(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2013년03월25일
(11) 등록번호 10-1246988
(24) 등록일자 2013년03월18일

(51) 국제특허분류(Int. Cl.)
G06T 13/00 (2011.01) *G06T 17/00* (2006.01)
(21) 출원번호 10-2008-7000381
(22) 출원일자(국제) 2006년07월11일
심사청구일자 2011년06월13일
(85) 번역문제출일자 2008년01월07일
(65) 공개번호 10-2008-0026597
(43) 공개일자 2008년03월25일
(86) 국제출원번호 PCT/US2006/026815
(87) 국제공개번호 WO 2007/008853
국제공개일자 2007년01월18일
(30) 우선권주장
11/181,197 2005년07월13일 미국(US)
(56) 선행기술조사문헌
JP2001051759 A
US06057833 A
전체 청구항 수 : 총 20 항

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
넬슨, 엘리자베스 케이.
미국 98052-6399 워싱톤주 레드몬드 원 마이크로소프트 웨이
재코브, 쿠르트 비.
미국 98052-6399 워싱톤주 레드몬드 원 마이크로소프트 웨이
(뒷면에 계속)
(74) 대리인
제일특허법인

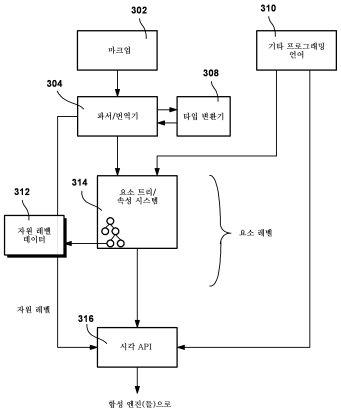
심사관 : 이주미

(54) 발명의 명칭 **애니메이션 간의 매끄러운 전환**

(57) 요 약

"제1" 애니메이션이 이미 실행되고 있는 시각 요소의 속성 상에서 "제2" 애니메이션이 시작될 때 화려한 미디어(예를 들어, UI의 시각 요소들의 애니메이션) 사이의 매끄러운 전환이 제공된다. 제2 애니메이션이 시작될 때, 애니메이션 시스템은 "제1" 애니메이션의 실행의 결과부터 얻어지는 속성의 현재값(즉, 스냅샷)이 저장되게 하며, 제1 애니메이션을 종료(terminate) 또는 해제(release)하고, 이어서 제2 애니메이션으로 하여금 스냅샷을 속성의 "from" 값으로서 사용하여 실행하게 한다. 제2 애니메이션이 정확히 제1 애니메이션이 끝난 시점에서 "시작"하기 때문에, 제1 애니메이션과 제2 애니메이션 간의 전환이 매끄럽다. 애니메이트되고 있는 동안에 속성의 기본값(base value) 및 스냅샷을 저장하기 위해 애니메이션이 트리거되어 있는 속성에 대해 애니메이션 저장 객체가 생성될 수 있다.

대 표 도 - 도3



(72) 발명자

칼킨스, 맷

미국 98052-6399 워싱톤주 레드몬드 원 마이크로소프트 웨이

힐버그, 마이클 제이.

미국 98052-6399 워싱톤주 레드몬드 원 마이크로소프트 웨이

특허청구의 범위

청구항 1

사용자 인터페이스 구성요소의 제1 애니메이션 및 상기 사용자 인터페이스 구성요소의 제2 애니메이션 간의 전환을 위하여 컴퓨터로 구현된 방법으로서,

상기 사용자 인터페이스 구성요소의 상기 제1 애니메이션을 시작하는 단계 - 상기 사용자 인터페이스 구성요소의 상기 제1 애니메이션은 제1 사용자 인터페이스 구성요소의 속성의 값(value of a property)을 조정하는 것을 포함하며, 상기 속성은 상기 값이 조정되는 경우 상기 사용자 인터페이스 구성요소의 크기에 영향을 주는 크기 속성(size property)임 - ;

상기 제2 애니메이션이 시작되고 상기 제1 애니메이션이 상기 사용자 인터페이스 구성요소의 상기 크기 속성의 상기 값을 여전히 조정하는 경우에 어떤 동작을 취할지 지정하는 핸드오프 거동(handoff behavior)을 설정하는 단계;

상기 제1 애니메이션이 실행 중인 동안에 상기 제2 애니메이션에 대한 트리거를 검출하는 단계 - 상기 제1 및 제2 애니메이션은 상기 사용자 인터페이스 구성요소의 상기 크기 속성의 상기 값을 조정하고, 상기 트리거는 상기 사용자 인터페이스 구성요소와의 사용자 상호작용에 응답하여 발생되는 이벤트임 - ;

상기 제1 애니메이션의 상기 실행 동안에 상기 사용자 인터페이스 구성요소의 상기 속성의 값을 재계산하는(recalculating) 단계;

상기 값이 재계산될 때마다 상기 속성의 상기 재계산된 값을 저장하는 단계; 및

상기 재계산된 값을 사용하는 단계를 포함하여, 상기 지정된 핸드오프 거동을 사용하여 상기 제2 애니메이션을 실행하도록 하는 단계

를 포함하는, 컴퓨터로 구현된 방법.

청구항 2

제1항에 있어서,

상기 속성에 대한 애니메이션 저장 객체를 할당하는 단계를 더 포함하며,

상기 애니메이션 저장 객체는 상기 제1 및 제2 애니메이션의 실행으로 얻어지는 상기 속성의 값을 저장하기 위한 것인, 컴퓨터로 구현된 방법.

청구항 3

제1항에 있어서,

상기 핸드오프 거동은,

상기 제1 애니메이션이 계속되고, 상기 제1 애니메이션의 렌더링 사이클 각각에 대하여 상기 제2 애니메이션이 상기 제1 애니메이션의 출력값을 그의 "from" 값으로서 사용한다는 것을 나타내는 작성 핸드오프 거동(compose handoff behavior) - 상기 제2 애니메이션의 출력값은 상기 속성을 렌더링하기 위한 값으로서 사용됨 - ;

상기 제1 애니메이션이 해제되고, "to" 및 "from" 값이 지정되지 않은 경우 상기 제2 애니메이션은 기본 "to" 및 "from" 값을 사용한다는 것을 나타내는 대체 핸드오프 거동(replace handoff behavior);

상기 제1 애니메이션의 상기 속성의 현재값이 획득되고 상기 제1 애니메이션이 해제되며, 상기 제2 애니메이션은, 지정되지 않은 경우, 상기 획득된 현재값을 상기 "to" 값에 대한 기본값 및 상기 "from" 값으로서 사용한다는 것을 나타내는 스냅샷 및 대체 핸드오프 거동(Snapshot and Replace handoff behavior);

상기 제1 애니메이션의 상기 속성의 현재값이 획득되고 상기 제1 애니메이션은 상기 속성의 상기 값에 영향을 주지 않고 계속 실행되며, 상기 제2 애니메이션은, 지정되지 않은 경우, 상기 획득된 현재값을 상기 "to" 값에 대한 기본값 및 상기 "from" 값으로서 사용하고, 상기 제2 애니메이션이 종료되는 경우 상기 제1 애니메이션은 상기 속성의 애니메이션을 재개하는 것을 나타내는 스냅샷 및 유지 핸드오프 거동(Snapshot and Retain handoff behavior);

상기 제1 애니메이션의 상기 속성의 현재값이 획득되고 상기 제1 애니메이션이 일시정지되며, 상기 제2 애니메이션은, 지정되지 않은 경우, 상기 획득된 현재값을 상기 "to" 값에 대한 기본값 및 상기 "from" 값으로서 사용하고, 상기 제2 애니메이션이 종료되는 경우 상기 제1 애니메이션은 일시정지될 때의 상기 제1 애니메이션의 상태로부터 상기 속성을 애니메이트하는 것을 나타내는 스냅샷 및 유지/일시정지 핸드오프 거동(Snapshot and Retain/Paused handoff behavior);

상기 제1 애니메이션 및 상기 제2 애니메이션의 가중 평균(weighted average)을 사용하는 것을 나타내는 블렌딩 핸드오프 거동(blend handoff behavior); 및

기존의 애니메이션이 완료될 때까지 대기 중인 애니메이션을 큐잉(queue up)하는 것을 나타내는 큐잉 핸드오프 거동(queue handoff behavior)

중 하나로부터 선택되는, 컴퓨터로 구현된 방법.

청구항 4

제1항에 있어서,

상기 제2 애니메이션이 실행 중인 동안, 상기 속성의 상기 값에 영향을 주지 않고 상기 제1 애니메이션이 실행될 수 있도록 하는 단계를 더 포함하는, 컴퓨터로 구현된 방법.

청구항 5

제1항에 있어서,

상기 제2 애니메이션이 실행 중인 동안, 상기 제1 애니메이션의 상기 실행을 일시정지시키는 단계를 더 포함하는, 컴퓨터로 구현된 방법.

청구항 6

제1항에 있어서,

상기 제1 및 제2 애니메이션은 복수의 계층으로 논리적으로 구성되고, 하나의 계층 내의 모든 애니메이션들이 종료되었을 경우 상기 계층은 선택적으로 제거될 수 있는, 컴퓨터로 구현된 방법.

청구항 7

제1항에 있어서,

상기 제2 애니메이션이 실행 중인 동안, 제3 애니메이션에 대한 트리거를 검출하는 단계를 더 포함하고,

상기 제1 및 제2 애니메이션은 하나의 계층의 일부이고 상기 제3 애니메이션은 다른 계층의 일부이며, 상기 제2 애니메이션으로부터 얻어지는 상기 속성의 값은 상기 제3 애니메이션에 대한 "from" 값으로서 사용되고, 상기 제3 애니메이션으로부터 얻어지는 상기 속성의 값은 시각 객체의 렌더링에서 상기 속성에 대한 렌더값(render value)으로서 사용되는, 컴퓨터로 구현된 방법.

청구항 8

명령어들을 저장하는 하나 이상의 컴퓨터 판독가능 저장 매체로서,

상기 명령어들은 컴퓨터에 의해 실행될 때 제1항의 방법을 구현하는, 하나 이상의 컴퓨터 판독가능 저장 매체.

청구항 9

제1 애니메이션 및 제2 애니메이션을 처리하는 시스템으로서,

디스플레이될 시각 객체들의 속성에 대한 값을 저장하는 속성 시스템;

디스플레이 장치를 통해 상기 시각 객체를 디스플레이하기 위해 사용되는 데이터를 제공하는 그래픽 서브시스템; 및

상기 디스플레이 장치를 통해 디스플레이되는 경우, 상기 시각 객체를 애니메이트하는 데 사용되는 상기 시각 객체의 속성의 일련의 값들을 계산하는 애니메이션 시스템

을 포함하며,

상기 속성은 애니메이션 동안 크기 속성의 값이 조정되는 경우에 상기 시각 객체의 크기에 영향을 주는 크기 속성이고, 상기 애니메이션 시스템은 상기 제1 애니메이션이 실행 중인 동안에 상기 제2 애니메이션에 대한 트리거를 검출하고 - 상기 트리거는 상기 시각 객체와의 사용자 상호작용에 응답하여 발생되는 이벤트임 - , 상기 제1 애니메이션의 상기 실행 동안 상기 속성의 값을 재계산하고, 상기 제1 애니메이션의 상기 실행으로부터 얻어지는 상기 속성의 상기 재계산된 값을 저장하며, 상기 속성의 상기 재계산되어 저장된 값을 "from" 값으로 사용하여 상기 제2 애니메이션을 시작하는 - 상기 "from" 값은 상기 제2 애니메이션의 시작점이고, 상기 제1 애니메이션 및 상기 제2 애니메이션은 그래픽 사용자 인터페이스 상의 시각 구조의 애니메이션임 - , 제1 애니메이션 및 제2 애니메이션을 처리하는 시스템.

청구항 10

제9항에 있어서,

상기 애니메이션 시스템은 상기 속성에 대한 애니메이션 저장 객체를 할당하고, 상기 애니메이션 저장 객체는 상기 제1 및 제2 애니메이션의 실행에서 얻어지는 상기 속성의 값을 저장하는, 제1 애니메이션 및 제2 애니메이션을 처리하는 시스템.

청구항 11

제9항에 있어서,

상기 애니메이션 시스템은 상기 제2 애니메이션이 시작될 때 상기 제1 애니메이션을 해제하기 위한 것인, 제1 애니메이션 및 제2 애니메이션을 처리하는 시스템.

청구항 12

제9항에 있어서,

상기 애니메이션 시스템은 상기 제2 애니메이션이 실행 중인 동안에 상기 속성의 상기 값에 영향을 주지 않고 상기 제1 애니메이션을 실행하도록 하기 위한 것인, 제1 애니메이션 및 제2 애니메이션을 처리하는 시스템.

청구항 13

제9항에 있어서,

상기 애니메이션 시스템은 상기 제2 애니메이션이 실행 중인 동안에 상기 제1 애니메이션의 상기 실행을 일시정지시키기 위한 것인, 제1 애니메이션 및 제2 애니메이션을 처리하는 시스템.

청구항 14

제9항에 있어서,

상기 애니메이션 시스템은 상기 제1 애니메이션을 실행하기 이전에 상기 속성 시스템으로부터 상기 속성에 대한 적어도 하나의 기본값을 획득하기 위한 것인, 제1 애니메이션 및 제2 애니메이션을 처리하는 시스템.

청구항 15

제9항에 있어서,

상기 애니메이션 시스템은 상기 제2 애니메이션이 실행 중인 동안 제3 애니메이션에 대한 트리거를 검출하고, 상기 제1 및 제2 애니메이션은 하나의 계층의 일부이고 상기 제3 애니메이션은 다른 계층의 일부이며, 상기 제2 애니메이션으로부터 얻어지는 상기 속성의 값은 상기 제3 애니메이션에 대한 "from" 값으로서 사용되고, 상기 제3 애니메이션으로부터 얻어지는 상기 속성의 값은 상기 시각 객체의 렌더링에서 상기 속성에 대한 렌더값으로서 사용되는, 제1 애니메이션 및 제2 애니메이션을 처리하는 시스템.

청구항 16

명령어들을 저장하는 하나 이상의 컴퓨터 판독가능 저장 매체로서,

상기 명령어들은 컴퓨터에 의해 실행될 때,

제1 애니메이션 및 제2 애니메이션을 처리하는 방법을 구현하고,

상기 방법은,

디스플레이될 시각 객체들의 속성에 대한 값을 저장하는 단계;

디스플레이 장치를 통해 상기 시각 객체를 디스플레이하기 위해 사용되는 데이터를 제공하는 단계; 및

상기 디스플레이 장치를 통해 디스플레이되는 경우, 상기 시각 객체를 애니메이트하는 데 사용되는 상기 시각 객체의 속성의 일련의 값들을 계산하는 단계

를 포함하며,

상기 속성은 애니메이션 동안 크기 속성의 값이 조정되는 경우에 상기 시각 객체의 크기에 영향을 주는 크기 속성이고, 상기 계산하는 단계는 상기 제1 애니메이션이 실행 중인 동안에 상기 제2 애니메이션에 대한 트리거를 검출하는 동작 - 상기 트리거는 상기 시각 객체와의 사용자 상호작용에 응답하여 발생되는 이벤트임 - , 상기 제1 애니메이션의 상기 실행 동안 상기 속성의 값을 재계산하는 동작, 상기 제1 애니메이션의 상기 실행으로부터 얻어지는 상기 속성의 상기 재계산된 값을 저장하는 동작, 상기 속성의 상기 재계산되어 저장된 값을 "from" 값으로 사용하여 상기 제2 애니메이션을 시작하는 동작 - 상기 "from" 값은 상기 제2 애니메이션의 시작점이고, 상기 제1 애니메이션 및 상기 제2 애니메이션은 그래픽 사용자 인터페이스 상의 시각 구조의 애니메이션임 - 을 더 포함하는,

하나 이상의 컴퓨터 판독가능 저장 매체.

청구항 17

시각 요소를 애니메이트하기 위해 컴퓨터로 구현된 방법으로서,

그래픽 사용자 인터페이스 상에서 디스플레이되는 시각 요소에 대한 트리거 가능 애니메이션들의 세트(a set of triggerable animations)를 정의하는 단계 - 상기 세트는 하나 이상의 트리거 가능 애니메이션들의 제1 서브셋 및 트리거 가능 애니메이션들의 하나 이상의 다른 서브셋을 포함하며, 상기 제1 서브셋의 상기 트리거 가능 애니메이션들 각각은 상기 시각 요소와의 사용자 상호 작용에 응답하여 발생되는 속성 트리거 또는 이벤트 트리거 중 어느 하나에 의해 트리거 가능하고, 상기 하나 이상의 다른 서브셋의 상기 트리거 가능 애니메이션들 각각은 상기 시각 요소와의 사용자 상호 작용에 응답하여 발생되는 속성 트리거에 의해 각각 트리거 가능함 - ;

재계산될 때 상기 시각 요소의 크기에 영향을 주는 상기 애니메이션들 각각 동안에 크기 속성값을 재계산하는 단계;

상기 재계산된 속성을 저장하는 단계; 및

상기 재계산된 값을 사용하여 후속 애니메이션이 개시되기 위한 "from" 위치를 결정하는 단계

를 포함하며,

하나의 서브셋의 애니메이션으로부터 다른 서브셋의 애니메이션으로 전환하기 위해 합성 핸드오프 거동(composition handoff behavior)이 사용되는, 컴퓨터로 구현된 방법.

청구항 18

제17항에 있어서,

상기 세트의 서브셋들 중 하나의 제2 애니메이션은, 시작될 때, 그 서브셋의 제1 애니메이션에 의해 영향을 받는 속성값의 스냅샷이 선택적으로 획득될 수 있도록 하고, 상기 제2 애니메이션이 시작될 때 상기 제1 애니메이션은 실행 중인, 컴퓨터로 구현된 방법.

청구항 19

제18항에 있어서,

상기 제2 애니메이션이 시작될 때 상기 제1 애니메이션은 해제되는, 컴퓨터로 구현된 방법.

청구항 20

명령어들을 저장하는 하나 이상의 컴퓨터 판독가능 저장 매체로서,

상기 명령어들은 컴퓨터에 의해 실행될 때 제17항의 방법을 구현하는, 하나 이상의 컴퓨터 판독가능 저장 매체.

명 세 서

배 경 기 술

[0001] 어떤 상호작용 시스템에서, 사용자 상호작용에 응답하여 "애니메이트(animate)"하는 것으로 반응하는 시각 요소가 구현될 수 있다. 예를 들어, 사용자 인터페이스(UI)에 디스플레이되는 버튼은 그 버튼이 확대 또는 축소되거나 회전, 기타 등등을 하도록 구현될 수 있다. 어떤 상황에서, 다수의 애니메이션이 단일의 시각 요소 상에 수행되는 것이 바람직할 수 있다. 어떤 시스템에서, 시각 요소의 현재 실행 중인 애니메이션이 다른 애니메이션으로 전환될 때 시각 요소의 디스플레이가 "글리치(glitch)"하는 것처럼 보일 수 있으며, 이는 어떤 시나리오에서 바람직하지 않을 수 있다. 이러한 배경 정보는 청구된 발명에 의해 해결되어야만 하는 문제점을 확인하기 위한 것이 아니다.

발명의 상세한 설명

[0002] 이하에서 실시예에 대한 설명 섹션에서 더 기술되는 일련의 개념들을 간단화된 형태로 소개하기 위해 이 요약이 제공된다. 이 요약은 청구된 발명의 주요 특징 또는 본질적인 특징을 확인하기 위한 것도 아니고 청구된 발명의 범위를 정하는 보조 수단으로서 사용하기 위한 것도 아니다.

[0003] 다양한 기술된 실시예들의 측면에 따르면, 화려한 미디어(예를 들어, UI의 시각 요소들의 애니메이션) 사이의 매끄러운 전환을 가능하게 해주는 구현이 제공된다. 한 측면에서, 다른 애니메이션이 이미 실행되고 있는 시각 요소의 속성에 대해 한 애니메이션이 시작될 때, "제1" 애니메이션의 실행의 결과부터 얻어지는 속성의 현재값(즉, 스냅샷)을 저장하고, 제1 애니메이션을 해제(release)하며, 이어서 스냅샷을 속성의 "from" 값으로서 사용하여 제2 애니메이션을 실행하도록 "제2" 애니메이션이 구성될 수 있다. 이 핸드오프 거동은 "스냅 및 대체(snap and replace)" 거동이라고 한다. 제2 애니메이션이 정확히 제1 애니메이션이 끝난 시점에서 "시작"하기 때문에, 제1 애니메이션과 제2 애니메이션 간의 전환이 매끄럽다.

[0004] 다른 측면에서, 애니메이션이 트리거되어 있는 속성에 대해 애니메이션 저장 객체가 생성된다. 애니메이션 저장 객체는 애니메이션이 목표로 하는 속성의 하나 이상의 기본값, 그에 부가하여 초기 애니메이션이 여전히 실행 중인 동안에 또하나의 애니메이션이 트리거되는 경우 그 속성의 임의의 스냅샷을 저장하기 위한 것이다.

[0005] 또다른 측면에서, 다수의 애니메이션 계층이 속성에 적용될 수 있으며, 계층들 간의 전환은 합성 핸드오프 거동(composition handoff behavior)을 갖는다. 어떤 구현들에서, 계층 내에서의 애니메이션들 간의 전환은 합성 또는 "스냅샷 및 대체" 거동을 가질 수 있다. 다른 구현들에서, 계층들 사이에서의 애니메이션들 간의 전환은 합성 또는 "스냅샷 및 대체" 거동을 가질 수 있다.

[0006] 실시예들은 컴퓨터 프로세스, 컴퓨터 시스템(모바일 핸드헬드 컴퓨팅 장치를 포함함) 또는 컴퓨터 프로그램 제품 등의 제조 물품으로서 구현될 수 있다. 컴퓨터 프로그램 제품은 컴퓨터 시스템에 의해 판독가능함과 동시에 컴퓨터 프로세스를 실행하는 명령어들의 컴퓨터 프로그램을 인코딩하는 컴퓨터 저장 매체일 수 있다. 컴퓨터 프로그램 제품은 또한 컴퓨터 시스템에 의해 판독가능함과 동시에 컴퓨터 프로세스를 실행하는 명령어들의 컴퓨터 프로그램을 인코딩하는 반송파를 통해 전파되는 신호일 수 있다.

실 시 예

[0016] 비제한적이고(non-limiting) 비전수적인(non-exhaustive) 실시예들이 첨부 도면들을 참조하여 기술되며, 달리 언급하지 않는 한 여러가지 도면에 걸쳐 유사한 참조 번호는 유사한 부분을 가리킨다.

[0017] 본 명세서의 일부를 형성하고 본 발명을 실시하는 구체적이고 예시적인 실시예를 도시하는 첨부 도면을 참조하여, 여러가지 실시예에 대해 이하에서 더 상세히 기술한다. 그렇지만, 실시예들은 많은 다른 형태로 구현될 수 있고 본 명세서에 기술된 실시예들로 제한되는 것으로 해석되어서는 안되며, 오히려 이들 실시예는 본 발명의 개시 내용이 철저하고 완전하며 당업자에게 본 발명의 범위를 충분히 전달하도록 제공된 것이다. 실시예들은 방법, 시스템 또는 장치로 실시될 수 있다. 그에 따라, 실시예들은 하드웨어 구현, 전적으로 소프트웨어 구현,

또는 소프트웨어 및 하드웨어 측면을 결합한 구현의 형태를 가질 수 있다. 따라서, 이하의 상세한 설명은 제한적인 의미로 보아서는 안된다.

[0018] 다양한 실시예의 논리적 동작은 (1) 컴퓨팅 시스템 상에서 실행되는 일련의 컴퓨터 구현 단계들로서, 및/또는 (2) 컴퓨팅 시스템 내의 상호연결된 기계 모듈로서 구현된다. 이 구현은 실시예를 구현하는 컴퓨팅 시스템의 성능 요건에 의존하는 선택의 문제이다. 그에 따라, 본 명세서에 기술된 실시예들을 구성하는 논리적 동작들은 다른 대안으로서 동작, 단계 또는 모듈이라고 한다.

[0019] 예시적인 운영 환경

[0020] 도 1은 애니메이션들 간의 매끄러운 전환이 구현될 수 있는 적합한 컴퓨팅 시스템 환경(100)의 일례를 도시하고 있다. 컴퓨팅 시스템 환경(100)은 적합한 컴퓨팅 환경의 일례에 불과하며, 본 발명의 용도 또는 기능성의 범위에 관해 어떤 제한을 암시하고자 하는 것이 아니다. 컴퓨팅 환경(100)이 예시적인 운영 환경(100)에 도시된 컴포넌트들 중 임의의 하나 또는 그 컴포넌트들의 임의의 조합과 관련하여 어떤 의존성 또는 요구사항을 갖는 것으로 해석되어서도 안된다.

[0021] 본 발명은 많은 기타 범용 또는 특수 목적의 컴퓨팅 시스템 환경 또는 구성에서 동작한다. 본 발명에서 사용하는 데 적합할 수 있는 잘 알려진 컴퓨팅 시스템, 환경 및/또는 구성의 예로는 퍼스널 컴퓨터, 서버 컴퓨터, 핸드-헬드 또는 랩톱 장치, 태블릿 장치, 멀티프로세서 시스템, 마이크로프로세서 기반 시스템, 셋톱 박스, 프로그램가능한 가전제품, 네트워크 PC, 미니컴퓨터, 메인프레임 컴퓨터, 전화 시스템, 상기 시스템들이나 장치들 중 임의의 것을 포함하는 분산 컴퓨팅 환경, 기타 등등이 있지만 이에 제한되는 것은 아니다.

[0022] 본 발명은 일반적으로 컴퓨터에 의해 실행되는 프로그램 모듈과 같은 컴퓨터 실행가능 명령어와 관련하여 기술될 수 있다. 일반적으로, 프로그램 모듈은 특정의 태스크를 수행하거나 특정의 추상 데이터 유형을 구현하는 루틴, 프로그램, 객체, 컴포넌트, 데이터 구조 등을 포함한다. 본 발명은 또한 통신 네트워크를 통해 연결되어 있는 원격 처리 장치들에 의해 태스크가 수행되는 분산 컴퓨팅 환경에서 실시될 수 있다. 분산 컴퓨팅 환경에서, 프로그램 모듈은 메모리 저장 장치를 비롯한 로컬 및 원격 컴퓨터 저장 매체 둘다에 위치할 수 있다.

[0023] 도 1을 참조하면, 본 발명을 구현하는 예시적인 시스템은 컴퓨터(110) 형태의 범용 컴퓨팅 장치를 포함한다. 컴퓨터(110)의 컴포넌트들은 처리 장치(120), 시스템 메모리(130), 및 시스템 메모리를 비롯한 각종의 시스템 컴포넌트들을 처리 장치(120)에 연결시키는 시스템 버스(121)를 포함할 수 있지만 이에 제한되는 것은 아니다. 시스템 버스(121)는 메모리 버스 또는 메모리 컨트롤러, 주변 장치 버스 및 각종 버스 아키텍처 중 임의의 것을 이용하는 로컬 버스를 비롯한 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, AGP(Accelerated Graphics Port) 버스, 그리고 메자닌 버스(Mezzanine bus)로도 알려진 PCI(Peripheral Component Interconnect) 버스 등을 포함하지만 이에 제한되는 것은 아니다.

[0024] 컴퓨터(110)는 통상적으로 각종 컴퓨터 판독가능 매체를 포함한다. 컴퓨터(110)에 의해 액세스 가능한 매체는 그 어떤 것이든지 컴퓨터 판독가능 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 포함할 수 있지만 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(110)에 의해 액세스될 수 있고 원하는 정보를 저장하는 데 사용될 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, 신호 내에 정보를 인코딩하도록 그 신호의 특성들 중 하나 이상을 설정 또는 변경시킨 신호를 의미한다. 예로서, 통신 매체는 유선 네트워크 또는 직접 배선 접속(direct-wired connection)과 같은 유선 매체, 그리고 음향, RF, 적외선, 기타 무선 매체와 같은 무선 매체를 포함하지만 이에 한정되는 것은 아니다. 상술된 매체들 중 임의의 것의 조합도 컴퓨터 판독가능 매체의 영역 안에 포함되는 것으로 한다.

[0025] 시스템 메모리(130)는 판독 전용 메모리(ROM)(131) 및 랜덤 액세스 메모리(RAM)(132)와 같은 휘발성 및/또는 비휘발성 메모리 형태의 컴퓨터 저장 매체를 포함한다. 시동 중과 같은 때에, 컴퓨터(110) 내의 구성요소들 사이의 정보 전송을 돕는 기본 루틴을 포함하는 기본 입/출력 시스템(BIOS)(133)은 통상적으로 ROM(131)에 저장되어 있다. RAM(132)은 통상적으로 처리 장치(120)가 즉시 액세스 할 수 있고 및/또는 현재 동작시키고 있는 데이터 및/또는 프로그램 모듈을 포함한다. 예로서, 도 1은 운영 체제(134), 애플리케이션 프로그램(135), 기타 프로그램 모듈(136) 및 프로그램 데이터(137)를 도시하고 있지만 이에 제한되는 것은 아니다.

[0026] 컴퓨터(110)는 또한 기타 이동식/비이동식, 휘발성/비휘발성 컴퓨터 저장 매체를 포함할 수 있다. 단지 예로서, 도 1은 비이동식·비휘발성 자기 매체에 기록을 하거나 그로부터 판독을 하는 하드 디스크 드라이브(141), 이동식·비휘발성 자기 디스크(152)에 기록을 하거나 그로부터 판독을 하는 자기 디스크 드라이브(151), CD-ROM 또는 기타 광 매체 등의 이동식·비휘발성 광 디스크(156)에 기록을 하거나 그로부터 판독을 하는 광 디스크 드라이브(155)를 포함한다. 예시적인 운영 환경에서 사용될 수 있는 기타 이동식/비이동식, 휘발성/비휘발성 컴퓨터 저장 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(141)는 통상적으로 인터페이스(140)와 같은 비이동식 메모리 인터페이스를 통해 시스템 버스(121)에 접속되고, 자기 디스크 드라이브(151) 및 광 디스크 드라이브(155)는 통상적으로 인터페이스(150)와 같은 이동식 메모리 인터페이스에 의해 시스템 버스(121)에 접속된다.

[0027] 위에서 설명되고 도 1에 도시된 드라이브들 및 이들과 관련된 컴퓨터 저장 매체는, 컴퓨터(110)에 대한 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 및 기타 데이터를 저장한다. 도 1에서, 예를 들어, 하드 디스크 드라이브(141)는 운영 체제(144), 애플리케이션 프로그램(145), 기타 프로그램 모듈(146), 및 프로그램 데이터(147)를 저장하는 것으로 도시되어 있다. 여기서 주의할 점은 이들 컴포넌트가 운영 체제(134), 애플리케이션 프로그램(135), 기타 프로그램 모듈(136), 및 프로그램 데이터(137)와 동일하거나 그와 다를 수 있다는 것이다. 이에 관해, 운영 체제(144), 애플리케이션 프로그램(145), 기타 프로그램 모듈(146) 및 프로그램 데이터(147)에 다른 번호가 부여되어 있다는 것은 적어도 이들이 다른 사본(copy)이라는 것을 나타내기 위한 것이다. 사용자는 태블릿(전자 디지타이저)(164), 마이크(163), 키보드(162) 및 통상적으로 마우스, 트랙볼(trackball) 또는 터치 패드라고 하는 포인팅 장치(161) 등의 입력 장치를 통해 명령 및 정보를 컴퓨터(110)에 입력할 수 있다. 다른 입력 장치(도시 생략)로는 조이스틱, 게임 패드, 위성 안테나, 스캐너 등을 포함할 수 있다. 이들 및 기타 입력 장치는 종종 시스템 버스에 결합된 사용자 입력 인터페이스(160)를 통해 처리 장치(120)에 접속되지만, 병렬 포트, 게임 포트 또는 USB(universal serial bus) 등의 다른 인터페이스 및 버스 구조에 의해 접속될 수도 있다. 모니터(191) 또는 다른 유형의 디스플레이 장치도 비디오 인터페이스(190) 등의 인터페이스를 통해 시스템 버스(121)에 접속될 수 있다. 모니터(191)는 또한 터치 스크린 인터페이스(192) 등의 인터페이스를 통해 컴퓨터 시스템(110)에 필기 등의 디지털화된 입력을 입력할 수 있는 터치 스크린 패널(193) 등과 일체화될 수 있다. 유의할 점은 모니터 및/또는 터치 스크린 패널이, 태블릿 유형 퍼스널 컴퓨터에서와 같이, 컴퓨팅 장치(110)가 포함되어 있는 하우징에 물리적으로 결합될 수 있다는 것이며, 여기서 터치 스크린 패널(193)은 본질적으로 태블릿(164)으로서 기능한다. 그에 부가하여, 컴퓨팅 장치(110) 등의 컴퓨터는 스피커(195) 및 프린터(196) 등의 기타 주변 출력 장치를 포함할 수 있고, 이들은 출력 주변장치 인터페이스(194)를 통해 접속될 수 있다.

[0028] 컴퓨터(110)는 원격 컴퓨터(180)와 같은 하나 이상의 원격 컴퓨터로의 논리적 접속을 사용하여 네트워크화된 환경에서 동작할 수 있다. 원격 컴퓨터(180)는 퍼스널 컴퓨터, 서버, 라우터, 네트워크 PC, 피어 장치 또는 기타 통상의 네트워크 노드일 수 있고, 통상적으로 컴퓨터(110)와 관련하여 상술된 구성요소들의 대부분 또는 그 전부를 포함하지만, 도 1에는 메모리 저장 장치(181)만이 도시되어 있다. 도 1에 도시된 논리적 접속으로는 LAN(171) 및 WAN(173)이 있지만, 기타 네트워크를 포함할 수도 있다. 이러한 네트워킹 환경은 사무실, 전사적 컴퓨터 네트워크(enterprise-wide computer network), 인트라넷, 및 인터넷에서 일반적인 것이다.

[0029] LAN 네트워킹 환경에서 사용될 때, 컴퓨터(110)는 네트워크 인터페이스 또는 어댑터(170)를 통해 LAN(171)에 접속된다. WAN 네트워킹 환경에서 사용될 때, 컴퓨터(110)는 통상적으로 인터넷과 같은 WAN(173)을 통해 통신을 설정하기 위한 모뎀(172) 또는 기타 수단을 포함한다. 내장형 또는 외장형일 수 있는 모뎀(172)은 사용자 입력 인터페이스(160) 또는 기타 적절한 메커니즘을 통해 시스템 버스(121)에 접속될 수 있다. 네트워크화된 환경에서, 컴퓨터(110) 또는 그의 일부와 관련하여 기술된 프로그램 모듈은 원격 메모리 저장 장치에 저장될 수 있다. 예로서, 도 1은 원격 애플리케이션 프로그램(185)이 메모리 장치(181)에 있는 것으로 도시하고 있지만 이에 제한되는 것은 아니다. 도시된 네트워크 접속은 예시적인 것이며 이 컴퓨터들 사이에 통신 링크를 설정하는 기타

수단이 사용될 수 있다는 것을 잘 알 것이다.

[0030] 예시적인 아키텍처

[0031] 한 측면은 일반적으로 컴퓨터 시스템을 통해 매끄럽고 복잡한 애니메이션 및/또는 미디어를 제공하는 것에 관한 것이다. 이를 위해, 도 2에 전체적으로 제공되어 있는 바와 같이, 미디어 통합 아키텍처(media integration architecture)(200)가 제공된다. 애플리케이션 프로그램, 제어 또는 기타 유사한 상위-레벨 프로그램 코드(예를 들어, 운영 시스템 컴포넌트의 사용자 인터페이스)(202)는 그래픽 정보에 액세스하기 위해(그래픽 정보를 기록 또는 판독하기 위해) 일련의 애플리케이션 프로그래밍 인터페이스(API)(204), 기타 등등을 통해 미디어 통합 계층 아키텍처(200)에 액세스한다. 유의할 점은, 본 명세서에 기술된 예들 중 다수가 API와 인터페이스하는 애플리케이션 프로그램을 언급하고 있지만, 기타 상위-레벨 프로그램 코드 및 컴포넌트(예를 들어, 운영 체제의 사용자 인터페이스)도 본 명세서에 기술된 하위-레벨 컴포넌트와 인터페이스할 수 있다는 것을 잘 알 것이다. 그 자체로서, 애플리케이션 프로그램, 사용자 인터페이스, 기타 등등이라고 말해지든지에 상관없이, 이러한 상위-레벨 프로그램 코드에 대한 어떠한 언급도 동등한 것으로 간주되어야만 한다.

[0032] 한 구현에서, 미디어 통합 아키텍처(200)는 상위-레벨 합성 및 애니메이션 엔진(high-level composition and animation engine)(206), 타이밍 및 애니메이션 컴포넌트(timing animation component)(208), 및 하위-레벨 합성 및 애니메이션 엔진(low-level composition and animation engine)(210)을 포함한다. 본 명세서에서 사용되는 바와 같이, 용어 "상위-레벨" 및 "하위-레벨"은 다른 컴퓨팅 시나리오에서 사용되는 것과 유사하며, 여기서 일반적으로 소프트웨어 컴포넌트가 상위 컴포넌트에 대해 하위일수록, 그 컴포넌트는 하드웨어에 더 가깝다. 따라서, 예를 들어, 상위-레벨 합성 및 애니메이션 엔진(206)으로부터 전송된 그래픽 정보는 하위-레벨 합성 및 애니메이션 엔진(210)에 수신될 수 있으며, 여기서 이 정보는 하드웨어를 비롯한 그래픽 서브시스템에 그래픽 데이터를 전송하는 데 사용된다.

[0033] 일반적으로, 상위-레벨 합성 및 애니메이션 엔진(본 명세서에서 상위-레벨 합성기 및 애니메이터(high-level compositor and animator) 또는 상위-레벨 엔진 또는 컴포넌트라고도 함)(206)은 애플리케이션 프로그램(202)에 의해 제공되는 그래픽 장면을 표현하는 디스플레이 요소 트리(display element tree)를 작성하는 반면, 타이밍 및 애니메이션 컴포넌트는 선언적(또는 기타) 애니메이션 및 타이밍 제어를 제공한다. 하위-레벨 합성 및 애니메이션 엔진(본 명세서에서 하위-레벨 합성기 및 애니메이터 또는 하위-레벨 엔진 또는 컴포넌트라고도 함)(210)은 다수의 애플리케이션의 장면들에 대한 렌더링을 작성하고, 렌더링 컴포넌트를 사용하여, 그래픽을 화면에 실제로 렌더링하는 것을 구현한다. 유의할 점은 여전히 상위 레벨에서 시간이 걸리거나 애플리케이션-관련된 렌더링을 하고 비트맵 등에 대한 참조를 하위 계층으로 전달할 수 있다는 것이다.

[0034] 상위-레벨 합성 및 애니메이션 엔진(206)은 요소 트리 구조를 작성하고 그 구조를 순회(traverse)하여, 하위-레벨 합성 및 애니메이션 엔진(210)으로 전달될 렌더링 명령어 및 간단한 애니메이션 구간(animation interval)을 생성한다. 상위-레벨 합성기에 의해 발생된 렌더링 명령어는 타이밍 및 애니메이션 정보를 포함할 수 있다. 하위-레벨 합성 및 애니메이션 엔진(210)은 렌더링 명령어 및 애니메이션 구간을 받아서 장면을 애니메이트, 렌더링 및 합성하는 일을 관리하고, 이어서 그 장면은 그래픽 서브시스템(예를 들어, 그래픽 소프트웨어 및 하드웨어)(212)에 제공된다. 다른 대안으로서 또는 로컬적으로 디스플레이된 출력에 부가하여, 상위-레벨 합성 및 애니메이션 엔진(206)(또는 이와 유사한 것)은 프린터(222) 기타 등등에 고정된 이미지 데이터(fixed image data)를 전송하기 위해 렌더링 및 애니메이션 명령어를 적절한 형식으로 하위-레벨 인쇄 코드(lower-level printing code)(220)에 제공할 수 있고 및/또는 원격 기계(228)로 전송하기 위해 렌더링 명령어 및 간단한 애니메이션 구간을 적절한 형식으로 하위-레벨 단말 전송 서버(lower-level terminal transport server)(226)에 제공할 수 있다. 유의할 점은 네트워크를 거쳐 더 풍성한 정보가 전달될 수도 있다는 것이다, 예를 들어, 어떤 네트워크 트래픽도 없이 원격 기계가 마우스 롤오버 효과(mouse rollover effects)를 로컬적으로 처리하게 하는 것이 바람직할 수 있다.

[0035] 이 구현에서, 미디어 통합 아키텍처(200)는 따라서 그래픽 처리를 다수의 레벨들로 분리하고, 이들 레벨 각각은 애플리케이션의 사용자 인터페이스, 기타 등등(202)이 매끄러운 애니메이션을 갖는 그래픽을 출력하는 것, 이 그래픽을 다른 애플리케이션의 그래픽과 합성하는 것, 및 비디오 프레임을 처리하는 것 모두를 할 수 있게 해주는 어떤 지능적인 그래픽 처리를 수행한다. 애니메이션 및/또는 합성은 오디오 출력과도 동기화될 수 있다. 예를 들어, 하위 레벨 컴포넌트에서 오디오를 프레임 레이트와 동기화시킴으로써, 오디오의 타이밍은 비디오 또는 그래픽의 타이밍에 맞게 본질적으로 정확할 수 있으며, 작업-스케쥴링된 복잡한 사전-처리가 리프레쉬 레이트(refresh rate)를 따라갈 수 있는 능력에 의존하지 않는다.

[0036] 도 3은 XAML-기반 코드 등의 마크업 코드(302)가 파서(parser)/번역기(translator)(304)에 의해 해석될 수 있는 한 구현을 나타낸 것이다. 일반적으로, 파서/번역기(304)는 요소 트리/속성 시스템(element tree/property system)(314)에 요소들을 추가하고, 이 요소들은 그 자신의 레이아웃을 행하는 시각 객체이다. 게다가, 마크업 코드 중 일부 또는 그 전부가 요구 시에 해석되기 보다는 컴파일될 수 있으며 그에 의해 효율성을 향상시킬 수 있다는 것에 유의한다.

[0037] 일반적으로, 요소는 속성 시스템, 트리거링 및/또는 레이아웃/프리젠테이션 시스템에 관여하는 요소 계층에서의 객체이다. 파서(304)는 태그들을 찾아내고 그 태그들이 요소 또는 자원 객체를 정의하는 데 도움이 되는지를 결정한다. 예를 들어, VisualBrush의 특수한 경우에, 동일한 태그가 그 태그들이 나타나는 상황에 따라, 예를 들어, 미국 특허 출원 제10/401,717호에 기술된 바와 같이, 복잡한 속성 구문(property syntax)으로 나타나는지 여부에 따라, 요소로 해석되거나 자원 객체로 해석될 수 있다.

[0038] 마크업에 인라인(inline)으로 존재하는 것에 부가하여, 자원 인스턴스(resource instance)는 다른 곳에(예를 들어, 로컬이거나 원격 네트워크 상에 있어 적절히 다운로드될 수 있는 마크업에 또는 파일에) 위치할 수 있으며, 이름(예를 들어, 텍스트 이름, 참조 또는 기타 적당한 식별자)에 의해 참조될 수 있다. 이와 같이, 장면 디자이너(scene designer)는, 복잡한 속성 구문에 의해 기술되는 요소들을 비롯하여, 요소 트리 내의 요소를 장면 전체에 걸쳐 재사용할 수 있다. 이하에서 기술하게 되는 바와 같이, 자원은 스토리보드(storyboard)를 포함할 수 있어, 스토리보드가 재사용될 수 있게 해준다.

[0039] 파서(304)는 필요에 따라 타입 변환기(type converter)(308)에 액세스함으로써 또한 지정된 파라미터를 객체 속성과 일치시킴으로써 복잡한 속성 구문으로 된 마크업을 처리하며, 그에 의해 장면 디자이너를 위해 복잡성을 처리한다. 따라서, 파서(304)는 객체를 설정할 뿐만 아니라 객체 상의 속성도 설정한다. 동일한 렌더링 모델이 요소 레벨과 API 레벨 사이에서 공유되기 때문에, 객체들 중 다수가 본질적으로 동일하다. 이것은 파싱/번역을 아주 효율적이도록 만들어주며, 또한 다른 유형의 프로그래밍 언어(예를 들어, C#과 같은 언어)가 마크업에서 그 자신의 구문으로 또한 그 역으로 용이하게 변환할 수 있게 해준다. 유의할 점은, 도 3에 나타낸 바와 같이, 다른 이러한 프로그래밍 언어(310)(컴파일된 마크업을 포함할 수 있음)는 요소들을 요소 트리(314)에 추가할 수 있거나, 시각 API 계층(316)과 직접 인터페이스할 수 있다.

[0040] 또한, 도 3에 나타낸 바와 같이, 요소 레벨 및 자원 레벨에서 프로그램하는 데 동일한 마크업(302)이 사용될 수 있다. 일반적으로, 요소 레벨은 장면 디자이너에게 완전한 프로그램가능성(full programmability), 상속(예를 들어, 스타일-시트 같은 특징)을 제공하는 속성 시스템의 사용, 및 트리거링(예를 들어, 이에 의해 요소는 사용자 입력 이벤트 또는 동작에 응답하여 그의 모습, 위치, 기타 등등을 변경하기 위한 첨부 코드를 가질 수 있음)을 제공한다. 그렇지만, 다양한 실시예들은 또한 자원-레벨 메카니즘도 제공하며, 이 자원-레벨 메카니즘에 의해 장면 디자이너는 본질적으로 요소 트리 및 프로그램을 직접 시각 API 계층으로 바로가게(shortcut) 해줄 수 있다. 요소-레벨 특징들이 필요하지 않은 많은 유형의 정적 형상, 이미지, 기타 등등에 있어서, 이것은 적절한 객체를 출력하는 보다 효율적이고 간단한 방법을 제공한다.

[0041] 애니메이션 및 미디어 출력을 제어하기 위해, 도 4 및 도 5를 참조하여 이하에 기술되는 바와 같이, 클록들을 포함하는 타이밍 트리가 유지된다. 일반적으로, 상위-레벨 합성기 및 애니메이터 엔진(206)은 올바른 출력을 렌더링하기 위해 하위 레벨들이 처리해야 하는 처리의 양을 상당히 간소화시킴과 동시에 처리해야 하는 데이터의 양을 상당히 감소시키는 복잡한 처리(때때로 컴파일이라고 함)를 수행한다. 그렇지만, 유의할 점은 상위 레벨에 의해 수행되는 처리의 양 및 유형이 하위 레벨의 부하(load), 구성 및 성능에 상당히 의존할 수 있다는 것이다. 예를 들어, 고성능 그래픽 하드웨어가 존재하는 경우, 상위 레벨은 더 적은 양의 처리를 할 수 있고, 그 역도 마찬가지이다. 상위-레벨 계층 및 하위-레벨 계층은 이들 인자에 적응적이다.

[0042] 일반적으로, 애니메이션은 상위-레벨 합성기 및 애니메이션 엔진(206)과 하위-레벨 합성기 및 애니메이션 엔진(210) 둘다에 의해 달성된다. 한 구현에서, 상위-레벨 엔진(206)은 장면을 순회하고 나중의 보간을 위해 구간을 갖는 애니메이션 파라미터를 갱신하며, 이들 간단화된 데이터 구조를 하위-레벨 엔진(210)으로 전달되는 명령어로 패키지화한다. 이것은 동기 및/또는 비동기 방식으로 행해질 수 있다. 구간 데이터는 타이밍 말단점(시작 및 종료 타이밍 데이터)는 물론, 렌더링 명령어에 대한 파라미터화된 값을 포함하는 것으로 생각될 수 있다. 유의할 점은 상위-레벨 엔진(204)이 요청된 보간의 일부 또는 그 전부를 수행할 수 있다는 것이다, 예를 들어, 보간 또는 기타 움직임 함수가 하위-레벨 엔진(210)이 처리하기에 너무 복잡한 경우, 또는 하위 레벨이 그에 부과되는 처리 요구를 따라갈 수 없는 경우, 상위-레벨 엔진이 계산들의 일부 또는 그 전부를 수행하고 원하는 결과를 달성하기 위해 하위 레벨에 간단화된 데이터, 명령어, 테셀레이션(tessellation)을 제공할 수

있다.

[0043] 하위 레벨이 보간을 수행하는 일반적인 경우에, 각각의 애니메이션 프레임에 대해, 하위-레벨 엔진(210)은 순간값을 획득하기 위해 파라미터 구간을 보간하고, 명령어(instruction)를 그래픽 장치에 의해 실행되는 렌더링 명령(rendering command)으로 디코딩한다. 그래픽 장치는 장면에 존재할 수 있는 임의의 비디오 프레임들을 추가하는 최종 장면을 작성한다. 디지털 저작권 관리에 의해 보호되는 컨텐츠 등의 기타 데이터도 추가될 수 있다.

[0044] 상위-레벨 엔진(206)은 따라서 장면 데이터-구조를 순회하고, 어떤 기간에 대한 각각의 애니메이트된 파라미터를 기술하는 구간을 계산하며, 이들 구간 및 간단화된 파라미터화된 드로잉 명령어를 하위-레벨 엔진(210)으로 전달한다. 파라미터 데이터는 시작 시간, 종료 시간, 보간기 및 보간 데이터를 포함한다. 예로서, 움직이는 것처럼 보이도록 이미지를 소거 및 재드로잉하지 않고, 상위-레벨 합성기및 애니메이션 엔진(206)은 이미지가 시간에 따라 어떻게 변해야 하는지, 예를 들어, 시작 좌표, 종료 좌표, 이미지가 이들 좌표 사이에서 이동해야 하는 시간량(구간), 및 선형 등의 움직임 함수(motion function)에 관하여 하위-레벨 합성기 및 애니메이션 엔진(210)에 명령할 수 있다(유의할 점은 애니메이션에 움직임이 필요하지 않다는 것이며, 그 이유는 정지한 객체가, 예를 들어, 그의 컬러 속성을 변경함으로써 애니메이트될 수 있기 때문이다). 하위-레벨 합성기 및 애니메이션 엔진(210)이 프레임들 간의 새로운 위치를 구하기 위해 보간을 하고, 이들을 그래픽 장치가 이해할 수 있는 드로잉 명령어로 변환하며, 이 명령들을 그래픽 장치로 전달한다. 상위-레벨 엔진(206)의 각각의 패스(pass)는 양호하게는 몇개의 프레임에 걸쳐 매끄러운 애니메이션을 수행하기 위해 하위-레벨 엔진(210)에 충분한 데이터를 제공한다.

[0045] 하위-레벨(예를 들어, 고속-틱(high-tick)) 엔진(210)은 상위-레벨 엔진(206)과 별도의 작업이다. 하위-레벨 엔진(210)은 상위-레벨 엔진(206)으로부터 장면을 기술하는 간단화된 파라미터화된 드로잉 명령어 및 파라미터 구간을 수신한다. 하위-레벨 엔진은 상위-레벨 엔진(206)에 의해 새로운 데이터 구조가 제공될 때까지 이들 데이터 구조를 유지하고 순회한다. 하위-레벨 엔진은 다수의 상위-레벨 엔진(206)에 서비스를 제공하여 각각에 대해 별도의 데이터 구조를 유지할 수 있다. 하위-레벨 엔진(210)과 상위-레벨 엔진(206) 간의 일대다 관계는 이 시스템이 다수의 장면을 동시에 매끄럽게 애니메이트할 수 있게 해준다.

[0046] 하위-레벨 엔진(210)은 상위-레벨 엔진이 제공하는 구간에 기초하여 본질적으로 순간적인 애니메이션 파라미터를 보간하고, 드로잉 명령어를 갱신하며, 모든 프레임에 대한 장면을 렌더링한다. 프레임들이 그래픽 하드웨어 화면 리프레쉬 레이트 등으로 프리젠테이션할 준비가 되도록 하기 위해 하위-레벨 엔진(210) 작업이 시스템 상에서 상위 우선순위로 실행된다. 하위-레벨 엔진(210)에 의해 수행되는 보간은 따라서 일반적으로 선형, 부분별 선형(piecewise linear), 큐빅 스플라인(cubic spline), 및 유사한 속도의 함수 등의 간단하고 고속인 함수로 제한된다.

[0047] 애니메이션 및 미디어와 관련하여, 도 4 및 도 5에 전체적으로 나타낸 바와 같이, 애플리케이션 프로그램(202) 등의 프로그램은 상위-레벨 컴포넌트(206)에 대해 클록 또는 클록 속성이라고 하는 타이밍 정보와 함께 애니메이션 속성값을 지정한다. 이하에서 기술하는 바와 같이, 기본적으로 임의의 독립적인 애니메이션 또는 미디어(예를 들어, 비디오 및 오디오 등의 선형 미디어)는 물론, 지정된 애니메이션을 조정하는 스토리보드는 상위-레벨 컴포넌트에 그것을 위해 유지되는 클록을 갖는다. 일반적으로, 저작자는 클록들을 동기화된 채로 유지하기 위해 적절한 경우 클록들로 인스턴스화되는 타임라인 데이터(timeline data)를 지정한다. 어떤 실시예에서, 스토리보드는 타임라인들의 그룹 또는 트리로 이루어지는 타임라인이다. 스토리보드는 그의 타임라인 트리가 다수의 요소들 상의 다수의 속성들을 대상으로 할 수 있도록 대상 지정 정보(targeting information)를 제공할 수 있다. 그에 부가하여, 어떤 실시예들에서 애니메이션은 타임라인의 유형이다.

[0048] 클록 속성은 주어진 클록과 나머지 타이밍 구조 간의 초기 동기화 관계를 정의하는 타이밍 속성을 포함한다. 도 5에 도시한 바와 같이, 상위-레벨 컴포넌트(206)는 애니메이션의 속성의 현재값을 결정하기 위해 상위-레벨 애니메이션 함수(520_H)(공용 언어 런타임(common language runtime, CLR) - 함수가 CLR 상에서 실행됨 - 이 함수의 동작을 제어할 수 있게 해주는 메타데이터의 형태로 정보를 제공하는 코드인 관리 코드(managed code)로 작성됨)를 호출할 수 있다. 고속 프레임 레이트 계산 동안에, 하위-레벨 컴포넌트(210)는 애니메이션의 현재 속성값을 결정하기 위해 엔진(414)에 의해 계산되는 과정을 갖는 유사한 (또는 동일한) 애니메이션 함수(520_L)를 호출한다. 유의할 점은, 대안의 구현에서, 애니메이션 함수가 예를 들어 하위-레벨 컴포넌트 내에 내장될 수 있다는 것이다.

[0049] 일반적으로, 애니메이션 및 선형 미디어는 동기화 프리미티브(primitive) 및 규칙(rule)에 의해 서로 관련되어

있는 일련의 클록과 연관되어 있다. 클록은 계층적으로 배열되어 있을 수 있다, 예를 들어, 애플리케이션 프로그램은 부모 클록을 가지며, 애플리케이션 프로그램의 애니메이트된 객체는 자식이고, 이는 차례로 다른 자식을 가질 수 있다. 클록의 속성이 정의되거나 수정될 때, 그 클록의 모든 자식이 영향을 받는다. 예를 들어, 부모 클록을 일시 정지시키면 그의 자식 클록 모두가 정지되고, 부모 클록의 속도를 2배로 하면 그의 자식 클록 모두의 속도가 2배로 된다.

[0050] 이들 클록 속성은 런타임 시에 애플리케이션에 의해 개시되는 상호적용적 제어 이벤트(interactive control event)를 포함하는 소스 이벤트에 의해 수정될 수 있다. 따라서, 각각의 클록이 애플리케이션에 의해 언제라도, 예를 들어, 사용자 입력에 응답하여 개별적으로 시작(start), 일시 정지(pause), 재개(resume), 및 정지(stop)될 수 있다는 점에서, 클록은 상호작용적이다. 그에 부가하여, 새로운 클록이 타이밍 구조에 추가될 수 있고, 기존의 클록들이 제거될 수 있다.

[0051] 상기한 미국 특허 출원 제10/693,822호에 기술된 바와 같이, 상위-레벨 타이밍 컴포넌트는 저장된 이벤트 리스트(시작(begin), 일시정지(pause), 기타 등등) 및 연관된 동기화 프리미티브에 기초하여 각각의 클록에 대한 구간 리스트(interval list)를 발생할 수 있다. 활성화 구간은 실세계 시간의 서로 다른 시점에서 주어진 클록에 의해 표현되는 시간을 기술하는 간단한 중첩하지 않는 세그먼트이다.

[0052] 도 4에 나타낸 바와 같이, 클록 속성들 중 적어도 일부는 클록의 타이밍 트리(402)에서 계층적으로 관련되어 있다. 그의 속성(즉, 클록 속성(404_1-404_3))을 갖는 3개의 이러한 클록이 도 4에 도시되어 있다. 그렇지만, 주어진 상황에서 더 많은 클록 및 대안의 타이밍 트리가 존재할 수 있다는 것을 잘 알 것이다. 예를 들어, 디스플레이될 애니메이트된 객체(또는 일련의 스토리보드-조정된 객체)에 대응할 수 있는 각각의 클록에 대해, 상위-레벨 컴포넌트(206)는 클록 속성으로부터 이벤트 리스트(예를 들어, 406_1)를 발생하는 이벤트 리스트 발생기(408)라고 하는 상태 기계(state machine)를 포함한다. 일반적으로, 이벤트 리스트 발생기(408)는 지정된 클록 속성들에 의해 처음에 스케쥴링되었던 이벤트들을 임의의 명시적인 상호작용적 이벤트(애니메이션과 관련하여 수신되는 일시정지 및 재개 요청 등)와 함께 이벤트 리스트로 그룹화한다. 각각의 애니메이션(또는 이하에 기술되는 바와 같이, 스토리보드를 통해 일군의 애니메이션)에 대해 클록이 유지되고, 따라서 각각의 독립적인 애니메이션/애니메이션 세트에 대응하는 이벤트 리스트는 물론 각각의 독립적인 선형 미디어에 대한 이벤트 리스트가 있다.

[0053] 상위-레벨 컴포넌트는 하위-레벨 컴포넌트(210)로 전달되는 대응하는 구간 리스트(예를 들어, 412_3)를 계산하기 위해 각각의 애니메이션 또는 미디어에 대해 이벤트 리스트(예를 들어, 406_3)를 사용하는 구간 발생기(410)를 포함한다. 차례로, 하위-레벨 컴포넌트(210)는, 그 객체에 대한 구간 데이터 및 현재 시간에 기초한 진행값을, 애니메이트된 객체의 변하는 속성에 대한 현재값을 결정하는 하위-레벨 애니메이션 함수 서브시스템(520_L)에 제공하는 등에 의해, 구간 데이터와 관련한 현재 시간에 기초하여 출력을 제어하는 하위-레벨 계산 엔진(414)을 포함한다. 예를 들어, 임의의 주어진 프레임에 대해, 하위-레벨 계산 엔진(414)은 구간 데이터 및 현재 시간(시스템 시간 또는 상대 시간일 수 있음)에 기초하여 애니메이트된 객체의 위치를 보간한다. 유의할 점은, 도 4 및 도 5에 도시된 바와 같이, 상위-레벨 타이밍 컴포넌트가 타이밍 트리(402) 및 이벤트 리스트(406_1-406_3)를 포함하는 반면, 하위-레벨 타이밍 컴포넌트(210)가 구간 리스트(412_1-412_3)를 포함한다는 것이다. 그렇지만, 이들 데이터 구조는 본질적으로 저장 장치 내의 아무 곳에나, 통상적으로는 고속 랜덤 액세스 메모리에 유지될 수 있다.

[0054] 요약하면, 상위-레벨 타이밍 컴포넌트(206)는 규칙 및 프리미티브를 동기화시킴으로써 (예를 들어, 계층적으로) 관련된 클록들의 트리(402)를 본다. 실시예들은 하위-레벨 타이밍 엔진에 의해 소비되는 활성화 구간(412)의 리스트를 컴파일하기 위해 상위-레벨 타이밍 컴포넌트(206) 내의 클록들을 사용한다. 하위-레벨 타이밍 엔진은 독립적인(예를 들어, 애니메이트된 객체별 또는 선형 미디어별) 클록들을 나타내는 구간 리스트(412)를 본다. 도 5에 도시된 바와 같이, 시스템 클록(524) 등의 둘다에 일관된 시간을 제공함으로써 다수의 레벨들이 동기화된 채로 있도록 하는 클록이 있다.

[0055] 애플리케이션(202)(또는 어떤 다른 소스)으로부터 수신된 상호작용적 변화에 응답하여, 상위-레벨 타이밍 컴포넌트는 그 변화와 직접 연관되어 있는 클록, 또한 동기화 프리미티브에 의해 지정된 바와 같이 간접적으로 연관되어 있는 임의의 클록의 상태를 갱신할 필요가 있다. 유의할 점은 단일의 클록에 대한 변화가 타이밍 구조 내의 다른 클록들에 영향을 줄 수 있다는 것이다. 예를 들어, 슬라이드쇼의 제공자가 전체 디스플레이를 일시 정

지할 수 있고, 그에 의해 현재 디스플레이되고 있는 임의의 애니메이션 및 선형 미디어가 일시 정지될 필요가 있다. 이하에서 기술하는 바와 같이, 일 실시예에 따르면, 스토리보드는 애니메이션 및/또는 미디어가 함께 그룹화되고 따라서 이러한 원하는 방식으로 조정될 수 있게 해준다.

[0056] 일반적으로, 객체에 대해 상호작용이 일어날 때마다, 필요한 경우 이벤트 리스트가 재생성되고 구간 리스트가 재계산되어 하위-레벨 타이밍 컴포넌트로 전송된다. 하위-레벨 컴포넌트(예를 들어, 초당 60회로 동작함)가 각각의 애니메이트된 객체에 대한 현재의 구간 리스트를 처리하기만 하면 되도록 상위-레벨 컴포넌트는 (예를 들어, 초당 6 내지 10회 정도의 주파수로) 이들 동작을 수행한다.

[0057] 이 구간 정보 및 객체에 대한 시작값 및 종료값에 부가하여, 부가적인 파라미터가 상위-레벨 컴포넌트로 전송될 수 있고 그 결과 구간 리스트에 변화가 있게 된다. 예를 들어, 애플리케이션은 이벤트에 대응하는 기간을 탐색할 수 있다. 애플리케이션은 탐색이 언제 행해져야 하는지 및 얼마나 행해져야 하는지를 지정한다. 미디어의 되감기 또는 고속감기와 같이, 애니메이션에 관한 탐색에 있어서, 애플리케이션은 이에 따라 특정의 시점으로 앞으로 또는 뒤로 점프할 수 있다. 역방향 이벤트(reverse event)는 애플리케이션이 지정할 수 있는 다른 유형의 스케쥴링된 또는 상호작용적 이벤트이다. 역방향 이벤트가 리스트에 있는 경우, 진행이 자동적으로 반대로 된다, 예를 들어, 100%에서 0%로 된다. 속도 변화도 지원된다, 예를 들어, 어떤 클록보다 낮은 모든 것이 실제 시간보다 더 빨리 또는 더 느리게 실행되도록 설정될 수 있다. 특정의 클록에 대해 재시작이 허용되는지 및/또는 현재 실행되지 않는 경우 다시 시작해야 하는지, 예를 들어, 1초에 시작하고, 10초에 다시 시작하고, 100초에 다시 시작하고, 이하 마찬가지로 해야 하는지, 그렇지만 현재 실행 중인 경우 선택적으로 재시작할지 재시작하지 말아야 할지를 나타내기 위해 다른 애트리뷰트(attribute)가 지정될 수 있다.

[0058] 애니메이션 및 미디어를 조정하는 스토리보드

[0059] 실시예들은, 그 중에서도 특히, 요소(예를 들어, 창, 패널 등의 FrameworkElement, 버튼 등의 컨트롤, 기타 등등)에 대해 타임라인 트리가 지정될 수 있게 해주는 객체들을 포함하는 Storyboard(스토리보드)에 관한 것이다. 타임라인은 요소의 속성값을 설정하고 요소 및 그의 서브트리에 대한 애니메이션을 제어한다. Storyboard는 기본적으로 시작, 정지 또는 일시 정지, 또는 탐색하는 데 사용될 수 있는 특정 유형의 타임라인 객체에 대응하며, 그에 의해 그 Storyboard와 연관되어 있는 애니메이트된 요소의 외관 및/또는 미디어의 거동을 제어한다. Storyboard는 마크업 및 코드 둘다로부터 이용가능하다.

[0060] 어떤 실시예들에서, 애니메이션 및 Storyboard에 대해 항상 트리거되는(always-triggered) 모델이 사용된다. 그에 부가하여, 이 모델은 핸드오프 거동(도 6 내지 도 9와 관련하여 이하에서 기술함)을 가능하게 해주기 위해 애니메이션 및 Storyboard를 트리거와 직접적으로 관계시킨다(이전의 구현에서와 같이 스토리보드 속성과 반대임). 핸드오프 거동은 기본적으로 트리거되는 개념이다. 이하에서 기술하는 바와 같이, 비지정된 "from" 값은 새로운 애니메이션이 트리거될 때 속성에 존재하는 애니메이션의 현재값 및 새로운 애니메이션의 핸드오프 거동에 기초하여 정의된다. 이것은 애니메이션의 "from" 값(지정되지 않은 경우)이 애니메이션이 트리거될 때만 정의된다는 것을 의미한다.

[0061] 애니메이션이 트리거될 때만 이 속성이 정의되기 때문에, 이 모델은 애니메이션을 트리거와 연관시킨다. 이 모델에서, 애니메이션은 트리거에 인라인(inline)으로 있거나 Resources에 정의되고 트리거될 때까지 사용되지 않는다.

[0062] 어떤 실시예에서, Storyboard는 요소와 연관되어 있음과 동시에 독립적으로 시작될 수 있는 상위-레벨 타임라인들의 집합체를 포함한다. Storyboard는 또한 Storyboard 내의 애니메이션이 Storyboard가 첨부되는 요소와 관련하여 그의 대상을 분석할 수 있게 해주는 타이밍 트리와 요소 트리 간의 연결점(connection point)을 제공한다. 마크업으로부터, Storyboard는 또한 상태 변화, 이벤트 또는 기타 동작들에 의해 활성화될 수 있고 트리거가 활성화될 때(예를 들어, 타임라인의 시작 또는 종료) 실행되는 일련의 확장가능 동작들을 포함하는 Trigger의 집합체에 선언될 수 있다. Storyboard는 또한 요소, Style 또는 Template의 Resources 속성에 선언될 수 있다.

[0063] 일반적으로, Storyboard 내의 타임라인들은 여러가지 서브트리, 요소 및 기타 객체(예를 들어, 텍스트 문자, 단어, 라인, 단락, 기타 등등)에 적용될 수 있는 애니메이션 템플릿을 나타낸다. 일 실시예에 따르면, 스토리보드(예를 들어, API와 연관된 객체)는 프로그램 저작자가 타임라인 내의 애니메이션을 (요소 속성으로) 그룹화함으로써 조정된 애니메이션 세트들을 스케쥴링할 수 있다. 환언하면, 스토리보드는 타이밍 트리를 요소 트리와 통합시켜, 동일한 타임라인을 통해 조정된 방식으로 서로 다른 애니메이션의 서로 다른 속성에 대한 타이밍

을 처리할 수 있다. 따라서, 스토리보드는 다양한 객체 및 속성에 영향을 주는 그렇지 않았으면 독립적인 타임라인들을 단일의 타임라인 트리로 결합시키는 방법을 제공하여, 프로그램 저작자가 복잡한 타이밍 이벤트 세트를 간단한 방식으로 구성 및 제어할 수 있게 해준다. 예를 들어, 개개의 애니메이션을 찾아내어 재시작해야만 하기보다는, Storyboard의 일부로서 타임라인 아래에 있는 애니메이션들의 그룹화는 Storyboard를 단순히 재시작함으로써 모든 애니메이션을 한꺼번에 재시작할 수 있게 해준다.

[0064] Storyboard는 애니메이션/비디오/오디오의 특정의 인스턴스에 대해 또는 특정의 스타일의 애니메이션에 대해 사용될 수 있다. 어떤 시나리오에서, Resource로서 선언된 단일의 Storyboard가 이들 요소에 대한 개별적인 트리거를 통해 다수의 Element에 적용될 수 있다. BeginStoryboard는 마크업 또는 코드 중 어느 하나로부터의 속성에 대해 애니메이션(또는 기타 미디어)을 시작한다. 마크업으로부터, BeginStoryboard는 FrameworkElement.BeginStoryboard를 이용하여 그의 Storyboard 속성에 할당된 Timeline의 트리를 순회한다. FrameworkElement.BeginStoryboard는 이어서 Storyboard의 Timeline 트리의 리프(leaf)들에 있는 임의의 애니메이션에 대해 FrameworkElement.BeginAnimation을 호출한다. 마크업에서, BeginStoryboard는 이벤트 및 속성 트리거 둘다에서 사용될 수 있다.

[0065] 마크업으로부터, 저작자는 BeginStoryboard를 사용하여 Storyboard를 시작할 수 있다. BeginStoryboard.Storyboard는 Storyboard를 포함하며, 이 Storyboard의 첨부된 속성은 다수의 요소에 대한 다수의 속성을 대상으로 할 수 있다. Storyboard는 애니메이션 및 MediaTimeline를 비롯한 모든 유형의 Timeline을 포함할 수 있다.

[0066] 이하의 예는 인라인 Storyboard를 나타낸 것이다. 컨테이너 객체(Button)가 애니메이트된다. 이 예에서, Storyboard.TargetName이 제공되지만, 어떤 실시예에서, 이것이 필요하지 않은데 그 이유는 애니메이션이 컨테이너를 대상으로 하기 때문이다. BeginStoryboard.Storyboard 태그는 자동적으로 제공되며 따라서 이들이 필요하지 않다.

[0067] 이 실시예에서, Storyboard.TargetName이 지정되지 않은 경우, 모든 애니메이션은 컨테이너 객체를 대상으로 한다. 게다가, 이들은 Source가 지정되지 않은 경우 트리거링 이벤트의 Source로서 컨테이너를 사용한다. 이들 2개의 가정은 컨테이너-대상 시나리오 및 컨테이너-트리거 시나리오에서 마크업의 간단화를 가능하게 해준다(TargetName 또는 Source를 가질 필요가 없다).

```
<Window>
    <Button Width="2" Name="myButton">
      <Button.Triggers>
        <EventTrigger Event="Loaded">
          <EventTrigger.Actions>
            <BeginStoryboard >
              <Storyboard>
                <DoubleAnimation To="4" Duration="0:0:1"
Storyboard.TargetProperty="Width"
```

[0068] Storyboard.TargetName="myButton"/>

[0069] // 이 경우에 TargetName이 필요하지 않은데 그 이유는 기본 대상(default target)이 컨테이너이기 때문이다.

```
              </Storyboard>
            </BeginStoryboard >
          </EventTrigger.Actions>
        </EventTrigger>
      </Button.Triggers>
    </Button>
</Window>
```

[0070]

[0071] 이하의 예는 Resource로서 정의된 Storyboard를 나타낸 것이다. 애니메이션이 Resources 섹션에 정의되기 때문에, 이는 다른 Framework Element의 애니메이션에서 재사용될 수 있다.

```
<Window>
    <Button Width="2">
      <Button.Resources>
        <DoubleAnimation To="4" Duration="0:0:1" x:Key="myAnimation" />
      </Button.Resources>
      <Button.Triggers>
        <EventTrigger Event="Loaded">
          <EventTrigger.Actions>
            <BeginStoryboard Storyboard ="{StaticResource myAnimation}"
TargetProperty="Width"/>
          </EventTrigger.Actions>
        </EventTrigger>
      </Button.Triggers>
    </Button>
</Window>
```

[0072]

[0073] 상기 예에 나타낸 바와 같이, 타이밍 트리를 요소 트리와 통합하는 것에 부가하여, 스토리보드는 이벤트 트리거의 사용이 그의 동작(예를 들어, 시작, 일시 정지, 탐색 및 정지 동작)을 제어할 수 있게 해준다. 그에 부가하여, 스토리보드는 속성 트리거와 함께 사용될 수 있다. 일반적으로, 속성 트리거는 true 또는 false 등의 어떤 상태/값에 대응하는 반면, 이벤트가 발생(fire)된다(따라서 false 값을 갖지 않는다). 예를 들어, 속성 트리거는 마우스가 요소 상에 놓여져 있을 때 true 값을 가질 수 있고, 요소 상에 놓여져 있지 않을 때 false 값을 가질 수 있다. 스토리보드(또는 스토리보드들)에서 시작, 정지, 일시 정지 및/또는 탐색을 개별적으로 행하기 위해 각각의 true 또는 false 값이 사용될 수 있다. 요소 상에서의 마우스 클릭 등의 이벤트가 발생되고 따라서 가변 상태를 갖지는 않지만 그와 마찬가지로 스토리보드의 동작을 제어할 수 있다. 이와 같이, 애니메이션은 디스플레이된 요소 트리 표현과의 사용자 상호작용을 통해 제어가능하다.

[0074] 이벤트 트리거는 이벤트가 호스트 요소에 대해 발생될 때 동작들이 행해지게 하는 그 이벤트를 그의 소스로서 받는다. 유의할 점은, 속성 트리거와 달리, 이벤트 트리거가 스코핑(scope)되지 않고, 동작을 수행하며, 이어서 그 동작이 그 자신의 수명을 결정한다는 것이다. 예를 들어, 하나의 이벤트는 타임라인을 시작할 수 있고, 그 타임라인은 그의 지속기간이 완료될 때까지 또는 다른 이벤트 트리거, 기타 등등이 그 타임라인을 정지시킬 때까지 실행할 수 있다.

[0075] 도 6은 일 실시예에 따른, 화려한 미디어 간의 매끄러운 전환을 제공할 수 있는 시스템(600)을 나타낸 것이다. 이 실시예에서, 시스템(600)은, 이전에 기술된 애플리케이션(202), 타이밍 시스템(208), 그래픽 서브시스템(212), 및 속성 시스템(314)에 부가하여, 핸드오프 시스템(604)을 갖는 애니메이션 시스템(602)을 포함한다. 다양한 실시예에 따른, 이러한 전환에서 사용되는 이들 컴포넌트의 기능들 중 일부에 대해 이하에서 기술한다.

[0076] 속성 시스템(314)은, 앞서 기술한 바와 같이, 애플리케이션(202)에 의해 인스턴스화되는 다양한 시각 객체(도시 생략)의 속성에 대한 값을 저장할 수 있다. 예를 들어, 속성 시스템(314)은, 객체 속성들 중에서도 특히, UI에 디스플레이될 버튼을 나타내는 객체의 길이 및 폭 속성에 대한 기본값을 저장할 수 있다.

[0077] 타이밍 서브시스템(208)은, 앞서 기술한 바와 같이, 애플리케이션(202)에 의해 생성된 애니메이션의 타이밍을 조정할 수 있다. 예를 들어, 타이밍 시스템(208)은 애니메이트되고 있는 속성들의 현재값을 계산하는 데 사용되는 타임라인 정보를 제공할 수 있다. 일 실시예에서, 타이밍 시스템(208)은 애니메이션를 발생하는 데 사용되는 트리거(예를 들어, 이벤트 트리거, 속성 트리거)를 발생하기 위해 상호작용적 요소(도시 생략)로부터 이벤트 및 속성 변화 정보를 수신할 수 있다. 예를 들어, 상호작용적 요소는 마우스 및/또는 키보드 동작을 모니터링하고 타이밍 시스템(208)에 정보를 제공하여 타이밍 시스템(208)이 타임라인을 시작하게 하고 애니메이션 시스템(602)에 트리거를 제공하게 할 수 있다.

[0078] 그래픽 서브시스템(212)은, 앞서 기술한 바와 같이, 디스플레이 장치 상에 애니메이션을 렌더링하는 데 필요한 데이터를 발생할 수 있다.

[0079] 이 실시예에서, 애니메이션 시스템(602)은 시각적인 애니메이트된 효과를 생성하는 속성값을 변화시키는 동작을 수행한다. 일 실시예에서, 애니메이션 시스템은 상위 레벨 및 하위 레벨 합성 및 애니메이션 엔진(206, 208)은 물론 핸드오프 시스템(604) 등의 컴포넌트를 포함한다. 또한, 애니메이션 시스템(602)은 애니메이트된 속성에 대한 지정된 "from" 및 "to" 값은 물론 애니메이션을 수행하는 지정된 기간을 애플리케이션(202)으로부터 수신할 수 있다. 어떤 애니메이션에서, 애플리케이션에 의해 지정되지 않을 때, 애니메이트된 속성에 대한 기본

"to" 및/또는 "from" 값이 사용될 수 있다.

[0080] 이 실시예에서, 다른 애니메이션이 어떤 속성에 대해 여전히 수행되고 있는 동안에 그 속성에 대해 애니메이션이 수행되어야 할 때 핸드오프 시스템(604)이 사용될 수 있다. 몇몇 실시예에서, 핸드오프 시스템(604)은 이하에서 표 1에 열거되어 있는 것 등의 몇가지 다른 핸드오프를 수행할 수 있다. 용어 "이전의(old)" 애니메이션은 애플리케이션(202)이 동일한 속성에 대해 다른 애니메이션을 시작할 때 특정의 속성에 대해 실행 중인 애니메이션을 말한다. 어떤 시나리오에서, 이전의 애니메이션은 2개 이상의 애니메이션의 "합성"(즉, 표 1에 정의된 바와 같은 합성)일 수 있다. 몇몇 실시예들에서, 애니메이션 시스템(602)은 하나 이상의 미리 정해진 시나리오에 대한 기본 핸드오프 거동을 제공하도록 구현될 수 있다. 이 실시예에서, 표 1은 애니메이션에서 설정될 수 있는 핸드오프 거동 속성의 여러가지 값을 기술하고 있다. 핸드오프 거동 속성은 기본적으로 애니메이션의 "from" 값의 값으로서 무엇을 사용할지 및 이전의 애니메이션으로 무엇을 할지를 기술한다.

표 1

[0081]

핸드오프	거동
작성(compose)	이전의 애니메이션이 계속되고 각각의 렌더링 사이클에 대해 새로운 애니메이션이 이전의 애니메이션의 출력값을 그의 "from" 값으로서 사용한다. 새로운 애니메이션의 출력값이 속성을 렌더링하는 값으로서 사용된다.
대체(replace)	이전의 애니메이션이 해제되고 값들이 지정되지 않은 경우 새로운 애니메이션이 기본 "to" 및 "from" 값을 사용한다.
스냅샷(snapshot) 및 대체	이전의 애니메이션의 현재값이 획득되고(즉, 스냅샷) 이전의 애니메이션이 해제된다. 이어서, 새로운 애니메이션은, 지정되지 않은 경우, "to" 값에 대한 기본값 및 "from" 값으로서 스냅샷을 사용한다.
스냅샷 및 유지(retain)	이전의 애니메이션의 스냅샷이 획득되고 이전의 애니메이션은 애니메이트된 속성의 값에 영향을 주지 않고 계속 실행될 수 있다. 새로운 애니메이션은, 지정되지 않은 경우, "to" 값에 대한 기본값 및 "from" 값으로서 스냅샷을 사용한다. 새로운 애니메이션이 종료될 때, 이전의 애니메이션은 속성의 애니메이션을 재개할 수 있다.
스냅샷 및 유지/일시정지(paused)	이전의 애니메이션의 스냅샷이 획득되고, 이전의 애니메이션이 일시 정지된다. 새로운 애니메이션은, 지정되지 않은 경우, "to" 값에 대한 기본값 및 "from" 값으로서 스냅샷을 사용한다. 새로운 애니메이션이 종료될 때, 이전의 애니메이션은 일시정지될 때 애니메이션의 상태로부터 속성을 애니메이트할 수 있다.
블렌딩(blend)	애니메이션의 가중 평균을 사용한다.
큐잉(queue)	기존의 애니메이션이 완료될 때까지 대기중인 애니메이션을 큐잉(queue up)한다.

[0082] 버튼 예를 한번 더 사용하여, 마우스 커서가 버튼 상으로 이동할 때 버튼이 특정의 기간에 걸쳐 특정의 크기로 확대(grow)된 다음에 마우스 커서가 버튼에서 벗어날 때 특정의 기간에 걸쳐 원래의 크기로 축소(shrink)되게 하도록 애플리케이션이 작성될 수 있다. 이 시나리오에서, 사용자는 특정의 기간이 완전히 경과하기 이전에 커서를 버튼 상으로 또 버튼 밖으로 재빨리 이동시킨다. 따라서, 애니메이션 "확대"는 "MouseOver" 시에 시작하고 애니메이션 "축소"는 확대 애니메이션이 완료되기 이전에 "MouseExit" 시에 시작한다.

[0083] 애니메이션 "확대"에 대한 핸드오프 거동이 상기 표 1에 기술된 "스냅샷 및 대체" 거동(지정되거나 기본값일 수 있음)인 경우, 커서가 버튼에서 벗어날 때, 애니메이션 "확대"의 결과 얻어지는 버튼의 길이 및 폭의 현재값이 스냅샷된다. 이 실시예에서, 스냅샷된 값은 애니메이션 저장 객체(606)에 저장된다. 이 실시예에서, 애니메이션 시스템(602)은 각각의 애니메이트된 속성에 대해 애니메이션 저장 객체(606)를 생성한다. 몇몇 실시예에서, 애니메이션 저장 객체(602)는 그래픽 서브시스템(212)에 의해 렌더링될 때 버튼이 확대 및 축소되는 것처럼 보이게 하는 길이 및 폭 속성의 변화에 대한 계산을 수행하도록 구성되어 있다.

[0084] 애니메이트된 속성에 대한 값이 스냅샷된 후에, 애니메이션 "확대"가 해제되고(즉, 애니메이션 "확대"가 사실상 종료되고), 애니메이션 "축소"가 시작되며, 이 때 길이 및 폭 속성에 대한 "from" 값은 스냅샷된 값이 된다. 버튼의 폭 및 길이 속성에 영향을 주는 어떤 다른 애니메이션도 시작되지 않는 경우, 애니메이션 "축소"가 완료되고, 애니메이션의 끝에서 버튼의 길이 및 폭 속성이 기본값에 있게 된다.

[0085] "스냅샷 및 대체" 거동은 애니메이션 "확대"와 "축소" 간의 전환이 사용자에게 매끄럽게 보이도록 한다. 즉, 버튼의 외관이 어떤 크기로 매끄럽게 확대되고 이어서 그 동일한 크기로 매끄럽게 축소된다. 상기한 스냅샷 기능을 갖지 않는 기타 시스템에서, 애플리케이션(202) 또는 애니메이션 시스템(602)은 애니메이션 "축소"에 대한 "from" 값(예를 들어, 애니메이션 "확대"의 기본 "from" 값 또는 목표값)을 제공해야만 할 가능성이 있으며, 이 결과 버튼의 외관에 예기치 않은 "글리치"가 생길 가능성이 있다. 즉, 버튼의 외관이 어떤 크기로 확대되지만,

이어서 갑자기 다른 크기로 변하는 것처럼 보이며, 이 크기로부터 버튼 크기가 축소되기 시작한다. 스냅샷(snapshotting)의 상기한 이점은 또한 이상에서 표 1에 기술한 "스냅샷 및 유지" 및 "스냅샷 및 보유/일시정지" 거동에 적용될 수 있다.

[0086] 이하는 "스냅샷 및 대체" 핸드오프 거동을 사용하는 코드의 일례이다. 이 버튼은 MouseEnter 시에 확대되고 MouseLeave 시에 축소되며, 이 경우 스냅샷 및 대체 거동은 기본 핸드오프 거동이다.

[0087]
```
<Grid xmlns="http://schemas.microsoft.com/winfx/avalon/2005"
xmlns:x="http://schemas.microsoft.com/winfx/xaml/2005">
 <Grid.Resources>
  <Style x:Key="GelButton" TargetType="{x:Type Button}">
   <Setter Property="Button.RenderTransform">
    <Setter.Value>
     <ScaleTransform Center="0,0" ScaleX="1" ScaleY="1" />
    </Setter.Value>
   </Setter>
   <Style.Triggers>
    <EventTrigger RoutedEvent="Mouse.MouseEnter">
     <EventTrigger.Actions>
      <BeginStoryboard>
       <Storyboard>
        <DoubleAnimation Duration="0:0:0.2"
```

[0088]
```
Storyboard.TargetProperty="RenderTransform.ScaleX" To="1.5"
AccelerationRatio=".9" />
        <DoubleAnimation Duration="0:0:0.2"
Storyboard.TargetProperty="RenderTransform.ScaleY" To="1.5"
AccelerationRatio=".9" />
       </Storyboard>
      </BeginStoryboard>
     </EventTrigger.Actions>
    </EventTrigger>
    <EventTrigger RoutedEvent="Mouse.MouseLeave">
     <EventTrigger.Actions>
      <BeginStoryboard>
       <Storyboard>
        <DoubleAnimation Duration="0:0:0.2"
```

[0089]
```
Storyboard.TargetProperty="RenderTransform.ScaleX" AccelerationRatio=".9" />
        <DoubleAnimation Duration="0:0:0.2"
Storyboard.TargetProperty="RenderTransform.ScaleY" AccelerationRatio=".9" />
       </Storyboard>
      </BeginStoryboard>
     </EventTrigger.Actions>
    </EventTrigger>
   </Style.Triggers>
  </Style>
 </Grid.Resources>
 <Button Style="{StaticResource GelButton}" Width="100" Height="100"/>
</Grid>
```

[0090] 예시적인 "스냅샷 및 대체" 동작 흐름

[0091] 도 7은 일 실시예에 따른, "스냅샷 및 대체" 거동을 수행하는 동작 흐름(700)을 나타낸 것이다. 동작 흐름(700)은 임의의 적당한 컴퓨팅 환경에서 수행될 수 있다. 예를 들어, 동작 흐름(700)은 시스템(600)(도 6) 등의 시스템에 의해 실행될 수 있다. 따라서, 동작 흐름(700)의 설명은 도 6의 컴포넌트들 중 적어도 하나를 참조할 수 있다. 그렇지만, 도 6의 컴포넌트들에 대한 이러한 참조는 어느 것이라도 단지 설명을 위한 것에 불과하며, 도 6의 구현이 동작 흐름(700)에 대한 비제한적인 환경이라는 것을 잘 알 것이다.

[0092] 블록(702)에서, 애니메이션에 대한 트리거가 검출된다. 일 실시예에서, 애니메이션 시스템(602)(도 6) 등의 애니메이션 시스템은 트리거(예를 들어, 이벤트 트리거 또는 속성 트리거)를 검출한다. 몇몇 실시예들에서, 애니메이션 시스템은 상호작용적 요소들(도 6과 관련하여 상기함)로부터 트리거의 통지를 수신한다. 이 예시적인 동작 흐름에서, 애니메이션과 연관된 속성(즉, 시각 객체가 "애니메이트"될 때 변하는 속성)에 대해 현재 실행 중인 다른 애니메이션이 없다.

[0093] 블록(704)에서, 애니메이션 저장 객체가 할당된다. 일 실시예에서, 애니메이션 시스템은 애니메이션과 연관된 속성에 대한 애니메이션을 처리하기 위해 애니메이션 저장 객체(606)(도 6) 등의 애니메이션 객체를 생성한다. 몇몇 실시예에서, 애니메이트될 각각의 속성에 대해 애니메이션 저장 객체가 생성될 수 있다. 속성에 대한 애니메이션들 모두가 완료되었을 때, 몇몇 실시예들에서 애니메이션 저장 객체가 삭제될 수 있다. 몇몇 실시예들에서, 애니메이션(들)이 채움값(fill value)(들)에 도달할 때, 이 채움값(들)이 애니메이션 저장 객체에 유지될 수 있지만 저장된 정보의 나머지 정보가 삭제될 수 있다.

[0094] 블록(706)에서, 트리거된 애니메이션과 연관된 속성의 기본값이 애니메이션 저장 객체에 저장된다. 일 실시예에서, 애니메이션 시스템은 애니메이션 저장 객체에 기본값을 저장한다. 기본값은 속성 시스템(314)(도 6) 등의 속성 시스템으로부터 획득된다.

[0095] 블록(708)에서, 애니메이션이 시작되고, 이 애니메이션이 애니메이션 저장 객체에 의해 유지된다. 일 실시예에서, 시각 객체가 애니메이트되는 것처럼 보이도록, 애니메이션 저장 객체는 사실상 애니메이션이 완료(또는 대체 또는 일시 정지)될 때까지 화면 리프레쉬 레이트로 속성에 대한 값을 재계산한다. 애니메이션 저장 객체는 또한 애니메이션이 실행 중인 동안 애니메이션을 유지한다(즉, 애니메이션이 유지 또는 유지/일시정지되면 애니메이션이 발견되어 계속될 수 있도록 애니메이션의 현재 상태를 정의하는 정보를 유지한다).

[0096] 블록(710)에서, 동일한 속성을 포함하는 제2 애니메이션에 대한 트리거는 제1 애니메이션이 완료되기 이전에 검출된다. 블록(702)에서와 같이, 일 실시예에서, 애니메이션 시스템은 이 다른 트리거를 검출한다. 이 예시적인 동작 흐름에서, 제2 애니메이션은 "스냅샷 및 대체" 거동에 대해 표 1에 정의된 것 등의 핸드오프 거동을 갖도록 구성되어 있다.

[0097] 블록(712)에서, 제1 애니메이션의 결과 얻어지는 속성의 현재값이 저장된다(즉, 스냅샷된다). 일 실시예에서, 스냅샷은 애니메이션 저장 객체에 저장된다. 예를 들어, 애니메이션 시스템은 제1 애니메이션에 다라 그 다음 화면 리프레쉬에 대한 속성에 대한 값을 계산하고 이 계산된 값(즉, 스냅샷)을 애니메이션 저장 객체에 저장할 수 있다.

[0098] 블록(714)에서, 제1 애니메이션이 해제된다. 일 실시예에서, 애니메이션 시스템은 애니메이션 저장 객체에 저장되었던 제1 애니메이션의 상태 정보를 삭제한다.

[0099] 블록(716)에서, 제2 애니메이션이 시작되고 제2 애니메이션이 애니메이션 저장 객체에 유지된다. 이 실시예에 따르면, 제2 애니메이션을 실행하고 있을 때 스냅샷이 객체의 "from" 값으로서 사용된다. 블록(708)의 동작과 유사하게, 시각 객체가 애니메이트되는 것처럼 보이도록, 애니메이션 시스템은 사실상 애니메이션이 완료(또는 대체 또는 일시 정지)될 때까지 화면 리프레쉬 레이트로 속성에 대한 값을 재계산한다. 애니메이션 저장 객체는 또한 제2 애니메이션이 실행 중인 동안 제2 애니메이션을 유지한다(즉, 제2 애니메이션이 유지 또는 유지/일시정지되면 제2 애니메이션이 발견되어 계속될 수 있도록 제2 애니메이션의 현재 상태를 정의하는 정보를 유지한다).

[0100] 동작 흐름(700)이 특정의 순서로 순차적으로 예시되고 기술되어 있지만, 다른 실시예들에서, 블록들에 기술된 동작들은 다른 순서로, 여러번, 및/또는 병렬로 수행될 수 있다. 게다가, 몇몇 실시예들에서, 블록들에 기술된 하나 이상의 동작들이 다른 블록으로 분리되거나, 생략되거나, 결합될 수 있다.

[0101] 도 8은 일 실시예에 따른, "스냅샷 및 대체" 핸드오프 거동(표 1)에 따라 애니메이션과 연관된 속성에 대한 값이 어떻게 획득되는지를 나타낸 것이다. 이 실시예에서, 애니메이트되지 않는 속성(802)이 기본값(804)을 갖는다. 일 실시예에서, 이 기본값은 이전에 기술된 속성 시스템(314)(도 3 및 도 6) 등의 속성 시스템에 저장된다.

[0102] 속성(802)에 영향을 주는 애니메이션(808)(도 8에 애니메이션 A로 표시됨)이 시작될 때, 기본값(804)이 일반적으로 애니메이션(808)에 대한 "from" 값(810)으로서 사용된다. 애니메이션(808)의 "to" 값(812)은 애니메이션(808)을 생성한 애플리케이션(예를 들어, 도 6의 애플리케이션(202) 등)에 의해 지정될 수 있다. 몇몇 시나리

오에서, "to" 및 "from" 값은 애플리케이션에 의해 지정되지 않은 채로 있을 수 있으며, 그 결과 기본적으로 기본값을 사용한다. 일 실시예에서, "to" 및 "from" 값은 애니메이션 저장 객체(606)(도 6) 등의 애니메이션 저장 객체에 저장되며, 이 애니메이션 저장 객체는 애니메이션(1108)이 시작될 때 애니메이션 시스템(602)(도 6) 등의 애니메이션 시스템에 의해 할당될 수 있다. 애니메이션(808)이 실행 중인 동안, 화면이 리프레쉬될 때마다 속성값(애니메이션(808)에 따라 변화됨)이 계산되고 렌더값(render value)(814)으로서 출력된다.

[0103] 이 예에서, 애니메이션(808)이 완료되기 이전에 애니메이션(816)(도 8에 애니메이션 B로 표시됨)이 시작된다. 이 예에서, 애니메이션(816)은 "스냅샷 및 대체" 거동을 갖도록 구성되어 있다. 시작될 때, 애니메이션(816)은 속성을 스냅샷하고 이 스냅샷을 애니메이션(816)의 "from" 값(818)으로서 사용한다. 애니메이션(816)의 "to" 값(820)은 애니메이션(816)을 생성한 애플리케이션에 의해 지정될 수 있다. 다른 시나리오에서, "to" 값(820)은 비지정되어, 기본 "to" 값이 사용되게 할 수 있다. 몇몇 실시예들에서, 이들 "to" 및 "from" 값은 상기한 애니메이션 저장 객체에 저장될 수 있다. 게다가, 애니메이션(816)만이 속성값에 영향을 주도록 애니메이션(808)이 해제된다. 애니메이션(816)이 실행 중인 동안에, 화면이 리프레쉬될 때마다 속성값(애니메이션(816)에 따라 변화됨)이 계산되고 렌더값(822)으로서 출력된다.

[0104] 핸드오프 거동이 "스냅샷 및 유지"이면, 애니메이션(808)은 해제되지 않는다. 오히려, 애니메이션(808)은 계속하여 실행할 수 있지만, 속성값이 애니메이션(816)만을 사용하여 계산된다(즉, 애니메이션(808)이 무시된다). 이어서, 애니메이션(816)이 완료될 때, 애니메이션(808)은 애니메이션(816)에 의해 발생된 최신 속성값을 사용하여 재개된다. 이와 유사하게, 핸드오프 거동이 "스냅샷 및 유지/일시 정지"이면, 애니메이션(808)이 해제되지 않는다. 오히려, 애니메이션(808)은 애니메이션(816)이 시작될 때 가졌던 상태에서 일시 정지될 수 있으며, 속성값이 애니메이션(816)만을 사용하여 계산된다. 이어서, 애니메이션(816)이 완료될 때, 애니메이션(808)은 애니메이션(808)이 일시 정지되었을 때의 속성값을 사용하여 재개된다.

[0105] 도 9는 일 실시예에 따른, 계층화된 애니메이션들 간의 관계를 나타낸 것이다. 이 실시예에서, 논리적으로 여러 층으로 배열되어 있는 몇개의 애니메이션을 가질 수 있는 애플리케이션(도 6의 애플리케이션(202) 등)에 의해 시각 객체(900)가 정의된다. 이 예에서, 시각 객체는 계층(902-1 내지 902-N)을 포함한다. 계층(902-1)은 애니메이션(904-1A) 및 애니메이션(904-1B)를 포함하고, 계층(902-2)은 애니메이션(904-2A) 및 애니메이션(904-2B)를 포함하며, 이하 마찬가지로하여, 계층(902-N)은 애니메이션(904-NA) 및 애니메이션(904-NB)를 포함한다. 이 예에서, N개의 계층이 있고 각각의 계층이 2개의 애니메이션을 갖지만, 계층의 수 및 계층마다의 애니메이션의 수는 애플리케이션 개발자의 설계 선택사항에 따라 다를 수 있다.

[0106] 일 실시예에서, (속성 트리거와 반대로) 이벤트 트리거에 의해 시작되는 애니메이션들 전부를 포함하도록 제1 계층(902-1)이 정의된다. 따라서, 계층(2 내지 N)의 애니메이션은 속성 트리거에 의해 시작된다. 몇몇 실시예들에서, 각각의 계층은 단일의 속성 트리거와 연관되어 있으며 그 속성 트리거에 의해 시작되는 애니메이션(들)을 포함한다. 게다가, 이 실시예에서, 실행 중인 애니메이션은 제1 계층(902-1)부터 제N 계층(902-N)로의 순서로 수행된다. 게다가, 이 실시예에서, 한 계층 내의 애니메이션들은 스냅샷(예를 들어, 표 1에 기술된 "스냅샷 및 대체", "스냅샷 및 유지", "스냅샷 및 유지/일시정지") 및/또는 합성을 포함하는 핸드오프 거동으로 정의될 수 있는 반면, 계층들 간의 전환은 합성 핸드오프 거동을 갖는다. 합성 핸드오프 거동의 예는 이하에서도 10과 관련하여 기술된다.

[0107] 몇몇 실시예들에서, 계층 내에서의 합성의 우선순위(precedence)는 사전 정의될 수 있다. 상위 우선순위 애니메이션에 의해 출력되는 값은 그 다음 하위 우선순위 애니메이션에 대한 "from" 값으로서 사용된다.

[0108] 일 실시예에서, 우선순위는 다음과 같이 정의된다. (1) 스타일로부터의 트리거된 애니메이션. 이들이 마크업 또는 코드에서 나타나는 순서가 계층 내의 다수의 "스타일" 트리거된 애니메이션에 대한 우선순위를 정의함, (2) 로컬 요소 트리거들로부터의 트리거된 애니메이션. 이들이 마크업 또는 코드에서 나타나는 순서가 계층 내의 다수의 "로컬 요소" 트리거된 애니메이션에 대한 우선순위를 정의함, 및 (3) 스토리보드 또는 메서드 호출로부터 적용되는 애니메이션. 이들이 적용되는 순서가 "스토리보드/메서드 호출 적용된" 애니메이션에 대한 우선순위를 정의함.

[0109] 몇몇 실시예들에서, 계층의 애니메이션(들) 전부가 종료될 때, 계층이 제거될 수 있다. 계층은 계층들의 "적층" 내의 임의의 위치로부터 제거될 수 있다.

[0110] 도 10은 일 실시예에 따른, 2개의 애니메이션의 합성을 나타낸 것이다. 이 실시예에서, 비애니메이트된 속성(1002)은 기본값(1004)을 갖는다. 일 실시예에서, 기본값은 앞서 기술한 속성 시스템(314)(도 3, 도 6) 등의

속성 시스템에 저장된다.

[0111] 속성에 영향을 미치는 애니메이션(1008)(도 10에 애니메이션 A로 표시됨)이 시작될 때, 기본값(1004)은 일반적으로 애니메이션(1008)에 대한 "from" 값(1010)으로서 사용된다. 애니메이션(1008)의 "to" 값(1012)은 애니메이션(1008)을 생성한 애플리케이션(예를 들어, 도 6의 애플리케이션(202) 등)에 의해 지정될 수 있다. 일 실시예에서, "to" 및 "from" 값은 애니메이션 저장 객체(606)(도 6) 등의 애니메이션 저장 객체에 저장된다. 도 10에 도시되어 있지 않지만, 애니메이션(1008)이 단독으로 실행되고 있는 동안, 화면이 리프레쉬될 때마다 속성값(애니메이션(1008)에 따라 변화됨)이 계산되고 그래픽 서브시스템(212)(도 3, 도 6) 등의 그래픽 서브시스템에 "렌더"값으로서 제공된다.

[0112] 이 예에서, 애니메이션(1008)이 완료되기 이전에 애니메이션(1016)(도 10에 애니메이션 B로 표시됨)이 시작된다. 이 예에서, 애니메이션(1016)은 "작성(compose)" 거동을 갖도록 구성되어 있다. "스냅 및 대체" 거동과 반대로, 속성의 순간값이 계속하여 갱신되도록 애니메이션(1008)이 계속하여 실행된다. 그렇지만, 애니메이션(1008)의 결과 얻어지는 속성의 순간값이 "렌더"값으로서 사용되지 않는다. 오히려, 애니메이션(1008)의 결과로서 계산되는 순간값이 일종의 중간값인 애니메이션(1016)의 "from" 값(1018)으로서 사용된다. 애니메이션(1016)의 "to" 값(1020)은 애니메이션(1016)을 생성한 애플리케이션에 의해 지정될 수 있다. 몇몇 실시예들에서, "to" 및 "from" 값은 상기한 애니메이션 저장 객체에 저장될 수 있다. 애니메이션(1016)의 결과 얻어지는 속성값은 이어서 렌더값(1022)로서 사용된다. 애니메이션(1008, 1016)이 실행 중인 동안, 화면이 리프레쉬될 때마다 속성값(애니메이션(1008, 1016)의 합성에 따라 변화됨)이 계산되고 렌더값(1022)으로서 출력된다.

산업상 이용 가능성

[0113] 본 명세서에 전체에 걸쳐 특정의 기술된 특징, 구조 또는 특성이 적어도 하나의 실시예에 포함되어 있음을 의미하는 "일 실시예", "한 실시예" 또는 "예시적인 실시예"가 언급되고 있다. 따라서, 이러한 어구의 사용은 단지 하나 이상의 실시예를 말할 수 있다. 게다가, 기술된 특징, 구조 또는 특성이 하나 이상의 실시예들에서 임의의 적당한 방식으로 결합될 수 있다.

[0114] 그렇지만, 당업자라면 본 발명이 구체적인 상세 중 하나 이상이 없이 또는 다른 방법, 자원, 재료, 기타 등등으로 실시될 수 있다는 것을 잘 알 것이다. 다른 예에서, 단지 본 발명의 측면들을 불명확하게 하는 것을 방지하기 위해 공지의 구조, 자원 또는 동작이 상세히 도시되거나 기술되어 있지 않다.

[0115] 예시적인 실시예 및 응용이 도시되고 기술되어 있지만, 본 발명이 상기한 정확한 구성 및 자원에 한정되지 않는다는 것을 잘 알 것이다. 청구된 발명의 범위를 벗어나지 않고 본 명세서에 기술된 방법 및 시스템의 구성, 동작 및 상세에서 당업자에게 명백한 다양한 수정, 변경 및 변형이 행해질 수 있다.

도면의 간단한 설명

[0007] 도 1은 여러가지 실시예들이 포함될 수 있는 예시적인 컴퓨터 시스템을 나타낸 블록도.

[0008] 도 2는 일 실시예에 따른 미디어 통합 계층 아키텍처(media integration layer architecture)를 나타낸 블록도.

[0009] 도 3은 일 실시예에 따른, 시각 API 계층과 인터페이스하기 위해 마크업 언어 코드를 인터프리트하는 컴포넌트를 나타낸 도면.

[0010] 도 4 및 도 5는 일 실시예에 따른, 클록 속성을 진행 데이터(progress data)를 결정하는 데 사용하기 위한 구간으로 변환하는 타이밍 컴포넌트를 갖는 2-레벨 아키텍처를 전체적으로 나타낸 블록도.

[0011] 도 6은 일 실시예에 따른, 화려한 미디어 간의 매끄러운 전환을 제공하는 시스템을 전체적으로 나타낸 블록도.

[0012] 도 7은 일 실시예에 따른, 핸드오프 동작을 수행하는 데 있어서의 동작 흐름을 전체적으로 나타낸 흐름도.

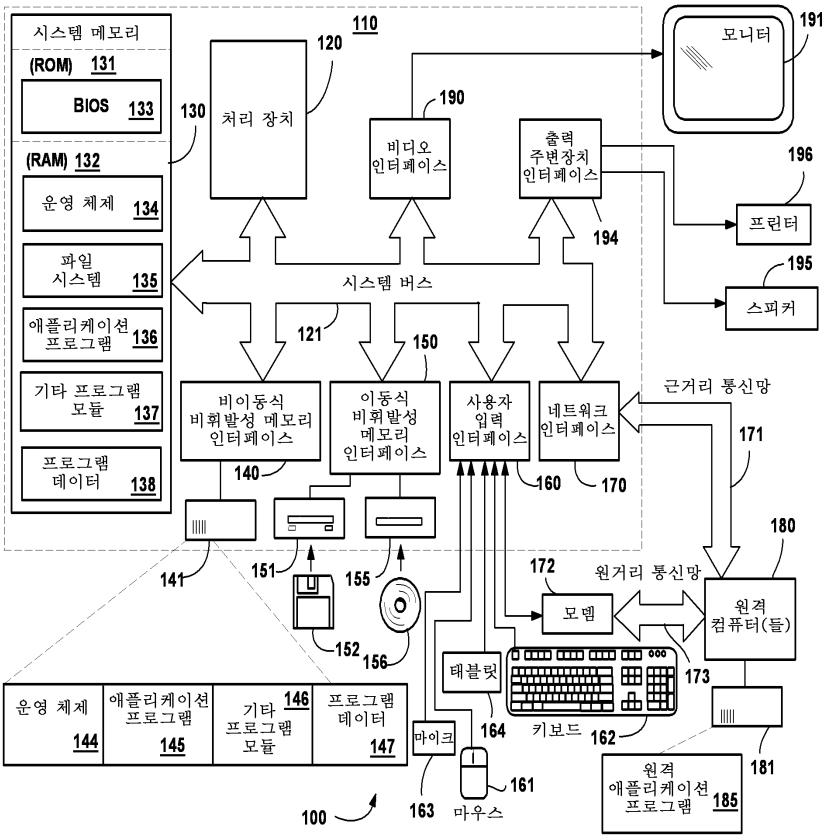
[0013] 도 8은 일 실시예에 따른, 핸드오프 동작을 사용하는 2개의 애니메이션 간의 전환을 나타낸 도면.

[0014] 도 9는 일 실시예에 따른, 계층화된 애니메이션들(layered animations) 간의 관계를 나타낸 도면.

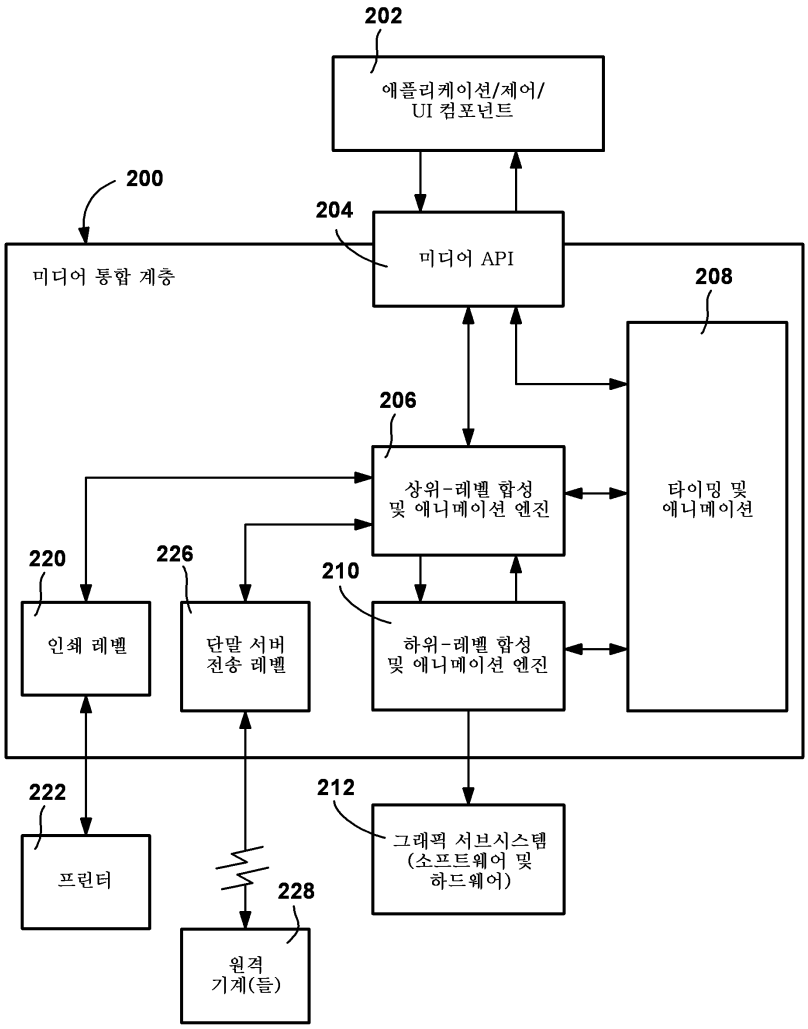
[0015] 도 10은 일 실시예에 따른, 2개의 애니메이션의 합성을 나타낸 도면.

도면

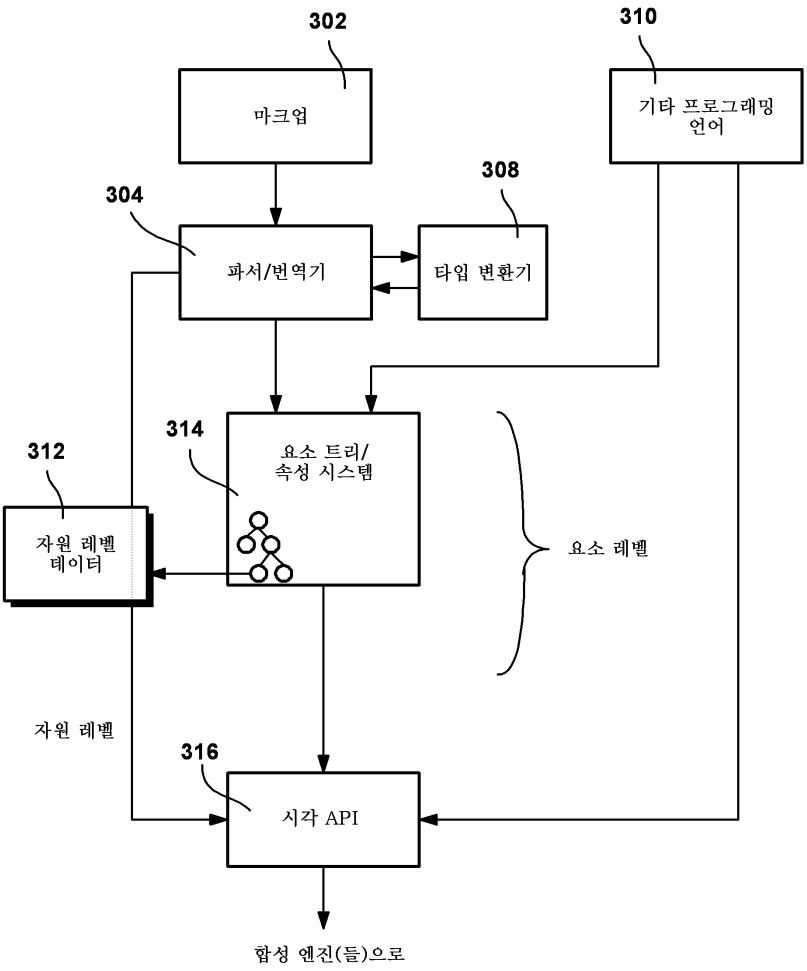
도면1


시스템 메모리
(ROM) 131
BIOS 133
(RAM) 132
운영 체제 134
파일 시스템 135
애플리케이션 프로그램 136
기타 프로그램 모듈 137
프로그램 데이터 138
130
110
처리 장치 120
190
비디오 인터페이스
출력 주변장치 인터페이스
194
191
모니터
196
프린터
195
스피커
시스템 버스
121
150
비이동식 비휘발성 메모리 인터페이스
이동식 비휘발성 메모리 인터페이스
사용자 입력 인터페이스
네트워크 인터페이스
160
170
근거리 통신망
171
140
141
151
155
152
156
172
원거리 통신망
모뎀
180
원격 컴퓨터(들)
태블릿
164
키보드
162
173
마이크
163
마우스
161
181
원격 애플리케이션 프로그램 185
운영 체제 144
애플리케이션 프로그램 145
기타 프로그램 모듈 146
프로그램 데이터 147
100

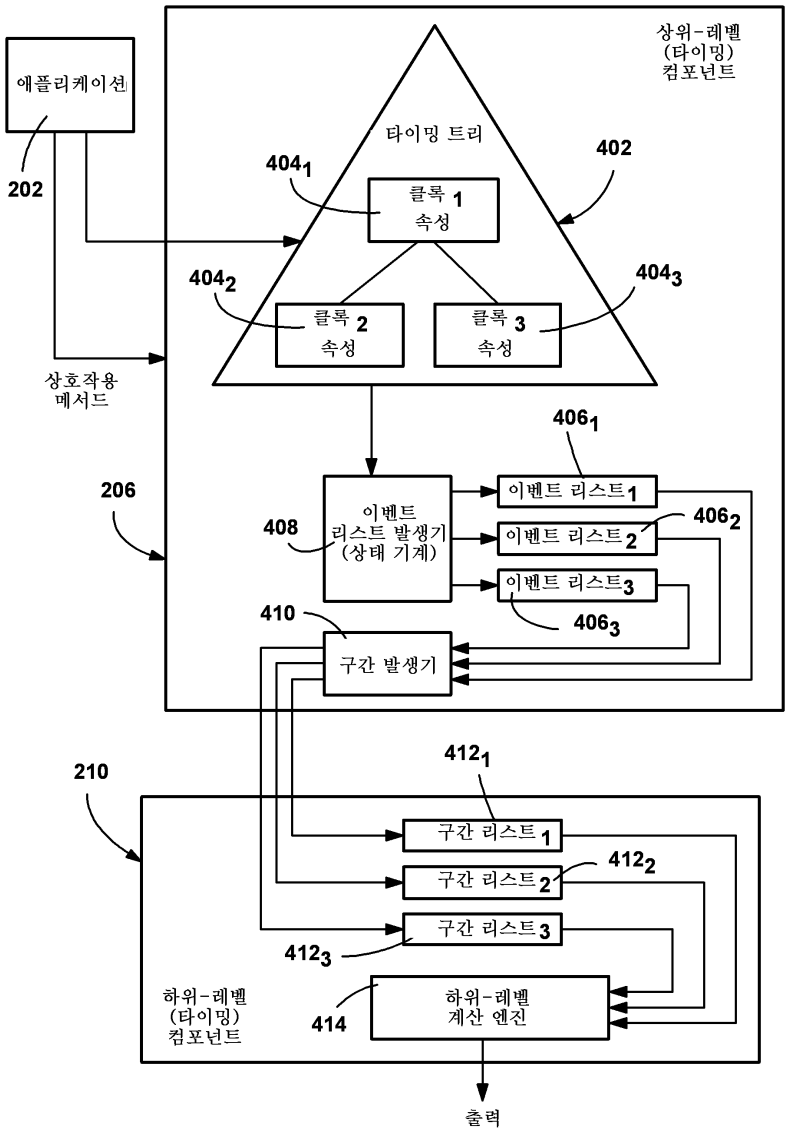
도면2


202
애플리케이션/제어/
UI 컴포넌트
200
204
미디어 API
미디어 통합 계층
208
206
상위-레벨 합성
및 애니메이션 엔진
타이밍 및
애니메이션
220
226
210
인쇄 레벨
단말 서버
전송 레벨
하위-레벨 합성
및 애니메이션 엔진
222
212
그래픽 서브시스템
(소프트웨어 및
하드웨어)
228
프린터
원격
기계(들)

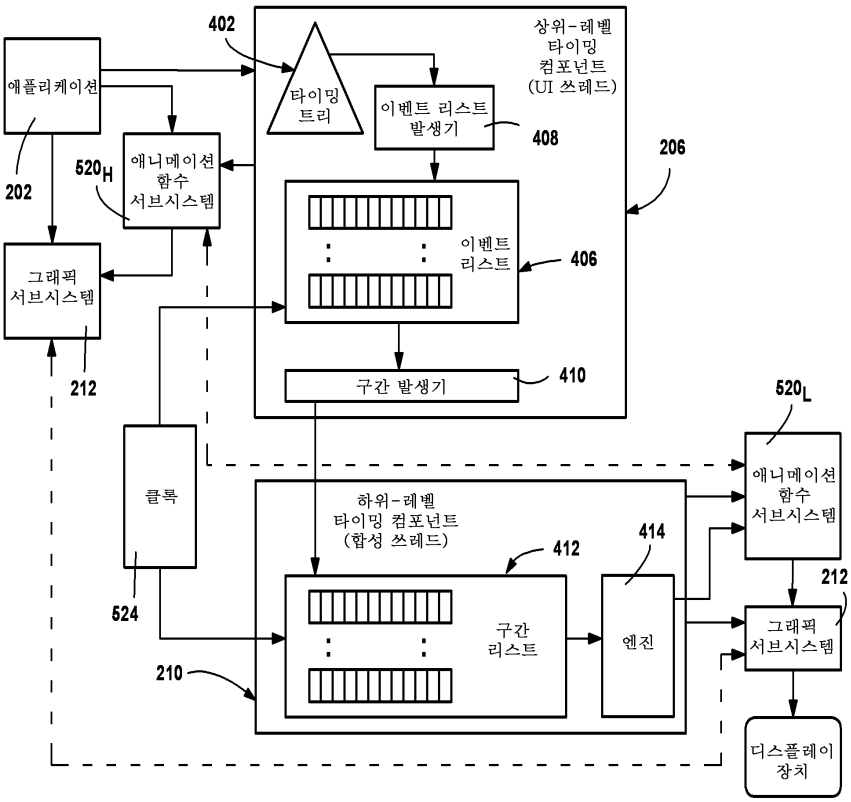
도면3


302
마크업
310
기타 프로그래밍 언어
304
308
파서/번역기
타입 변환기
314
요소 트리/
속성 시스템
312
자원 레벨
데이터
요소 레벨
자원 레벨
316
시각 API
합성 엔진(들)으로

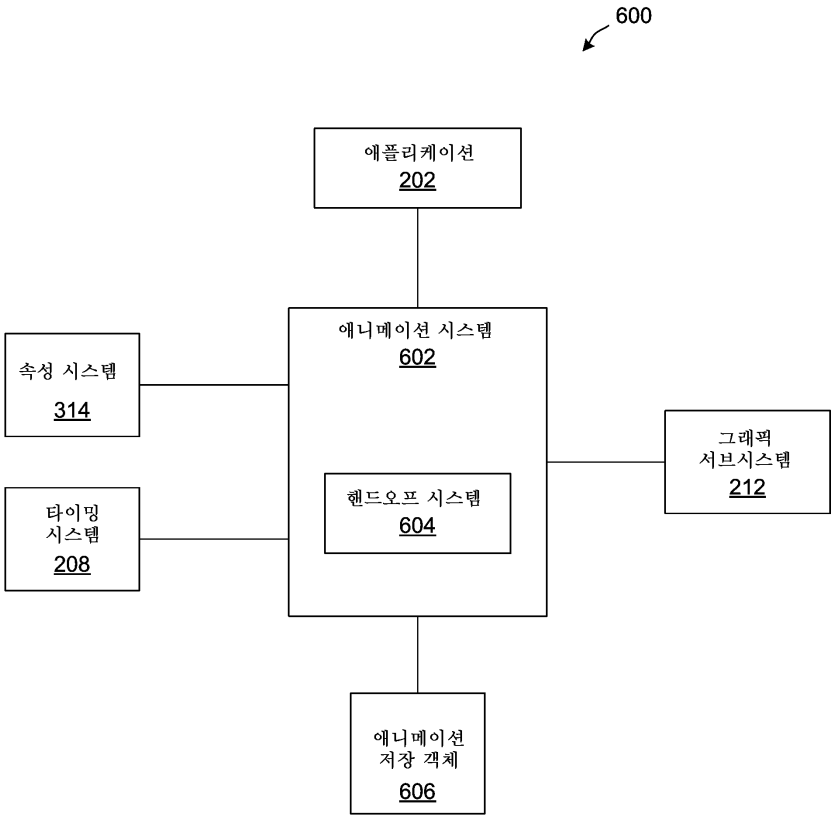
도면4


애플리케이션
202
상위-레벨
(타이밍)
컴포넌트
타이밍 트리
402
404_1
클록 1
속성
404_2
클록 2
속성
404_3
클록 3
속성
상호작용
메서드
206
406_1
이벤트 리스트1
406_2
이벤트 리스트2
이벤트 리스트3
406_3
408
이벤트
리스트 발생기
(상태 기계)
410
구간 발생기
210
412_1
구간 리스트1
412_2
구간 리스트2
구간 리스트3
412_3
하위-레벨
(타이밍)
컴포넌트
414
하위-레벨
계산 엔진
출력

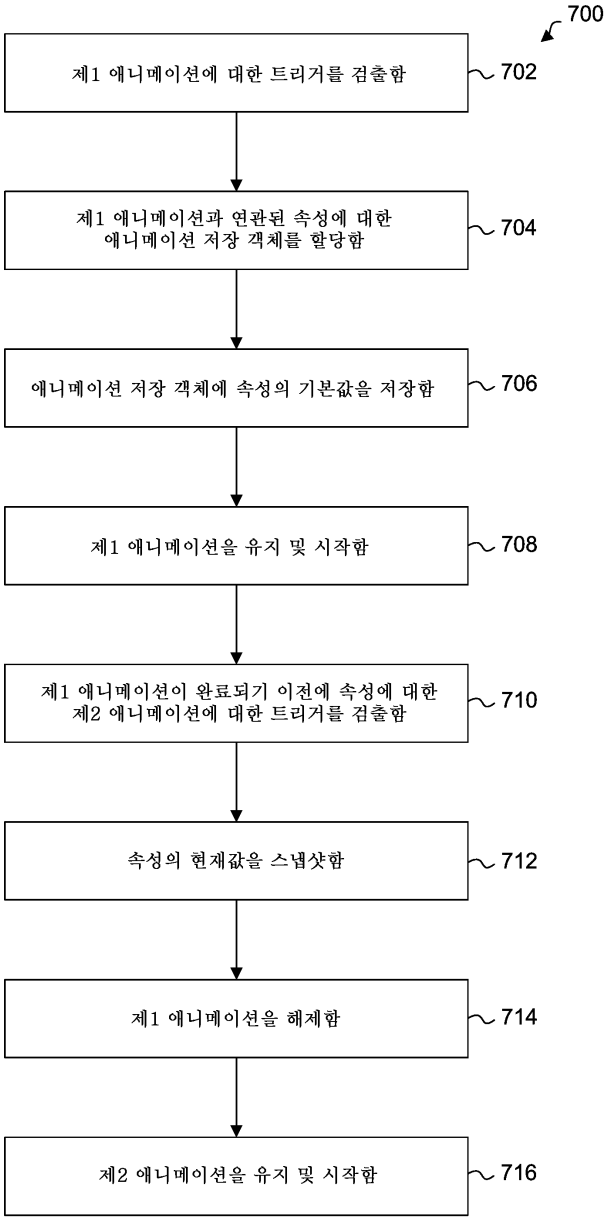
도면5


402
상위-레벨
타이밍
컴포넌트
(UI 쓰레드)
애플리케이션
타이밍
트리
이벤트 리스트
발생기
408
206
202
520H
애니메이션
함수
서브시스템
이벤트
리스트
406
그래픽
서브시스템
212
구간 발생기
410
520L
클록
하위-레벨
타이밍 컴포넌트
(합성 쓰레드)
412
414
애니메이션
함수
서브시스템
212
524
구간
리스트
엔진
그래픽
서브시스템
210
디스플레이
장치

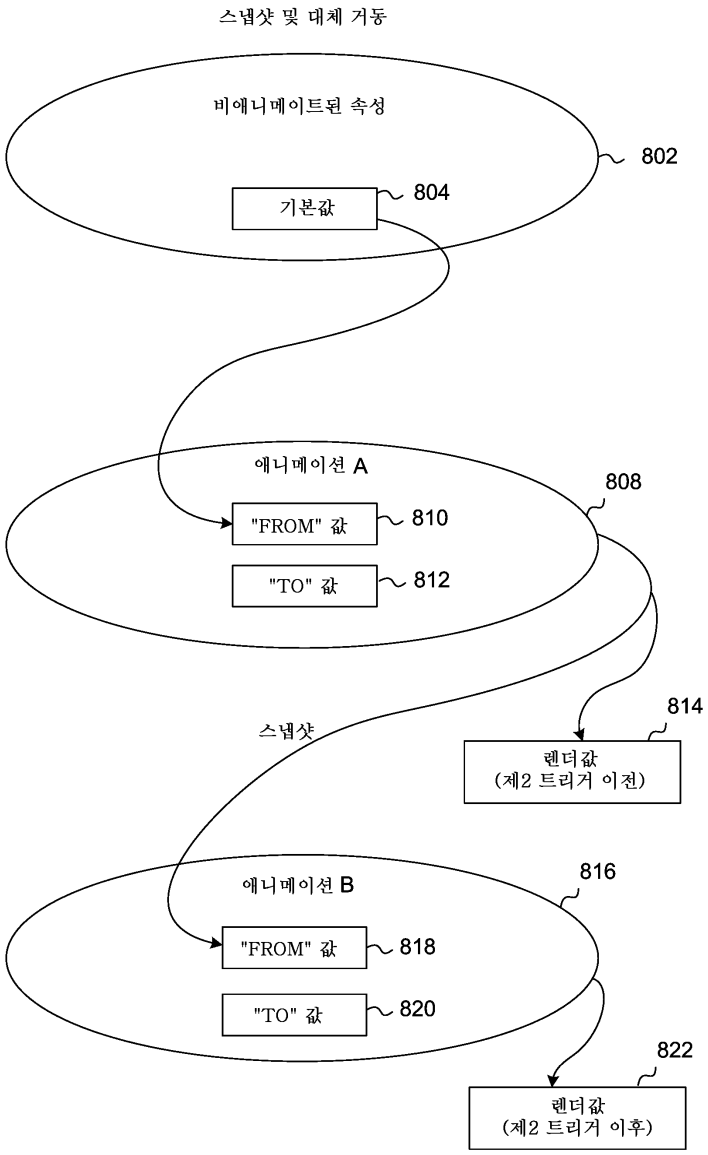
도면6


600
애플리케이션
202
애니메이션 시스템
602
속성 시스템
314
타이밍
시스템
208
핸드오프 시스템
604
그래픽
서브시스템
212
애니메이션
저장 객체
606

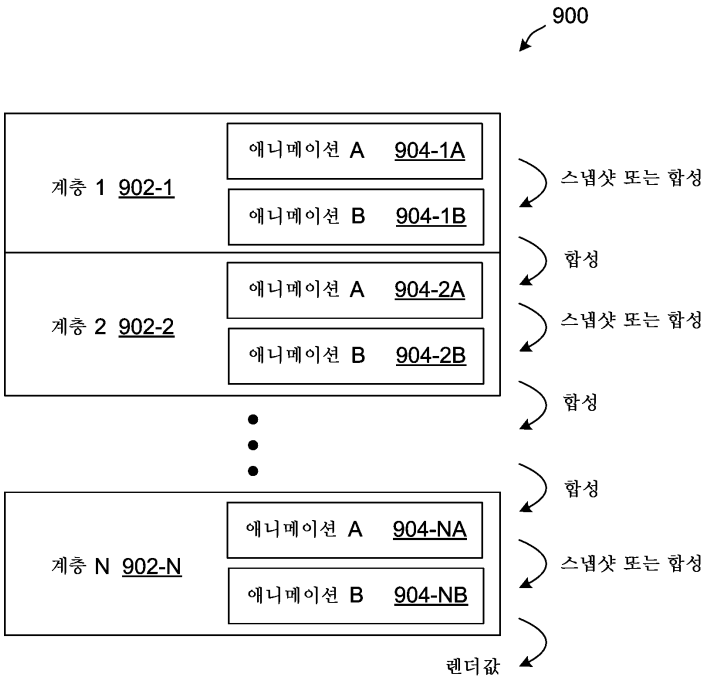
도면7


700
제1 애니메이션에 대한 트리거를 검출함
702
제1 애니메이션과 연관된 속성에 대한 애니메이션 저장 객체를 할당함
704
애니메이션 저장 객체에 속성의 기본값을 저장함
706
제1 애니메이션을 유지 및 시작함
708
제1 애니메이션이 완료되기 이전에 속성에 대한 제2 애니메이션에 대한 트리거를 검출함
710
속성의 현재값을 스냅샷함
712
제1 애니메이션을 해제함
714
제2 애니메이션을 유지 및 시작함
716

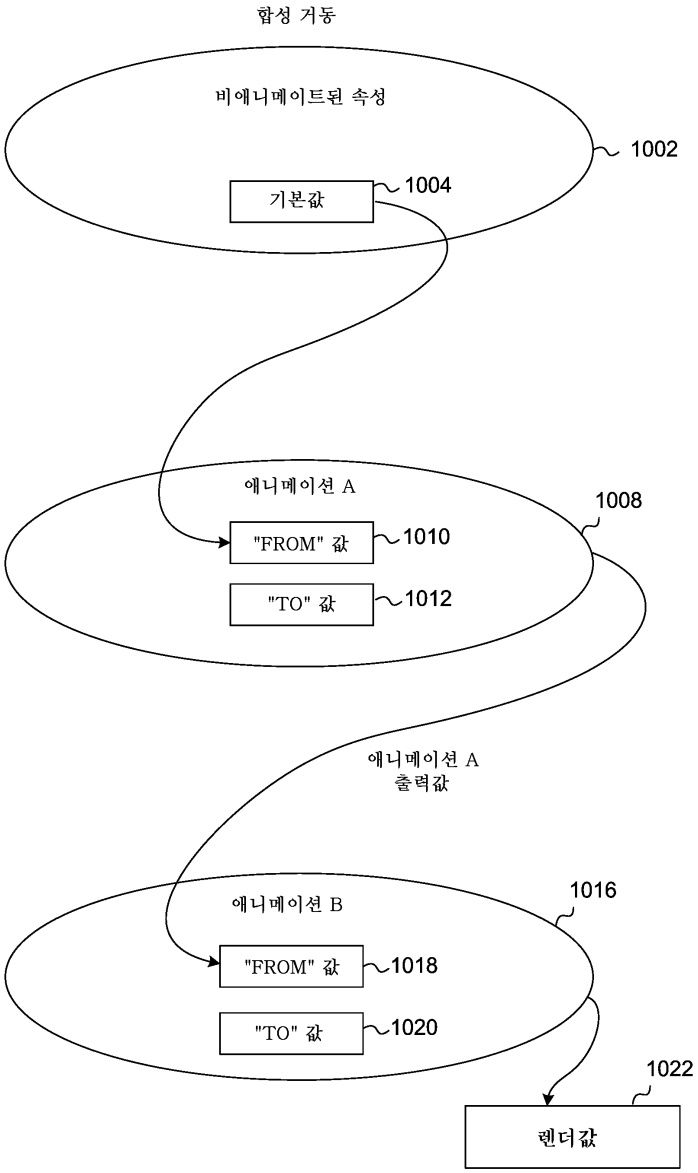
도면8


스냅샷 및 대체 거동
비애니메이트된 속성
802
기본값
804
애니메이션 A
808
"FROM" 값
810
"TO" 값
812
스냅샷
814
렌더값
(제2 트리거 이전)
애니메이션 B
816
"FROM" 값
818
"TO" 값
820
822
렌더값
(제2 트리거 이후)

도면9


900
계층 1 902-1
애니메이션 A 904-1A
애니메이션 B 904-1B
스냅샷 또는 합성
합성
계층 2 902-2
애니메이션 A 904-2A
애니메이션 B 904-2B
스냅샷 또는 합성
합성
합성
계층 N 902-N
애니메이션 A 904-NA
애니메이션 B 904-NB
스냅샷 또는 합성
렌더값

도면10


합성 거동
비애니메이트된 속성
1002
기본값
1004
애니메이션 A
1008
"FROM" 값
1010
"TO" 값
1012
애니메이션 A
출력값
애니메이션 B
1016
"FROM" 값
1018
"TO" 값
1020
1022
렌더값