랙 1 1902(1), 오디오를 위한 트랙 2 1902(2), 및 오디오를 위한 트랙 3 1902(3)을 포함하는 복수의 트랙을 포함한다. 트랙 1 1902(1)은 비디오 효과(1906)가 적용되는 비디오 파일(1904)를 포함한다. 트랙 2 1902(2)는 오디오 파일(1908)을 포함하고, 트랙 3 1902(3)도 오디오 파일(1910)을 포함한다. 오디오 파일들(1908, 1910) 사이에 크로스 페이드 효과(1912)가 적용되는 것으로 도시되어 있다. 사용자 인터페이스 뷰에 의해 어느 세그먼트들이 형성될 것인지를 결정하기 위하여, 타임라인(1900)에서 표현되는 각 타임라인 객체의 시작 및 끝에 "가상" 라인들이 그려질 수 있으며, 그 일례가 아래 도면에 도시된다.

<199> 도 20은 타임라인 정렬기에 의해 도 19의 타임라인(1900)을 복수의 세그먼트로 분할함으로써 형성되는 세그먼트화된 타임라인(2000)을 나타내는 다른 사용자 인터페이스 뷰의 도면이다. 타임라인(2000)은 복수의 점선을 포함하는데, 이들 각각은 타임라인 객체의 시작 및 끝을 나타낸다.

<200> 도시된 바와 같이, 이들 점선은 복수의 세그먼트 2002(1)-2002(6)를 나타낸다. 예를 들어, 세그먼트 2002(1)은 오디오 파일(1908)의 시작에서 시작하여 비디오 파일(1904)의 시작에서 끝난다. 세그먼트 2 2002(2)는 비디오 파일(1904)의 시작에서 시작하여 비디오 효과(1906)의 시작에서 끝난다. 세그먼트 3 2002(3)은 비디오 효과(1906)의 시작에서 시작하여 크로스 페이드 효과(1912)의 시작에서 끝난다. 세그먼트 4 2002(4)는 크로스 페이드 효과(1912)의 시작에서 시작하여 크로스 페이드 효과(1912)의 끝에서 끝난다. 세그먼트 5 2002(5)는 크로스 페이드 효과(1912)의 끝에서 시작하여 비디오 효과(1906)의 끝에서 끝난다. 최종 세그먼트인 세그먼트 6 2002(6)은 오디오 파일(1910) 및 비디오 파일(1904)의 끝에서 끝난다. 따라서, 도 20의 연속 점선들의 각 쌍은 하나의 세그먼트를 정의하는데, 이 세그먼트에서 타임라인 객체들은 그 세그먼트가 변경되지 않는 동안 렌더링된다.

<201> 도 21은 도 19 및 20에 각각 도시된 타임라인들(1900, 2000)을 제공하도록 구성된 객체 모델을 이용하여 표현되는 예시적인 타임라인(2100)의 도면이다. 타임라인(2100)은 병렬 노드(2102) 및 3개의 자식 리프 노드를 포함하는데, 이 리프 노드들은 비디오 파일 리프 노드(2104), 오디오 파일 리프 노드(2106) 및 또 하나의 오디오 파일 리프 노드(2108)로서 도시된다.

<202> 비디오 파일 리프 노드(2104)는 포인터(2114)에 의해 참조되는 비디오 파일(즉, 비디오 파일 1904)에 대한 "7"의 시작 시간(2110) 및 "58"의 중지 시간(2112)를 지정한다. 비디오 파일 리프 노드(2104)는 또한 "17"의 시작 시간(2118) 및 "39"의 중지 시간(2120)을 갖는 비디오 효과(2116)를 지정한다.

<203> 오디오 파일 리프 노드(2106)는 포인터(2126)에 의해 참조되는 오디오 파일(즉, 오디오 파일 1908)에 대한 "0"

의 시작 시간(2122) 및 "33"의 중지 시간(2124)를 지정한다. 오디오 파일 리프 노드(2108)는 포인터(2132)에 의해 참조되는 오디오 파일(즉, 오디오 파일 1910)에 대한 "20"의 시작 시간(2128) 및 "58"의 중지 시간(2130)을 지정한다. 병렬 노드(2102)는 오디오 파일 리프 노드들(2106, 2108)에 적용될 "20"의 시작 시간(2136) 및 "33"의 중지 시간(2138)을 갖는 크로스 페이드 효과(2134)를 지정한다. 타임라인(2100)을 정렬하기 위한 다양한 기술이 이용될 수 있으며, 그에 대한 추가 설명은 아래의 도면들과 관련하여 발견될 수 있다.

<204> **예시적인 프로시저**

<205> 다음은 전술한 시스템 및 장치를 이용하여 구현될 수 있는 정렬 기술을 설명한다. 프로시저들 각각의 양태들은 하드웨어, 펌웨어 또는 소프트웨어, 또는 이들의 조합으로 구현될 수 있다. 프로시저들은 하나 이상의 장치에 의해 수행되는 동작들을 지정하는 한 세트의 블록들로 도시되며, 각각의 블록에 의한 동작을 수행하도록 도시된 순서로 제한될 필요는 없다. 아래의 설명 중 일부에서는 도 1-21의 환경 및 시스템을 참조한다.

<206> 도 22는 미디어 타임라인이 타임라인 정렬기에 의해 렌더링을 위한 복수의 세그먼트로 정렬되는 예시적인 구현에서의 프로시저(2200)를 나타내는 흐름도이다. 미디어 타임라인이 수신된다(블록 2202). 예를 들어, 미디어 타임라인은 애플리케이션 프로그래밍 인터페이스에서 애플리케이션으로부터 수신될 수 있다.

<207> 타임라인 정렬기가 실행되어 미디어 타임라인에 포함된 타임라인 객체들을 검사한다(블록 2204). 예를 들어, 타임라인 정렬기는 노드들(예를 들어, 리프 노드, 병렬 노드, 시퀀스 노드) 및 효과들, 예를 들어 크로스 페이드 등과 같은 태스크의 수행을 위한 지속 기간을 지정하는 메타데이터를 위해 타임라인 객체들을 검사할 수 있다.

<208> 타임라인 정렬기는 메타데이터에 의해 지정되는 바와 같은 타임라인 객체에 대한 시작 시간 및 중지 시간을 취득한다(블록 2206). 이어서, 타임라인 정렬기는 시작 시간에 대한 입력 및 중지 시간에 대한 다른 입력을 어레이로 만든다(블록 2208). 이어서, 다른 객체가 이용 가능한지에 대한 판정이 이루어진다(판정 블록 2210). 그러한 경우(판정 블록 2210에서 "예"인 경우), 다른 타임라인 객체가 검사되고(블록 2204), 시작 및 중지 시간이 취득되고(블록 2206), 또한 어레이에 입력된다(블록 2208).

<209> 타임라인 객체들 각각이 처리된 경우(판정 블록 2210에서 "아니오"), 어레이 내의 입력들은 시간순으로 정렬된다(블록 2212). 예를 들어, 입력들은 시간에 기초하여 오름차순으로 배열될 수 있다. 이어서, 정렬된 어레이는 각각의 "시작" 항목을 수집하고 각각의 "중지" 항목을 버리기 위해 "워킹(walking)"되어, 시간을 구성하고 타임라인 객체들이 당해 세그먼트 동안 이용되는 세그먼트들에 도달한다.

<210> 예를 들어, 도 18-21의 이전 예에 계속하여, 세그먼트들을 "이해(figure out)"하기 위하여, 타임라인 정렬기는 도 21의 타임라인(2100)의 각각의 노드 및 효과를 어레이 안에 입력한다. 어레이는 "정렬기 요소 어레이"로서 지칭될 수 있다. 타임라인(2100) 내의 각 노드 및 효과에 대해, 어레이 내로 2개의 입력이 이루어진다. 타임라인 객체의 제1 입력은 객체의 시작 시간에 대한 것이고, 타임라인 객체의 제2 입력은 객체의 중지 시간에 대한 것이다. 미디어 타임라인(2100)으로부터 정렬기 요소 어레이를 채운 후, 어레이는 요소들이 시간에 기초하여 오름차순으로 배열되도록 정렬된다.

<211> 도 23은 도 22의 프로시저(2200)를 이용하여 타임라인(2100)으로부터 형성된 예시적인 정렬기 요소 어레이(2300)의 도면이다. 정렬기 요소 어레이(2300)는 복수의 요소(2302-2318)를 갖는데, 이들 중 2개는 도 21의 미디어 타임라인(2100)의 리프 노드들 및 요소들 각각에 대한 것이다. 정렬기 요소 어레이(2300)로부터 세그먼트를 취득하기 위하여, 타임라인 정렬기는 어레이를 "워킹"하여, 미디어 타임라인(2100)으로부터 "0", "7", "17", "20", "33", "39" 및 "58"과 같은 상이한 시간 값들을 발견한다. 이들 값은 세그먼트들 2002(1)-2002(6)을 설명하는 데 이용된 도 20의 점선들의 값들에 대응한다는 점에 유의해야 한다. 따라서, 각 세그먼트는 어레이에서 취득된 연속적인 상이한 값들 사이에 정의되는데, 이것은 도 23에서 세그먼트 1 2002(1), 세그먼트 2 2002(2), 세그먼트 3 2002(3), 세그먼트 4 2002(4), 세그먼트 5 2002(5) 및 세그먼트 6 2002(6)을 정의하는 괄호로 도시된다.

<212> 타임라인 정렬기는 시간순으로 정렬된 어레이(2300)를 워킹하여 시작 입력에 의해 참조되는 각 항목을 수집하고 중지 입력을 가진 각 항목을 버린다. 예를 들어, 세그먼트 1 2002(1)은 "0"초와 "7"초 사이에 정의된다. 이 세그먼트 동안 어느 타임라인 객체가 이용되는지를 결정하기 위하여, 타임라인 정렬기는 A1(즉, 오디오 파일 1908)에 대한 입력(2302)에 주목한다. 이어서, 타임라인 정렬기는 다음 입력(2304)을 검사하여 또 하나의 시작 시간을 발견하고 검사를 중지한다. 따라서, 세그먼트 1 2002(1)은 타임라인 정렬기에 의해 결정된 바와 같이 7초 동안 재생되는 오디오 파일(1908)만을 포함한다.

<213> 유사한 기술을 이용하여, 타임라인 정렬기(126)는 미디어 타임라인(2100)의 임의의 특정 시점에 어느 타임라인 객체가 이용되는지를 결정할 수 있다. 예를 들어, 어느 타임라인 객체가 39초와 58초 사이의 세그먼트 6 2002(6)에 적용될 수 있는지를 결정하기 위하여, 타임라인 정렬기 모듈은 다음과 같이 어레이(2300)를 트래버스 할 수 있다.

<214> 오디오 파일(A1) 1908, 시작 -> [A1]

<215> 비디오 파일(V1) 1904, 시작 -> [A1,V1]

<216> 비디오 효과(E1) 1906, 시작 -> [A1,V1,E1]

<217> 오디오 파일(A2) 1910, 시작 -> [A1,V1,E1]

<218> 크로스 페이드 효과(E2) 1912, 시작 -> [A1,V1,E1,A2,E2]

<219> 오디오 파일(A1) 1908, 중지 -> [V1,E1,A2,E2]

<220> 크로스 페이드 효과(E2) 1912, 중지 -> [V1,E1,A2]

<221> 비디오 효과(E1) 1906, 중지 -> [V1,A2]

<222> 비디오 파일(V1) 1904, 중지 -> [A2]

<223> 오디오 파일(A2) 1910, 중지 -> []

<224> 어레이를 트래버스하여 특정 시간에 어느 타임라인 객체가 렌더링되는지를 발견하기 위하여, 타임라인 정렬기는 요청된 시간보다 큰 시간 표시를 가진 요소에 도달할 때까지 어레이를 통해 계속한다. 예를 들어, 32초에 어느 타임라인 객체가 이용되는지를 결정하기 위하여, 타임라인 정렬기는 "33"초의 대응하는 표시를 갖는 입력(231 0)에 도달할 때까지 계속할 수 있다. 따라서, 요소들(2310, 2312)은 "32"초에 어느 타임라인 객체가 필요한지에 대한 결정에 이용되지 않는다. 이러한 방식으로, 동일한 시간 표시를 갖는 다수의 타임라인 객체가 렌더링을 위해 수집될 수 있다.

<225> 타임라인 정렬기는 또한 타임라인에서 "특수" 사례들의 표현을 위해 다양한 로직을 이용할 수 있다. 예를 들어, 도 7과 관련하여 전술한 바와 같이, 타임라인 객체 모델은 노드에 대한 메타데이터가 그 노드가 몇 회 루프 처리되는지, 즉 렌더링되는지를 지시하는 특징(예를 들어, 루프 카운트 726)을 지원할 수 있다. 이러한 사례를 설명하기 위하여, 타임라인 정렬기는 어레이 내에 동일 노드에 대한 다수의 입력을 행할 수 있으며, 시간을 적절히 조정할 수 있다. 그러나, 몇몇 사례에서, 미디어 타임라인은 시작 또는 중지 시간의 표시를 포함하지 않을 수 있으며, 따라서 타임라인 정렬기는 타임라인을 정렬하기 위한 추가 기능을 적용하게 되는데, 그에 대한 추가 설명은 아래의 도면들과 관련하여 발견될 수 있다.

<226> 도 24는 지정된 시작 및 중지 시간을 갖지 않는 순차 재생 리스트를 제공하도록 구성되는 시퀀스 노드의 도면이다. 도 24에는, 도 4 및 10과 각각 관련하여 설명된 시퀀스 노드들(402, 1102)과 유사한 시퀀스 노드(2402)가 도시되어 있다. 시퀀스 노드(2402)는 시퀀스 노드(2402)의 자식들인 복수의 리프 노드(2404, 2406, 2408)를 갖는다. 복수의 리프 노드(2404-2408) 각각은 각각의 미디어(2426-2430)를 참조하는 각각의 포인터(2420-2424)를 갖는 각각의 메타데이터(2414-2418)를 포함한다.

<227> 시퀀스 노드(2402)는 노드들 상에서 시작 또는 중지 시간이 지정되지 않은 재생 리스트에 대해 도시된다. 오히려, 복수의 리프 노드(2404-2408) 각각은 하나씩 연속적으로 렌더링된다. 따라서, 이들 복수의 노드(2404-2408)를 정렬하기 위하여, 타임라인 정렬기(126)는 "무한 시간"에 대한 구조를 이용하여, 노드가 완료시까지 렌더링된 후 다음 노드가 순차적으로 렌더링됨을 지시하는데, 이에 대한 추가 설명은 아래의 도면과 관련하여 발견될 수 있다.

<228> 도 25는 시작 또는 중지 시간 표시를 갖지 않은 타임라인 객체들을 포함하는 미디어 타임라인이 정렬되는 예시적인 구현에서의 프로시저(2500)를 나타내는 흐름도이다. 미디어 타임라인이 애플리케이션 프로그래밍 인터페이스를 통해 수신된다(블록 2502).

<229> 이어서, 타임라인 정렬기(126)는 수신된 미디어 타임라인으로부터 타임라인 객체(예를 들어, 노드 또는 효과)를 검사한다(블록 2504). 이어서, 타임라인 객체에 대한 시작 및 중지 시간이 지정되어 있는지에 대한 판정이 이루어진다(판정 블록 2506). 그러한 경우(판정 블록 2506에서 "예"), 시작 시간 및 중지 시간이 어레이에 입력된다(블록 2508). 따라서, 시작 및 중지 시간이 이용 가능한 경우, 이들 시간은 도 22와 관련하여 전술한 바와

같이 어레이 내에 입력될 수 있다.

<230> 타임라인 객체에 대해 시작/중지 시간이 지정되지 않은 경우(판정 블록 2506에서 "아니오"), 적용이 가능할 경우에 임의의 이전 "무한" 시간을 해결하는 "무한" 시간에 대한 구조가 사용된다(블록 2512). 이 구조는 단일 변수와 같은 다양한 구성을 취할 수 있다. 예를 들어, 도 24의 미디어 타임라인을 다시 참조하면, 리프 노드(2404)는 타임라인 정렬기에 의해 "0"의 시작 시간 및 "x"의 중지 시간을 부여 받을 수 있다. 리프 노드(2406)는 "x"의 시작 시간 및 "2x"의 중지 시간을 부여 받을 수 있고, 리프 노드(2408)은 "2x"의 시작 시간 및 "3x"의 중지 시간을 부여 받을 수 있다. 이 예에서, 값 "x"는 리프 노드들(2404, 2406, 2408) 사이에서 반드시 동일한 양의 시간을 나타내지는 않는다는 점에 유의해야 한다. 예를 들어, 리프 노드(2404)는 리프 노드(2406) 및 리프 노드(2408)와 다른 지속 기간을 가질 수 있다. 따라서, 이 예에서 변수는 대응하는 리프 노드가 렌더링이 완료될 때까지 렌더링될 것이라는 것을 나타낸다. 각각의 노드가 완료시까지 렌더링될 것임을 지시하는 "무한" 시간을 지정함으로써, 타임라인 정렬기는 리프 노드들(2404-2408) 서로의 비교를 통해 전술한 바와 같이 시퀀스 노드(2402)를 어레이로 정렬할 수 있다.

<231> 도 26은 타임라인 정렬기에 의해 정렬된 바와 같은 도 24의 미디어 타임라인(2400)으로부터 형성된 예시적인 어레이(2600)의 도면이다. 어레이(2600)는 복수의 입력(2602-2612)을 포함하는데, 이들 중 2개는 도 24의 리프 노드들(2404-2408) 각각에 대해 형성된다. 도 22 및 23과 관련하여 전술한 바와 같이, "무한" 시간의 구조를 이용함으로써, 시작 및 중지 시간을 지정하지 않은 리프 노드들(2404-2408)이 복수의 세그먼트 2614(1)-2614(3)로 여전히 정렬될 수 있다.

<232> 그러나, 몇몇 예에서는, 노드들에 대해 시작 및 중지 시간이 지정되지 않는 경우에도, 노드들에 대한 시작 및 중지 시간이 판정된다. 예를 들어, 도 24를 다시 참조하면, 시퀀스 노드(2402)는 각각의 리프 노드들(2404-2408)에 의해 참조되는 미디어(2426-2430)의 출력 사이에 사용되는 효과(2412)를 기술하는 메타데이터(2410)를 포함한다. 따라서, 효과(2412)는 시퀀스 노드(2402)의 자식들로부터 유래하는 미디어(2426-2430)에 적용된다.

<233> 효과(2412)는 지속 기간(2432)을 갖는다. 지속 기간(2432)은 시퀀스 내의 둘 이상의 노드 사이에 요구되는 중복의 양을 지정하는 데 사용될 수 있다. 예를 들어, 시퀀스 내의 제2 입력, 즉 미디어(2426)는 효과(2412)의 지속 기간(2432) 동안 중복되도록 출력될 수 있다. 따라서, 시퀀스 노드(2402)의 출력 지속 기간은 리프 노드들(2404-2408) 상에 지정된 시간들 및 효과(2412)의 지속 기간(2432)에 의해 지정되는 중복의 함수가 된다. 그러나, 이 예에서 리프 노드들(2404-2408)은 시작 및 중지 시간을 지정하지 않으므로, 타임라인 정렬기는 이들 시작 및 중지 시간을 결정하여, 효과(2412)를 언제 적용할지를 결정한다.

<234> 예를 들어, 타임라인 정렬기는 먼저 정렬기 어레이를 이용하여 세그먼트를 구축한 후, 세그먼트가 유효한지 무효인지, 예를 들어 세그먼트가 무한 지속 기간, 무한 오프셋 중지 위치 등을 가진 둘 이상의 소스를 포함하는지를 결정하기 위해 검사할 수 있다. 세그먼트가 무효인 경우, 타임라인 정렬기는 지속 기간을 계산하고 어레이를 재구축할 수 있는데, 그 일례가 아래에 보여진다.

<235> 0<1<2...<N<N+1<무한대-K-1<무한대-K<무한대-K+1<2무한대-K-1<

<236> 위의 식에서, 임의의 수 "N"에 대해 다음이 참으로 유지된다.

<237> -N<무한대

<238> 또한, 위의 식에서, 임의의 수 K에 대해 다음이 참으로 유지된다.

<239> 무한대-K-1<무한대-K 및

<240> 무한대+K<2무한대

<241> 예를 들어, 타임라인 정렬기는 도 26의 어레이를 구성할 때 효과(2412)를 언제 렌더링할지를 결정하기 위해 다른 노드의 시작 또는 중지 시간에 종속하는 효과(예를 들어, 효과 2412)를 만날 수 있다. 예를 들어, 효과(2412)는 효과(2412)가 리프 노드(2404)의 렌더링의 종료 전에 적용될 지속 기간(2432)을 갖는 것으로 지정할 수 있다. 따라서, 타임라인 정렬기는 리프 노드(2404)의 미디어 소스에 조회하여 효과(2412)의 시작 및 중지 시간을 결정하고, 시작 및 중지 시간을 어레이에 적용한 후, 타임라인을 재정렬하여 세그먼트들에 대해 임의의 변경이 행해졌는지를 결정할 수 있는데, 이에 대한 추가 설명은 아래의 도면과 관련하여 발견될 수 있다. 타임라인 정렬기는 다음의 경우에, 즉 (1) 세그먼트가 "무한"(즉, 미지정) 지속 기간을 가진 둘 이상의 소스를 포함할 때, 또는 (2) 세그먼트가 오프셋(예를 들어, 효과에 대해 "x-오프셋")으로서 지정된 시간을 가진 소스를 포함할 때 소스 노드들에 대한 지속 기간을 계산할 수 있다. 세그먼트가 무한 지속 기간을 가진 하나의 소스를

포함하는 경우, 타임라인은 지속 기간을 계산하지 않고 파일을 재생할 수 있다.

<242> 도 27은 타임라인 정렬기가 각각의 메타데이터에서 시작/중지 시간이 지정되지 않는 어레이 내의 타임라인 객체들에 대하여 미디어 소스들로부터 시작/중지 시간을 결정하는 예시적인 구현에서의 프로시저(2700)를 나타내는 흐름도이다. 미디어 타임라인이 세그먼트에 도달하도록 정렬된다(블록 2702). 예를 들어, 미디어 타임라인은 도 25의 프로시저(2500)에 따라 정렬될 수 있으며, 따라서 미디어 타임라인은 복수의 "무한" 시간 구조를 포함한다.

<243> 타임라인 정렬기는 세그먼트에서 지정된 각 타임라인 객체에 대한 미디어 소스를 취득하고(블록 2704), 미디어 소스로부터 시작/중지 시간을 취득한다(블록 2706). 예를 들어, 타임라인 정렬기는 지속 기간, 특정 시작/중지 시간을 기술하는 메타데이터 등으로부터 상대적인 시작/중지 시간을 결정할 수 있다. 이어서, 타임라인 정렬기는 다른 세그먼트에 도달하도록 입력들을 재정렬한다(블록 2708). (블록 2702의) 세그먼트와 (블록 2708의) 세그먼트를 비교하여, 이들이 동일한지, 즉 동일한 타임라인 객체들을 포함하는지를 판정한다(판정 블록 2710). 세그먼트들이 동일하지 않은 경우(판정 블록 2710에서 "아니오"), 세그먼트들이 동일할 때까지 프로시저(2700)의 일부(블록 2704-2710)가 반복된다. 세그먼트들이 동일한 경우(블록 2710에서 "예"), 세그먼트가 렌더링된다(블록 2712). 이러한 방식으로, 미디어 타임라인이 정렬되는 방법에 영향을 미칠 수 있는 미디어 타임라인에 대하여 이루어지는 변경들과 같은 미디어 타임라인에 대한 변경 및 갱신이 처리될 수 있다. 또한, 이러한 기술은 렌더링 동안 미디어 타임라인에 대해 이루어지는 임의의 갱신을 처리할 수 있다.

<244> **예시적인 운영 환경**

<245> 본 명세서에서 설명되는 다양한 컴포넌트 및 기능은 다양한 개별 컴퓨터를 이용하여 구현된다. 도 28은 참조번호 2802로 참조되는 컴퓨터를 포함하는 컴퓨터 환경(2800)의 대표적인 예의 컴포넌트들을 나타낸다. 컴퓨터(2802)는 도 1의 컴퓨터(102)와 같거나 다를 수 있다. 도 28에 도시된 컴포넌트는 단지 예일 뿐, 본 발명의 기능의 범위에 관해 어떠한 제한을 암시하고자 하는 것이 아니며, 본 발명은 도 28에 도시된 특징들에 반드시 종속하는 것은 아니다.

<246> 일반적으로, 많은 기타 범용 또는 특수 목적의 컴퓨팅 시스템 구성이 이용될 수 있다. 본 발명에 사용하기에 적합하고 잘 알려진 컴퓨팅 시스템, 환경 및/또는 구성의 예로는 퍼스널 컴퓨터, 서버 컴퓨터, 핸드-헬드 또는 랩톱 장치, 멀티프로세서 시스템, 마이크로프로세서 기반 시스템, 셋톱 박스, 프로그램가능한 가전제품, 네트워크 PC, 네트워크-레디 장치, 미니컴퓨터, 메인프레임 컴퓨터, 상기 시스템들이나 장치들 등 중의 임의의 것을 포함하는 분산 컴퓨팅 환경 기타 등등이 있지만 이에 제한되는 것은 아니다.

<247> 컴퓨터의 기능은 많은 경우에 컴퓨터에 의해 실행되는 소프트웨어 컴포넌트와 같은 컴퓨터 실행 가능 명령에 의해 구현된다. 일반적으로, 소프트웨어 컴포넌트는 특정 태스크를 수행하거나 특정 추상 데이터 유형을 구현하는 루틴, 프로그램, 개체, 컴포넌트, 데이터 구조 등을 포함한다. 또한 통신 네트워크를 통해 연결되어 있는 원격 처리 장치들에 의해 태스크가 수행될 수도 있다. 분산 컴퓨팅 환경에서, 소프트웨어 컴포넌트는 로컬 및 원격 컴퓨터 저장 매체 둘 다에 위치할 수 있다.

<248> 명령 및/또는 소프트웨어 컴포넌트는 컴퓨터의 일부이거나 컴퓨터에 의해 판독될 수 있는 다양한 컴퓨터 판독가능 매체에 상이한 시간들에 저장된다. 프로그램들은 일반적으로 예를 들어 플로피 디스크, CD-ROM, DVD, 또는 피변조 신호(modulated signal)와 같은 소정 형태의 통신 매체 상에 분산된다. 그곳들로부터 프로그램들은 컴퓨터의 부 메모리에 인스톨 또는 로딩된다. 실행시, 프로그램들은 적어도 부분적으로 컴퓨터의 주 전자 메모리에 로딩된다.

<249> 설명의 목적으로, 운영 체제와 같은 프로그램 및 기타 실행 가능 프로그램 컴포넌트는, 이들이 다양한 시간에 컴퓨터의 상이한 저장 컴포넌트들에 위치하고 컴퓨터의 데이터 프로세서(들)에 의해 실행된다는 것이 인식되지만, 본 명세서에서는 개별 블록으로 예시된다.

<250> 도 28을 참조하면, 컴퓨터(2802)의 컴포넌트는 처리 장치(2804), 시스템 메모리(2806), 및 시스템 메모리를 포함하는 각종 시스템 컴포넌트를 처리 장치(2804)에 결합하는 시스템 버스(2808)를 포함하지만 이에 제한되는 것은 아니다. 시스템 버스(2808)는 메모리 버스 또는 메모리 컨트롤러, 주변 버스 및 각종 버스 아키텍처 중 임의의 것을 이용하는 로컬 버스를 포함하는 몇몇 유형의 버스 구조 중 어느 것이라도 될 수 있다. 예로서, 이러한 아키텍처는 ISA(industry standard architecture) 버스, MCA(micro channel architecture) 버스, EISA(Enhanced ISA) 버스, VESA(video electronics standard association) 로컬 버스 그리고 메자닌 버스(mezzanine bus)로도 알려진 PCI(peripheral component interconnect) 버스 등을 포함하지만 이에 제한되는 것

은 아니다.

<251> 컴퓨터(2802)는 통상적으로 각종 컴퓨터 판독가능 매체를 포함한다. 컴퓨터(2802)에 의해 액세스 가능한 매체는 그 어떤 것이든지 컴퓨터 판독 가능 매체가 될 수 있고, 이러한 컴퓨터 판독가능 매체는 휘발성 및 비휘발성 매체, 이동식 및 비이동식 매체를 포함한다. 예로서, 컴퓨터 판독가능 매체는 컴퓨터 저장 매체 및 통신 매체를 포함하지만 이에 제한되는 것은 아니다. 컴퓨터 저장 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터와 같은 정보를 저장하는 임의의 방법 또는 기술로 구현되는 휘발성 및 비휘발성, 이동식 및 비이동식 매체를 포함한다. 컴퓨터 저장 매체는 RAM, ROM, EEPROM, 플래시 메모리 또는 기타 메모리 기술, CD-ROM, DVD(digital versatile disk) 또는 기타 광 디스크 저장 장치, 자기 카세트, 자기 테이프, 자기 디스크 저장 장치 또는 기타 자기 저장 장치, 또는 컴퓨터(2802)에 의해 액세스되고 원하는 정보를 저장할 수 있는 임의의 기타 매체를 포함하지만 이에 제한되는 것은 아니다. 통신 매체는 통상적으로 반송파(carrier wave) 또는 기타 전송 메커니즘(transport mechanism)과 같은 피변조 데이터 신호(modulated data signal)에 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터 등을 구현하고 모든 정보 전달 매체를 포함한다. "피변조 데이터 신호"라는 용어는, 신호내에 정보를 인코딩하도록 그 신호의 특성들 중 하나 이상을 설정 또는 변경시킨 신호를 의미한다. 예로서, 통신 매체는 유선 네트워크 또는 직접 배선 접속과 같은 유선 매체, 그리고 음향, RF, 적외선, 기타 무선 매체와 같은 무선 매체를 포함한다. 상술된 매체들의 모든 조합이 또한 컴퓨터 판독가능 매체의 영역 안에 포함되는 것으로 한다.

<252> 시스템 메모리(2806)는 판독 전용 메모리(ROM)(2810) 및 랜덤 액세스 메모리(RAM)(2812)와 같은 휘발성 및/또는 비휘발성 메모리 형태의 컴퓨터 저장 매체를 포함한다. 시동 중과 같은 때에 컴퓨터(2802) 내의 구성요소들 사이의 정보 전송을 돕는 기본 루틴을 포함하는 기본 입/출력 시스템(BIOS)(2814)은 통상적으로 ROM(2810)에 저장되어 있다. RAM(2812)은 통상적으로 처리 장치(2804)가 즉시 액세스 할 수 있고 및/또는 현재 처리 장치(2804)에 의해 동작되고 있는 데이터 및/또는 소프트웨어 컴포넌트를 포함한다. 예로서, 도 28은 운영 체제(2816), 애플리케이션 프로그램(2818), 소프트웨어 컴포넌트(2820) 및 프로그램 데이터(2822)를 도시하고 있지만 이에 제한되는 것은 아니다.

<253> 컴퓨터(2802)는 또한 기타 이동식/비이동식, 휘발성/비휘발성 컴퓨터 저장 매체를 포함한다. 단지 예로서, 도 28은 비이동식·비휘발성 자기 매체에 기록을 하거나 그로부터 판독을 하는 하드 디스크 드라이브(2824), 이동식, 비휘발성 자기 디스크(2828)에 기록을 하거나 그로부터 판독을 하는 자기 디스크 드라이브(2826), CD-ROM 또는 기타 광 매체 등의 이동식·비휘발성 광 디스크(2832)에 기록을 하거나 그로부터 판독을 하는 광 디스크 드라이브(2830)를 포함한다. 예시적인 운영 환경에서 사용될 수 있는 기타 이동식/비이동식, 휘발성/비휘발성 컴퓨터 기억 매체로는 자기 테이프 카세트, 플래시 메모리 카드, DVD, 디지털 비디오 테이프, 고상(solid state) RAM, 고상 ROM 등이 있지만 이에 제한되는 것은 아니다. 하드 디스크 드라이브(2824)는 통상적으로 데이터 매체 인터페이스(2834)와 같은 비이동식 메모리 인터페이스를 통해 시스템 버스(2808)에 접속되고, 자기 디스크 드라이브(2826) 및 광 디스크 드라이브(2830)는 통상적으로 이동식 메모리 인터페이스에 의해 시스템 버스(2808)에 접속된다.

<254> 위에서 설명되고 도 28에 도시된 드라이브들 및 이들과 관련된 컴퓨터 저장 미디어는, 컴퓨터를(2802)를 위해 컴퓨터 판독가능 명령어, 데이터 구조, 소프트웨어 컴포넌트 및 기타 데이터를 저장한다. 도 28에서, 예를 들어, 하드 디스크 드라이브(2824)는 운영 체제(2816'), 애플리케이션 프로그램(2818'), 소프트웨어 컴포넌트(2820') 및 프로그램 데이터(2822')를 저장하는 것으로 도시되어 있다. 여기서 주의할 점은 이 컴포넌트들이 운영 체제(2816), 애플리케이션 프로그램(2818), 소프트웨어 컴포넌트(2820) 및 프로그램 데이터(2822)와 동일하거나 그와 다를 수 있다는 것이다. 이에 관해, 운영 체제(2816'), 애플리케이션 프로그램(2818'), 소프트웨어 컴포넌트(2820') 및 프로그램 데이터(2822')에 다른 번호가 부여되어 있다는 것은 적어도 이들이 다른 사본(copy)이라는 것을 나타내기 위한 것이다. 사용자는 키보드(2835) 및 마우스, 트랙볼(trackball) 또는 터치 패드로서 일반적으로 지칭되는 포인팅 장치(도시되지 않음) 등의 입력 장치를 통해 명령 및 정보를 컴퓨터(2802)에 입력할 수 있다. 다른 입력 장치로는 소스 주변 장치(스트리밍 데이터를 제공하는 마이크(2838) 또는 카메라(2840) 등), 조이스틱, 게임 패드, 위성 안테나, 스캐너 등을 포함할 수 있다. 이들 및 기타 입력 장치는 종종 시스템 버스에 결합된 입출력(I/O) 인터페이스(2842)를 통해 처리 장치(2804)에 접속되지만, 병렬 포트, 게임 포트 또는 USB(universal serial bus) 등의 다른 인터페이스 및 버스 구조에 의해 접속될 수도 있다. 모니터(2844) 또는 다른 유형의 디스플레이 장치도 비디오 어댑터(2846) 등의 인터페이스를 통해 시스템 버스(2808)에 접속된다. 모니터(2844) 외에, 컴퓨터는 다른 주변 렌더링 장치(예를 들어, 스피커) 및 하나 이상의 프린터를 포함할 수 있고, 이들은 I/O 인터페이스(2842)를 통해 접속될 수 있다.

<255> 컴퓨터는 원격 장치(2850)와 같은 하나 이상의 원격 컴퓨터로의 논리적 접속을 사용하여 네트워크화된 환경에서 동작할 수 있다. 원격 컴퓨터(2850)는 퍼스널 컴퓨터, 네트워크-레디 장치, 서버, 라우터, 네트워크 PC, 피어 장치 또는 다른 공통 네트워크 노드일 수 있고, 통상적으로 컴퓨터(2802)와 관련하여 상술된 구성요소의 대부분 또는 그 전부를 포함한다. 도 28에 도시된 논리적 접속으로는 LAN(2852) 및 WAN(2854)이 포함된다. 도 28에 도시된 WAN(2854)은 인터넷이지만, WAN(2854)은 또한 다른 네트워크일 수도 있다. 이러한 네트워킹 환경은 사무실, 전사적 컴퓨터 네트워크(enterprise-wide computer), 인트라넷 등에서 일반적인 것이다.

<256> LAN 네트워킹 환경에서 사용될 때, 컴퓨터(2802)는 네트워크 인터페이스 또는 어댑터(2856)를 통해 LAN(2852)에 접속된다. WAN 네트워킹 환경에서 사용될 때, 컴퓨터(2802)는 통상적으로 인터넷(2854) 상에서의 통신을 설정하기 위한 모뎀(2858) 또는 기타 수단을 포함한다. 내장형 또는 외장형일 수 있는 모뎀(2858)은 입출력 인터페이스(2842) 또는 기타 적절한 메커니즘을 통해 시스템 버스(2808)에 접속될 수 있다. 네트워크화된 환경에서, 컴퓨터(2802) 또는 그의 일부와 관련하여 기술된 프로그램 모듈은 원격 장치(2850)에 저장될 수 있다. 예로서, 도 28은 원격 소프트웨어 컴포넌트(2860)가 원격 장치(2850)에 있는 것을 도시하고 있지만 이에 제한되는 것은 아니다. 도시된 네트워크 접속은 예시적인 것이며 컴퓨터들 사이에 통신 링크를 설정하는 기타 수단이 사용될 수 있다는 것을 이해할 것이다.

<257> **결론**

<258> 본 발명은 구조적 특징 및/또는 방법론적 동작들에 고유한 언어로 설명되었지만, 첨부된 청구항들에 정의된 본 발명은 설명된 특정 특징 또는 동작으로 한정되는 것은 아니다. 오히려, 특정 특징 및 동작은 청구된 발명을 구현하는 예시적인 형태로서 개시되는 것이다.

도면의 간단한 설명

<10> 도 1은 컴퓨터가 복수의 미디어에 대한 액세스를 제공하는 예시적인 구현에서의 환경을 나타내는 도면.

<11> 도 2는 소프트웨어로 구현되는 시스템이 복수의 미디어의 제시를 제어하기 위해 미디어 파운데이션과 상호작용하는 애플리케이션을 포함하는 예시적인 구현에서의 시스템의 하이 레벨 블록도.

<12> 도 3은 미디어 타임라인이 제시를 위해 미디어의 출력을 제공하는 복수의 노드를 포함하는 트리로서 도시되는 예시적인 구현의 도면.

<13> 도 4는 시퀀스 노드, 및 시퀀스 노드의 자식인 복수의 리프 노드를 나타내는 예시적인 구현의 도면.

<14> 도 5는 시퀀스 노드 및 시퀀스 노드의 자식인 복수의 노드가 복수의 노드 각각의 실행을 위한 타이밍 정보를 지정하는 메타데이터를 포함하는 예시적인 구현의 도면.

<15> 도 6은 병렬 노드 및 병렬 노드의 자식인 복수의 리프 노드를 나타내는 예시적인 구현의 도면.

<16> 도 7은 노드에 포함될 수 있는 메타데이터의 예를 나타내는 예시적인 구현에서의 노드의 도면.

<17> 도 8은 복수의 간단한 효과를 지정하는 노드에 포함되는 메타데이터를 나타내는 예시적인 구현에서의 노드의 도면.

<18> 도 9는 둘 이상의 자식 노드의 출력에 복합 효과를 제공하는 병렬 노드를 나타내는 예시적인 구현의 도면.

<19> 도 10은 전이 효과가 선행 노드에 의해 참조되는 미디어의 출력과 후속 노드에 의해 참조되는 미디어의 출력 간의 효과를 제공하도록 지정되는 예시적인 구현의 도면.

<20> 도 11은 복수의 효과 메타데이터를 포함하는 메타데이터를 갖는 노드를 나타내는 예시적인 구현의 도면.

<21> 도 12는 노드들에 포함되는 메타데이터에 기초하여 미디어 타임라인의 노드들이 로딩되는 동적 로딩의 예시적인 구현에서의 미디어 타임라인의 도면.

<22> 도 13은 미디어 타임라인의 노드들이 노드 소스에 의해 필요에 따라 정의되고 구현되는 동적 로딩의 예시적인 구현에서의 미디어 타임라인의 도면.

<23> 도 14는 미디어 타임라인에 발생하는 변경이 변경에 의해 영향을 받을 수 있는 노드들에 통신될 수 있도록 노드에 의해 이벤트들이 제공되는 예시적인 구현에서의 미디어 타임라인의 도면.

<24> 도 15는 ASX 파일 확장자에 의해 식별되는 윈도우 미디어 재생기 재생 리스트 파일에 의해 기술되는 하나의 시

퀀스 노드 및 세 개의 리프 노드를 포함하는 미디어 타임라인을 나타내는 예시적인 구현의 도면.

<25> 도 16은 실행 가능 시간 언어(XTL) 파일에 의해 기술되는 2개의 자식 시퀀스 노드를 갖는 병렬 노드를 포함하는 미디어 타임라인을 나타내는 예시적인 구현의 도면.

<26> 도 17은 특정 시간 간격들로 제1 및 제2 미디어의 출력을 나타내고 제1 및 제2 미디어 간의 전이 효과를 포함하는 예시적인 구현의 도면.

<27> 도 18은 도 17의 크로스 페이드 효과를 구현하기에 적합한 예시적인 구현에서의 미디어 타임라인의 도면.

<28> 도 19는 정렬을 위한 예시적인 타임라인의 사용자 인터페이스 뷰의 도면.

<29> 도 20은 타임라인 정렬기에 의해 도 19의 타임라인을 복수의 세그먼트로 분할함으로써 형성되는 세그먼트화된 타임라인을 나타내는 다른 사용자 인터페이스 뷰의 도면.

<30> 도 21은 도 19 및 20에 도시된 타임라인을 제공하도록 구성되는 객체 모델을 이용하여 표현되는 예시적인 타임라인의 도면.

<31> 도 22는 미디어 타임라인이 타임라인 정렬기에 의해 렌더링하기 위해 복수의 세그먼트로 정렬되는 예시적인 구현에서의 프로시저를 나타내는 흐름도.

<32> 도 23은 도 22의 프로시저를 이용하여 도 21의 타임라인으로부터 형성되는 예시적인 정렬기 요소의 도면.

<33> 도 24는 지정된 시작 및 중지 시간을 갖지 않는 순차적인 재생 리스트를 제공하도록 구성되는 시퀀스 노드의 도면.

<34> 도 25는 시작 및 중지 시간 지시를 갖지 않는 타임라인 객체를 포함하는 미디어 타임라인이 정렬되는 예시적인 구현에서의 프로시저를 나타내는 흐름도.

<35> 도 26은 타임라인 정렬기에 의해 정렬됨에 따라 도 24의 미디어 타임라인으로부터 형성되는 예시적인 어레이의 도면.

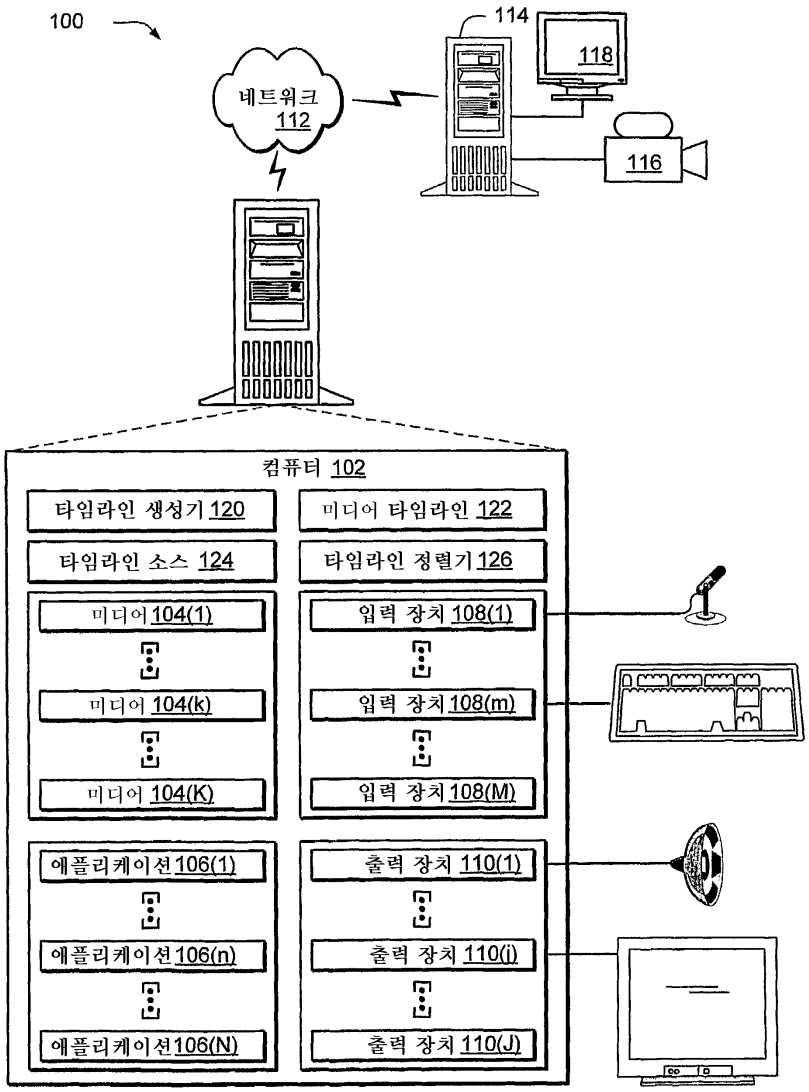
<36> 도 27은 타임라인 정렬기가 각각의 메타데이터에서 지속 기간이 지정되지 않는 어레이 내의 타임라인 객체들에 대해 미디어 소스로부터 지속 기간을 취득하는 예시적인 구현에서의 프로시저를 나타내는 흐름도.

<37> 도 28은 예시적인 운영 환경의 도면.

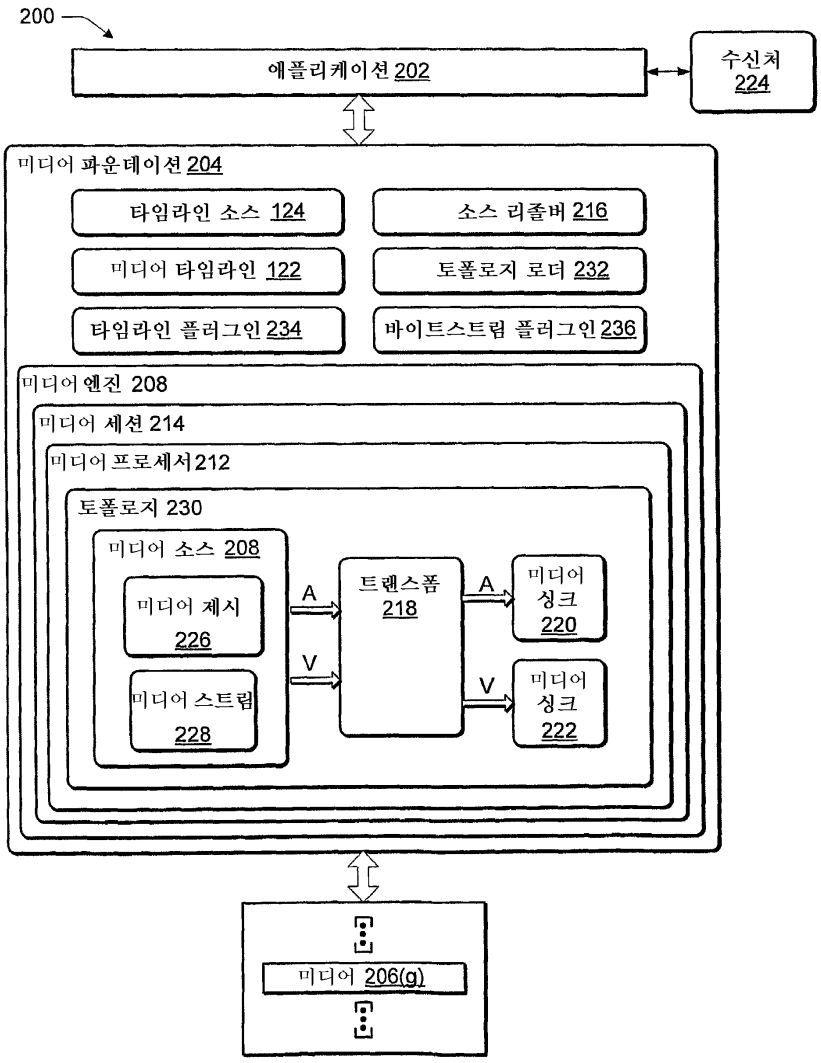
<38> 명세서 및 도면 전반에서 동일한 컴포넌트 및 특징을 참조하기 위해 동일한 번호가 사용된다.

도면

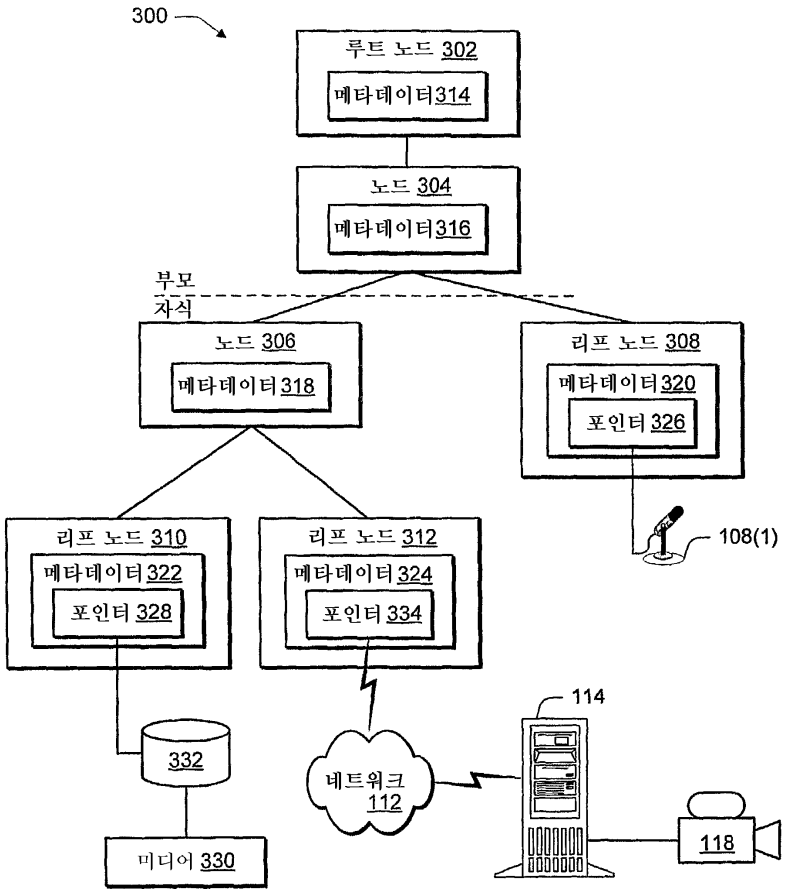
도면1


100
114
118
네트워크
112
116
컴퓨터 102
타임라인 생성기 120
미디어 타임라인 122
타임라인 소스 124
타임라인 정렬기 126
미디어 104(1)
미디어 104(k)
미디어 104(K)
입력 장치 108(1)
입력 장치 108(m)
입력 장치 108(M)
애플리케이션 106(1)
애플리케이션 106(n)
애플리케이션 106(N)
출력 장치 110(1)
출력 장치 110(j)
출력 장치 110(J)

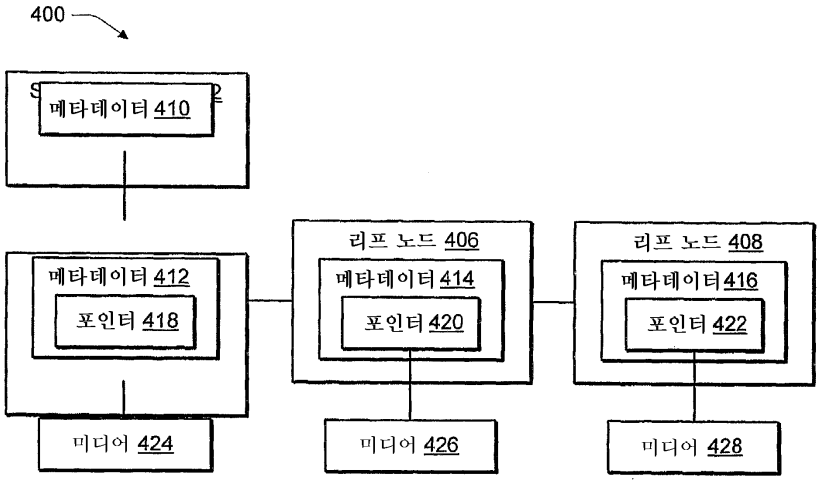
도면2


200
애플리케이션 202
수신처 224
미디어 파운데이션 204
타임라인 소스 124
소스 리졸버 216
미디어 타임라인 122
토폴로지 로더 232
타임라인 플러그인 234
바이트스트림 플러그인 236
미디어 엔진 208
미디어 세션 214
미디어 프로세서 212
토폴로지 230
미디어 소스 208
미디어 제시 226
미디어 스트림 228
A
V
트랜스폼 218
A
V
미디어 싱크 220
미디어 싱크 222
미디어 206(g)

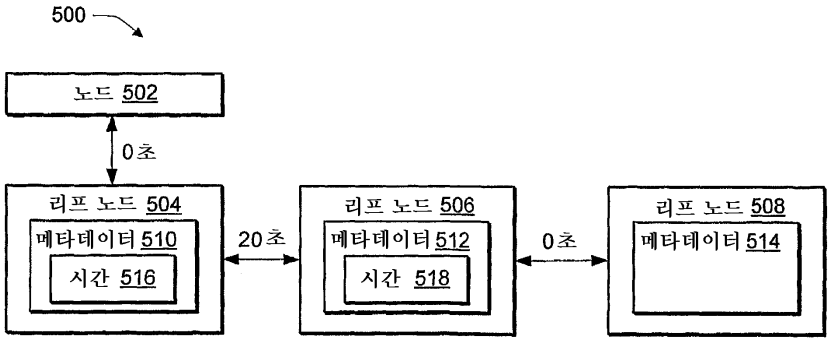
도면3


300
루트 노드 302
메타데이터314
노드 304
메타데이터316
부모
자식
노드 306
메타데이터 318
리프 노드 308
메타데이터320
포인터 326
108(1)
리프 노드 310
메타데이터322
포인터 328
리프 노드 312
메타데이터 324
포인터 334
114
332
네트워크
112
118
미디어 330

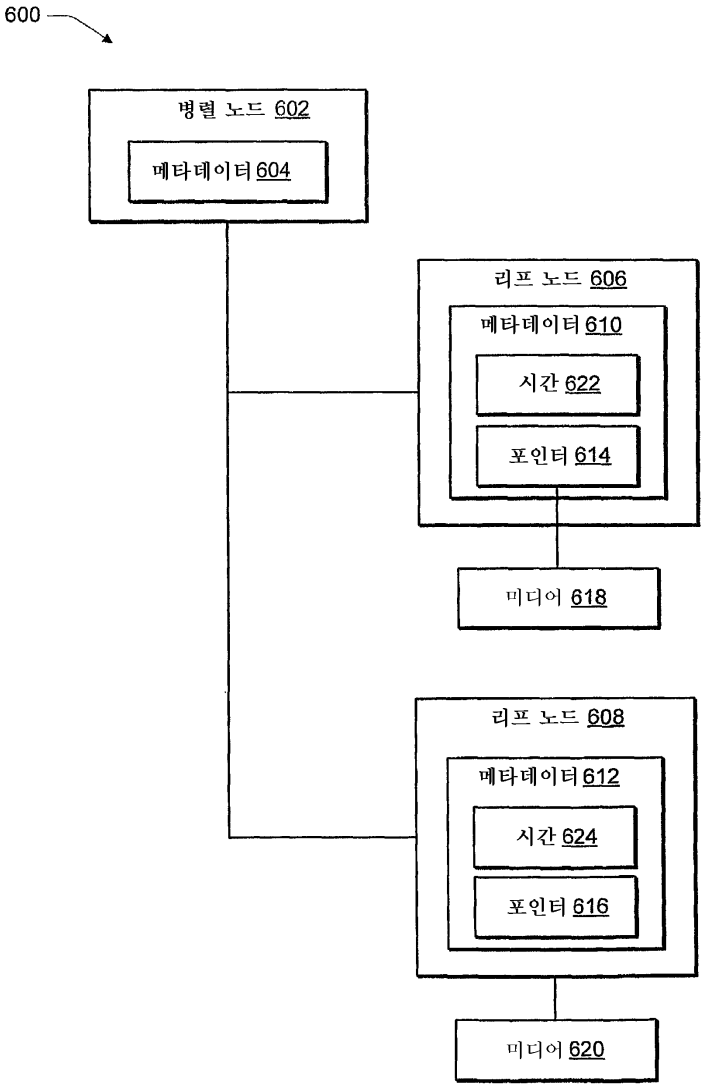
도면4


400
메타데이터 410
메타데이터 412
포인터 418
리프 노드 406
메타데이터 414
포인터 420
리프 노드 408
메타데이터416
포인터 422
미디어 424
미디어 426
미디어 428

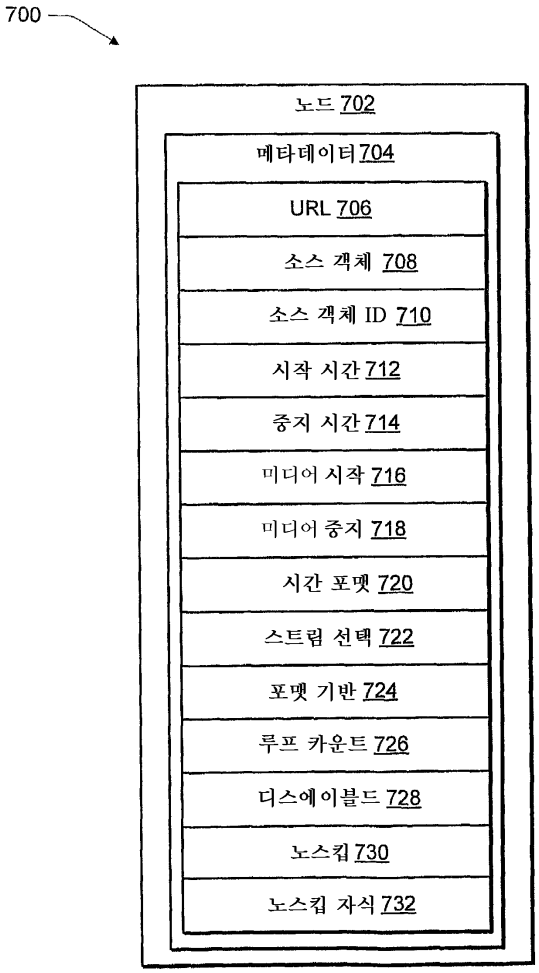
도면5


500
노드 502
0초
리프 노드 504
메타데이터 510
시간 516
20초
리프 노드 506
메타데이터 512
시간 518
0초
리프 노드 508
메타데이터 514

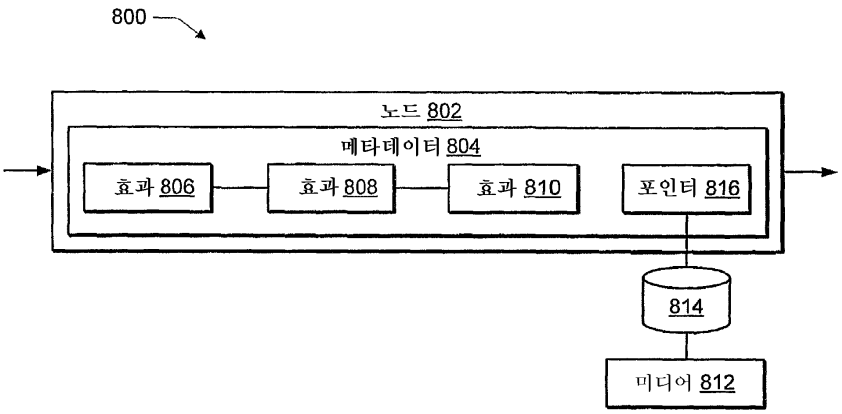
도면6


600
병렬 노드 602
메타데이터 604
리프 노드 606
메타데이터 610
시간 622
포인터 614
미디어 618
리프 노드 608
메타데이터 612
시간 624
포인터 616
미디어 620

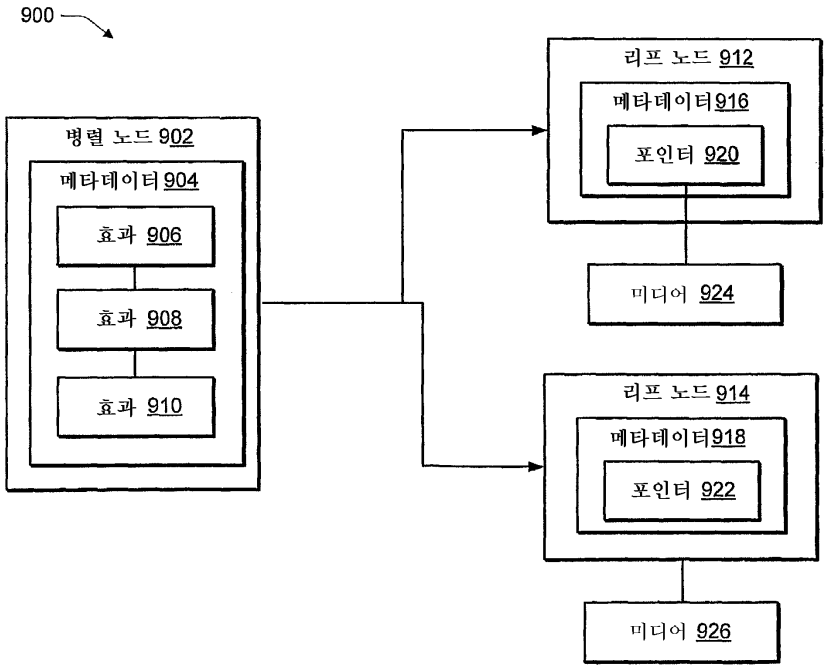
도면7


700
노드 702
메타데이터 704
URL 706
소스 객체 708
소스 객체 ID 710
시작 시간 712
중지 시간 714
미디어 시작 716
미디어 중지 718
시간 포맷 720
스트림 선택 722
포맷 기반 724
루프 카운트 726
디스에이블드 728
노스킵 730
노스킵 자식 732

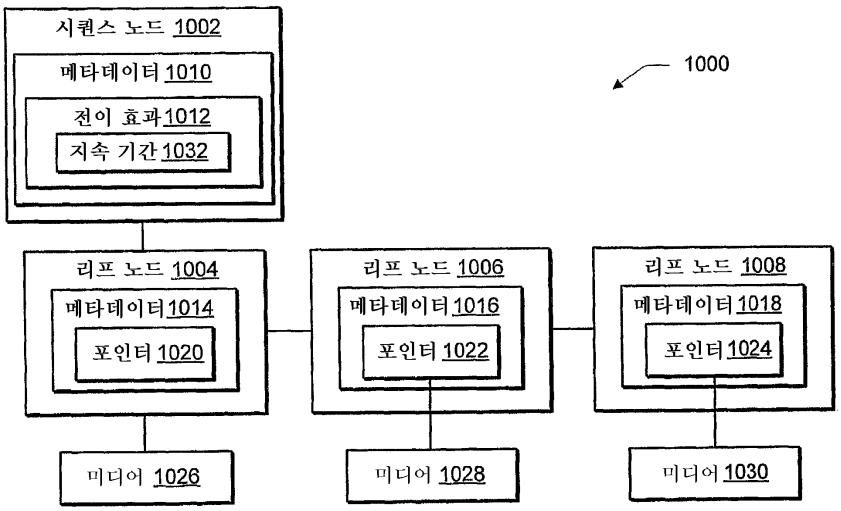
도면8


800
노드 802
메타데이터 804
효과 806
효과 808
효과 810
포인터 816
814
미디어 812

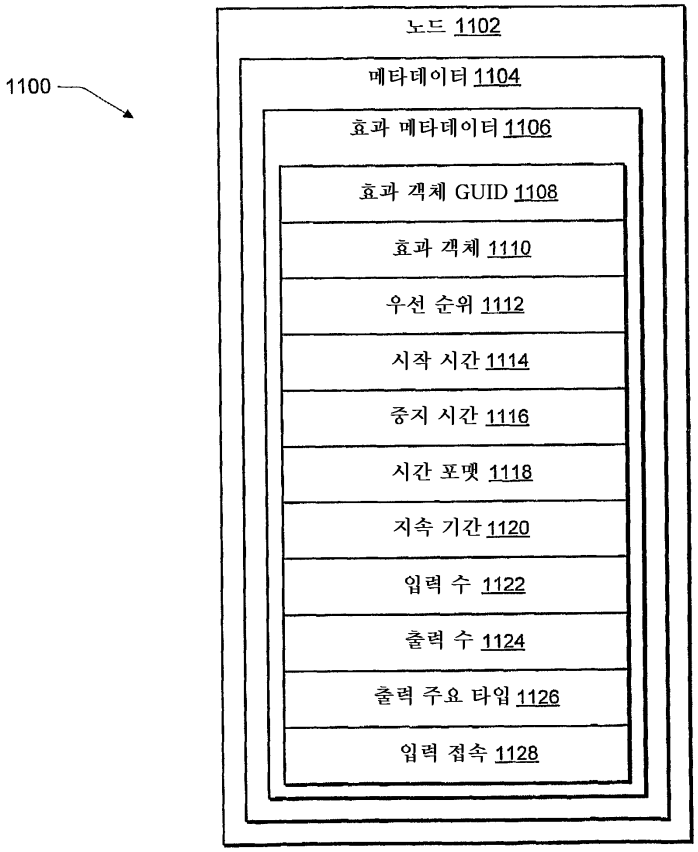
도면9


900
병렬 노드 902
메타데이터 904
효과 906
효과 908
효과 910
리프 노드 912
메타데이터 916
포인터 920
미디어 924
리프 노드 914
메타데이터 918
포인터 922
미디어 926

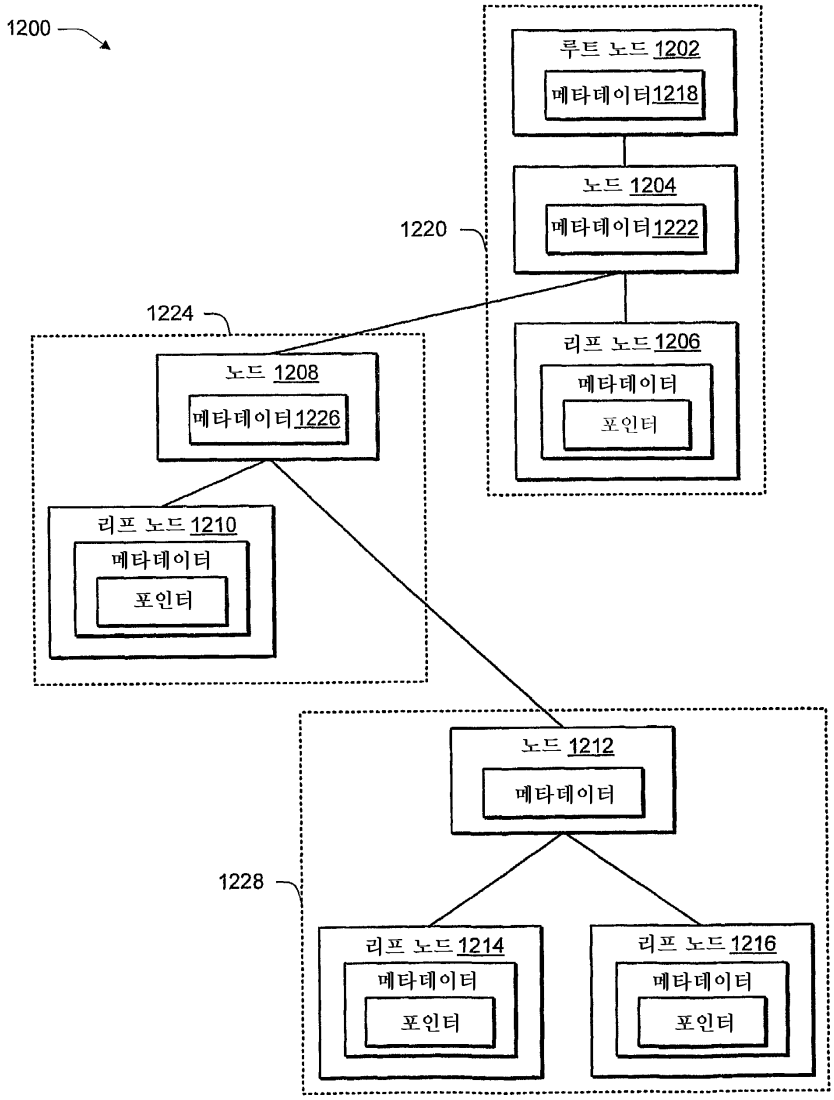
도면10


시퀀스 노드 1002
메타데이터 1010
전이 효과 1012
지속 기간 1032
1000
리프 노드 1004
메타데이터 1014
포인터 1020
미디어 1026
리프 노드 1006
메타데이터 1016
포인터 1022
미디어 1028
리프 노드 1008
메타데이터 1018
포인터 1024
미디어 1030

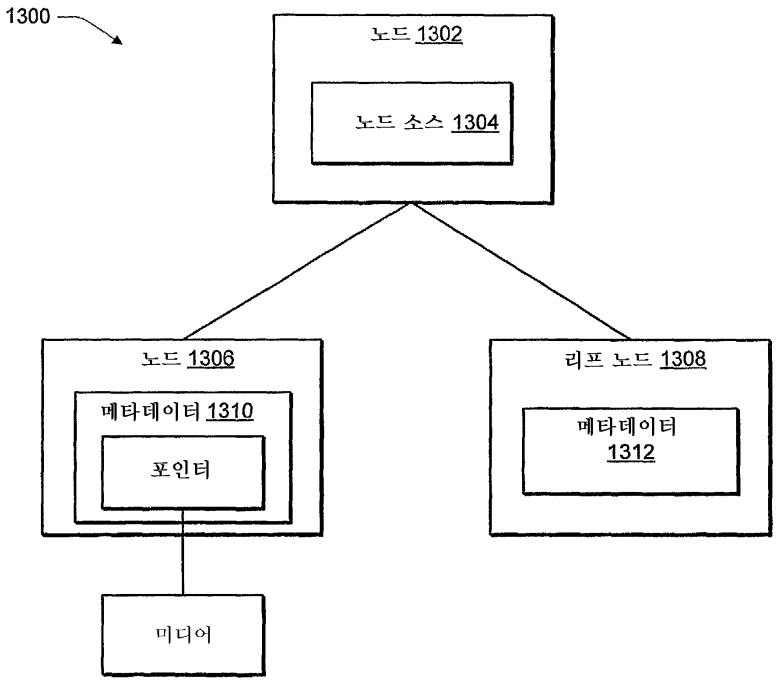
도면11


1100
노드 1102
메타데이터 1104
효과 메타데이터 1106
효과 객체 GUID 1108
효과 객체 1110
우선 순위 1112
시작 시간 1114
중지 시간 1116
시간 포맷 1118
지속 기간 1120
입력 수 1122
출력 수 1124
출력 주요 타입 1126
입력 접속 1128

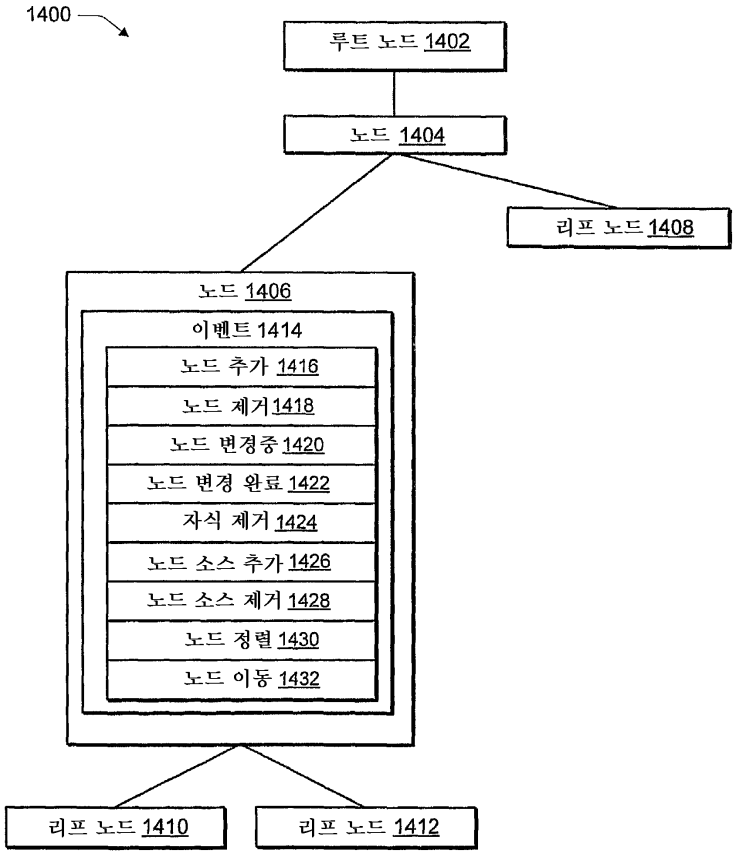
도면12


1200
루트 노드 1202
메타데이터1218
노드 1204
메타데이터1222
1220
1224
노드 1208
메타데이터1226
리프 노드1206
메타데이터
포인터
리프 노드 1210
메타데이터
포인터
노드 1212
메타데이터
1228
리프 노드 1214
메타데이터
포인터
리프 노드 1216
메타데이터
포인터

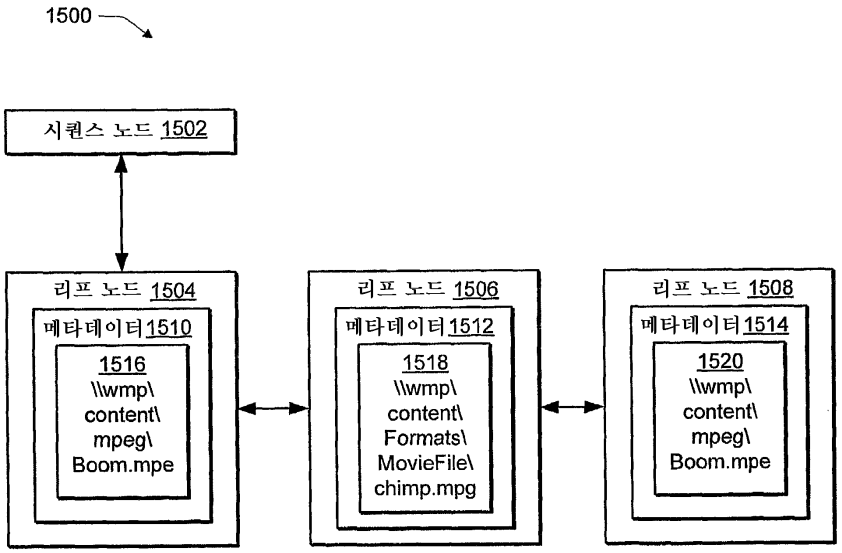
도면13


1300
노드 1302
노드 소스 1304
노드 1306
메타데이터 1310
포인터
리프 노드 1308
메타데이터 1312
미디어

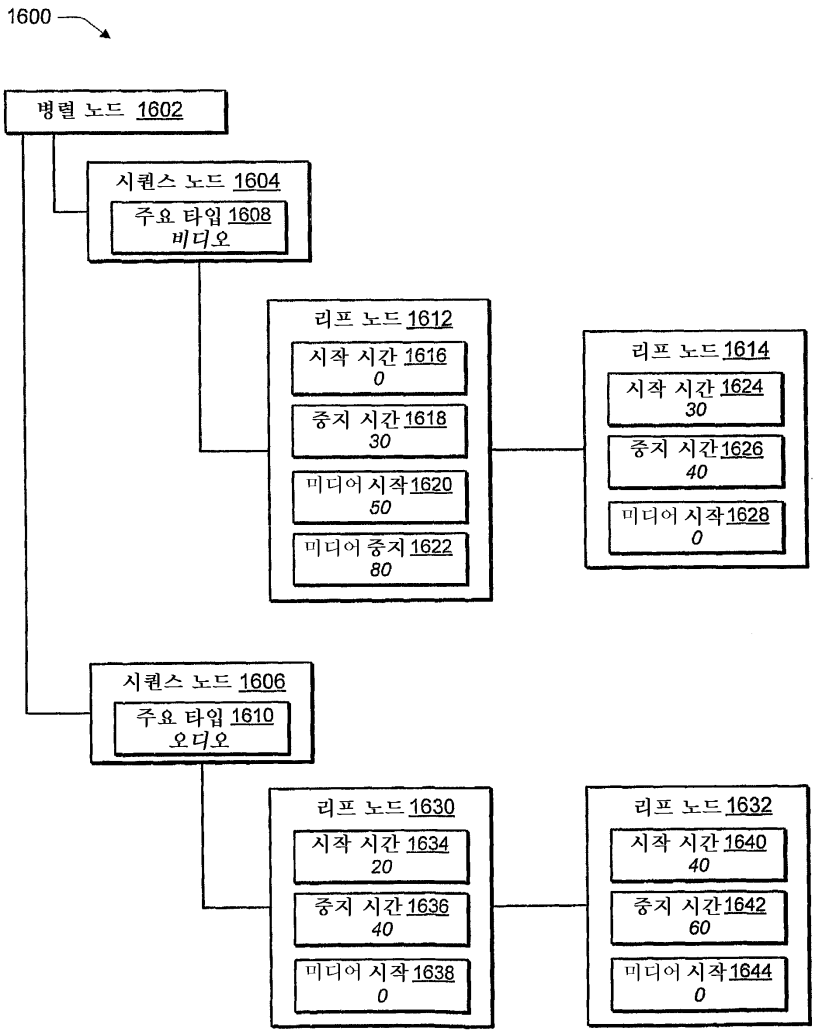
도면14


1400
루트 노드 1402
노드 1404
리프 노드 1408
노드 1406
이벤트 1414
노드 추가 1416
노드 제거 1418
노드 변경중 1420
노드 변경 완료 1422
자식 제거 1424
노드 소스 추가 1426
노드 소스 제거 1428
노드 정렬 1430
노드 이동 1432
리프 노드 1410
리프 노드 1412

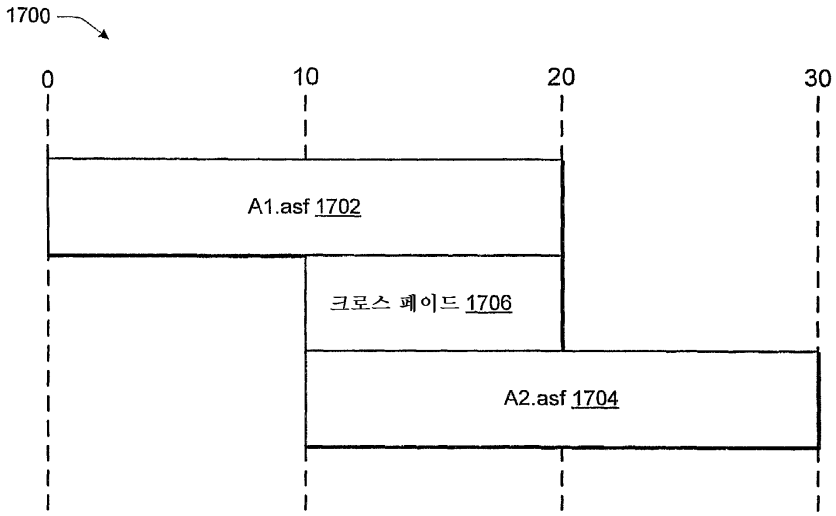
도면15


1500
시퀀스 노드 1502
리프 노드 1504
메타데이터 1510
1516
\\wmp\
content\
mpeg\
Boom.mpe
리프 노드 1506
메타데이터 1512
1518
\\wmp\
content\
Formats\
MovieFile\
chimp.mpg
리프 노드 1508
메타데이터 1514
1520
\\wmp\
content\
mpeg\
Boom.mpe

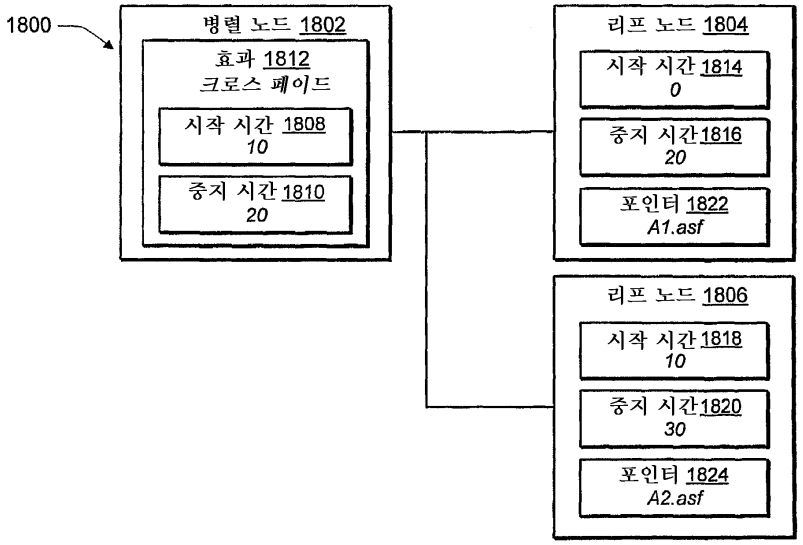
도면16


1600
병렬 노드 1602
시퀀스 노드 1604
주요 타입 1608
비디오
리프 노드 1612
시작 시간 1616
0
중지 시간 1618
30
미디어 시작 1620
50
미디어 중지 1622
80
리프 노드 1614
시작 시간 1624
30
중지 시간 1626
40
미디어 시작 1628
0
시퀀스 노드 1606
주요 타입 1610
오디오
리프 노드 1630
시작 시간 1634
20
중지 시간 1636
40
미디어 시작 1638
0
리프 노드 1632
시작 시간 1640
40
중지 시간 1642
60
미디어 시작 1644
0

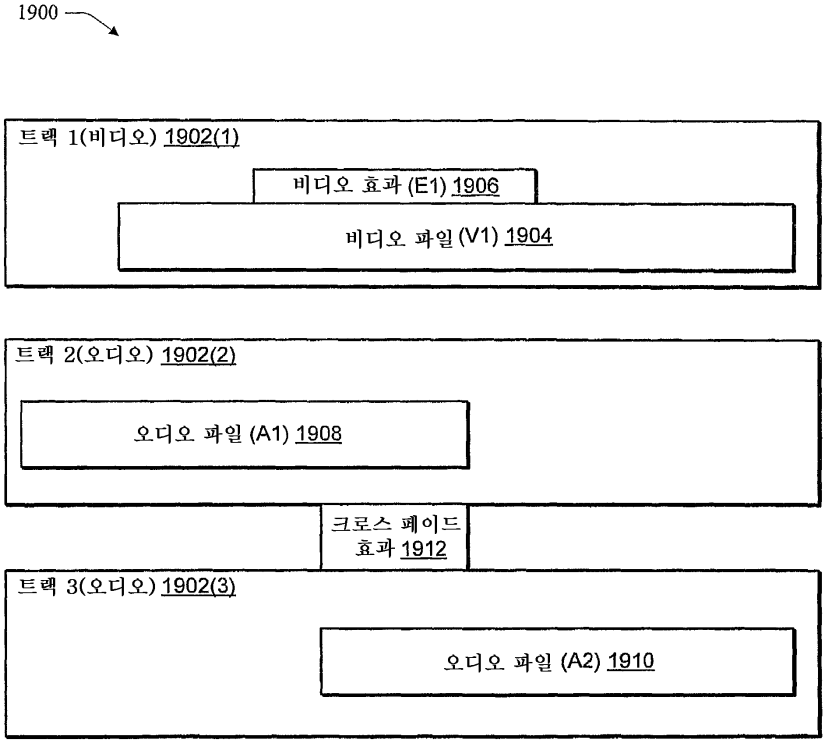
도면17


1700
0
10
20
30
A1.asf 1702
크로스 페이드 1706
A2.asf 1704

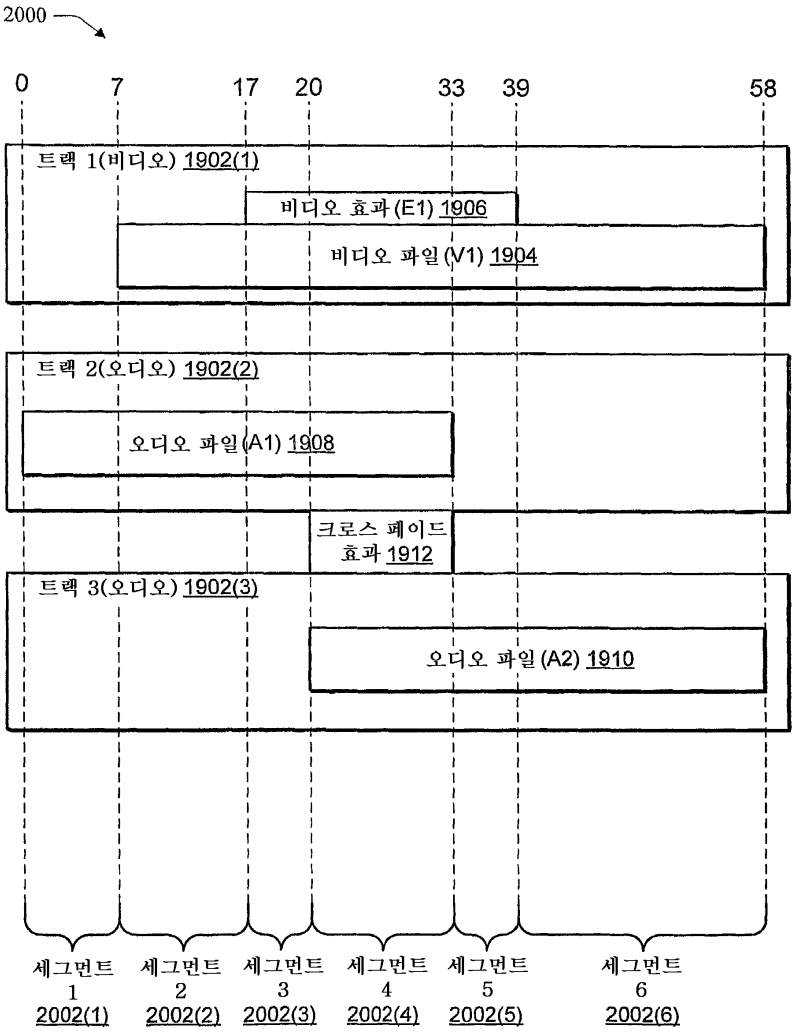
도면18


1800
병렬 노드 1802
효과 1812
크로스 페이드
시작 시간 1808
10
중지 시간 1810
20
리프 노드 1804
시작 시간 1814
0
중지 시간 1816
20
포인터 1822
A1.asf
리프 노드 1806
시작 시간 1818
10
중지 시간 1820
30
포인터 1824
A2.asf

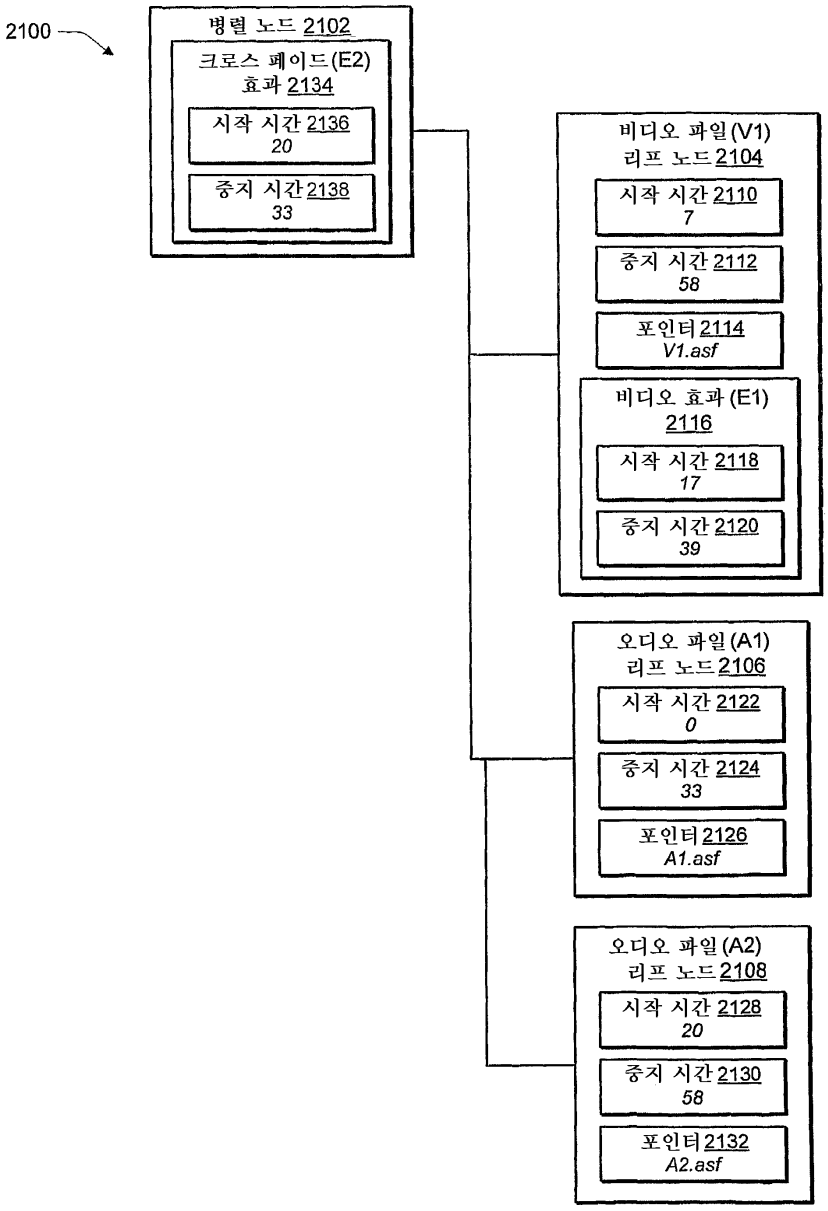
도면19


1900
트랙 1(비디오) 1902(1)
비디오 효과 (E1) 1906
비디오 파일 (V1) 1904
트랙 2(오디오) 1902(2)
오디오 파일 (A1) 1908
크로스 페이드 효과 1912
트랙 3(오디오) 1902(3)
오디오 파일 (A2) 1910

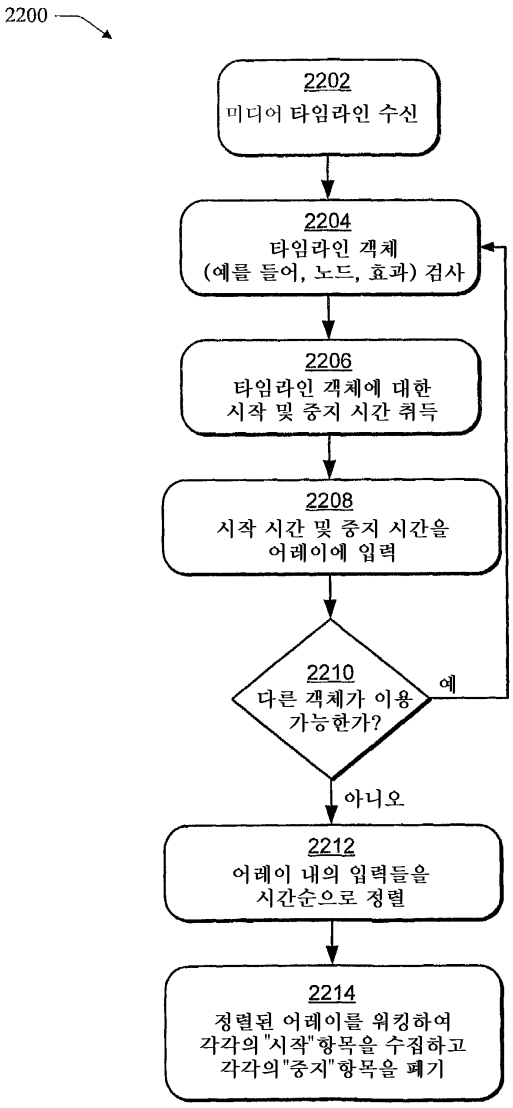
도면20


2000
0
7
17
20
33
39
58
트랙 1(비디오) 1902(1)
비디오 효과 (E1) 1906
비디오 파일 (V1) 1904
트랙 2(오디오) 1902(2)
오디오 파일 (A1) 1908
크로스 페이드 효과 1912
트랙 3(오디오) 1902(3)
오디오 파일 (A2) 1910
세그먼트 1 2002(1)
세그먼트 2 2002(2)
세그먼트 3 2002(3)
세그먼트 4 2002(4)
세그먼트 5 2002(5)
세그먼트 6 2002(6)

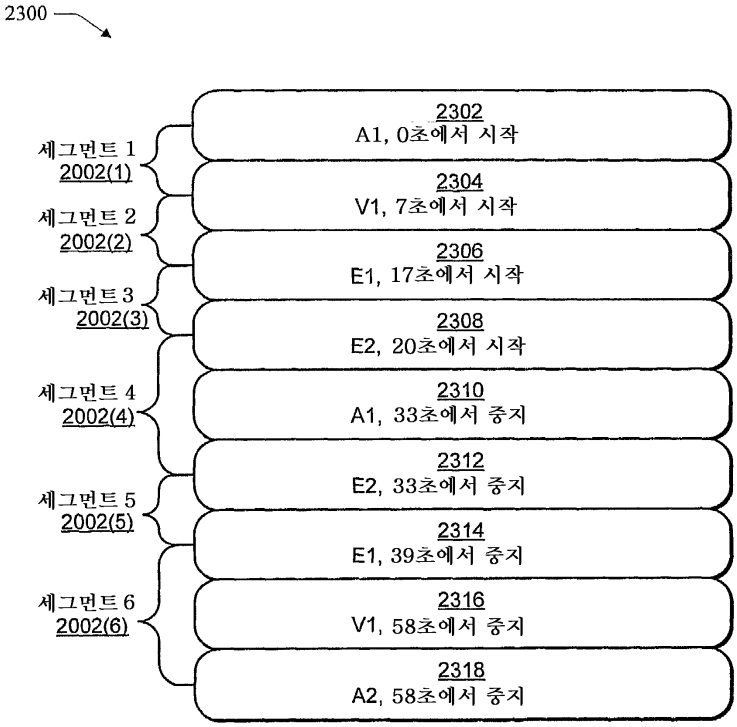
도면21


2100
병렬 노드 2102
크로스 페이드(E2) 효과 2134
시작 시간 2136
20
중지 시간 2138
33
비디오 파일(V1) 리프 노드 2104
시작 시간 2110
7
중지 시간 2112
58
포인터 2114
V1.asf
비디오 효과 (E1) 2116
시작 시간 2118
17
중지 시간 2120
39
오디오 파일(A1) 리프 노드 2106
시작 시간 2122
0
중지 시간 2124
33
포인터 2126
A1.asf
오디오 파일(A2) 리프 노드 2108
시작 시간 2128
20
중지 시간 2130
58
포인터 2132
A2.asf

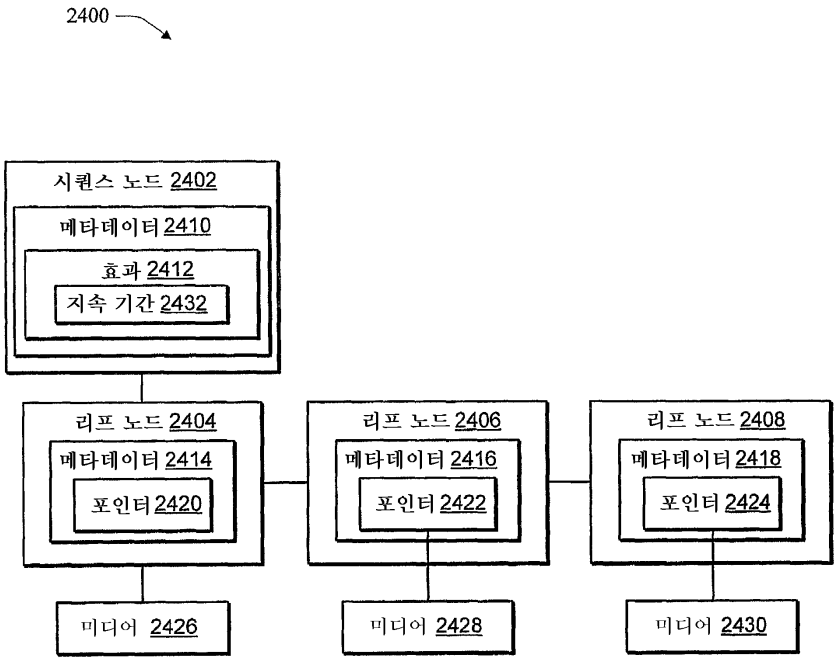
도면22


2200
2202
미디어 타임라인 수신
2204
타임라인 객체
(예를 들어, 노드, 효과) 검사
2206
타임라인 객체에 대한
시작 및 중지 시간 취득
2208
시작 시간 및 중지 시간을
어레이에 입력
2210
다른 객체가 이용
가능한가?
예
아니오
2212
어레이 내의 입력들을
시간순으로 정렬
2214
정렬된 어레이를 워킹하여
각각의 "시작" 항목을 수집하고
각각의 "중지" 항목을 폐기

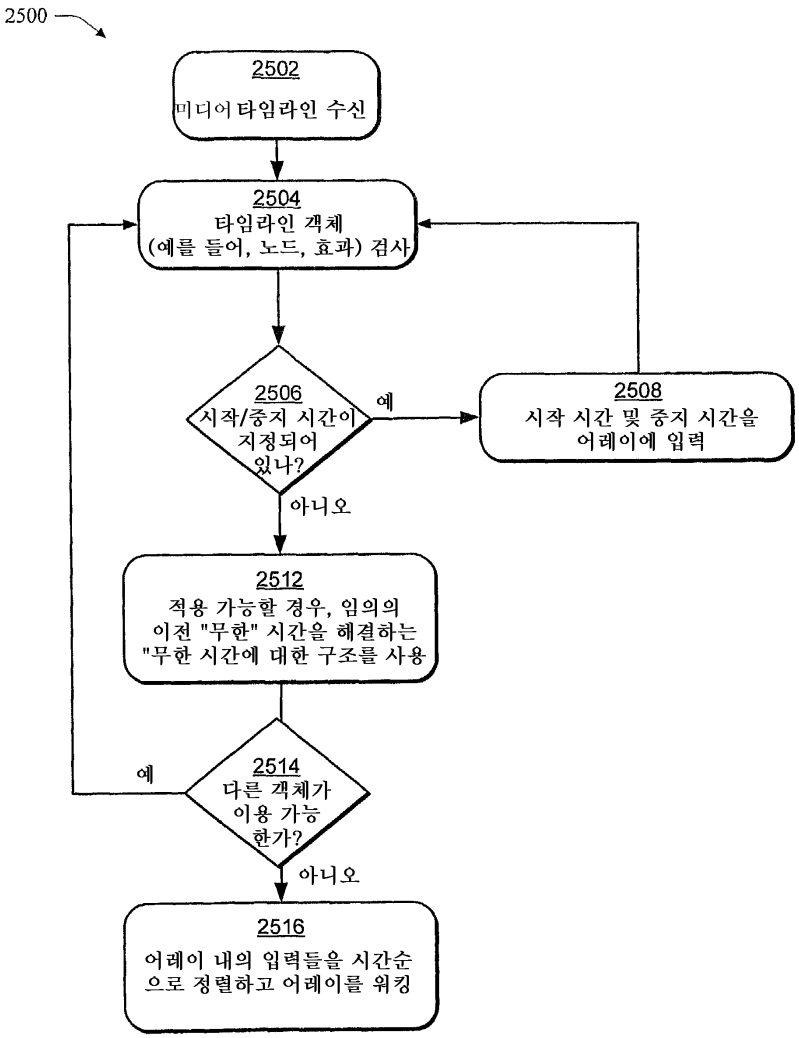
도면23


2300
세그먼트 1
2002(1)
세그먼트 2
2002(2)
세그먼트 3
2002(3)
세그먼트 4
2002(4)
세그먼트 5
2002(5)
세그먼트 6
2002(6)
2302
A1, 0초에서 시작
2304
V1, 7초에서 시작
2306
E1, 17초에서 시작
2308
E2, 20초에서 시작
2310
A1, 33초에서 중지
2312
E2, 33초에서 중지
2314
E1, 39초에서 중지
2316
V1, 58초에서 중지
2318
A2, 58초에서 중지

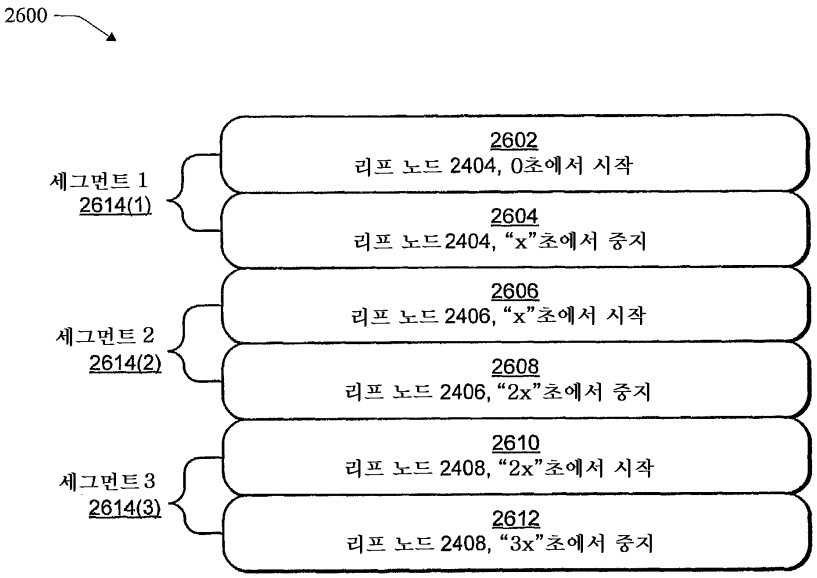
도면24


2400
시퀀스 노드 2402
메타데이터 2410
효과 2412
지속 기간 2432
리프 노드 2404
메타데이터 2414
포인터 2420
리프 노드 2406
메타데이터 2416
포인터 2422
리프 노드 2408
메타데이터 2418
포인터 2424
미디어 2426
미디어 2428
미디어 2430

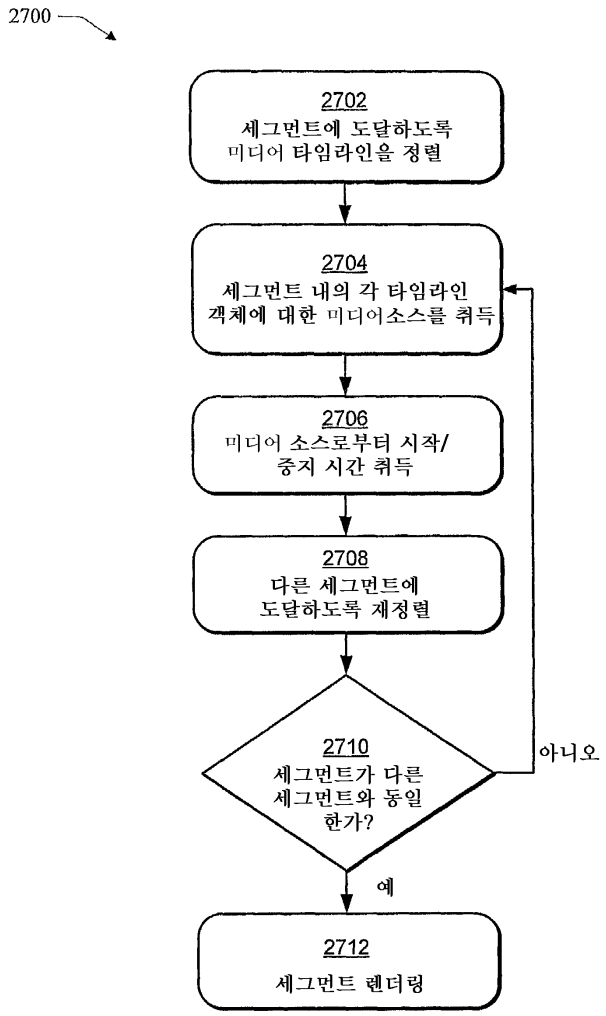
도면25


2500
2502
미디어 타임라인 수신
2504
타임라인 객체 (예를 들어, 노드, 효과) 검사
2506
시작/중지 시간이 지정되어 있나?
예
2508
시작 시간 및 중지 시간을 어레이에 입력
아니오
2512
적용 가능할 경우, 임의의 이전 "무한" 시간을 해결하는 "무한 시간에 대한 구조를 사용
2514
다른 객체가 이용 가능한가?
예
아니오
2516
어레이 내의 입력들을 시간순으로 정렬하고 어레이를 워킹

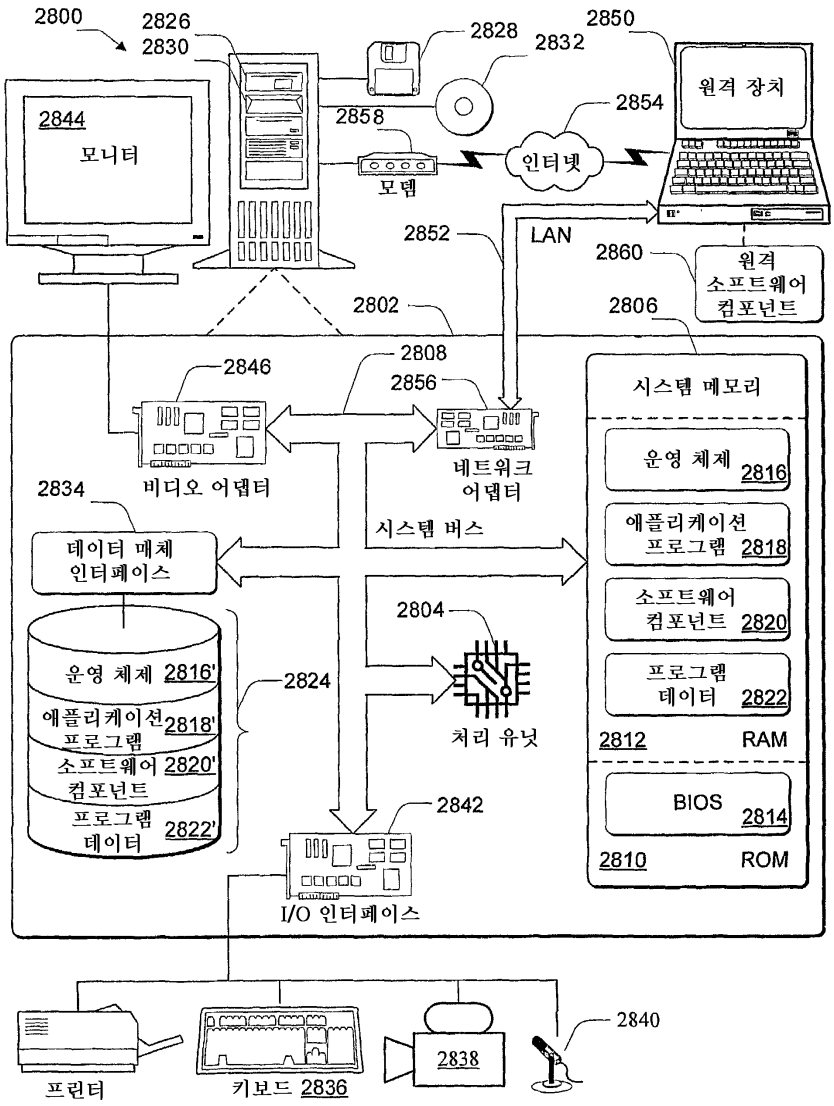
도면26


2600
세그먼트 1
2614(1)
2602
리프 노드 2404, 0초에서 시작
2604
리프 노드 2404, "x"초에서 중지
세그먼트 2
2614(2)
2606
리프 노드 2406, "x"초에서 시작
2608
리프 노드 2406, "2x"초에서 중지
세그먼트 3
2614(3)
2610
리프 노드 2408, "2x"초에서 시작
2612
리프 노드 2408, "3x"초에서 중지

도면27


2700
2702
세그먼트에 도달하도록
미디어 타임라인을 정렬
2704
세그먼트 내의 각 타임라인
객체에 대한 미디어소스를 취득
2706
미디어 소스로부터 시작/
중지 시간 취득
2708
다른 세그먼트에
도달하도록 재정렬
2710
세그먼트가 다른
세그먼트와 동일
한가?
아니오
예
2712
세그먼트 렌더링

도면28


2800
2826
2830
2828
2832
2850
원격 장치
2844
모니터
2858
2854
인터넷
모뎀
2852
LAN
2860
원격
소프트웨어
컴포넌트
2802
2806
2808
2846
2856
시스템 메모리
운영 체제 2816
애플리케이션
프로그램 2818
소프트웨어
컴포넌트 2820
프로그램
데이터 2822
2812 RAM
BIOS 2814
2810 ROM
비디오 어댑터
네트워크
어댑터
시스템 버스
2834
데이터 매체
인터페이스
2804
처리 유닛
2824
운영 체제 2816'
애플리케이션
프로그램 2818'
소프트웨어 2820'
컴포넌트
프로그램
데이터 2822'
2842
I/O 인터페이스
2840
2838
프린터
키보드 2836