(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101223555 B
(45)授权公告日 2012.11.14

(21)申请号 200680025782.8

(22)申请日 2006.07.11

(30)优先权数据
11/181,197 2005.07.13 US

(85)PCT申请进入国家阶段日
2008.01.14

(86)PCT申请的申请数据
PCT/US2006/026815 2006.07.11

(87)PCT申请的公布数据
WO2007/008853 EN 2007.01.18

(73)专利权人 微软公司
地址 美国华盛顿州

(72)发明人 E·K·尼尔森 K·B·雅各布 M·考金斯 M·J·希尔伯格

(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 陈斌

(51)Int.Cl.
G06T 13/00(2011.01)

(56)对比文件
US 6057833 A,2000.05.02, 说明书第4栏-第12栏及附图3-7.
CN 1244690 A,2000.02.16, 全文.

审查员 赵小宁

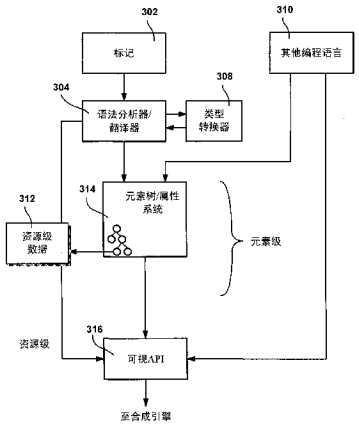
权利要求书 2 页 说明书 17 页 附图 10 页

(54)发明名称

用于动画间的平滑过渡的方法和系统

(57)摘要

当“第二”动画在“第一”动画已为其运行的可视元素的属性上启动时,提供富媒体(例如,UI可视元素的动画)间的平滑过渡。当该第二动画被启动时,动画系统引发由运行“第一”动画产生的该属性的当前值(即,快照)被存储,终止或释放该第一动画,并在随后引发第二动画使用此快照作为该属性的“来自”值而运行。因为第二动画正好在第一动画结束的那一点处“开始”,所以第一和第二动画之间的过渡很平滑。可以为已为其触发动画的属性创建动画存储对象,以存储基值并存储在被动画播放时的快照。



CN 101223555 B

1. 一种用于实现可视对象的第一动画和所述可视对象的第二动画之间过渡的方法，所述方法包括：

启动所述可视对象的第一动画，其中所述可视对象的第一动画包括调整所述可视对象的属性的值；

在所述第一动画的运行期间以屏幕刷新率重新计算所述可视对象的所述属性的值，并存储所重新计算的值；

在所述第一动画完成之前检测有关所述第二动画的触发，其中所述第二动画将调整所述可视对象的同一属性的值；

执行传递行为，所述传递行为指定了当所述第二动画被启动而所述第一动画仍在调整所述可视对象的所述属性的值时要采取什么动作；以及

使用所指定的传递行为来引发所述第二动画运行。

2. 如权利要求 1 所述的方法，其特征在于，还包括为所述属性分配动画存储对象，其中所述动画存储对象要存储引起所述第一和第二动画运行的所述属性的值。

3. 如权利要求 1 所述的方法，其特征在于，还包括当所述第二动画被启动时释放所述第一动画。

4. 如权利要求 1 所述的方法，其特征在于，还包括在所述第二动画运行的同时，允许所述第一动画在不影响所述属性的所述值的情况下运行。

5. 如权利要求 1 所述的方法，其特征在于，还包括在所述第二动画运行的同时暂停所述第一动画的所述运行。

6. 如权利要求 1 所述的方法，其特征在于，有关一属性的动画被逻辑上组织成多层，其中在一层内的所有动画已退出的情况下选择性地移除所述层。

7. 如权利要求 1 所述的方法，其特征在于，所述传递行为选自以下之一：

合成传递行为，所述合成传递行为指示所述第一动画继续，并且对所述第一动画的每个再现循环而言，所述第二动画使用所述第一动画的输出值作为其“来自”值来启动所述第二动画，其中所述第二动画的输出值被用作再现所述属性的值；

替换传递行为，所述替换传递行为指示所述第一动画被释放，并且当“至”值和“来自”值未被指定时所述第二动画使用基“至”和“来自”值，所述“至”值是第二动画期间所述“来自”值要改变到的值；

快照和替换传递行为，所述快照和替换传递行为指示所述第一动画的所述属性的当前值被获取并且所述第一动画被释放，所述第二动画使用所获得的当前值作为所述“来自”值并且在未被指定时作为有关“至”值的基值；

快照和保留传递行为，所述快照和保留传递行为指示所述第一动画的所述属性的当前值被获取并且所述第一动画保持运行而不影响所述属性的值，所述第二动画使用所获得的当前值作为所述“来自”值并且在未被指定时作为有关“至”值的基值，当所述第二动画结束时，所述第一动画恢复所述属性的动画；

快照和保留 / 暂停传递行为，所述快照和保留 / 暂停传递行为指示所述第一动画的所述属性的当前值被获取并且所述第一动画被暂停，所述第二动画使用所获取的当前值快照作为所述“来自”值并且在未被指定时作为有关“至”值的基值，当所述第二动画结束时，所述第一动画从暂停时的所述第一动画的状态中动画播放所述属性；

混合传递行为，所述混合传递行为指示使用所述第一动画和所述第二动画的加权平均值；以及

队列传递行为，所述队列传递行为指示排队未决动画直到现有动画完成。

8. 一种用于处理第一动画和第二动画的系统，所述系统包括：

属性系统，存储有关要被显示的可视对象的属性的值；

图形子系统，提供用于经由显示设备来显示所述可视对象的数据；以及

动画系统，计算在通过显示设备显示时动画播放所述可视对象中使用的所述可视对象的属性的一系列值，其中所述动画系统在所述第一动画的运行期间以屏幕刷新率重新计算所述可视对象的所述属性的值并存储所重新计算的值；在所述第一动画完成之前检测有关所述第二动画的触发，其中所述第二动画调整所述可视对象的同一属性的值；执行传递行为，所述传递行为指定了当所述第二动画被启动而所述第一动画仍在调整所述可视对象的所述属性的值时要采取什么动作；以及使用所述属性的所存储的所重新计算的值作为“来自”值来启动所述第二动画。

9. 如权利要求 8 所述的系统，其特征在于，所述动画系统为所述属性分配动画存储对象，所述动画存储对象要存储引起所述第一和第二动画运行的所述属性的值。

10. 如权利要求 8 所述的系统，其特征在于，所述动画系统在所述第二动画被启动时释放所述第一动画。

11. 如权利要求 8 所述的系统，其特征在于，所述动画系统在所述第二动画运行的同时，允许所述第一动画在不影响所述属性的所述值的情况下运行。

12. 如权利要求 8 所述的系统，其特征在于，所述动画系统在所述第二动画运行的同时暂停所述第一动画的所述运行。

13. 如权利要求 8 所述的系统，其特征在于，所述动画系统在运行所述第一动画之前从所述属性系统获取有关所述属性的至少一个基值。

用于动画间的平滑过渡的方法和系统

技术领域

[0001] 本发明涉及在计算机系统上提供平滑、复杂的动画和/或媒体。

背景技术

[0002] 在某些交互式系统中，响应于用户交互，可视元素可被实现以通过“动画”来做出响应。例如，在用户界面(UI)内正显示的按钮可被实现以在用户将光标放置在该按钮上时，该按钮可以增大、缩小或旋转。在某些情况下，可能期望具有在单个可视元素上执行的多种动画。在某些系统中，可视元素的显示在从可视元素当前运行的动画过渡至另一动画时可能出现“失灵”，而这在某些场景中是不期望的。此背景信息并不旨在标识必须由所要求的主题解决的问题。

发明内容

[0003] 本概述用于引入一些概念的简化形式，这些概念在下面的详细描述中被描述。本概述并不旨在标识所要求主题的关键特征或基本特征，也不旨在用于确定所要求的主题的范围。

[0004] 根据描述的各个实施例的各个方面，提供用于使能富媒体（例如，UI可视元素的动画）间平滑过渡的实现。一方面，当动画在另一动画已为其运行的可视元素的属性上启动时，该“第二”动画可被配置成存储由运行“第一”动画产生的该属性的当前值（即，快照），释放该第一动画并在随后使用此快照作为该属性的“来自(from)”值而运行第二动画。这一传递(handoff)行为被称为“快照和替换”行为。因为第二动画正好在第一动画结束的那一点处“开始”，所以第一和第二动画之间的过渡很平滑。

[0005] 另一方面，为已为其触发动画的属性创建动画存储对象。该动画存储对象要存储动画以其为目标的属性的一个或多个基值，另外在起始动画仍在运行时有另一动画被触发的情况下还要存储该属性的任何快照。

[0006] 再一方面，动画的多层可应用于一属性，其中各层之间的过渡具有合成传递行为。在某些实现中，一层内各动画间的过渡可以具有合成或“快照和替换”行为。在其它实现中，各层之间的动画间过渡可以具有合成或“快照和替换”行为。

[0007] 根据本发明的一方面，提供了一种用于可视对象的第一动画和所述可视对象的第二动画之间过渡的计算机实现方法，所述方法包括：启动所述可视对象的第一动画，其中所述可视对象的第一动画包括调整所述可视对象的属性的值；在所述第一动画完成之前检测有关所述第二动画的触发，其中所述第二动画调整所述可视对象的同一属性的值；根据所述第一动画计算用于下一屏幕刷新的、所述属性的值，并存储所计算的值；以及在所述第二动画运行期间，引发所述第二动画使用所存储的所述属性的值作为“来自”值，所述“来自”值用于启动所述第二动画。

[0008] 根据本发明的另一方面，提供了一种用于处理第一动画和第二动画的系统，所述系统包括：属性系统，存储有关要被显示的可视对象的属性的值；图形子系统，提供用于经

由显示设备来显示所述可视对象的数据；以及动画系统，计算在通过显示设备显示时动画播放所述可视对象中使用的所述可视对象的属性的一系列值，其中所述动画系统在所述第一动画完成之前检测有关所述第二动画的触发，其中所述第一和第二动画调整所述可视对象的同一属性；根据所述第一动画计算用于下一屏幕刷新的、所述属性的值并存储所计算的值；并且使用所述存储的所述属性的值作为“来自”值来启动所述第二动画。

[0009] 根据本发明的再一方面，提供了一种用于动画播放可视元素的计算机实现方法，所述方法包括：为可视元素定义一组可触发动画集，所述集包括一个或多个可触发动画的第一子集以及可触发动画的一个或多个其他子集，其中所述第一子集的所述可触发动画的每一个可由事件触发或属性触发来触发，所述一个或多个其他子集的所述可触发动画的每一个各自可由属性触发来触发，并且合成传递行为被用于从一个子集的动画过渡到另一子集的动画，并且其中所述从一个子集的第一动画过渡到该子集的第二动画是通过以下步骤来完成的：在所述第一动画完成之前检测关于所述第二动画的触发，其中所述第一和第二动画调整可视对象的同一属性的值；根据所述第一动画计算用于下一屏幕刷新的、所述属性的值并存储所计算的值；以及使用所存储的所述属性的值作为“来自”值来启动所述第二动画。

[0010] 各实施例可被实现为计算机进程、计算机系统（包括移动手持计算设备）或被实现为诸如计算机程序产品的制造物品。计算机程序产品可以是可由计算机系统读取并编码用以执行计算机进程的指令的计算机程序的计算机存储介质。计算机程序产品也可以是可由计算系统读取并编码用以执行计算机进程的指令的计算机程序的载体上的传播信号。

附图说明

[0011] 参考以下的附图描述了非限制性和非穷尽的实施例，除非特别指出，否则在附图中类似的编号在各附图中指代相同的部分。

[0012] 图 1 是表示了其中可结合各实施例的一示例性计算机系统的框图。

[0013] 图 2 是表示了根据一个实施例的媒体整合层架构的框图。

[0014] 图 3 是根据一个实施例用以对具有可视 API 层的接口解释标记语言代码的组件的表示。

[0015] 图 4 和图 5 是根据一个实施例的具有用于把时钟属性数据转变为在确定前进数据内使用的时间间隔的两级架构的一般表示的框图。

[0016] 图 6 是根据一个实施例用于提供富媒体之间平滑过渡的系统的一般表示的框图。

[0017] 图 7 是根据一个实施例在执行传递操作的操作流程的一般表示的框图。

[0018] 图 8 是根据一个实施例使用传递操作在两动画之间过渡的表示。

[0019] 图 9 是根据一个实施例在分层动画之间的关系的表示。

[0020] 图 10 是根据一个实施例的两个动画合成的表示。

具体实施方式

[0021] 如下将参考形成其一部分并示出了用于实践本发明的特定示例性实施例的附图对各实施例进行更为全面的描述。然而各实施例可以被实现为许多不同的形式，并且不应被解释为受到在此描述的各实施例的限制；相反地，提供这些实施例以使得本公开透彻和

完全，并且能向本领域普通技术人员完整地传达本发明的范围。各实施例可实践为方法、系统或设备。因此，各实施例采取硬件实现、全部软件实现、或者组合了软件和硬件方面的实现的形式。因此，以下的详细描述没有限制性的意义。

[0022] 各实施例的逻辑操作被实现为 (1) 在计算系统上运行的一系列计算机实现的步骤，和 / 或 (2) 在计算系统内互连的机器模块。实现是取决于实现各实施例的计算系统的性能要求的选择问题。因此，组成在此描述各实施例的逻辑操作也可被称为操作、步骤或模块。

[0023] 示例性操作环境

[0024] 图 1 示出了在其上可实现动画间平滑过渡的一合适的计算系统环境 100 的示例。计算系统环境 100 只是一合适的计算环境的一例，而不是要提出对本发明使用范围或功能性进行任何限制。计算环境 100 也不应解释成对于在示范操作环境 100 中所示出的任一组件或其组合有任何依赖或要求。

[0025] 本发明是可用多个其它通用或专用计算系统环境或配置运行的。可以适用于该本发明的公知的计算系统、环境、和 / 或配置的示例包括，但不局限于，个人计算机、服务器计算机、手持设备或膝上型设备、写字板设备、多处理器系统、基于微处理器的系统、机顶盒、可编程消费者电子设备、网络 PC、小型机、大型计算机、包括任何诸如以上系统或设备等的分布式计算环境等。

[0026] 本发明可以在正由计算机执行的诸如程序模块的计算机可执行指令的一般上下文中被描述。一般地，程序模块包括完成特殊任务或执行特殊抽象数据类型的例行程序、程序、对象、组件、数据结构等。本发明也可以在分布的计算环境中实践，在此由通过通信网络而链接的远程处理设备来执行任务。在分布式计算环境中，程序模块可以定位于包括记忆体储存设备的本地和远程计算机存储介质。

[0027] 参见图 1，对于实现本发明的示范的系统包括一以计算机 110 形式出现的通用计算设备。计算机 110 的组件包括，但不局限于，处理单元 120、系统存储器 130、以及将包括该系统存储器在内的各种系统组件耦合至处理单元 120 的系统总线 121。上述系统总线 121 可以是几种总线结构类型中的任何一种，包括存储总线或存储控制器、外围总线和使用各种总线体系结构的任一种的局域总线。举例来说，而非限制，此类体系结构包括工业标准体系结构 (ISA) 总线、微通道体系结构 (MCA) 总线、增强型 ISA(EISA) 总线、视频电子标准技术协会 (VESA) 局域总线、加速图形端口 (AGP) 总线和也被称为 Mezzanine 总线的外围部件互连 (PCI) 总线。

[0028] 计算机 110 通常包括各种计算机可读介质。计算机可读介质可以是任何计算机 110 能够访问的可用介质，包括易失性的和非易失性的介质、可移动的和不可移动的介质。举例来说，而非限制，计算机可读介质可以包含计算机存储介质和通信介质。计算机存储介质包括能以任何方法或技术实现的易失性的和非易失性的、可移动的和不可移动的介质，用于存储诸如计算机可读指令、数据结构、程序模块或其它数据等信息。计算机存储介质包括，但不局限于，RAM、ROM、EEPROM、闪存或其它存储技术，CD-ROM、数字化多功能光盘 (DVD) 或其它光盘存储、盒式磁带、磁带、磁盘存储器或其它磁存储设备，或任何其它可以被用来存储想要的信息并且可以被计算机 110 访问的介质。通信介质通常具体体现为诸如载波或其它传送机制的已调制数据信号中的计算机可读指令、数据结构、程序模块或其它数据，也

包括任何信息传递介质。术语“已调制数据信号”是指在该信号中以编码信息的方式来设置或改变其一个或多个特征的信号。举例来说，而非限制，通信介质包括诸如有线网或直线连接的有线介质，和诸如声音、射频、红外线和其它无线介质的无线介质。任何以上所述的组合也应该包括在计算机可读介质的范围之内。

[0029] 系统存储器 130 包括以诸如只读存储器 (ROM) 131 和随机存取存储器 (RAM) 132 的易失性和 / 或非易失性存储器的形式的计算机存储介质。包含如在启动期间帮助在计算机 110 内各元件之间传送信息的基本例行程序的基本输入 / 输出系统 133(BIOS)，通常存储在 ROM 131 中。RAM 132 通常包含可以被处理单元 120 立即访问和 / 或当前操作的数据和 / 或程序模块。作为示例而非限制，图 1 示出了操作系统 134、应用程序 135、其它程序模块 136 和程序数据 137。

[0030] 计算机 110 还可以包括其它可移动 / 不可移动、易失性 / 非易失性的计算机存储介质。仅作为示例，图 1 示出了从不可移动、非易失性磁性介质读出或写入不可移动、非易失性磁性介质的硬盘驱动器 141、从可移动、非易失性磁盘 152 读出或写入可移动、非易失性磁盘 152 的磁盘驱动器 151、以及从诸如 CD ROM 或其它光学介质的可移动、非易失性光盘 156 读出或写入可移动、非易失性光盘 156 的光盘驱动器 155。其它可以使用在示例的操作环境中的可移动 / 不可移动、易失性 / 非易失性计算机存储介质包括，但不局限于，盒式磁带、闪存卡、数字多功能光盘、数字录像带、固态 RAM、固态 ROM 等。硬盘驱动器 141 通常通过诸如接口 140 的不可移动存储接口连接到系统总线 121，磁盘驱动器 151 和光盘驱动器 155 通常通过诸如接口 150 的可移动存储接口连接到系统总线 121。

[0031] 以上讨论并且在图 1 中示出的驱动器及它们相关的计算机存储介质为计算机 110 提供了计算机可读指令、数据结构、程序模块和其它数据的存储。在图 1 中，例如，示出硬盘驱动器 141 存储操作系统 144、应用程序 145、其它程序模块 146、和程序数据 147。需要注意的是这些组件可以和操作系统 134、应用程序 135、其它程序模块 136 和程序数据 137 相同，也可以和它们不同。在此对操作系统 144、应用程序 145、其它程序模块 146 和程序数据 147 给出了不同的标号来说明至少它们是不同的副本。用户可以通过诸如写字板（电子数字转换器）164、麦克风 163、键盘 162 和定位设备 161 的输入设备把指令和信息输入到计算机 110 中，定位设备 161 通常指如鼠标、跟踪球或触摸板。其它输入设备（图 1 中未示）可以包括操纵杆、游戏垫、圆盘式卫星天线、扫描仪等等。这些和其它输入设备通常由用户输入接口 160 连接到处理单元 120，上述输入接口 160 和系统总线 121 相耦合。但是上述和其它输入设备也可以由其它接口和总线结构连接到处理单元 120，诸如，并行端口、游戏端口或通用串行总线 (USB)。监视器 191 或其它类型显示设备也可以通过诸如视频接口 190 的接口连接到系统总线 121。监视器 191 也可以和触摸屏面板 193 等集成在一起。该触摸屏面板 193 等可以经由接口（例如触摸屏接口 192）将诸如手写笔迹的数字化输入到计算机系统 110。需要注意的是监视器和 / 或触摸屏画板可以物理地耦合到包含了计算机 110，诸如写字板式个人计算机的一外壳，其中触摸屏面板 193 实质上用为写字板 164。另外，计算机（例如，计算设备 110）也包括其它外围输出设备，诸如可以是通过一输出外围设备接口 194 等连接的扬声器 195 和打印机 196。

[0032] 计算机 110 可以在网络化的环境中运行，该环境使用与诸如远程计算机 180 的一台或多台远程计算机的逻辑连接。远程计算机 180 可以是个人计算机、服务器、路由器、网

络个人计算机、对等设备或其它公共网络节点，通常包括以上描述的和计算机 110 相关的多个或全部元件，尽管在图 1 中只示出了记忆存储设备 181。在图 1 中描绘的逻辑连接包括局域网 (LAN)171 和广域网 (WAN)173，但是也可以包括其它网络。这样的网络环境在办公室、企业范围的计算机网络、内联网和因特网中是普遍的。

[0033] 当在 LAN 网络环境中使用时，计算机 110 通过网络接口或适配器 170 连接到 LAN 171。当在 WAN 网络环境中使用时，计算机 110 通常包括调制解调器 172 或通过诸如因特网的 WAN 173 建立通信的其他装置。调制解调器 172 可以是内置的或外置的，可以通过用户输入接口 160 或其它适当的机制连接到系统总线 121。在一网络化的环境中，所描述的和计算机 110 相关的程序模块或其中的各个部分可以存储在远程记忆体存储设备内。举例说明，但非限制，图 1 示出了驻留在存储设备 181 上的远程应用程序 185。可以理解的是所示的网络连接是示例的，也可以使用在计算机间建立通信链路的其他装置。

[0034] 示例性架构

[0035] 一方面一般涉及在计算机系统上提供平滑、复杂的动画和 / 或媒体。为此，如图 2 中所一般表现的，提供媒体整合架构 200。应用程序、控制或其它类似的较高级程序代码（例如，操作系统组件的用户接口）202 通过一组应用编程接口 (API)204 等访问媒体整合层架构 200 以访问（读或写）图形信息。需要注意的是虽然在此描述的例子中的多个例子涉及与 API 连接的应用程序，但可以理解的是其它较高级程序代码和组件（例如，操作系统的用户接口）也可用以与在此描述的较低级组件相连接。如此，任何涉及对这样的较高级程序代码的引用，不管是否涉及应用程序、用户接口等，都应该被同等考虑。

[0036] 在一实现中，媒体整合架构 200 包括高级合成和动画引擎 206、定时和动画组件 208、以及低级合成和动画引擎 210。如在此使用的，术语“高级”和“低级”类似于那些在其它计算情况中使用的，其中一般地，相对于较高级组件而言越低级的软件组件是越靠近硬件的组件。如此，例如，发送自高级合成和动画引擎 206 的图形信息可在低级合成和动画引擎 210 处被接收，在 210 处信息被用以发送图形数据至包括硬件的图形子系统。

[0037] 一般地，高级合成和动画引擎（在此也称为高级合成器和动画播放器或高级引擎或组件）206 构建显示元素树以表现由应用程序 202 提供的图形场景，而定时和动画组件提供说明性的（或其它）动画和定时控制。低级合成和动画引擎（在此也称为低级合成器和动画播放器或低级引擎或组件）210 组成用以多个应用场景的再现，并利用再现组件来实现对于屏幕真实的图形再现。注意，在较高层上依然可能做消耗时间的或特定应用的再现，并发送涉及位图等至较低层。

[0038] 高级合成和动画引擎 206 构建元素树结构并遍历该结构，创建被发送至低级合成和动画引擎 210 的再现指令和简单动画时间间隔。由高级合成器生成的再现指令可包括定时和动画信息。低级合成和动画引擎 210 采用再现指令和动画时间间隔并管理以后提供给图形子系统（举例来说，图形软件和硬件）212 的场景的动画播放、再现和合成。除了本地显示的输出以外，高级合成和动画引擎 206（或此外一类似的）可以合适的格式提供再现和动画指令给较低级打印代码 220 以发送固定的图形数据至打印机 222 等，和 / 或可以合适的格式提供再现指令和简单动画时间间隔给较低级终端传送服务器 226 以传送至远程机器 228。注意大量的信息也可通过网络发送，举例来说，在没有任何网络流通量的情况下，让远程机器本地处理鼠标滚动影响是理想的。

[0039] 在该实现中，媒体整合架构 200 把图形处理分割成多级，并且这些级中的每级执行某些智能图形处理，这些处理一起允许应用程序的用户接口 202 等输出具有平滑动画的图形、用其它应用图形合成这些图形、以及与视频帧一起工作。动画和 / 或合成也可和音频输出同步。例如，通过同步音频与在低级组件上的帧速率，音频的定时实质上可与视频或图形的定时一致，并且不依赖于任务调度、复杂预处理的能力以保持刷新速率。

[0040] 图 3 表示了在其中诸如基于 XAML 的代码的标记代码 302 可由语法分析器 / 翻译器 304 解释的一个实现。一般地，语法分析器 / 翻译器 304 添加元素至元素树 / 属性系统 314 ；元素是实行自己布局的可视对象。此外，注意标记代码的某些或所有可被按需编译而不是解释，从而提高效率。

[0041] 一般地，元素是在参与属性系统、触发和布局 / 表现系统的元素层内的对象。语法分析器 304 查找标记并且确定是否那些标记帮助定义元素或资源对象。在 VisualBrush 的特殊情况中，例如，相同的标记可被解释成元素或也被解释成资源对象，这取决于那些标记出现地方的上下文，例如，如在美国专利申请序号为 10/401,717 中描述的那样，取决于是否在复杂属性句法中出现。

[0042] 除了在标记内的内联表现以外，资源实例也可定位在其它任何位置（例如，在标记内或文件内，该文件可以是在本地或在远程网络上并能适当地下载），并能由名称所引用，（例如，文本名称、引用或其它合适的标识符）。通过这种方式，场景设计者可在整个场景内重用元素树中的元素，包括由复杂属性句法描述的元素。正如以下将详述的那样，资源可以包括故事板，以允许各故事板的重新使用。

[0043] 语法分析器 304 按需通过访问类型转换器 308 来处理在复杂属性句法内的标记，并且也通过把特定的参数与对象属性匹配来处理在复杂属性句法内的标记，从而为场景设计者处理复杂性。如此，语法分析器 304 不仅仅建立对象，也设置对象上的属性。由于相同的再现模型在元素级和 API 级之间共享，对象中的多个对象实质上是相同的。这使得语法分析 / 翻译具有高效率，并且也允许不同类型的编程语言（例如，类似 C# 等语言）可容易地从标记转换成自己的句法，并且反之亦然。注意的是如图 3 中所表现的，另一个这样的编程语言 310（可包括编译过的标记）能添加元素至元素树 314，或能直接与可视 API 层 316 连接。

[0044] 还如图 3 中所表现的，相同的标记 302 可被用以在元素级和资源级编程。一般地，元素级给场景设计者全部的可编程性、提供继承的属性系统的使用（例如，如特征的式样表）、以及触发（例如，由此元素可具有附加的代码以响应用户输入事件或动作来更改它的外观、位置等）。然而，各实施例还提供资源层机制，场景设计者实质上可由此简化元素树以及直接对可视 API 层编程。对于不需要元素级特征的许多类型的静态形状、图形等，这提供一种更有效率和轻便的方式输出适当的对象。

[0045] 为了控制动画和媒体输出，如以下参照图 4 和图 5 描述的那样，包括时钟的定时树也被维持。一般地，高级合成器和动画引擎 206 执行复杂处理（有时指编译），该复杂处理大大简化处理量并且大大减少较低级需要处理的数据量以再现正确的输出。然而，需要的注意的是，由较高级执行的处理的数量和类型在很大程度上可取决于较低级的装载、配置和能力。例如，如果高能力图形硬件存在，较高级可进行较少数量的处理，反之亦然。高级和低级层被自适应于这些因素。

[0046] 一般地，动画通过高级合成器和动画引擎 206 以及低级合成器和动画引擎 210 完成。在一实现中，高级引擎 206 遍历场景并且用关于稍后的插补的时间间隔更新动画参数，以及把这些简化的数据结构封装至发送到较低级引擎 210 的指令中。这可以同步和 / 或异步的方式完成。时间间隔数据可被考虑成包括定时端点（开始和结束定时数据）和用于再现指令的参数化的值。注意的是高级引擎 204 可执行请求的插补中的某些插补或所有插补，例如，如果一插补或其它运动函数对于较低级引擎 210 而言太复杂而不能处理，或较低级不能跟上放置其中的处理需要，较高级引擎可执行计算中的某些计算或所有计算并提供给较低级简化的数据、指令、格局等以实现理想的结果。

[0047] 在典型的情况下，当较低级执行插补时，对于动画的每个帧，低级引擎 210 插补参数时间间隔以获得即时值，并解码指令为由图形设备执行的再现命令。图形设备通过添加任何可在场景中出现的视频帧以合成最终的场景。其它数据也可被添加，诸如由数字化版权管理保护的内容。

[0048] 高级引擎 206 于是遍历场景数据结构、计算描述一段时间内每个动画的参数的时间间隔、以及发送这些时间间隔和简化的参数化的画图指令至低级引擎 210。参数数据包括开始时间、结束时间、插补算子和插补数据。作为示例，高级合成器和动画引擎 206 可指示低级合成器和动画引擎 210 关于图形应该随时间的推移如何改变，例如，开始坐标、结束坐标、图形应该在坐标之间移动的时间量（时间间隔）、以及诸如线性的运动函数（注意运动不是动画所要求的，如静态对象可通过更改例如它的颜色属性而被动画播放），而不是擦除和重画图像以使它看上去移动。低级合成器和动画引擎 210 将插补以确定帧之间新的位置，把这些转换成图形设备可理解的画图指令，并发送命令至图形设备。高级引擎 206 的每个发送较好地提供足够数据给低级引擎 210 以执行随着几个帧的推移的平滑的动画。

[0049] 低级（例如，快速记号）引擎 210 是从高级引擎 206 中分离的任务。低级引擎 210 从高级引擎 206 接收简化的参数化的画图指令和描述场景的参数时间间隔。低级引擎维持和遍历这些数据结构直到新的数据结构由高级引擎 206 提供。低级引擎可服务多个高级引擎 206，对于每个高级引擎维持独自的数据结构。在低级引擎 210 和高级引擎 206 之间一对多的关系允许系统同步地平滑地动画播放多个场景。

[0050] 低级引擎 210 实质上基于高级引擎的提供的时间间隔来插补即时动画参数，更新画图指令和再现每一帧的场景。低级引擎 210 任务在系统上以高优先级运行以确保帧准备好诸如以图形硬件屏幕刷新速率显示。由低级引擎 210 执行的插补一般因此限制于简单、快速的函数，诸如线性、分段线性、三次样条以及类似速度的那些函数。

[0051] 关于动画和媒体，一般如图 4 和图 5 中显示的，诸如应用程序 202 的程序结合定时信息（被称为时钟或时钟属性）对高级组件 206 指定动画属性值。如以下描述的，实质上任何独立动画或媒体（例如，诸如视频和音频的线性媒体）以及协调指定的动画的故事板将具有对于其在高级组件上维持的时钟。一般地，作者指定例示成时钟以合适地保持它们同步的时间线数据。在某些实施例中，故事板是由一组时间线或时间线树组成的时间线。故事板可以提供目标信息，以使其时间线树能够以有关多个元素的多种属性为目标。此外，动画在某些实施例中是各类时间线。

[0052] 时钟属性包括定义给定的时钟和定时结构的其余部分之间初始同步关系的定时属性。如图 5 所示，高级组件 206 可调用高级动画函数 520H（例如，以管理的代码书写，该

代码以元数据的形式提供信息，该元数据允许通用语言运行时间，其上可运行元数据以控制它的操作）以确定动画属性的当前值。在快速帧速率计算期间，低级组件 210 调用类似（或相同）的具有由引擎 414 计算的进程的动画函数 520L 以确定动画的当前属性值。注意的是在可供选择的实现中，动画函数可构建进例如较低级组件中。

[0053] 一般地，动画和线性媒体与一组时钟关联，该组时钟通过同步图元和规则相互相关。时钟可被分层次地安排，例如，应用程序具有父时钟，而应用程序的动画的对象是子层，其依次可具有其它子层。当时钟的属性被定义或修改时，该时钟的任何子时钟可被影响。例如，暂停父时钟可暂停其子时钟的任何一个，以及加倍父时钟的速度可加倍子时钟的任何一个的速度。

[0054] 这些时钟属性可由包括由应用程序在运行时间启动的交互式的控制事件的资源事件所修改。如此，时钟是交互的，通过这种方式每个时钟可各自地由应用程序在任意时间，例如，响应于用户输入而启动、暂停、继续执行以及停止。此外，新的时钟可被添加至定时结构，并且现有的时钟可被移除。

[0055] 如在上述美国专利申请 10/693,822 中描述的，高级定时组件可基于存储的事件列表（开始、暂停等）和关联的同步图元对于每个时钟生成时间间隔列表。激活时间间隔是描述在真实世界时间内的不同点上由给定的时钟表示的时间的直接的、非重叠片断。

[0056] 如图 4 中所示的，时钟属性中的至少某些属性可在时钟的定时树 402 内被分层次地关联。三个这样的时钟与它们的属性在图 4 中示出，即时钟属性 404_1-404_3，然而，可以理解的是在给定的情况下可存在许多更多的时钟和可供替代的定时树。对每个时钟，例如可对应于要显示的动画对象（或故事板协调的对象组），高级组件 206 包括状态机，称为事件列表生成器 408，可从时钟属性产生事件列表（例如，406_1）。一般地，事件列表生成器 408 把由特定的时钟属性初始安排的事件与任何诸如暂停和继续执行接收到的关于动画的请求的显式的交互事件一起分组成一个事件列表。时钟对于每个动画（或如以下描述的通过故事板的动画组）维持，并且这样就存在相应于每个独立的动画 / 动画组的事件列表以及存在一个对于每个独立线性媒体的事件列表。

[0057] 高级组件包括对于每个动画或媒体使用事件列表（例如，406_3）的时间间隔生成器 410 以计算发送至低级组件 210 的相应的时间间隔列表（例如，412_3）。依次地，低级组件 210 包括基于关于时间间隔数据的当前时间来控制输出的低级计算引擎 414，该引擎通过诸如基于对于该对象的时间间隔数据和当前时间提供进程值至确定动画对象的变化属性的当前值的低级动画功能子系统 520L 来控制输出。例如，对于任何给定的帧，低级计算引擎 414 基于时间间隔数据和当前时间（可以是系统时间或相对时间）来插补动画对象的位置。注意的是如图 4 和图 5 中所示，高级定时组件包括定时树 402 和事件列表 406_1-406_3，而低级定时组件 210 包括时间间隔列表 412_1-412_3，然而这些数据结构实质上可维持在存储器，一般在高速随机存取存储器内的任何位置。

[0058] 总而言之，高级定时组件 206 查看同步规则和图元所相关（例如，分层次地）的时钟的树 402。各实施例在高级定时组件 206 内使用时钟以编译由低级定时引擎消耗的活动时间间隔列表 412。低级定时引擎查看表现独立（例如，每一动画对象或每一线性媒体）时钟的时间间隔列表 412。如图 5 中所示，存在提供一致的时间给高级定时组件和低级定时组件的诸如系统时钟 524 的时钟以使多个级保持同步。

[0059] 响应于从应用程序202(或某些其它源)接收到的交互更改,高级定时组件需要更新时钟的状态,该状态直接涉及该更改,和任何如同步图元所指定的那样间接涉及的更改。注意的是对于单时钟的更改可影响在定时结构内其它时钟是可能的。例如,幻灯片的陈述者可暂停整个显示,由此当前在显示的任何动画和线性媒体需要被暂停。如以下描述的,根据一个实施例,Storyboard(故事板)允许动画和/或媒体被一起分组并且以这种理想的方式协调。

[0060] 一般地,任何时候发生关于对象的交互动作,事件列表按需重新生成并且时间间隔列表被重新计算并被发送至较低级定时组件。较高级组件执行这些操作(例如,以每秒大约6到10次数量级的频率)以使较低级组件(例如,每秒操作60次)只需要处理对于每个动画的对象当前的时间间隔列表。

[0061] 除了一个对象的该时间间隔信息和开始以及结束值,额外的参数可被发送至较高级组件并导致对于时间间隔列表的更改。例如,应用程序可及时寻找对应于一事件的时间段。该应用程序指定寻找何时将要发生、以及到什么程度。象倒带或快进媒体,通过在动画上的寻找,应用程序可及时前跳或回跳至特定点。倒转事件是应用程序可指定的另一类型的计划的或交互的事件。当倒转事件在表中时,进程自动倒转,例如,从百分之百至百分之零。速度更改也被支持,例如,低于某个时钟的每个事物可被设置成运行速度高于或低于实际时间。其它属性也可被指定,诸如对于特定时钟表明是否允许重新开始,和/或如果当前不运行是否再开始,例如,在一秒处启动,在10秒处再开始,在100秒处再开始等,但是如果当前运行,重启或不重启是可任选的。

[0062] 用于协调动画和媒体的故事板

[0063] 各实施例尤其涉及故事板,该故事板包括允许对于元素(例如,诸如窗体、面板的FrameworkElement(框架元素),诸如按钮等的控件)特定时间线树的对象。时间线设置元素的属性值,并且控制元素和它子树的动画。Storyboard基本上对应于可被启动、结束或暂停、或用以寻找的特定类型的时间线对象,从而控制与该Storyboard关联的动画元素和/或媒体行为的外观。Storyboard可从标记和代码中获得。

[0064] 在某些实施例中,对于动画和故事板使用总被触发模型。此外,该模型将动画和故事板放入与触发的直接关系中(与先前实现中的Storyboard属性相反)从而允许传递行为(将在如下结合图6至图9描述)。传递行为是根本的触发概念。正如下文将描述的,未指定的“来自”值基于在新动画被触发以及新动画的传递行为时刻在属性上存在的当前值而被定义。这意味着动画的“来自”值(在未指定时)仅在触发动画时被定义。

[0065] 因为该属性仅在触发动画时被定义,所以该模型将动画与触发相关联。在此模型中,动画可以内联于触发,或者在Resource(资源)中定义并且直到触发才被使用。

[0066] 在某些实施例中,Storyboard包括可与元素相关联并且能独立开始的顶级时间线集合。Storyboard也在定时树和元素树之间提供连接点,其中该元素树允许在Storyboard内的动画相对该元素解析它们的目标,且该Storyboard依附于该元素。从标记中,Storyboard也可在Trigger(触发)的集合中声明,该Trigger可由状态更改、事件、或其它动作激活并包含在Trigger被激活时将要被执行的一组可扩展动作(例如,时间线的开始或停止)。Storyboard也可在元素的Resource属性、Style(式样)或Template(模板)中声明。

[0067] 一般地，在 Storyboard 内的时间线表示可被应用于各种子树、元素和其它对象（例如，文本字母、单词、线、段落等）的动画模板。根据本发明，故事板（例如，通过 API 访问的对象）允许程序作者通过对时间线内的动画分组（成为元素属性）来安排协调的动画组。换句话说，故事板把定时树和元素树整合，通过相同的时间线以协调的方式提供能力以操作对于不同动画的不同属性的定时。如此，故事板提供一种方法以将影响各种对象和属性的独立的时间线以其他方式组成一单个时间线树，使程序作者以直接的方式组织和控制复杂定时事件组。例如，作为 Storyboard 一部分的时间线下的动画分组允许通过简单重启 Storyboard 来一次性重启所有动画，而不是必须寻找和重启单独的动画。

[0068] Storyboard 可在动画 / 视频 / 音频的特定实例上使用，或被用以特定式样的动画。在某些场景中，被声明为 Resource 的单个 Storyboard 可经由多个 Element（元素）上各自的触发而被应用于这些元素。BeginStoryboard（开始故事板）在来自标记或代码的属性上启动动画（或其他媒体）。从标记中，开始故事板利用 FrameworkElement.BeginStoryboard（框架元素 . 开始故事板）来走过分配给其故事板属性的各时间线的树。FrameworkElement.BeginStoryboard 随后调用居于 Timeline（时间线）的 Storyboard 的树的叶子上的任何动画上的 FrameworkElement.BeginAnimation（框架元素 . 开始动画）。在标记中，BeginStoryboard 可用于事件和属性触发。

[0069] 从标记中，作者可以使用 BeginStoryboard 来启动 Storyboard。BeginStoryboard.Storyboard（开始故事板 . 故事板）包含其附加属性提供了让多种属性以多个元素为目标的能力的 Storyboard。该 Storyboard 可以含有所有类型的 Timeline，包括动画和 MediaTimeline（媒体时间线）。

[0070] 以下的示例示出了内联 Storyboard。容器对象 (Button（按钮）) 被动画播放。在此示例中供应 Storyboard.TargetName（故事板 . 目标名），但是在某些实施例中则并不要求供应，这是由于为该动画以该容器为目标。因为 BeginStoryboard.Storyboard 标签被自动供应，所以它们不是必须的。

[0071] 在此实施例中，所有的动画在 Storyboard.TargetName 未被指定时都以容器对象为目标。此外，它们在源未被指定时将容器用作触发事件的 Source（源）。这两个假设允许在以容器为目标和触发容器的场景中标记的简化（无需具有 TargetName 或 Source）。

[0072]

```
<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"
                                          Storyboard.TargetName="myButton"/>
                                          //TargetName is not required in this case
because the default target is the container.
              </Storyboard>
            </BeginStoryboard >
          </EventTrigger.Actions>
        </EventTrigger>
      </Button.Triggers>
    </Button>
</Window>
```

[0073] 以下的示例示出了被定义为 Resource 的 Storyboard。因为动画被定义在 Resource 部分，所以它可以在其他 FrameworkElement 的动画中被重新使用。

[0074]

```
<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>
```

[0075] 如上各示例所示，除了整合定时树和元素树之外，故事板允许使用事件触发来控制它们的操作（例如，启动、暂停、寻找和停止动作）。此外，故事板可用属性触发来使用。一般说来，属性触发与诸如真或假的某种状态 / 值相对应，而事件则激发 (fire)（于是就不具有假值）。例如，属性触发可以在鼠标在一元素上盘旋时具有真值，并在不再盘旋时具有假值。每个真或假的状态都可分别用来开始、停止、暂停和 / 或在故事板（或多个故事板）内寻找。诸如鼠标点击一元素的事件激发，且因而不具有可变状态，但仍能控制故事板的操作。以此方式，动画可经由与显示的元素树表示的用户交互动作而受到控制。

[0076] 事件触发将在事件在主控元素上激发时引发动作出现的该事件作为它的源。注

意，与属性触发不同，事件触发并不界定范围（scoped）；它执行操作，随后该操作确定其自身的生命周期。例如，一个事件可以启动一时间线，并且该时间线可以运行直到其持续时间完成，或直到另一事件触发或类似的事件停止该时间线。

[0077] 图 6 根据一个实施例示出了一种在富媒体间提供平滑过渡的系统 600。在此实施例中，系统 600 除了先前描述的应用程序 202、定时系统 208、图形子系统 212 和属性系统 314 之外，还包括具有传递系统 604 的动画系统 602。如下将描述根据各实施例在这些过渡中使用的这些组件的某些功能。

[0078] 如前所述，属性系统 314 可以存储有关由应用程序 202 示例的各可视对象（未示出）的属性的值。例如，属性系统 314 除了对象属性之外，可以存储有关表示 UI 中显示的按钮的一对象的长度和宽度属性的基值。

[0079] 如前所述，定时系统 208 可以协调由应用程序 202 创建的各动画的定时。例如，定时系统 208 可以提供在计算正被动画播放的各属性当前值中使用的时间线信息。在一个实施例中，定时系统 208 可以接收来自交互元素（未示出）的事件和属性变更信息以生成在动画生成中使用的触发（例如，事件触发、属性触发等）。例如，交互元素可以监视鼠标和 / 或键盘活动，并且将信息提供给定时系统 208 以引发该定时系统 208 开始时间线并将触发提供给动画系统 602。

[0080] 如前所述，图形子系统 212 可以生成需要用来在显示设备上再现动画的数据。

[0081] 动画系统 602 在此实施例中执行改变用于创建可视动画性效果的属性值的操作。在一个实施例中，动画系统包括诸如高和低级合成和动画引擎 206 和 208 的组件，以及传递系统 604。同样地，动画系统 602 可以从应用程序 202 接收为动画性属性指定的“来自”和“至（to）”值，以及为执行这些动画指定的时间段。在某些实施例中，有关动画性属性的默认“至”和 / 或“来自”值可在应用程序未指定时使用。

[0082] 传递系统 604 在此实施例中可在动画要在一属性上执行同时其他动画仍然在该属性上执行时使用。在某些实施例中，传递系统 604 可以执行若干种不同的传递，诸如下表 1 中所列。术语“旧”动画指的是当应用程序为一特定属性启动另一动画时正在对该相同属性运行的动画。在某些场景中，旧动画可以是两个或更多其他动画的“合成”（即，如表 1 定义的合成）。在某些实施例中，动画系统 602 可以被实现以为一个或多个预先确定的场景提供默认的传递行为。在此实施例中，表 1 描述了能够在动画中设置的传递活动属性的各个值。传递行为属性基本上描述了使用什么作为该动画的“来自”值的值以及要对旧动画做出的动作。

[0083] 表 1

[0084]

传递	行为
合成	旧动画继续，并且对每个再现循环而言，新动画使用旧动画的输出值作为其“来自”值。新动画的输出值被用作再现该属性的值。
替换	旧动画被释放，并且当各值未被指定时新动画使用基“至”和“来自”值。
快照和替换	旧动画的当前值被获取（即，快照）并且旧动画被释放。新动画随后使用快照作为“来自”值并且在未被指定时作为有关“至”值的基值。

[0085]

快照和保留	旧动画的快照被获取并且旧动画被允许继续运行而不影响动画性属性的值。新动画使用快照作为“来自”值并且在未被指定时作为有关“至”值的基值。当新动画结束时，旧动画随后被允许恢复该属性的动画。
快照和保留/暂停	旧动画的快照被获取并且旧动画被暂停。新动画使用快照作为“来自”值并且在未被指定时作为有关“至”值的基值。当新动画结束时，旧动画随后被允许从暂停时的动画状态中动画播放该属性。
混合	使用动画的加权平均值
队列	排队未决动画直到现有动画完成。

[0086] 再次使用按钮示例，应用程序可被写入以引发按钮在鼠标光标移动到按钮上时经一特定时间段增大至一特定大小，并在鼠标光标离开该按钮时经一特定时间段缩小回原始大小。在此场景下，用户然后在该特定时间段完全经过之前将光标快速移至并移开该按钮。于是，一当“MouseOver（鼠标在其上）”，“增大”动画会开始，，随后在增大动画完成之前一当“MouseExit（鼠标退出）”，“缩小”动画开始。

[0087] 如果有关“增大”动画的传递行为是如上表 1 所述的“快照和替换”行为（可以是指定或默认的），那么在光标离开按钮之时，由“增大”动画产生的按钮的长度和宽度的当前将被快照。在此实施例中，快照的值存储在动画存储对象 606 内。动画系统 602 在此实施例中为每个动画播放属性创建动画存储对象 606。在某些实施例中，动画存储对象 602 被配置用以为使按钮在被图形子系统 212 再现时出现增大和缩小的属性长度和宽度的改变执行计算。

[0088] 在有关动画性属性的值被快照之后，“增大”动画被释放（即，“增大”动画实际上被终止），并且“缩小”动画将利用作为快照值的长度和宽度的有关属性的“来自”值启动。如果没有影响按钮宽度和长度属性的其他动画被启动，“缩小”动画就完成并在动画结束处，按钮的长度和宽度属性将处于基值。

[0089] “快照和替换”行为使得在“增大”和“缩小”动画之间的过渡在用户看来是平滑的。也就是说，按钮看上去是平滑地增大至某一大小并在随后从同一大小平滑地缩小。在不具有前述快照能力的其他系统中，应用程序 202 或动画系统 602 很可能不得不提供有关“缩小”动画的“来自”值（例如，基“来自”值或者“增大”动画的目标值），从而很可能导致按钮看上去意外“失灵”。也就是说，按钮看上去增大到某一大小，随后看上去突然改变至一个不同的大小，并由此按钮大小开始缩小。快照的前述优点也应用于如上表 1 所述的“快照和保留”和“快照和保留 / 暂停”行为。

[0090] 如下是使用“快照和替换”传递行为的代码的示例。该代码在 MouseEnter（鼠标进入）上增大并在 MouseLeave（鼠标离开）上缩小，其中快照和替换行为是默认的传递行为。

[0091]

```
<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"
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"
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>
```

[0092] 示例性“快照和替换”操作流程

[0093] 图 7 根据一个实施例示出了用于执行“快照和替换”行为的操作流程 700。操作流程 700 可以在任何合适的计算环境内执行。例如，操作流程 700 可由诸如系统 600（图 6）的系统执行。因此，操作流程 700 的描述可能涉及图 6 的至少一个组件。然而，对图 6 组件的任何这类参考仅出于描述的目的，并且应该理解图 6 的实现是操作流程 700 的一个非限制性环境。

[0094] 在框 702，检测到有关一动画的触发。在一个实施例中，诸如动画系统 602（图 6）的动画系统检测到该触发（例如，一事件触发或属性触发）。在某些实施例中，动画系统可以从交互元素（以上结合图 6 所述）接收触发的通知。在此示例性操作流程中，没有其他动画当前正为与该动画相关联的属性运行（即，该属性在可视对象被“动画播放”时改变）。

[0095] 在框 704，分配动画存储对象。在一个实施例中，动画系统创建诸如动画存储对象 606（图 6）的动画对象来处理与该动画相关联的属性的各动画。在某些实施例中，可为每个要动画播放的属性创建动画存储对象。当有关一属性的所有动画都已被完成时，在某些实施例中就可以删除该动画存储对象。在某些实施例中，当动画到达其充满值时，该充满值会保留在该动画存储对象内，但存储信息的其余信息则被删除。

[0096] 在框 706，将与被触发动画相关联的属性的基值存储在动画存储对象内。在一个实施例中，动画系统将该基值存储在动画存储对象内。基值从诸如属性系统 314（图 6）的属性系统中获取。

[0097] 在框 708，启动动画并由动画存储对象保持该动画。在一个实施例中，动画存储对象实际上将以屏幕刷新率重新计算有关该属性的值直到该动画完成（或被替换或被暂停），以使得可视对象看上去被动画播放。动画存储对象还可以在该动画正运行的同时保持该动画（即，维持定义该动画当前状态的信息以使得该动画在被保留或保留 / 暂停的情况下可以被定位并继续）。

[0098] 在框 710，在第一动画完成之前检测到涉及同一属性的第二动画的触发。如框 702，动画系统在一个实施例中检测这另外的触发。在此示例性操作流程中，第二动画被配置为具有传递行为，诸如表 1 中定义的“快照和替换”行为。

[0099] 在框 712，保存由第一动画产生的属性的当前值（即，快照）。在一个实施例中，快照被存储在动画存储对象内。例如，动画系统能够根据第一动画计算有关下一屏幕刷新率的属性的值并将该算出的值（即，快照）存储在动画存储对象内。

[0100] 在框 714，释放第一动画。在一个实施例中，动画系统删除存储在动画存储对象内的第一动画的状态信息。

[0101] 在框 716，启动第二动画并由动画存储对象保持该第二动画。根据该实施例，快照被用作运行第二动画时该属性的“来自”值。与框 708 的操作类似，动画系统实际上将以屏幕刷新率重新计算有关该属性的值直到该动画完成（或被替换或被暂停），以使得可视对象看上去被动画播放。动画存储对象还可以在该第二动画正运行的同时保持该第二动画（即，维持定义该第二动画当前状态的信息以使得该第二动画在被保留或保留 / 暂停的情况下可以被定位并继续）。

[0102] 虽然以特定的次序顺序地示出并描述了操作流程 700，但是在其它实施例中，在各框内描述的操作也可以按不同次序，多次和 / 或并行执行。此外，在某些实施例中，在各框中描述的一个或多个操作可以被分成另一框，或者被省略或合并。

[0103] 图 8 根据一个实施例示出了如何根据“快照和替换”传递行为（表 1）获取与一动画相关联的有关属性的值。在此实施例中，未被动画播放的属性 802 具有基值 804。在一个实施例中，该基值被存储在诸如先前描述的属性系统 314（图 3、图 6）的属性系统内。

[0104] 当启动影响属性 802 的动画 808（在图 8 中被指示为动画 A）时，基值 804 通常被用作动画 808 的“来自”值 810。动画 808 的“至”值 812 可由创建该动画 808 的应用程序

（诸如，图 6 中的应用程序 202）指定。在某些场景中，该应用程序可能未指定“至”值和“来自”值，从而导致使用基值作为默认。在一个实施例中，“至”值和“来自”值被存储在诸如动画存储对象 606（图 6）的动画存储对象内，该动画存储对象在动画 808 被启动时可由诸如动画系统 602（图 6）的动画系统分配。当动画 808 正运行时，在每次屏幕被刷新时计算属性的值（随着动画 808 改变）并输出作为再现值 814。

[0105] 在此示例中，动画 816（在图 8 中被指示为动画 B）在动画 808 完成之前被启动。在此示例中，动画 816 已被配置为具有“快照和替换”行为。当被启动时，动画 816 快照该属性并使用该快照作为动画 816 的“来自”值 818。动画 816 的“至”值 820 可由创建该动画 816 的应用程序指定。在其他场景中，“至”值 820 可能未被指定，从而导致对基“至”值的使用。在某些实施例中，这些“至”值和“来自”值被存储在前述动画存储对象内。此外，释放动画 808 以使得只有动画 816 影响属性的值。当动画 816 运行时，在每次屏幕被刷新时计算属性的值（随着动画 816 改变）并将其作为再现值 822 输出。

[0106] 如果传递行为是“快照和保留”，动画 808 就不会被释放。相反地，动画 808 可被允许继续运行，但是会仅使用动画 816（即，忽略动画 808）来计算属性的值。随后当动画 816 完成时，可以使用由动画 816 生成的属性的最新值来恢复动画 808。类似地，如果传递行为是“快照和保留 / 暂停”，动画 808 不会被释放。相反地，动画 808 可被允许在动画 816 启动时它所具有的状态下暂停，并且仅使用动画 816 来计算属性的值。随后当动画 816 完成时，可以使用动画 808 暂停时的属性来恢复动画 808。

[0107] 图 9 是根据一个实施例在分层动画之间关系的表示。在此实施例中，可视对象 900 由应用程序（诸如，图 6 的应用程序 202）定义，该应用程序具有逻辑排列在各层内的各式动画。在此示例中，可视对象包括层 902-1 至 902-N。层 902-1 包括动画 904-1A 和动画 904-1B ；层 902-2 包括动画 904-2A 和动画 904-2B ；并且如此类推直至层 902-N 包括动画 904-NA 和动画 904-NB。虽然在此示例中存在 N 层并且每层包括两个动画，但是层数和每层的动画数可以取决于应用程序开发人员的设计选择而改变。

[0108] 在一个实施例中，第一层 902-1 被定义以包含由事件触发（与属性触发相对）启动的所有动画。于是，层 2 至 N 的动画可由属性触发启动。在某些实施例中，每层与单个属性触发相关联并且包括由该属性触发启动的动画。此外，在此实施例中，正运行的动画可以按从第一层 902-1 到第 N 层 902-N 的属性被执行。另外，在此实施例中，一层内的动画可用包括快照（例如，表 1 中描述的“快照和替换”、“快照和保留”、“快照和保留 / 暂停”）和 / 或合成的传递行为定义，而各层间的过渡则具有合成传递行为。如下将结合图 10 描述合成传递行为的示例。

[0109] 在某些实施例中，可以预定义一层内的合成优先级。由较高优先级动画输出的值将被用作下一较低优先级动画的“来自”值。

[0110] 在一个实施例中，优先级如下定义 ：(1) 来自式样的触发的动画，带有在其中它们出现在定义该层内多个“式样”触发的动画的优先级的标记或代码中的次序 ；(2) 来自本地元素触发的触发的动画，带有在其中它们出现在定义该层内多个“本地元素”触发的动画的优先级的标记或代码中的次序 ；以及 (3) 从故事板或方法调用所应用的动画，带有在其中它们被应用以定义有关该“被应用的故事板 / 方法调用”的优先级的次序。

[0111] 在某些实施例中，当一层的所有动画都退出时，就移除该层。可以从层“堆栈”中

的任何位置移除一层。

[0112] 图 10 根据一个实施例示出了两个动画的合成。在此实施例中，未被动画播放的属性 1002 具有基值 1004。在一个实施例中，该基值被存储在诸如先前描述的属性系统 314（图 3、图 6）的属性系统内。

[0113] 当启动影响属性的动画 1008（在图 10 中被指示为动画 A）时，基值 1004 通常被用作动画 1008 的“来自”值 1010。动画 1008 的“至”值 1012 可由创建该动画 1008 的应用程序（诸如，图 6 中的应用程序 202）指定。在一个实施例中，“至”值和“来自”值被存储在诸如动画存储对象 606（图 6）的动画存储对象内。虽然未在图 10 中示出，但是当动画 1008 自行运行时，在每次屏幕被刷新时计算该属性的值（随着动画 1008 改变）并将其作为“再现”值提供给诸如图形子系统 212（图 3、图 6）的图形子系统。

[0114] 在此示例中，动画 1016（在图 10 中被指示为动画 B）在动画 1008 完成之前被启动。在此示例中，动画 1016 已被配置为具有“合成”行为。与“快照和替换”行为相反，动画 1008 继续运行以使得该属性的瞬时值持续更新。然而，由动画 100 产生的该属性的瞬时值不用作“再现”值。相反地，作为动画 1008 结果而计算的瞬时值作为一种中间值而被用作动画 106 的“来自”值 1018。动画 1016 的“至”值 1020 可由创建动画 1016 的应用程序指定。在某些实施例中，这些“至”值和“来自”值被存储在前述动画存储对象内。由动画 1016 产生的属性的值随后可用作再现值 1022。当动画 1008 和 1016 运行时，在每次屏幕被刷新时计算属性的值（随着动画 1008 和 1016 的合成改变）并将其作为再现值 1022 输出。

[0115] 本说明书通篇引述了“一个实施例”、“一实施例”或“一示例性实施例”，这意指特定的说明特征、结构或特性是涵盖在本发明至少一个实施例中的。于是，这种短语的使用可引用到不止一个实施例中。进一步，在一个或多个实施例中，所述特征、结构或特性能以任何合适的方式组合。

[0116] 然而，本领域熟练的技术人员会认识到本发明也可不用一个或多个特定细节，或者采用其它方法、资源、材料等来实践。仅仅为了避免模糊本发明的方面，这里未示出和详述其它情况下我们熟知的结构、资源或操作。

[0117] 虽然已经示出并描述了本发明的示例性实施例和应用形式，但是应当认识到本发明不局限于上述的精确配置和资源。本领域普通技术人员也对这里所述本发明方法和系统的排列、操作和细节做出显而易见的各种修改、变更或变化而不背离本发明所声明的范围。

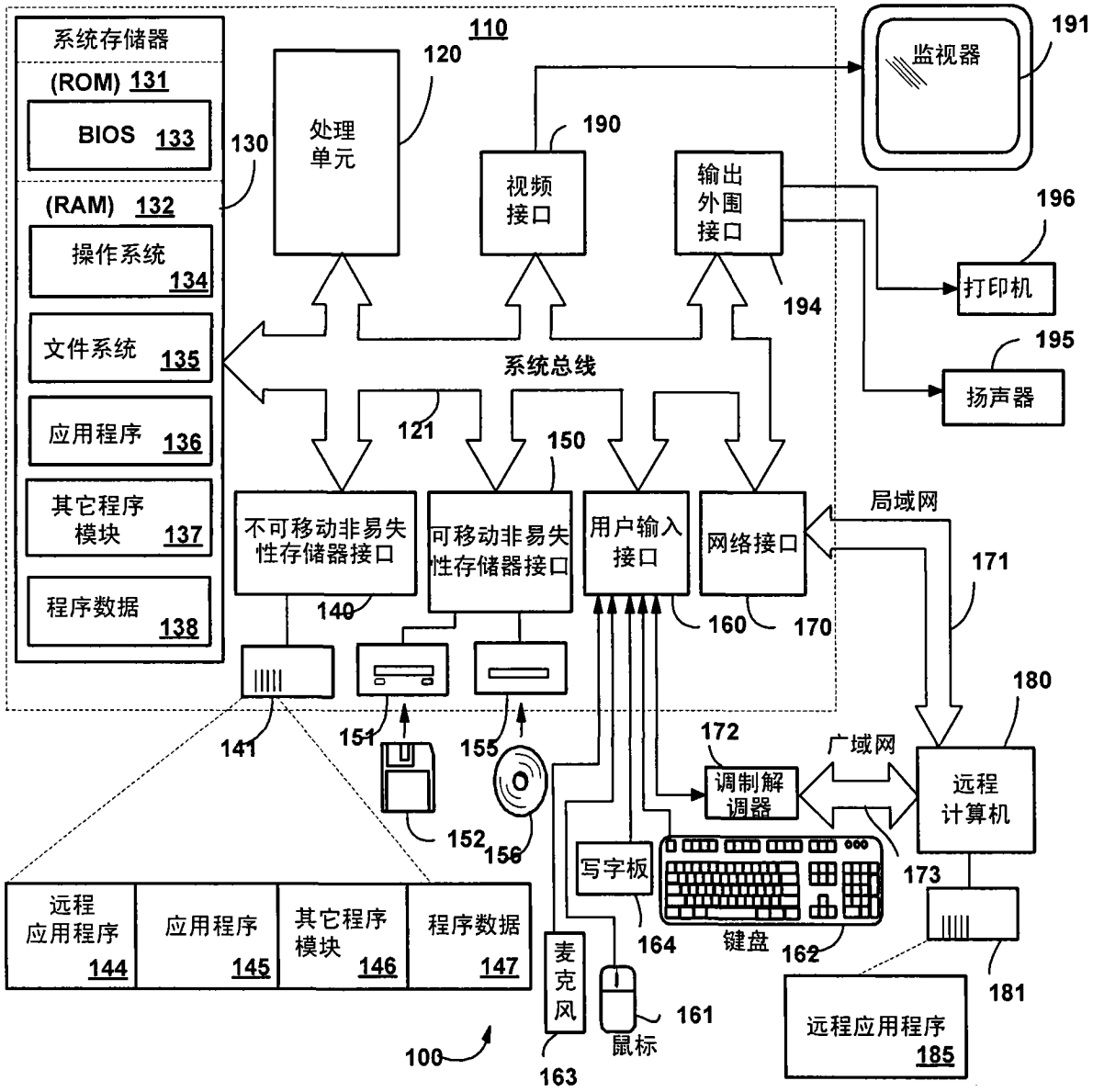

系统存储器
(ROM) 131
BIOS 133
(RAM) 132
操作系统 134
文件系统 135
应用程序 136
其它程序模块 137
程序数据 138
130
处理单元
120
110
视频接口
190
监视器
191
输出外围接口
194
196
打印机
195
扬声器
系统总线
121
150
不可移动非易失性存储器接口
可移动非易失性存储器接口
用户输入接口
网络接口
160
170
局域网
171
140
141
151
155
152
156
172
调制解调器
广域网
远程计算机
180
写字板
164
键盘
162
173
181
远程应用程序 185
麦克风
163
161
鼠标
100
远程应用程序 144
应用程序 145
其它程序模块 146
程序数据 147

图 1

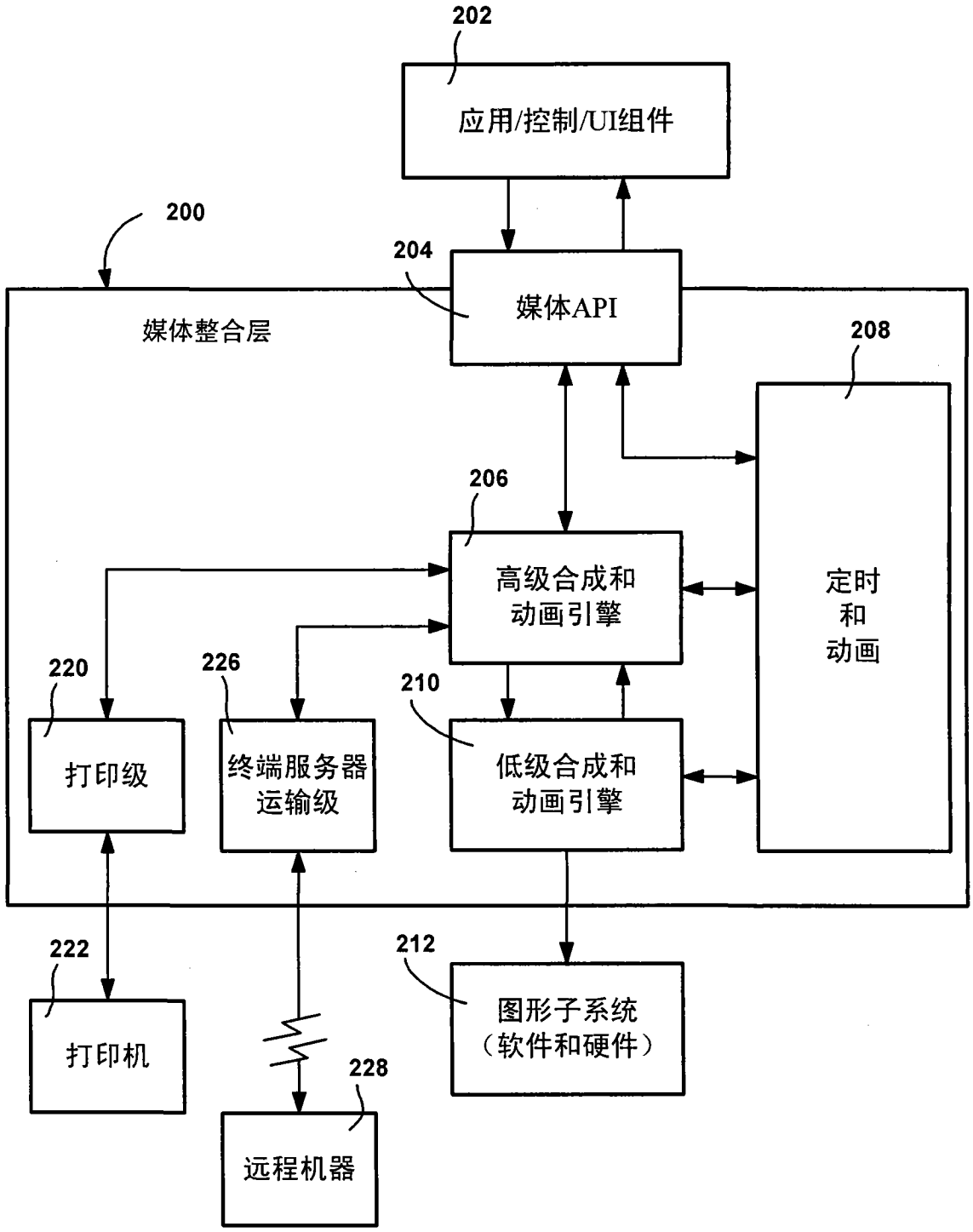

202
应用/控制/UI组件
200
204
媒体API
媒体整合层
208
206
高级合成和
动画引擎
定时
和
动画
220
226
210
打印级
终端服务器
运输级
低级合成和
动画引擎
222
212
图形子系统
（软件和硬件）
228
打印机
远程机器

图 2

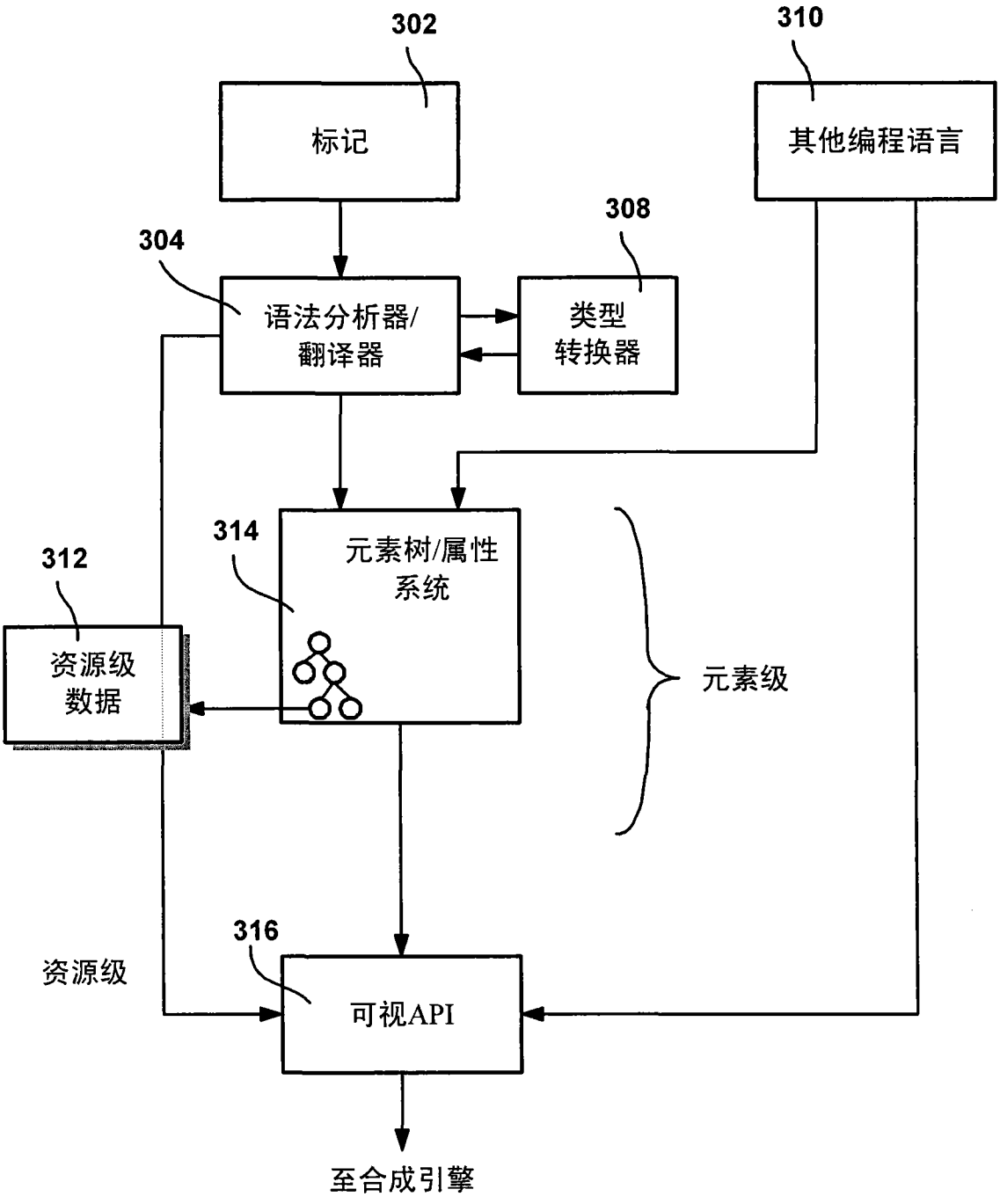

302
标记
310
其他编程语言
308
304
语法分析器/
翻译器
类型
转换器
314
元素树/属性
系统
312
资源级
数据
元素级
316
资源级
可视API
至合成引擎

图 3

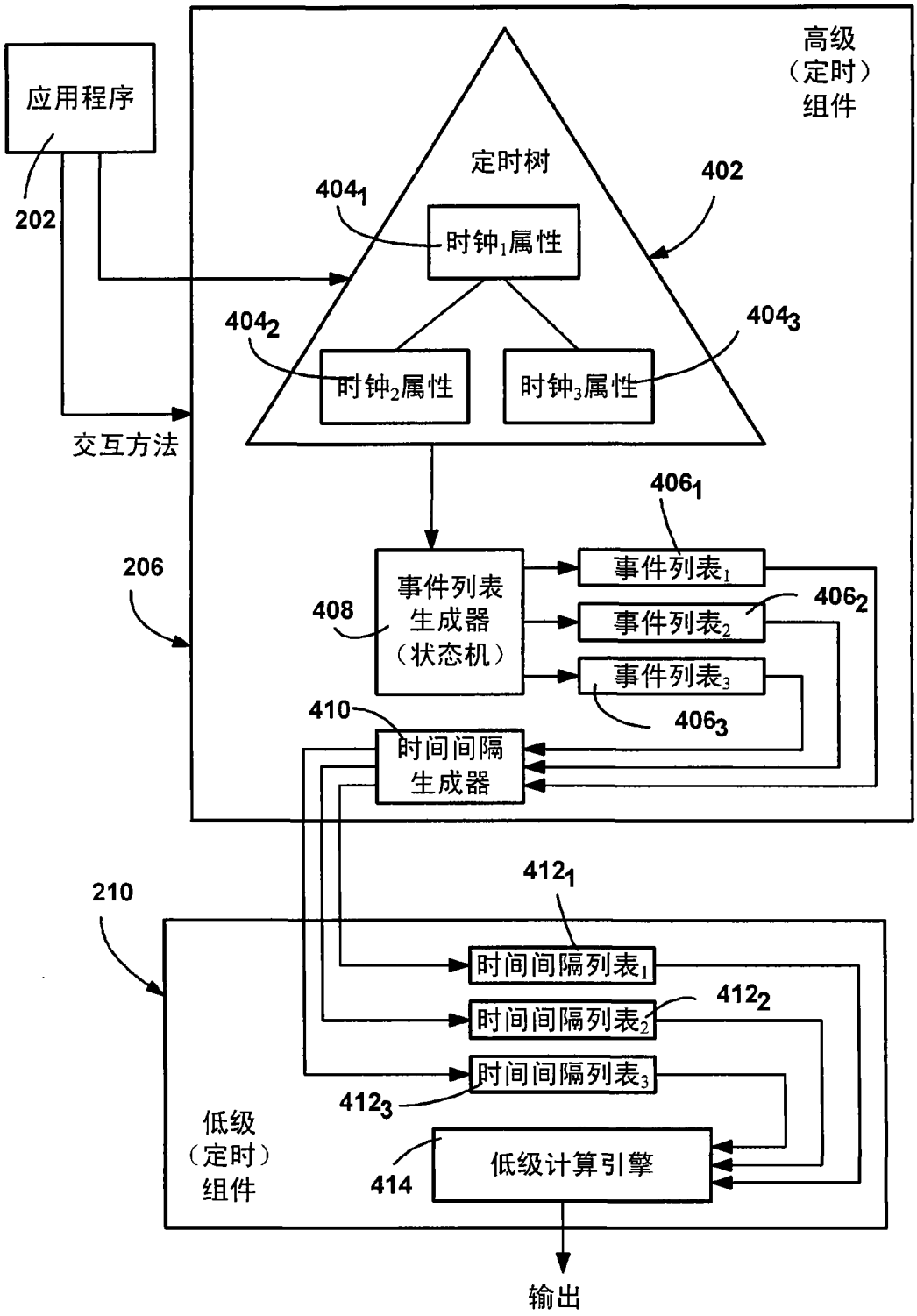

应用程序
202
高级
（定时）
组件
定时树
402
404_1
时钟$_1$属性
404_2
404_3
时钟$_2$属性
时钟$_3$属性
交互方法
406_1
206
事件列表
生成器
（状态机）
408
事件列表$_1$
事件列表$_2$
事件列表$_3$
406_2
406_3
410
时间间隔
生成器
210
412_1
时间间隔列表$_1$
时间间隔列表$_2$
412_2
时间间隔列表$_3$
412_3
低级
（定时）
组件
414
低级计算引擎
输出

图 4

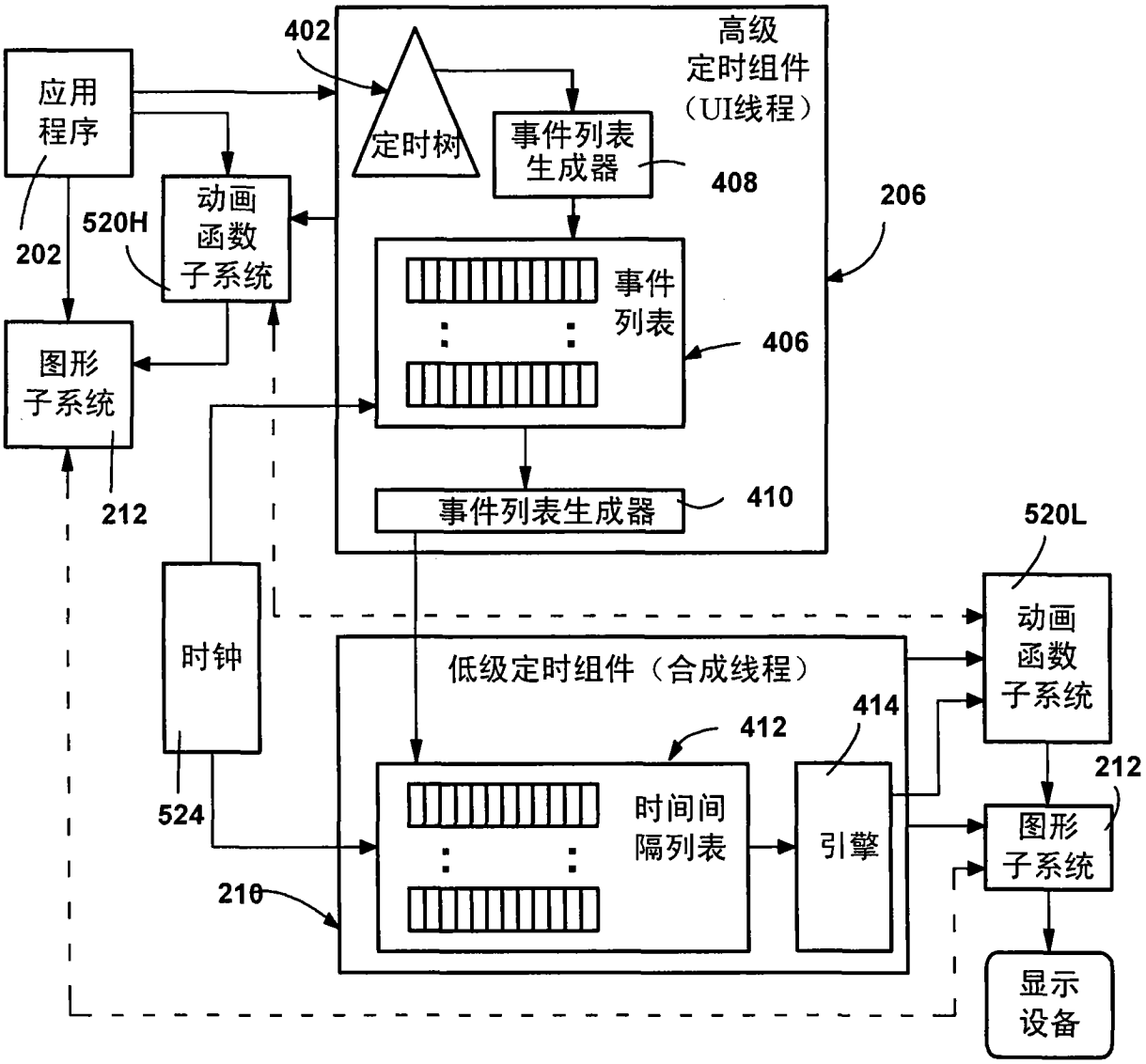

应用程序
202
图形子系统
212
动画函数子系统
520H
402
定时树
高级定时组件（UI线程）
事件列表生成器
408
206
事件列表
406
事件列表生成器
410
时钟
524
低级定时组件（合成线程）
412
414
时间间隔列表
引擎
210
520L
动画函数子系统
212
图形子系统
显示设备

图 5

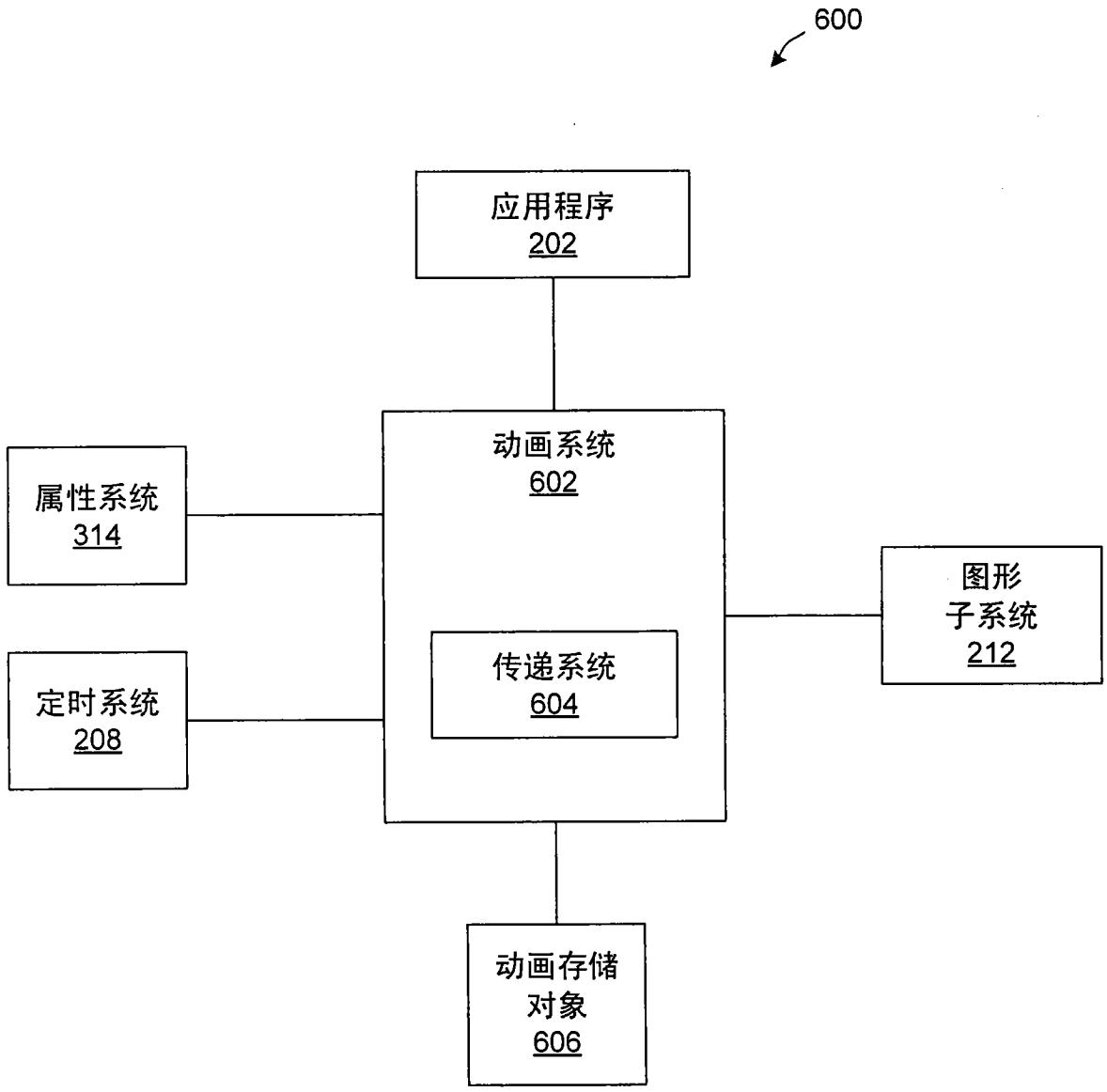

600
应用程序
202
动画系统
602
属性系统
314
图形
子系统
212
传递系统
604
定时系统
208
动画存储
对象
606

图 6

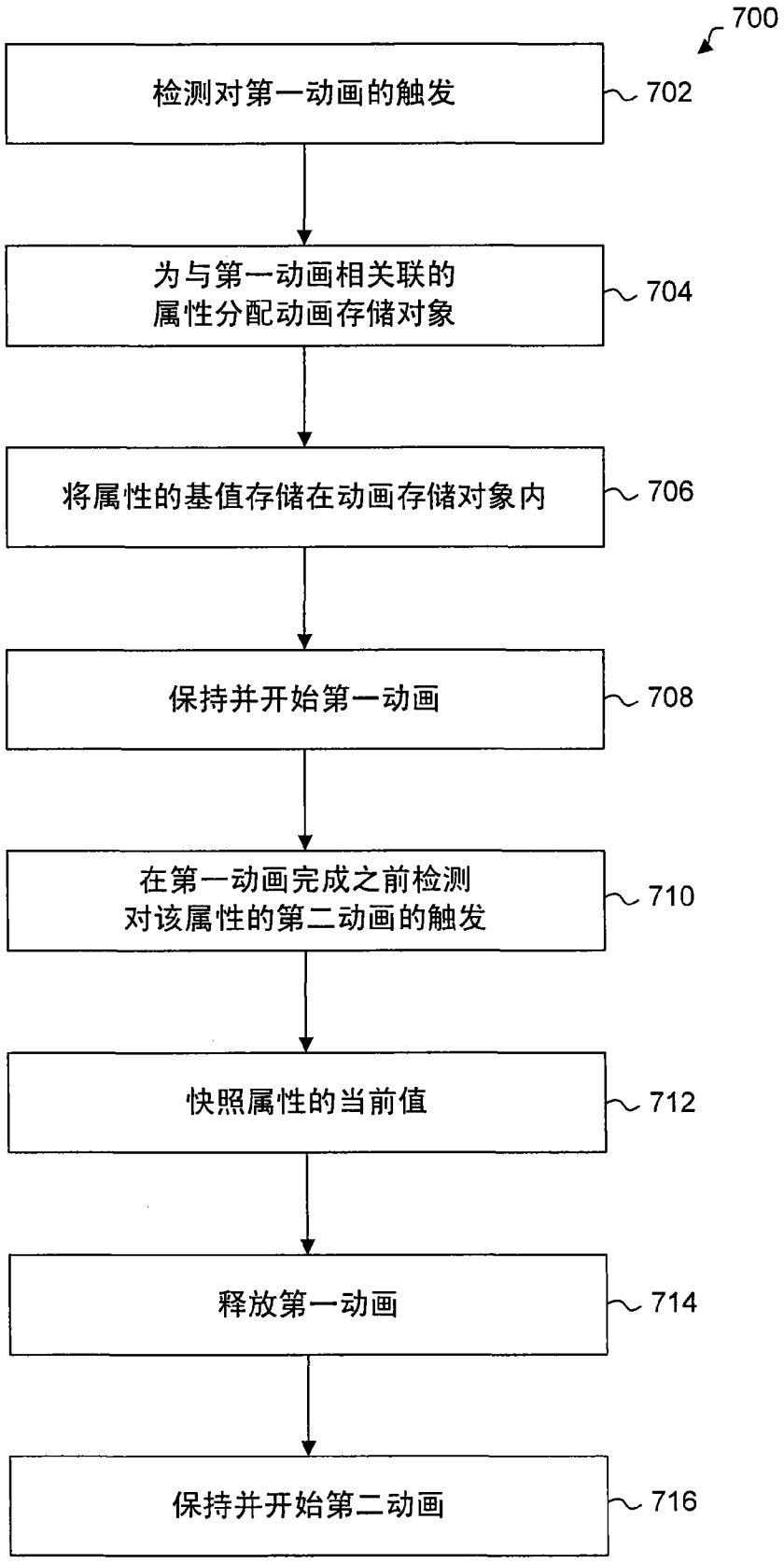

700
检测对第一动画的触发
702
为与第一动画相关联的
属性分配动画存储对象
704
将属性的基值存储在动画存储对象内
706
保持并开始第一动画
708
在第一动画完成之前检测
对该属性的第二动画的触发
710
快照属性的当前值
712
释放第一动画
714
保持并开始第二动画
716

图 7

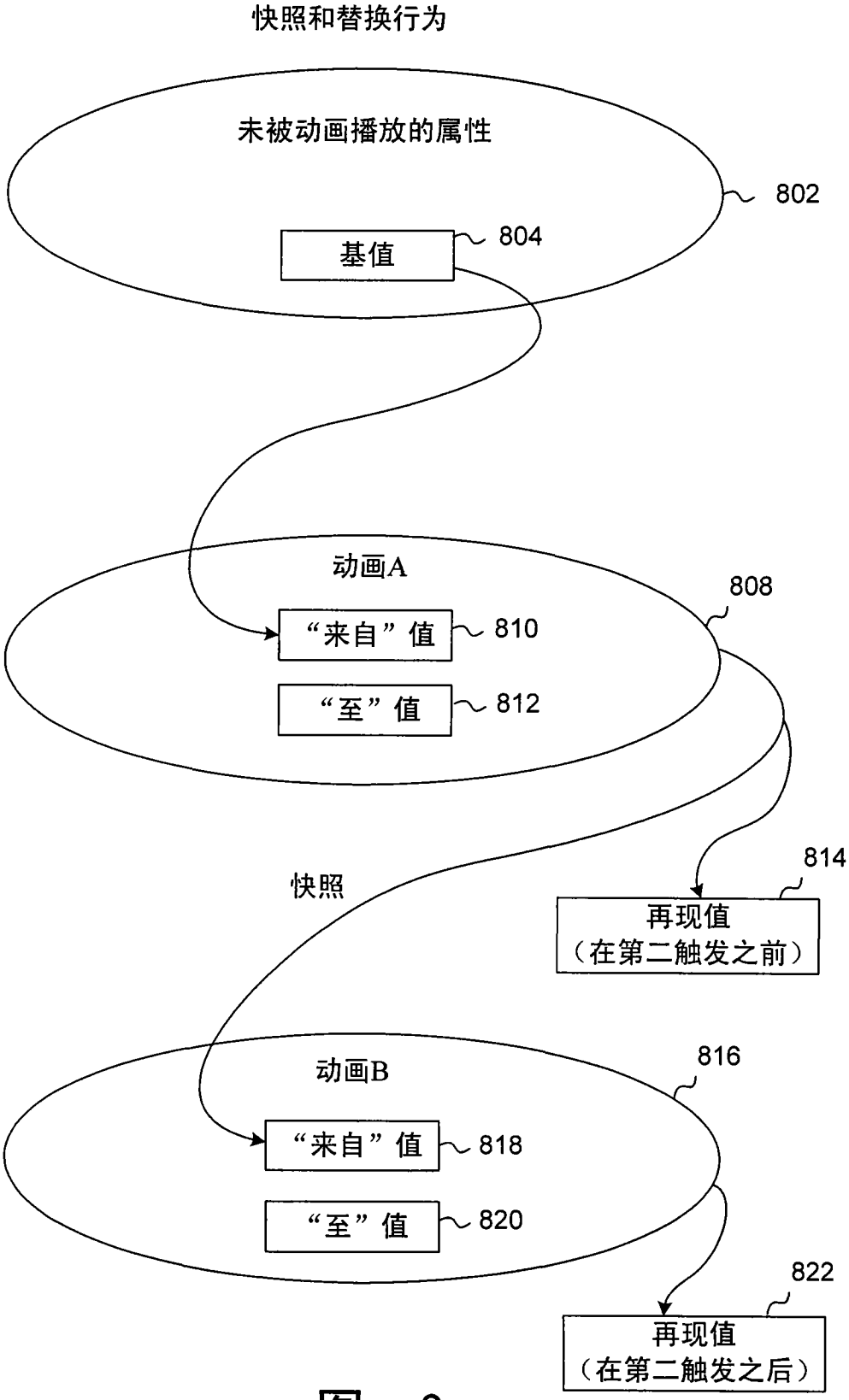

快照和替换行为
未被动画播放的属性
802
基值
804
动画A
808
“来自”值
810
“至”值
812
快照
814
再现值
（在第二触发之前）
动画B
816
“来自”值
818
“至”值
820
822
再现值
（在第二触发之后）

图 8

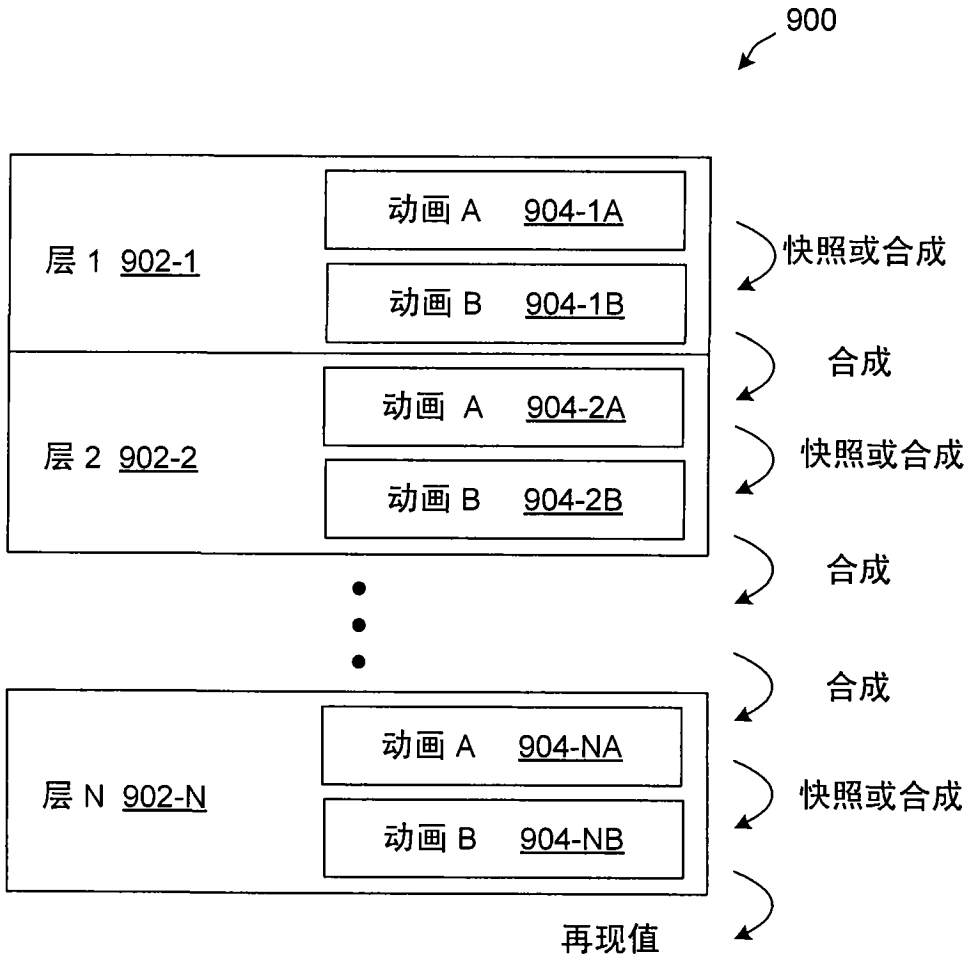

900
层 1 902-1
动画 A 904-1A
动画 B 904-1B
层 2 902-2
动画 A 904-2A
动画 B 904-2B
层 N 902-N
动画 A 904-NA
动画 B 904-NB
快照或合成
合成
快照或合成
合成
合成
快照或合成
再现值

图 9

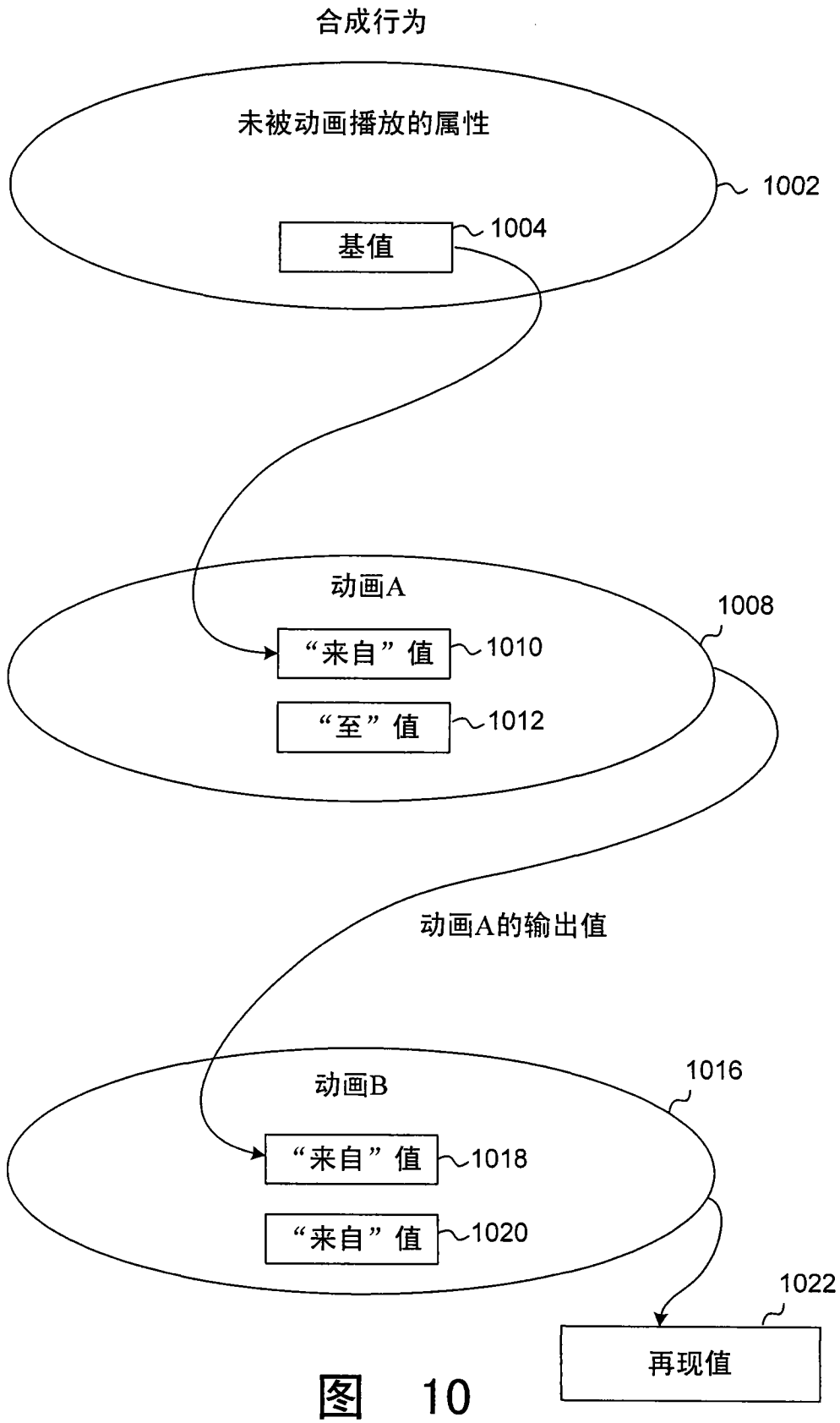

合成行为
未被动画播放的属性
1002
基值
1004
动画A
1008
“来自”值
1010
“至”值
1012
动画A的输出值
动画B
1016
“来自”值
1018
“来自”值
1020
1022
再现值

图 10