(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 102422264 B
(45)授权公告日 2014.06.11

(21)申请号 201080020598.0
(22)申请日 2010.03.12
(30)优先权数据
61/210,332 2009.03.16 US
12/566,660 2009.09.24 US
(85)PCT国际申请进入国家阶段日
2011.11.11
(86)PCT国际申请的申请数据
PCT/US2010/027118 2010.03.12
(87)PCT国际申请的公布数据
WO2010/107669 EN 2010.09.23
(73)专利权人 苹果公司
地址 美国加利福尼亚
(72)发明人 B·A·摩尔 J·H·沙法尔
(74)专利代理机构 北京市金杜律师事务所 11256
代理人 王茂华
(51)Int.Cl.
G06F 9/44(2006.01)
G06F 3/0488(2013.01)
(56)对比文件
US 5627959 A,1997.05.06, 全文.
CN 101052939 A,2007.10.10, 全文.
CN 101198925 A,2008.06.11, 全文.
审查员 白露霜

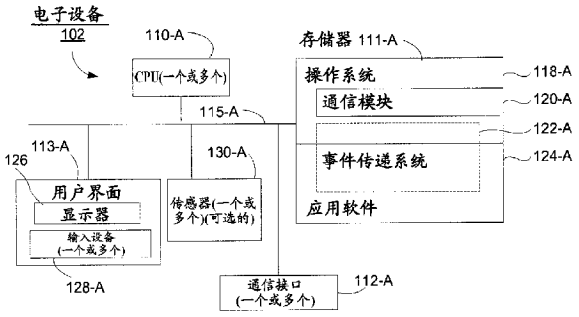
权利要求书2页 说明书20页 附图13页

(54)发明名称

事件识别

(57)摘要

公开了一种执行包括具有多个视图的视图分层结构的软件的方法，所述方法显示视图分层结构的一个或多个视图。所述方法执行与特定视图关联的软件元件，其中每个特定视图包括事件识别器。每个事件识别器具有基于子事件的事件定义和事件处理器，所述事件处理器指定关于目标的动作，并被配置成响应于事件识别，把动作发送给所述目标。所述方法检测子事件的序列，把视图分层结构的视图之一识别为选中视图，所述选中视图确定视图分层结构中的哪些视图是主动涉及视图。所述方法把相应子事件传递给每个主动涉及视图的事件识别器，其中视图分层结构中的主动涉及视图的每个事件识别器处理相应子事件，之后处理子事件序列中的下一个子事件。



CN 102422264 B

1. 一种用于事件识别的方法，包括：

在被配置成执行包括具有多个视图的视图分层结构的软件的电子设备处：

显示视图分层结构的一个或多个视图；

执行一个或多个软件元件，每个软件元件与特定视图关联，其中每个特定视图包括一个或多个事件识别器，每个事件识别器具有：

基于一个或多个子事件的事件定义，以及

事件处理器，其中事件处理器：

指定关于目标的动作，以及

被配置成响应于事件识别器检测到与事件定义对应的事件，把所述动作发送给所述目标；

检测一个或多个子事件的序列；以及

把视图分层结构的视图之一识别为选中视图，其中选中视图确定视图分层结构中的哪些视图是主动涉及视图；

把相应子事件传递给视图分层结构内的每个主动涉及视图的一个或多个事件识别器，以及

在视图分层结构中的主动涉及视图的每个事件识别器处，处理相应子事件，之后处理子事件序列中的下一个子事件。

2. 按照权利要求 1 所述的方法，其中视图分层结构中的主动涉及视图的每个事件识别器同时处理一个或多个子事件的序列。

3. 按照权利要求 1 所述的方法，其中视图分层结构内的第一个主动涉及视图被配置成阻止向与所述第一个主动涉及视图相关的事件识别器传递相应子事件。

4. 按照权利要求 1 所述的方法，其中视图分层结构内的第一个主动涉及视图被配置成阻止向与所述第一个主动涉及视图相关的事件识别器传递相应子事件，除非所述第一个主动涉及视图是选中视图。

5. 按照权利要求 1 所述的方法，其中视图分层结构内的第二个主动涉及视图被配置成阻止向与第二个主动涉及视图相关的事件识别器和与第二个主动涉及视图的先辈相关的事件识别器传递相应子事件。

6. 按照权利要求 1 所述的方法，其中所述多个事件识别器中的至少一个是具有姿态定义和姿态处理器的姿态识别器。

7. 按照权利要求 1 所述的方法，其中电子设备还包含被配置成检测多触摸姿态的触摸敏感表面。

8. 按照权利要求 1 所述的方法，其中一个或多个子事件的序列包括原始触摸事件。

9. 按照权利要求 1 所述的方法，其中每个事件识别器具有一组事件识别状态，所述一组事件识别状态至少包括事件可能状态、事件不可能状态和事件被识别状态。

10. 按照权利要求 9 所述的方法，还包括：

如果事件识别器进入事件可能状态，那么事件处理器开始其对应动作的准备以便传递给目标。

11. 按照权利要求 10 所述的方法，还包括：

如果事件识别器进入事件被识别状态，那么事件处理器完成其对应动作的准备以便传

递给目标。

12. 按照权利要求 1 所述的方法，其中多个事件识别器并行地独立处理一个或多个子事件的序列。

13. 按照权利要求 1 所述的方法，其中多个事件识别器中的一个或多个事件识别器适于推迟传递子事件序列的一个或多个子事件，直到在所述事件识别器识别该事件之后为止。

14. 按照权利要求 1 所述的方法，其中：

多个事件识别器中的一个或多个事件识别器，其中包含独占事件识别器，被配置成进行独占事件识别，以及

在独占事件识别器任意之一识别事件之后，事件传递系统阻止视图分层结构中的主动涉及视图的任何非独占事件识别器识别相应的事件。

15. 按照权利要求 14 所述的方法，其中一个或多个独占事件识别器包括事件例外列表。

16. 按照权利要求 1 所述的方法，其中事件定义定义用户输入操作。

17. 一种用于事件识别的设备，包括：

用于显示视图分层结构的一个或多个视图的装置；

用于执行一个或多个软件元件的装置，每个软件元件与特定视图关联，其中每个特定视图包括一个或多个事件识别器，每个事件识别器具有：

基于一个或多个子事件的事件定义，以及

事件处理器，其中事件处理器：

指定关于目标的动作，以及

被配置成响应于事件识别器检测到与事件定义对应的事件，把所述动作发送给所述目标；

用于检测一个或多个子事件的序列的装置；

用于把视图分层结构的视图之一识别为选中视图的装置，其中选中视图确定视图分层结构中的哪些视图是主动涉及视图；

用于把相应子事件传递给在视图分层结构内的每个主动涉及视图的一个或多个事件识别器的装置；以及

用于处理相应子事件，之后处理子事件序列中的下一个子事件的装置。

18. 按照权利要求 17 所述的设备，还包括：

用于同时处理一个或多个子事件的序列的装置。

19. 按照权利要求 17 所述的设备，其中视图分层结构内的第一个主动涉及视图被配置成阻止向与所述第一个主动涉及视图相关的事件识别器传递相应子事件。

20. 按照权利要求 17 所述的设备，其中视图分层结构内的第一个主动涉及视图被配置成阻止向与所述第一个主动涉及视图相关的事件识别器传递相应子事件，除非所述第一个主动涉及视图是选中视图。

事件识别

技术领域

[0001] 公开的实施例一般涉及用户界面处理。更具体地，公开的实施例涉及识别用户界面事件的设备和方法。

背景技术

[0002] 计算设备一般包括可用于与计算设备交互的用户界面。用户界面可包括显示器和/或用于与用户界面的各个方面交互的输入设备，诸如键盘、鼠标和触摸敏感表面。在以触摸敏感表面作为输入设备的一些设备中，在特定的上下文中（例如，在第一应用程序的特定模式下），第一组基于触摸的姿态（例如，下述之中的两个或两个以上：轻击，双击，水平轻扫，垂直轻扫，捏合，张开，双指轻扫）被识别为正确的输入，而在其它上下文中（例如，在不同的应用程序中和/或第一应用程序内的不同模式或上下文中），其它不同的多组基于触摸的姿态被识别为正确的输入。结果，识别和响应基于触摸的姿态所需的软件和逻辑会变得复杂，从而每次更新应用程序或者在计算设备中增加新的应用程序时，都需要修正。在利用除基于触摸的姿态之外的输入源的用户界面中，也会出现这些和类似的问题。

[0003] 从而，希望的是具有识别基于触摸的姿态和事件，以及来自其它输入源的姿态和事件的综合性架构或机制，所述综合性架构或机制能够容易地适用于计算设备上的所有应用程序的几乎所有上下文或模式，并且当更新应用程序或者在计算设备中增加新的应用程序时，不需要或几乎不需要任何修正。

发明内容

[0004] 为了解决上述缺陷，一些实施例提供一种方法，所述方法在被配置成执行包括具有多个视图的视图分层结构的软件的电子设备处：显示视图分层结构的一个或多个视图；执行一个或多个软件元件，每个软件元件与特定视图关联，其中每个特定视图包括一个或多个事件识别器。每个事件识别器具有基于一个或多个子事件的事件定义，和事件处理器，其中事件处理器指定关于目标的动作，和被配置成响应于事件识别器检测到与事件定义对应的事件，把所述动作发送给所述目标。所述方法还检测一个或多个子事件的序列，和把视图分层结构的视图之一识别为选中视图，其中选中视图确定视图分层结构中的哪些视图是主动涉及视图。所述方法还把相应子事件传递给视图分层结构内的每个主动涉及视图的事件识别器，其中视图分层结构中的主动涉及视图的每个事件识别器处理相应子事件，之后处理子事件序列中的下一个子事件。

[0005] 一些实施例提供一种方法，所述方法在被配置成执行包括具有多个视图的视图分层结构的软件的电子设备处：显示视图分层结构的一个或多个视图；执行一个或多个软件元件，每个软件元件与特定视图关联，其中每个特定视图包括一个或多个事件识别器。每个事件识别器具有基于一个或多个子事件的事件定义，和事件处理器，其中事件处理器指定关于目标的动作，和被配置成响应于事件识别器检测到对应于事件定义的事件，把所述动作发送给目标。所述方法还检测一个或多个子事件的序列，和把视图分层结构的视图之一

识别为选中视图，其中选中视图确定视图分层结构中的哪些视图是主动涉及视图。所述方法还把相应子事件传递给视图分层结构内的每个主动涉及视图的事件识别器，和在视图分层结构中的主动涉及视图的事件识别器处处理相应子事件的同时，做出子事件识别判定。

[0006] 一些实施例提供一种计算机可读存储介质，所述计算机可读存储介质保存由计算机系统或设备的一个或多个处理器执行的一个或多个程序，所述一个或多个程序包括显示具有多个视图的视图分层结构的一个或多个视图的一个或多个应用程序。所述一个或多个应用程序包括一个或多个软件元件，每个软件元件与特定视图关联，其中每个特定视图包括一个或多个事件识别器。每个事件识别器具有基于一个或多个子事件的事件定义，和事件处理器，其中事件处理器：指定关于目标的动作，和被配置成响应于事件识别器检测到与事件定义对应的事件，把所述动作发送给所述目标；事件管理指令在被计算机系统或设备的一个或多个处理器执行时，使计算机系统或设备：检测一个或多个子事件的序列；把视图分层结构的视图之一识别为选中视图，其中选中视图确定视图分层结构中的哪些视图是主动涉及视图；和把相应子事件传递给视图分层结构内的每个主动涉及视图的事件识别器，其中视图分层结构中的主动涉及视图的每个事件识别器处理相应子事件，之后处理子事件序列中的下一个子事件。

[0007] 一些实施例提供一种设备，所述设备包括显示器，一个或多个处理器，存储器和保存在存储器中的一个或多个程序，所述一个或多个程序被配置成显示具有多个视图的视图分层结构的一个或多个视图。所述一个或多个程序包括一个或多个软件元件，每个软件元件与特定视图关联，其中每个特定视图包括一个或多个事件识别器。每个事件识别器具有基于一个或多个子事件的事件定义，和事件处理器，其中事件处理器指定关于目标的动作，和被配置成响应于事件识别器检测到与事件定义对应的事件，把所述动作发送给所述目标。所述设备的程序还包括事件传递程序，当被所述设备的一个或多个处理器执行时，所述事件传递程序使所述设备检测一个或多个子事件的序列；把视图分层结构的视图之一识别为选中视图，其中选中视图确定视图分层结构中的哪些视图是主动涉及视图；和在视图分层结构中的主动涉及视图的事件识别器处理相应子事件的同时，做出子事件识别判定。

[0008] 在一些实施例中，提供一种设备，所述设备包含一个或多个处理器，存储器，和保存在存储器中的一个或多个程序，所述一个或多个程序被配置成管理具有多个编程层次的编程分层结构的一个或多个编程层次的执行。所述一个或多个程序包括一个或多个软件元件，每个软件元件与特定的编程层次关联，其中每个特定的编程层次包括一个或多个事件识别器。事件识别器具有基于一个或多个子事件的事件定义，和事件处理器器，其中事件处理器指定关于目标的动作，并被配置成响应于事件识别器检测到与事件定义对应的事件，把所述动作发送给所述目标。所述设备还包括事件传递程序，在被所述设备的一个或多个处理器执行时，所述事件传递程序使所述设备检测一个或多个子事件的序列；把编程分层结构的编程层次之一识别为选中层次，其中选中层次确定编程分层结构中的哪些编程层次是主动涉及编程层次；和把相应子事件传递给编程分层结构内的每个主动涉及编程层次的事件识别器，其中编程分层结构中的主动涉及编程层次的每个事件识别器处理相应子事件，之后处理子事件序列中的下一个子事件。

附图说明

[0009] 图 1A 和 1B 是图解说明按照一些实施例的电子设备的方框图。

[0010] 图 2 是按照一些实施例的示例性电子设备的输入 / 输出处理栈的示图。

[0011] 图 3A 图解说明按照一些实施例的示例性视图分层结构。

[0012] 图 3B 和 3C 是图解说明按照一些实施例的示例性事件识别器方法和数据结构的方框图。

[0013] 图 4A 和 4B 是图解说明按照一些实施例的示例性状态机的流程图。

[0014] 图 4C 图解说明按照一些实施例的关于示例性的一组子事件的图 4A 和 4B 的示例性状态机。

[0015] 图 5A-5C 图解说明按照一些实施例的利用示例性的事件识别器状态机的示例性子事件序列。

[0016] 图 6A 和 6B 是按照一些实施例的事件识别方法流程图。

[0017] 附图中，相同的附图标记表示对应的部分。

具体实施方式

[0018] 下面详细参考实施例，附图中图解说明了实施例的例子。在下面的详细说明中，陈述了众多的具体细节，以便透彻理解本发明。不过，对本领域的技术人员来说，显然可在没有这些具体细节的情况下实践本发明。在其它情况下，为了避免不必要地模糊实施例的特征，没有详细说明公知的方法、过程、组件、电路和网络。

[0019] 另外要明白，尽管这里可以使用术语第一、第二等来描述各个要素，不过这些要素不应受这些术语限制。这些术语只是用于区分一个要素和另一个要素。例如，第一触点可被称为第二触点，类似地，第二触点可被称为第一触点，而不脱离本发明的范围。第一触点和第二触点都是触点，不过它们不是同一触点。

[0020] 在本发明的说明书中使用的术语只是用于说明特定的实施例，并不意图限制本发明。在本发明的说明书和所附权利要求书中使用的单数形式意图还包括复数形式，除非上下文明确地另有说明。另外应明白这里使用的“和 / 或”指的是包括关联的列举项目中的一个或多个的任何和所有可能组合。还要明白的是当用在本说明书中时，术语“包括”和 / 或“包含”指定所陈述的特征、整数、步骤、操作、元件和 / 或组件的存在，但是不排除一个或多个其它的特征、整数、步骤、操作、元件、组件和 / 或它们的组的存在或添加。

[0021] 这里使用的术语“如果”可被解释成意味着“当 ... 时”、“一旦 ...”，“响应于确定 ...”或者“响应于检测到 ...”，这取决于上下文。类似地，短语“如果确定 ...”或者“如果检测到 [规定的条件和事件]”可被解释成意味着“一旦确定 ...”或“响应于确定 ...”或“一旦检测到 [规定的条件和事件]”或者“响应于检测到 [规定的条件和事件]”，这取决于上下文。

[0022] 如上所述，在以触摸敏感表面作为输入设备的一些设备中，在特定的上下文中（例如，在第一应用程序的特定模式下），第一组基于触摸的姿态（例如，下述之中的两个或两个以上 :轻击，双击，水平轻扫，垂直轻扫）被识别为正确的输入，在其它上下文中（例如，在不同的应用程序中和 / 或第一应用程序内的不同模式或上下文中），其它不同的多组基于触摸的姿态被识别为正确的输入。结果，识别和响应基于触摸的姿态所需的软件和逻辑会变得复杂，并且每次更新应用程序或者在计算设备中添加新的应用程序时，都需要修正。

[0023] 在下面说明的实施例中，基于触摸的姿态是事件。一旦识别了预定事件（例如，与应用程序的当前上下文中的正确输入对应的事件），关于该事件的信息被传递给该应用程序。在本文的上下文中，所有基于触摸的姿态都对应于事件。此外，每个相应的事件被定义为一系列的子事件。在具有多触摸显示设备（这里通常称为“屏幕”）或其它多触摸敏感表面，并且接受基于多触摸的姿态的设备中，定义基于多触摸的事件的子事件包括多触摸子事件（要求两根或两根以上的手指同时接触设备的触摸敏感表面）。例如，在具有多触摸敏感显示器的设备中，当用户的手指首次触摸屏幕时，可开始子事件的相应多触摸序列。当一根或多根另外的手指随后或者同时触摸屏幕时，可发生另外的子事件，另外当触摸屏幕的所有手指在屏幕内移动时，可发生其它子事件。当从屏幕抬起用户的最后一根手指时，所述序列结束。

[0024] 当利用基于触摸的姿态来控制在具有触摸敏感表面的设备中运行的应用程序时，触摸具有时间和空间特征。时间特征（称为相位）指示触摸何时刚刚开始，触摸是在移动还是固定不动的，和触摸何时结束－即，何时从屏幕抬起手指。触摸的空间特征是其中发生触摸的一组视图或用户界面窗口。其中检测到触摸的视图或窗口可对应于编程或视图分层结构内的编程层次。例如，其中检测到触摸的最低层次视图可被称为选中视图，并且至少部分根据开始基于触摸的姿态的初始触摸的选中视图，可确定被识别为正确输入的一组事件。

[0025] 图 1A 和 1B 是图解说明按照一些实施例的电子设备 102 和 104 的方框图。电子设备 102 和 104 可以是任意电子设备，包括但不限于桌上型计算机系统，膝上型计算机系统，移动电话机，智能电话机，个人数字助手或导航系统。电子设备还可以是具有被配置成呈现用户界面的触摸屏显示器（例如，显示器 156，图 1B）的便携式电子设备，具有被配置成呈现用户界面的触摸屏显示器的计算机，具有触摸敏感表面和被配置成呈现用户界面的显示器的计算机，或者任何其它形式的计算设备，包括但不限于：消费电子设备，移动电话机，视频游戏系统，电子音乐播放器，平板 PC，电子书阅读系统，ebook，PDA，电子记事本，电子邮件设备，膝上型计算机或其它计算机，信息站计算机，自动贩卖机，智能设备等等。电子设备 102 和 104 可分别包括用户界面 113-A 和 113-B。

[0026] 在一些实施例中，电子设备 102 和 104 包括触摸屏显示器。在这些实施例中，用户界面 113（即，113-A 或 113-B）包括屏幕上 (on-screen) 键盘（未示出），用户使用所述屏幕上键盘与电子设备 102 和 104 交互。另一方面，键盘可以独立于和不同于电子设备 102 和 104。例如，键盘可以是与电子设备 102 或 104 耦接的有线或无线键盘。

[0027] 在一些实施例中，电子设备 102 可包括与电子设备 102 耦接的显示器 126 和一个或多个输入设备 128-A（例如，键盘、鼠标、跟踪球、麦克风、物理按钮（一个或多个）、触摸板等等）。在这些实施例中，输入设备 128-A 中的一个或多个可视情况独立于和不同于电子设备 102。例如，所述一个或多个输入设备可包括下述中的一个或多个：键盘、鼠标、触控板、跟踪球和电子笔，所述输入设备任意之一可视情况与电子设备分离。可选的是，设备 102 或 104 可包括一个或多个传感器 130，诸如一个或多个加速度计、陀螺仪、GPS 系统、扬声器、红外 (IR) 传感器、生物识别传感器、照相机等。注意作为输入设备 128 或作为传感器 130 的各种示例性设备的上述说明对这里说明的实施例的操作无关紧要，并且这里描述成输入设备的任意输入或传感器设备可同样被描述成传感器，反之亦然。在一些实施例中，一个或多个传感器 130 产生的信号被用作检测事件的输入源。

[0028] 在一些实施例中，电子设备 104 包括与电子设备 104 耦接的触摸敏感显示器 156（即，具有触摸敏感表面的显示器）和一个或多个输入设备 128-B。在一些实施例中，触摸敏感显示器 156 具有检测两个或两个以上不同的并发（或者至少部分并发）触摸的能力，在这些实施例中，有时把显示器 156 称为多触摸显示器或者多触摸敏感显示器。

[0029] 在这里讨论的电子设备 102 或 104 的一些实施例中，输入设备 128 被置于电子设备 102 或 104 中。在其它实施例中，输入设备 128 中的一个或多个独立于和不同于电子设备 102 或 104；例如，输入设备 128 中的一个或多个可利用电缆（例如，USB 电缆）或无线连接（例如，蓝牙连接），与电子设备 102 或 104 耦接。

[0030] 当使用输入设备 128 时，或者当分别在电子设备 102 或 104 的触摸敏感显示器 156 上进行基于触摸的姿态时，用户产生由电子设备 102 或 104 的一个或多个 CPU 110 处理的一系列子事件。在一些实施例中，电子设备 102 或 104 的所述一个或多个 CPU 110 处理所述一系列子事件，以识别事件。

[0031] 电子设备 102 或 104 一般分别包括一个或多个单核芯或多核芯处理单元（“CPU”）110，以及一个或多个网络接口或其它通信接口 112。电子设备 102 或 104 分别包括存储器 111 和用于互连这些组件的一个或多个通信总线 115。通信总线 115 可包括互连和控制系统组件（这里未说明）之间的通信的电路（有时称为芯片集）。如上简要所述，电子设备 102 和 104 可视情况分别包括用户界面 113，用户界面 113 包括显示器 126 和多触摸显示器 156。此外，电子设备 102 和 104 一般包括输入设备 128（例如，键盘、鼠标、触摸屏、触摸敏感表面、多触摸屏幕、数字小键盘等等）。在一些实施例中，输入设备包括屏幕上输入设备（例如，显示设备的触摸敏感表面）。存储器 111 可包括高速随机存取存储器，诸如 DRAM、SARM、DDR RAM 或其它随机存取固态存储器设备；并且可包括非易失性存储器，诸如一个或多个磁盘存储设备，光盘存储设备，闪速存储设备，或者其它非易失性固态存储设备。存储器 111 可视情况包括远离 CPU（一个或多个）110 的一个或多个存储设备。存储器 111，或者另一方面，存储器 111 内的非易失性存储设备（一个或多个）包含计算机可读存储介质。在一些实施例中，存储器 111 保存下述程序，模块和数据结构，或者它们的子集：

[0032] • 包括用于处理各种基本系统服务和用于执行依赖于硬件的任务的各过程的操作系统 118；

[0033] • 通信模块 120（分别在电子设备 102 和 104 中），用于通过电子设备 102 或 104 的一个或多个相应通信接口 112（有线或无线）和一个或多个通信网络，诸如因特网、其它广域网、局域网、城域网等等，分别把电子设备 102 或 104 连接到其它设备；

[0034] • 事件传递系统 122（分别在电子设备 102 和 104 中），在各个备选实施例中，事件传递系统 122 可在操作系统 118 内或者在应用软件 124 中实现；不过，在一些实施例中，事件传递系统 122 的一些方面可在操作系统 118 中实现，而其它方面可在应用软件 124 中实现；和

[0035] • 分别在应用软件 124 中的一个或多个应用程序（例如，电子邮件应用程序，web 浏览器应用程序，文本消息接发应用程序等等）。

[0036] 每个上述元件可被保存在前面提及的存储设备中的一个或多个中，并且对应于实现这里说明的功能的一组指令。所述一组指令可由一个或多个处理器（例如，一个或多个 CPU 110）执行。上述模块或程序（即，指令集）不需要被实现成独立的软件程序、过程或模

块，从而在各个实施例中，可以组合或者以其它方式重新排列这些模块的各个子集。在一些实施例中，存储器 111 可保存上面识别的模块和数据结构的子集。此外，存储器 111 可保存上面没有说明的另外的模块和数据结构。

[0037] 图 2 是按照本发明的一些实施例的示例性电子设备或装置（例如，设备 102 或 104）的输入 / 输出处理栈 200 的示图。设备的硬件（例如，电子电路）212 处于输入 / 输出处理栈 200 的底层。硬件 212 可包括各种硬件接口组件，诸如在图 1A 和 / 或 1B 中描述的组件。硬件 212 还可包括上述传感器 130 中的一个或多个。输入 / 输出处理栈 200 的所有其它元件 (202-210) 是处理从硬件 212 接收的输入和产生通过硬件用户接口（例如，显示器，扬声器，设备振动促动器中的一个或多个）呈现的各种输出的软件过程，或者软件过程的各部分。

[0038] 驱动器或一组驱动器 210 与硬件 212 通信。驱动器 210 能够接收和处理从硬件 212 接收的输入数据。核心操作系统（“OS”）208 能够与驱动器（一个或多个）210 通信。核心 OS 208 能够处理从驱动器（一个或多个）210 接收的原始输入数据。在一些实施例中，驱动器 210 可被认为是核心 OS 208 的一部分。

[0039] 一组 OS 应用编程接口（“OS API”）206 是与核心 OS 208 通信的软件过程。在一些实施例中，API 206 包含在设备的操作系统中，不过位于在核心 OS 208 之上的层次。API 206 是为供在这里讨论的电子设备或装置上运行的应用程序使用而设计的。用户界面 (UI) API 204 可利用 OS API206。在设备上运行的应用软件（“应用程序”）202 能够利用 UI API 204，以便与用户通信。UI API 204 又能够与层次较低的元件通信，最终与例如多触摸显示器 156 的各个用户接口硬件通信。

[0040] 虽然每层输入 / 输出处理栈 200 能够利用在它下面的那层，不过并不总是要求如此。例如，在一些实施例中，应用程序 202 可以偶尔与 OS API206 通信。通常，在 OS API 层 206 或其之上的各层不能直接访问核心 OS 208，驱动器（一个或多个）210 或硬件 212，因为这些层被认为是专用的。层 202 中的应用程序和 UI API 204 通常把调用引导到 OS API 206，OS API 206 再访问核心 OS 208，驱动器（一个或多个）210 和硬件 212 各层。

[0041] 换句话说，电子设备 102 或 104 的一个或多个硬件元件 212，和在设备上运行的软件，诸如例如驱动器 210（示于图 2 中），核心 OS（操作系统）208（示于图 2 中），操作系统 API 软件 206（示于图 2 中），及应用程序和用户界面 API 软件 204（示于图 2 中）检测在一个或多个输入设备 128 和 / 或触摸敏感显示器 156 处的输入事件（其对应于姿态中的子事件），并生成或更新（保存在设备 102 或 104 的存储器中的）数据结构，一组当前活动的事件识别器利用所述数据结构判定输入事件是否和何时对应于待传递给应用程序 202 的事件。事件识别方法，设备和计算机程序产品的实施例在下面更详细说明。

[0042] 图 3A 描述示例性的视图分层结构 300，在本例中，视图分层结构 300 是显示在最外层视图 302 中的搜索程序。最外层视图 302 一般包含用户可以直接与之交互的整个用户界面，并包括从属视图，例如，

[0043] • 搜索结果面板 304，它对搜索结果分组，并且能够被垂直滚动；

[0044] • 接受文本输入的搜索栏 306；和

[0045] • 对应用程序分组以便快速访问的主功能行 (home row)310。

[0046] 在本例中，每个从属视图包括较低层次的从属视图。在其它例子中，在分层结构的

不同分枝中，分层结构 300 中的视图层次的数目可不同，一个或多个从属视图具有较低层次的从属视图，一个或多个其它从属视图没有任何这种较低层次的从属视图。继续图 3A 中所示的例子，搜索结果面板 304 包含每个搜索结果的独立从属视图 305（从属于面板 304）。这里，本例在称为地图视图 305 的从属视图中，显示一个搜索结果。搜索栏 306 包括这里称为清除内容图标视图 307 的从属视图，当用户对视图 307 中的清除内容图标执行特定动作（例如，单次触摸或轻击姿态）时，清除内容图标视图 307 清除搜索栏的内容。主功能行 310 包括分别对应于联系人应用程序，电子邮件应用程序，web 浏览器和 iPod 音乐界面的从属视图 310-1、310-2、310-3 和 310-4。

[0047] 触摸子事件 301-1 表示在最外层视图 302 中。假定触摸子事件 301-1 的位置在搜索结果面板 304 和地图视图 305 之上，该触摸子事件还分别以 301-2 和 301-3 的形式表现在这些视图之上。该触摸子事件的主动涉及视图包括视图搜索结果面板 304，地图视图 305 和最外层视图 302。下面参考图 3B 和 3C，提供关于子事件传递和主动涉及视图的其它信息。

[0048] 视图（和对应的编程层次）可被嵌套。换句话说，视图可包括其它视图。从而，与第一视图相关的软件元件（一个或多个）（例如，事件识别器）可包括或者链接到与第一视图内的视图相关的一个或多个软件元件。虽然一些视图可与应用程序关联，不过，其它视图可与高层 OS 元件，诸如图形用户界面、窗口管理器等关联。

[0049] 为了简化后续讨论，一般将只提及视图和视图分层结构，不过必须明白在一些实施例中，所述方法可使用具有多个编程层次的编程分层结构，和 / 或视图分层结构进行操作。

[0050] 图 3B 和 3C 描述与事件识别器相关的示例性方法和结构。图 3B 描述当事件处理器与视图的分层结构内的特定视图相关时的事件处理用方法和数据结构。图 3C 描述当事件处理器与编程层次的分层结构内的特定层次相关时的事件处理用方法和数据结构。事件识别器全局方法 312 和 350 分别包括选中视图和选中层次判定模块 314 和 352，活动事件识别器判定模块 316 和 354，及子事件传递模块 318 和 356。

[0051] 选中视图和选中层次判定模块 314 和 352 分别提供用于判定在一个或多个视图，例如，图 3A 中描述的具有三个主要分枝的示例性视图分层结构 300 内，子事件发生在何处的软件过程。

[0052] 图 3B 的选中视图判定模块 314 接收与子事件 -- 例如，在最外层视图 302，搜索结果（地图视图 305）和搜索结果面板视图 304 上的表示成 301-1 的用户触摸 -- 有关的信息。选中视图判定模块 314 把选中视图识别为分层结构中应处理该子事件的最底层视图。在多数情况下，选中视图是其中发生起始子事件（即，构成事件或潜在事件的子事件序列中的第一个子事件）的最底层视图。一旦选中视图被识别，那么它将接收与为其识别了该选中视图的相同触摸或输入源相关的所有子事件。

[0053] 在一些实施例中，图 3C 的选中层次判定模块 352 可利用类似的进程。

[0054] 事件识别器全局方法 312 和 350 的活动事件识别器判定模块 316 和 354 分别判定视图分层结构内的哪个或哪些视图应接收特定的子事件序列。图 3A 描述接收子事件 301 的一组示例性活动视图 302、304 和 305。在图 3A 的例子中，活动事件识别器判定模块 304 会判定顶层视图 302，搜索结果面板 304 和地图视图 305 是主动涉及视图，这是因为这些视图包含用子事件 301 表示的触摸的物理位置。注意即使触摸子事件 301 完全局限在与地图

视图 305 相关的区域内，搜索结果面板 304 和顶层视图 302 仍会保留在主动涉及视图中，因为搜索结果面板 304 和顶层视图 302 是地图视图 305 的先辈。

[0055] 在一些实施例中，活动事件识别器判定模块 316 和 354 利用类似的进程。

[0056] 子事件传送模块 318 把子事件传递给主动涉及视图的事件识别器。以图 3A 的例子为例，用触摸标记 301-1、301-2 和 301-3，在分层结构的不同视图中表示用户的触摸。在一些实施例中，表示该用户触摸的子事件数据由子事件传递模块 318 传递给在主动涉及视图 -- 即，顶层视图 302，搜索结果面板 304 和地图视图 305-- 处的事件识别器。此外，视图的事件识别器能够接收开始于该视图内的事件的后续子事件（例如，当初始子事件发生于该视图内时）。换句话说，视图能够接收与在该视图内开始的用户交互相关的子事件，即使所述子事件是在该视图之外继续进行的。

[0057] 在一些实施例中，在与子事件传递模块 318 使用的进程类似的进程中，子事件传递模块 356 把子事件传递给主动涉及编程层次的事件识别器。

[0058] 在一些实施例中，为每个主动涉及的事件识别器生成独立的事件识别器结构 320 或 360，并将其保存在设备的存储器中。事件识别器结构 320 和 360 一般分别包括事件识别器状态 334、374（下面当参照图 4A 和 4B 时更详细说明）并且分别包括事件识别器特有代码 338、378，事件识别器特有代码 338、378 分别具有状态机 340、380。事件识别器结构 320 还包括视图分层结构参考 336，而事件识别器结构 360 包括编程分层结构参考 376。特定事件识别器的每个实例严格地参考一个视图或编程层次。（特定事件识别器的）视图分层结构参考 336 或编程分层结构参考 376 被用于确定哪个视图或编程层次在逻辑上与相应的事件识别器耦接。

[0059] 视图元数据 341 和层次元数据 381 分别包括关于视图或层次的数据。视图元数据或层次元数据至少包括会影响对事件识别器的子事件传递的下述属性：

[0060] • 停止属性 342、382，当对视图或编程层次设定停止属性 342、382 时，该属性阻止向与视图或编程层次相关的事件识别器及其在视图或编程分层结构中的先辈传递子事件。

[0061] • 跳过属性 343、383，当对视图或编程层次设定跳过属性 343、383 时，该属性阻止向与视图或编程层次相关的事件识别器传递子事件，但是允许向其在视图或编程层次中的先辈传递子事件。

[0062] • NoHit（非选中）跳过属性 344、384，当对视图设定 NoHit 跳过属性 344、384 时，该属性阻止向与该视图相关的事件识别器传递子事件，除非该视图是选中视图。如上所述，选中视图判定模块 314 把选中视图（或者在选中层次判定模块 352 的情况下为选中层次）识别为分层结构中应处理该子事件的最底层视图。

[0063] 事件识别器结构 320 和 360 分别包括元数据 322、362。在一些实施例中，元数据 322、362 包括指示事件传递系统应如何进行对主动涉及事件识别器的子事件传递的可配置属性、标记和列表。在一些实施例中，元数据 322、362 包括指示事件识别器如何相互交互的可配置属性、标记和列表。在一些实施例中，元数据 322、362 包括指示是否把子事件传递给视图或编程分层结构之中各不相同的层次的可配置属性、标记和列表。在一些实施例中，事件识别器元数据 322、362 和视图或层次元数据（分别为 341、381）的组合可被用于配置事件传递系统：a) 进行对主动涉及事件识别器的子事件传递，b) 指示事件识别器如何相互交互，和 c) 指示子事件是否和何时被传递给视图或编程分层结构中的各个层次。

[0064] 注意，在一些实施例中，相应的事件识别器把事件识别动作 333、373 发送给它的由事件识别器的结构 320、360 的字段指定的相应目标 335、375。把动作发送给目标完全不同于把子事件发送（和延期发送）给相应的选中视图或层次。

[0065] 保存在对应事件识别器的相应事件识别器结构 320、360 中的元数据属性至少包括：

[0066] • 独占性标记 324、364，当对事件识别器设定独占性标记 324、364 时，该标记指示一旦事件识别器识别到事件，事件传递系统应停止把子事件传递给主动涉及视图或编程层次的任何其它事件识别器（在例外列表 326、366 中列举的任何其它事件识别器除外）。当子事件的接收导致特定事件识别器进入独占状态（该独占状态由其对应的独占性标记 324 或 364 指示）时，那么后续子事件只被传递给处于独占状态的该事件识别器（以及在例外列表 326、366 中列举的任何其它事件识别器）。

[0067] • 一些事件识别器结构 320、360 可包括独占性例外列表 326、366。当包含在相应事件识别器的事件识别器结构 320、360 中时，该列表 326、366 指示即使在相应的事件识别器已进入独占状态之后，也将继续接收子事件的一组事件识别器（如果有的话）。例如，如果单击事件的事件识别器进入独占状态，并且当前涉及的视图包括双击事件的事件识别器，那么列表 320，360 会列出双击事件识别器，以致即使在检测到单击事件之后，也能够识别双击事件。因而，独占性例外列表 326、366 允许事件识别器识别共享公共子事件序列的不同事件，例如，单击事件识别不排除其它事件识别器对双击或三击事件的后续识别。

[0068] •一些事件识别器结构 320、360 可包括等待列表 327、367。当包含在相应事件识别器的事件识别器结构 320、360 中时，该列表 327、367 指示在相应的事件识别器能够识别相应事件之前，必须进入事件不可能状态或事件取消状态的一组事件识别器（如果有的话）。实际上，与具有等待列表 327、367 的事件识别器相比，列出的事件识别器具有更高的事件识别优先权。

[0069] • 延迟触摸开始标记 328、368，当对事件识别器设定延迟触摸开始标记 328、368 时，该标记使事件识别器延迟把子事件（包括触摸开始或手指向下子事件，和后续事件）发送给事件识别器的相应选中视图或层次，直到已判定子事件序列不对应于该事件识别器的事件种类之后为止。该标记可用于在已识别姿态的情况下，防止选中视图或层次不断查看任意子事件。当事件识别器未能识别事件时，触摸开始子事件（和后续的触摸结束子事件）可被传递给选中视图或层次。在一个例子中，向选中视图或层次传递这样的子事件使用户界面短暂地突出显示某个对象，而不调用与该对象相关的动作。

[0070] • 延迟触摸结束标记 330、370，当对事件识别器设定延迟触摸结束标记 330、370 时，该标记使事件识别器延迟把子事件（例如，触摸结束子事件）发送给事件识别器的相应选中视图或层次，直到已判定子事件序列不对应于该事件识别器的事件种类为止。该标记可用于在较迟识别姿态的情况下，防止选中视图或层次按照触摸结束子事件行动。只要未发送触摸结束子事件，就能够向选中视图或层次发送触摸取消。如果识别出事件，那么执行应用程序的对应动作，然后向选中视图或层次传递触摸结束子事件。

[0071] • 触摸取消标记 332、372，当对事件识别器设定触摸取消标记 332、372 时，当判定子事件序列不对应于该事件识别器的事件种类时，该标记使事件识别器向事件识别器的相应选中视图或选中层次发送触摸或输入取消。发送给选中视图或选中层次的触摸或输入取

消指示在前的子事件（例如，触摸开始子事件）已被取消。触摸或输入取消可使输入源处理器的状态（参见图 4B）进入输入序列取消状态 460（下面讨论）。

[0072] 在一些实施例中，非独占事件识别器也可使用例外列表 326、366。特别地，当非独占事件识别器识别了事件时，后续子事件不被传递给与目前活动的视图相关的独占事件识别器，在识别该事件的事件识别器的例外列表 326、366 中列举的那些独占事件识别器除外。

[0073] 在一些实施例中，事件识别器可被配置成一起利用触摸取消标记和延迟触摸结束标记以防止不需要的子事件被传递给选中视图。例如，单击姿态的定义和双击姿态的前半部分相同。一旦单击事件识别器成功地识别了单击，就可能发生不希望的动作。如果设定了延迟触摸结束标记，那么在单击事件被识别之前，可防止单击事件识别器向选中视图发送子事件。另外，单击事件识别器的等待列表可识别双击事件识别器，从而防止单击事件识别器识别单击，直到双击事件识别器已进入事件不可能状态为止。等待列表的使用可避免在进行双击姿态时，执行与单击相关的动作。相反，将响应于双击事件的识别，只执行与双击相关的动作。

[0074] 下面特别说明在触摸敏感表面上的用户触摸的形式，如上所述，触摸和用户姿态可包括不必是瞬时的动作，例如，触摸可包括使手指贴着显示器移动或停留一段时间的动作。不过，触摸数据结构定义在特定时间的触摸的状态（或者更一般地，任何输入源的状态）。于是，保存在触摸数据结构中的值可随着单一触摸的进展而变化，从而使所述单一触摸在不同时刻的状态能够被传递给应用程序。

[0075] 每个触摸数据结构可包含各种字段。在一些实施例中，触摸数据结构可包括至少对应于图 3B 的触摸特有字段 339 或图 3C 的输入源特有字段 379 的数据。

[0076] 例如，图 3B 中的“对视图的初始触摸”字段 345（图 3C 中的“对层次的初始触摸”字段 385）可指示触摸数据结构是否定义对特定视图的初始触摸（因为例示了实现该视图的软件元件）。“时间戳”字段 346、386 可指示与触摸数据结构有关的特定时间。

[0077] 可选地，“信息”字段 347、387 可用于指示触摸是否是基本触摸。例如，“信息”字段 347、387 可指示触摸是否是轻扫，如果是，那么所述轻扫朝向哪个方向。轻扫是一根或多根手指沿着直线方向的快速拖动。API 实现（下面讨论）能够判定触摸是否是轻扫，并通过“信息”字段 347、387，把该信息传给应用程序，从而使应用程序减少在触摸是轻扫的情况下一直以来必需的一些数据处理。

[0078] 可选地，图 3B 中的“轻击计数”字段 348（图 3C 中的“事件计数”字段 388）可指示在初始触摸的位置，已顺序进行了多少次轻击。轻击可被定义成在特定位置，手指对触摸敏感面板的快速按下和抬起。如果在面板的相同位置快速地接连再次按下和松开手指，那么会发生多个连续轻击。事件传递系统 124 能够计数轻击，并通过“轻击计数”字段 348 把该信息传送给应用程序。在相同位置的多次轻击有时被认为是针对支持触摸的界面的有用且易于记忆的命令。从而，通过计数轻击，事件传递系统 124 能够再次从应用程序减轻一些数据处理。

[0079] “阶段”字段 349、389 可指示基于触摸的姿态目前处于的特定阶段。阶段字段 349、389 可具有各种值，诸如“触摸阶段开始”，“触摸阶段开始”指示触摸数据结构定义以前的触摸数据结构一直未涉及的新触摸。“触摸阶段移动”值可指示已定义的触摸已移离先前的位

置。“触摸阶段固定”值可指示触摸一直停留在相同的位置。“触摸阶段结束”值可指示触摸已结束（例如，用户从多触摸显示器的表面抬起他 / 她的手指）。“触摸阶段取消”值可指示触摸已被设备取消。取消的触摸可以是不一定是由用户结束的，不过设备已决定忽略的触摸。例如，设备可判定触摸是非有意生成的（即，由于把便携式支持多触摸的设备放在口袋中而生成的），从而由于该原因而忽略该触摸。“阶段”字段 349、389 的每个值可以是整数。

[0080] 从而，每个触摸数据结构可定义在特定时间，关于触摸（或者其它输入源）发生了什么事（例如，触摸是否固定不动，正在移动等等），以及与触摸相关的其它信息（诸如位置）。因而，每个触摸数据结构能够定义特定触摸在特定时刻的状态。涉及相同时间的一个或多个触摸数据结构可被添加到能够定义在某个时刻特定视图正在接收的所有触摸的状态的触摸事件数据结构中（如上所述，一些触摸数据结构也可涉及已结束，从而不再被接收的触摸）。随着时间的过去，多个触摸事件数据结构可被发送给实现视图的软件，以便向软件提供描述正在该视图中发生的触摸的连续信息。

[0081] 处理复杂的基于触摸的姿态（视情况包括多触摸姿态）的能力会增大各个软件元件的复杂性。在一些情况下，这种额外的复杂性是实现高级的所需的界面特征所必需的。例如，游戏需要处理在不同视图中发生的多个并发触摸的能力，因为游戏常常需要在相同时间按下多个按钮，或者使加速计数据和在触摸敏感表面上的触摸相结合。不过，一些较简单的应用程序和 / 或视图（及它们的相关软件元件）不需要高级界面特征。例如，利用各单一触摸，而不是多触摸功能，就可令人满意地操作简单的软按钮（即，显示在触摸敏感显示器上的按钮）。在这些情况下，基础的 OS 可能向与预定可以仅仅用各单一触摸（例如，对软按钮的单一触摸或轻击）操作的视图相关的软件元件发送不必要或者过多的触摸数据（例如，多触摸数据）。由于软件元件需要处理该数据，因此需要描述处理多个触摸的软件元件的所有复杂性的特征，即使它与仅仅涉及各单一触摸的视图相关联。这会增大设备用软件的开发成本，因为传统上在鼠标界面环境（即，各种按钮等）中易于编程的软件元件在多触摸环境中会更加复杂。

[0082] 不过应明白，关于评估和处理触摸敏感表面上的用户触摸的复杂性的上述讨论也适用于分别用输入设备 128 和 158 操作电子设备 102 和 104 的所有形式的用户输入（并非所有用户输入都始发于触摸屏上），例如，在有或没有单个或多个键盘按压或保持，触摸板上的用户移动、轻击、拖动、滚动等，铁笔输入，口头指令，检测到的眼睛运动，生物特征输入，检测到的用户生理变化，和 / 或它们的组合的情况下，协调鼠标移动和鼠标按钮按压，这些用户输入可用作为与定义要识别的事件的子事件对应的输入。

[0083] 图 4A 描述包含 4 种状态的事件识别器状态机 400。通过根据接收的子事件，管理事件识别器状态机 400 中的状态转变，事件识别器有效地表述事件定义。例如，可用一系列的两个子事件，或者视情况，三个子事件有效地定义轻击姿态。首先，应检测触摸，这将是子事件 1。例如，触摸子事件可以是用户的手指触摸在包括具有状态机 400 的事件识别器的视图中的触摸敏感表面。其次，其间触摸基本上未沿任何方向移动（例如，触摸位置的任何移动都小于预先定义的阈值，所述阈值可被测定为距离（例如，5mm），或者测定为显示器上的像素数（例如，5 个像素）），并且延迟足够短的可选测定延迟可充当子事件 2。最后，触摸的终止（例如，用户手指从触摸敏感表面的抬起）将充当子事件 3。通过把事件识别器状态机

400 编码成根据收到这些子事件，在各种状态之间转变，事件识别器状态机 400 有效地表述轻击姿态事件定义。

[0084] 与事件种类无关，事件识别器状态机 400 从事件识别开始状态 405 开始，并且取决于收到什么子事件，可前进到任意的剩余状态。为了使事件识别器状态机 400 的讨论更容易，将讨论从事件识别开始状态 405 到事件被识别状态 415，事件可能状态 410 和事件不可能状态 420 的直接路径，之后说明从事件可能状态 410 引出的路径。

[0085] 从事件识别开始状态 405 开始，如果收到自身包括对于事件的事件定义的子事件，那么事件识别器状态机 400 将转变成事件被识别状态 415。

[0086] 从事件识别开始状态 405 开始，如果收到不是事件定义中的第一个子事件的子事件，那么事件识别器状态机 400 将转变成事件不可能状态 420。

[0087] 从事件识别开始状态 405 开始，如果收到为给定事件定义中的第一子事件，而不是最后一个子事件的子事件，那么事件识别器状态机 400 将转变成事件可能状态 410。如果收到的下一个子事件是给定事件定义中的第二子事件，但不是最后一个子事件，那么事件识别器状态机 400 将保持事件可能状态 410。只要收到的子事件序列依旧是该事件定义的一部分，事件识别器状态机 400 就可保持事件可能状态 410。如果在任何时候，事件识别器状态机 400 处于事件可能状态 410，而事件识别器状态机 400 收到不是该事件定义的一部分的子事件，那么事件识别器状态机 400 将转变成事件不可能状态 420，从而判定当前事件（如果有的话）不是与该事件识别器（即，对应于状态 400 的事件识别器）对应的那种事件。另一方面，如果事件识别器状态机 400 处于事件可能状态 410，并且事件识别器状态机 400 收到事件定义中的最后一个子事件，那么事件识别器状态机 400 将转变成事件被识别状态 415，从而完成成功的事件识别。

[0088] 图 4B 描述输入源处理进程 440 的一个实施例，输入源处理进程 440 具有表示视图如何接收关于相应输入的信息的有限状态机。注意当在设备的触摸敏感表面上存在多个触摸时，每个触摸是具有它自己的有限状态机的独立输入源。在本实施例中，输入源处理进程 440 包括四种状态 :输入序列开始状态 445，输入序列继续状态 450，输入序列结束状态 455 和输入序列取消状态 460。输入源处理进程 440 可被相应的事件识别器使用，例如，当输入将被传递给应用程序时，但是仅仅在检测到输入序列完成之后。输入源处理进程 440 可以和应用程序一起使用，该应用程序不能取消或撤消响应于传递给该应用程序的输入序列而产生的变化。

[0089] 从输入序列开始状态 445 开始，如果收到独自完成输入序列的输入，那么输入源处理进程 440 将转变成输入序列结束状态 455。

[0090] 从输入序列开始状态 445 开始，如果收到指示输入序列被终止的输入，那么输入源处理进程 440 将转变成输入序列取消状态 460。

[0091] 从输入序列开始状态 445 开始，如果收到作为输入序列中的第一输入，而不是最后一个输入的输入，那么输入源处理进程 440 将转变成输入序列继续状态 450。如果收到的下一个输入是输入序列中的第二输入，那么输入处理进程 440 将保持输入序列继续状态 450。只要正在传递的子事件序列依旧是给定输入序列的一部分，输入源处理进程 440 就可保持在输入序列继续状态 450。如果在任何时候，输入源处理进程 440 处于输入序列继续状态 450，而输入源处理进程 440 收到不是该输入序列的一部分的输入，那么输入源处理进程

440 将转变成输入序列取消状态 460。另一方面，如果输入源处理进程 440 处于输入序列继续状态 450，并且输入源处理进程 440 收到给定输入定义中的最后一个输入，那么输入源处理进程 440 将转变成输入序列结束状态 455，从而成功地接收一组子事件。

[0092] 在一些实施例中，可为特定视图或编程层次实现输入源处理进程 440。在这种情况下，某些子事件序列会导致转变到输入取消状态 460。

[0093] 例如，考虑图 4C，图 4C 设想仅仅由主动涉及的视图输入源处理器 480（下面称为“视图 480”）表示的主动涉及视图。视图 480 包括垂直轻扫事件识别器，其仅仅由作为其事件识别器之一的垂直轻扫事件识别器 468（下面称为“识别器 468）”表示。在这种情况下，作为其定义的一部分，识别器 468 要求检测 :1) 手指向下 465-1 ;2) 可选的短延迟 465-2 ;3) 至少 N 个像素的垂直轻扫 465-3 ;和 4) 手指抬起 465-4。

[0094] 对本例来说，识别器 468 还使其延迟触摸开始标记 328 和触摸取消标记 332 被设定。现在考虑传递给识别器 468 以及视图 480 的下述子事件序列 :

[0095] • 子事件序列 465-1 :检测手指向下，它对应于识别器 468 的事件定义

[0096] • 子事件序列 465-2 :测量延迟，它对应于识别器 468 的事件定义

[0097] •子事件序列 465-3 :手指进行与垂直滚动相容，但是小于 N 个像素，因而不对应于识别器 468 的事件定义的垂直轻扫移动

[0098] • 子事件序列 465-4 :检测手指抬起，它对应于识别器 468 的事件定义

[0099] 这里，识别器 468 会成功地把子事件 1 和 2 识别成其事件定义的一部分，因而紧接在子事件 3 的传递之前，会处于事件可能状态 472。由于识别器 468 使其延迟触摸开始标记 328 被设定，因此，初始的触摸子事件不被发送给选中视图。相应地，在子事件 3 的传递之前，视图 480 的输入源处理进程 440 会仍然处于输入序列开始状态。

[0100] 一旦完成向识别器 468 传递子事件 3，识别器 468 的状态就转变成事件不可能状态 476，重要的是，识别器 468 现在已判定该子事件序列不对应于它特有的垂直轻扫姿态事件种类（即，它已确定该事件不是垂直轻扫。换句话说，在本例中，不会发生作为垂直轻扫的识别 474）。视图输入源处理器 480 的输入源处理系统 440 也将更新其状态。在一些实施例中，当事件识别器发送指示它已开始识别事件的状态信息时，视图输入源处理器 480 的状态会从输入序列开始状态 482 进入输入序列继续状态 484。当在没有事件被识别出的情况下，结束触摸或输入时，视图输入源处理器 480 进入输入序列取消状态 488，这是因为已设定了事件识别器的触摸取消标记 322。另一方面，如果事件识别器的触摸取消标记 322 未被设定，那么当触摸或输入结束时，视图输入源处理器 480 进入输入序列结束状态 486。

[0101] 由于事件识别器 468 的触摸取消标记 322 被设定，因此当事件识别器 468 转变成事件不可能状态 476 时，该识别器将向与该事件识别器对应的选中视图发送触摸取消子事件或消息。结果，视图输入源处理器 480 将转变成输入序列取消状态 488。

[0102] 在一些实施例中，子事件 465-4 的传递并不与识别器 468 做出的任何事件识别决定关系密切，尽管视图输入源处理器 480 的其它事件识别器（如果有的话）可继续分析子事件序列。

[0103] 下表以汇总表格的格式，给出了与上面说明的事件识别器 468 的状态以及视图输入源处理器 480 的状态相关的示例性子事件序列 465 的处理。在本例中，由于识别器 468 的触摸取消标记 332 被设定，因此视图输入源处理器 480 的状态从输入序列开始状态 445 进

入输入序列取消状态 488 ：

[0104]

子事件序列 465	状态 :识别器 468	状态 :视图 480
传递开始之前	事件识别开始 470	
检测手指向下 465-1	事件可能 472	输入序列开始 482
测量延迟 46-2	事件可能 472	输入序列继续 484
检测手指垂直轻扫 465-3	事件不可能 476	输入序列继续 484
检测手指抬起 465-4	事件不可能 476	输入序列取消 488

[0105] 参见图 5A，仔细考虑正由包括多个事件识别器的视图接收的子事件序列 520 的例子。对本例来说，在图 5A 中描述了两个事件识别器，滚动事件识别器 580 和轻击事件识别器 590。为了便于举例说明，使图 3A 中的视图搜索结果面板 304 与子事件序列 520 的接收，及滚动事件识别器 580 和轻击事件识别器 590 中的状态转变相联系。注意在本例中，子事件序列 520 定义在触摸敏感显示器或触控板上的轻击手指姿态，不过，相同的事件识别技术可以应用在无数的上下文（例如，检测鼠标按钮按下）中，和 / 或应用在利用各编程层次的编程分层结构的实施例中。

[0106] 在第一子事件被传递给视图搜索结果面板 304 之前，事件识别器 580 和 590 分别处于事件识别开始状态 582 和 592。在触摸 301 之后（触摸 301 作为检测手指向下子事件 521-1 被传递给视图搜索结果面板 304 的主动涉及事件识别器，作为触摸子事件 301-2（以及被传递给地图视图 305 的主动涉及事件识别器，作为触摸子事件 301-3），滚动事件识别器 580 转变成事件可能状态 584，类似地，轻击事件识别器 590 转变成事件可能状态 594。这是因为轻击和滚动的事件定义都从触摸敏感表面上的触摸开始，诸如检测手指向下。

[0107] 轻击和滚动姿态的一些定义可视情况包括初始触摸和事件定义中的任何下一步之间的延迟。在这里讨论的所有例子中，轻击和滚动姿态的示例性事件定义识别在第一触摸子事件（检测手指向下）之后的延迟子事件。

[0108] 因而，当测量延迟子事件 521-2 被传递给事件识别器 580 和 590 时，事件识别器 580 和 590 都分别保持事件可能状态 584 和 594。

[0109] 最后，检测手指抬起子事件 521-3 被传递给事件识别器 580 和 590。在这种情况下，因为关于轻击和滚动的事件定义不同，所以事件识别器 580 和 590 的状态转变不同。就滚动事件识别器 580 来说，保持在事件可能状态的下一个子事件应是检测移动。不过，由于传递的子事件是检测手指抬起 521-3，因此滚动事件识别器 580 转变成事件不可能状态 588。不过，轻击事件定义结束于手指抬起子事件。因而，在传递了检测手指抬起子事件 521-3 之后，轻击事件识别器 590 转变成事件识别状态 596。

[0110] 注意在一些实施例中，如上关于图 4B 和 4C 所述，在图 4B 中讨论的输入源处理进程 440 可在视图层次用于各种目的。下表以汇总表格的格式给出了与事件识别器 580、590 和输入源处理进程 440 相关的子事件序列 520 的传递 ：

[0111]

子事件序列 520	状态：滚动事件识别器 580	状态：轻击事件识别器 590	状态：输入源处理进程 440
传递开始之前	事件识别开始 582	事件识别开始 592	
检测手指向下 521-1	事件可能 584	事件可能 594	输入序列开始 445
测量延迟 521-2	事件可能 584	事件可能 594	输入序列继续 450
检测手指抬起 521-3	事件不可能 588	事件被识别 596	输入序列结束 455

[0112] 参见图 5B，仔细考虑正由包括多个事件识别器的视图接收的子事件序列 530 的另一个例子。对本例来说，在图 5B 中描述了两个事件识别器，滚动事件识别器 580 和轻击事件识别器 590。为了便于举例说明，使图 3A 中的视图搜索结果面板 304 与子事件序列 530 的接收，以及滚动事件识别器 580 和轻击事件识别器 590 中的状态转变相联系。注意在本例中，子事件序列 530 定义在触摸敏感显示器上的滚动手指姿态，不过，相同的事件识别技术可以应用在无数的上下文（例如，检测鼠标按钮按下，鼠标移动和鼠标按钮释放）中，和 / 或应用在利用各编程层次的编程分层结构的实施例中。

[0113] 在第一子事件被传递给视图搜索结果面板 304 的主动涉及事件识别器之前，事件识别器 580 和 590 分别处于事件识别开始状态 582 和 592。在对应于触摸 301 的子事件的传递之后（如上所述），滚动事件识别器 580 转变成事件可能状态 584，类似地，轻击事件识别器 590 转变成事件可能状态 594。

[0114] 当测量延迟子事件 531-2 被传递给事件识别器 580 和 590 时，事件识别器 580 和 590 都分别转变成事件可能状态 584 和 594。

[0115] 之后，检测手指移动子事件 531-3 被传递给事件识别器 580 和 590。在这种情况下，因为关于轻击和滚动的事件定义不同，所以事件识别器 580 和 590 的状态转变不同。就滚动事件识别器 580 来说，保持在事件可能状态的下一个子事件是检测移动，从而当滚动事件识别器 580 收到检测手指移动子事件 531-3 时，它保持在事件可能状态 584。不过，如上所述，轻击的定义结束于手指抬起子事件，从而轻击事件识别器 590 转变成事件不可能状态 598。

[0116] 最后，检测手指抬起子事件 531-4 被传递给事件识别器 580 和 590。轻击事件识别器已处于事件不可能状态 598，从而不发生任何状态转变。滚动事件识别器 580 的事件定义结束于检测到手指抬起。由于传递的子事件是检测手指抬起子事件 531-4，因此滚动事件识别器 580 转变成事件被识别状态 586。注意，触摸敏感表面上的手指移动会产生多个移动子事件，于是，在抬起之前，并且持续到抬起为止，可识别滚动。

[0117] 下表以汇总表格的格式，给出了与事件识别器 580、590 和输入源处理进程 440 相关的子事件序列 530 的传递：

[0118]

子事件序列 530	状态：滚动事件识别器 580	状态：轻击事件识别器 590	状态：输入源处理进程 440
传递开始之前	事件识别开始 582	事件识别开始 592	
检测手指向下 531-1	事件可能 584	事件可能 594	输入序列开始 445
测量延迟 531-2	事件可能 584	事件可能 594	输入序列继续 450
检测手指移动 531-3	事件可能 584	事件不可能 598	输入序列继续 450
检测手指抬起 531-4	事件被识别 586	事件不可能 598	输入序列结束 455

[0119] 参见图 5C，仔细考虑正由包括多个事件识别器的视图接收的子事件序列 540 的另一个例子。对本例来说，在图 5C 中描述了两个事件识别器，双击事件识别器 570 和轻击事件识别器 590。为了便于举例说明，使图 3A 中的地图视图 305 与子事件序列 540 的接收，及双击事件识别器 570 和轻击事件识别器 590 中的状态转变相联系。注意在本例中，子事件序列 540 定义在触摸敏感显示器上的双击姿态，不过，相同的事件识别技术可以应用在无数的上下文（例如，检测鼠标双击）中，和 / 或应用在利用各编程层次的编程分层结构的实施例中。

[0120] 在第一子事件被传递给地图视图 305 的主动涉及事件识别器之前，事件识别器 570 和 590 分别处于事件识别开始状态 572 和 592。在与触摸子事件 301 相关的子事件被传递给地图视图 304 之后（如上所述），双击事件识别器 570 和轻击事件识别器 590 分别转变成事件可能状态 574 和 594。这是因为轻击和双击的事件定义都从触摸敏感表面上的诸如检测手指向下之类的触摸开始。

[0121] 当测量延迟子事件 541-2 被传递给事件识别器 570 和 590 之后，事件识别器 570 和 590 都分别保持在事件可能状态 574 和 594。

[0122] 之后，检测手指抬起子事件 541-3 被传递给事件识别器 570 和 590。在这种情况下，因为关于轻击和双击的示例性事件定义不同，所以事件识别器 570 和 590 的状态转变不同。就轻击事件识别器 590 来说，事件定义中的最终子事件是检测手指抬起，从而轻击事件识别器 590 转变成事件被识别状态 596。

[0123] 不过，由于延迟已开始，因此双击识别器 570 保持在事件可能状态 574，而不管用户最终会做什么。可是，双击的完整事件识别定义需要后面是完整的轻击子事件序列的另一个延迟。这造成像已处于事件被识别状态 576 的轻击事件识别器 590，和仍然处于事件可能状态 574 的双击识别器 570 之间的含糊情形。

[0124] 因而，在一些实施例中，事件识别器可以实现如上关于图 3B 和 3C 讨论的独占性标记和独占性例外列表。这里，轻击事件识别器 590 的独占性标记 324 会被设定，另外，轻击事件识别器 590 的独占性例外列表 326 会被配置成在轻击事件识别器 590 进入事件被识别

状态 596 之后，继续允许把子事件传递给某些事件识别器（诸如双击事件识别器 570）。

[0125] 在轻击事件识别器 590 保持在事件被识别状态 596 的同时，子事件序列 540 继续被传递给双击事件识别器 570，这种情况下，测量延迟子事件 541-1 检测手指向下 541-5 和测量延迟 541-6，使双击事件识别器 570 保持在事件可能状态 574；序列 540 的最后子事件的传递，检测手指抬起 541-7 使双击事件识别器 570 转变成事件被识别状态 576。

[0126] 此时，地图视图 305 采用事件识别器 570 识别的双击事件，而不是轻击事件识别器 590 识别的单击事件。采用双击事件的决定是综合考虑到轻击事件识别器 590 的独占性标记 324 被设定，轻击事件识别器 590 的包括双击事件的独占性例外列表 326，及轻击事件识别器 590 和双击事件识别器 570 都成功地识别它们各自的事件种类的事实做出的。

[0127] 下表以汇总表格的格式，给出了与事件识别器 570、590 和子事件处理进程 440 相关的子事件序列 540 的传递：

[0128]

子事件序列 540	状态：双击事件识别器 570	状态：轻击事件识别器 590	状态：输入源处理进程 440
传递开始之前	事件识别开始 572	事件识别开始 592	
检测手指向下 541-1	事件可能 574	事件可能 594	输入序列开始 445
测量延迟 541-2	事件可能 574	事件可能 594	输入序列继续 450
检测手指抬起 541-3	事件可能 574	事件被识别 596	输入序列继续 450
测量延迟 541-4	事件可能 574	事件被识别 596	输入序列继续 450
检测手指向下 541-5	事件可能 574	事件被识别 596	输入序列继续 450
测量延迟 541-6	事件可能 574	事件被识别 596	输入序列继续 450
检测手指抬起 541-7	事件被识别 574	事件被识别 596	输入序列结束 455

[0129] 在另一个实施例中，在图 5C 的事件情形下，单击姿态不被识别，因为单击事件识别器具有识别双击事件识别器的等待列表。结果，在双击事件识别器进入事件不可能状态之前（如果发生过的话），单击姿态不被识别。在其中双击姿态被识别的这个例子中，单击事件识别器会保持在事件可能状态，直到识别出双击姿态为止，此时，单击事件识别器会转变成事件不可能状态。

[0130] 现在仔细考虑图 6A 和 6B，图 6A 和 6B 是图解说明按照一些实施例的事件识别方法的流程图。在电子设备处执行方法 600，在一些实施例中，该电子设备可以是电子设备 102 或 104，如上所述。在一些实施例中，电子设备可包括被配置成检测多触摸姿态的触摸敏感

表面。另一方面，电子设备可包括被配置成检测多触摸姿态的触摸屏。

[0131] 方法 600 被配置成执行包括具有多个视图的视图分层结构的软件。方法 600 显示 608 视图分层结构的一个或多个视图，执行 610 一个或多个软件元件。每个软件元件与特定视图关联，每个特定视图包括一个或多个事件识别器，诸如在图 3B 和 3C 中分别作为事件识别器结构 320 和 360 说明的那些事件识别器。

[0132] 每个事件识别器通常包括基于一个或多个子事件的事件定义，其中事件定义可被实现成状态机，例如参见图 3B 的状态机 340。事件识别器通常还包括事件处理器，事件处理器规定关于目标的动作，并被配置成响应于事件识别器检测到与事件定义对应的事件，把动作发送给目标。

[0133] 在一些实施例中，多个事件识别器中的至少一个是具有姿态定义和姿态处理器的姿态识别器，如在图 6A 的步骤 612 中所示。

[0134] 在一些实施例中，事件定义定义用户姿态，如在图 6A 的步骤 614 中所示。

[0135] 另一方面，事件识别器具有一组事件识别状态 616。这些事件识别状态至少包括事件可能状态，事件不可能状态和事件被识别状态。

[0136] 在一些实施例中，如果事件识别器进入事件可能状态，那么事件处理器开始其对应动作的准备 618，以便传递给目标。如上关于图 4A 和图 5A-5C 中的例子所述，为每个事件识别器实现的状态机通常包括初始状态，例如，事件识别开始状态 405。收到构成事件定义的初始部分的子事件会触发到事件可能状态 410 的状态改变。因而，在一些实施例中，当事件识别器从事件识别开始状态 405 转变成事件可能状态 410 时，事件识别器的事件处理器开始准备其特定动作，以便在成功识别事件之后，传递给事件识别器的目标。

[0137] 另一方面，在一些实施例中，如果事件识别器进入事件不可能状态 620，那么事件处理器可以终止其对应对作的准备 620。在一些实施例中，终止对应动作包括取消事件处理器的对应动作的任何准备。

[0138] 图 5B 的例子提供了这种实施例的信息，因为轻击事件识别器 590 可能已开始其动作的准备 618，不过随后，一旦检测手指移动子事件 531-3 被传递给轻击事件识别器 590，识别器 590 就转变成事件不可能状态 598、578。此时，轻击事件识别器 590 会终止它已开始为其做准备 618 的动作的准备 620。

[0139] 在一些实施例中，如果事件识别器进入事件被识别状态，那么事件处理器完成其对应动作的准备 622，以便传递给目标。图 5C 的例子图解说明了这种实施例，因为地图视图 305 的主动涉及事件识别器识别出双击，在一些实现中，所述双击会是开始选择和 / 或执行地图视图 305 描述的搜索结果的事件。这里，在双击事件识别器 570 成功识别由子事件序列 540 构成的双击事件之后，地图视图 305 的事件处理器完成其动作的准备 622，即，指示它已收到激活命令。

[0140] 在一些实施例中，事件处理器把其对应动作传递 624 给与事件识别器相关的目标。继续图 5C 的例子，准备的动作，即，地图视图 305 的激活命令会被传递给与地图视图 305 相关的特定目标，所述目标可以是任何适当的编程方法或对象。

[0141] 另一方面，多个事件识别器可以并行地独立处理 626 一个或多个子事件的序列。

[0142] 在一些实施例中，一个或多个事件识别器可被配置成独占事件识别器 628，如上分别关于图 3B 和 3C 的独占性标记 324 和 364 所述。当事件识别器被配置成独占事件识别器

时，在独占事件识别器识别了事件之后，事件传递系统阻止视图分层结构中的主动涉及视图的任何其它事件识别器（在识别该事件的事件识别器的例外列表 326、366 中列举的那些事件识别器除外）接收（相同子事件序列的）后续子事件。此外，当非独占事件识别器识别了事件时，事件传递系统阻止视图分层结构中的主动涉及视图的任何独占事件识别器接收后续子事件，除识别该事件的事件识别器的例外列表 326、366 中列举的那些事件识别器之外（如果有的话）。

[0143] 在一些实施例中，独占事件识别器可包括 630 事件例外列表，如上分别关于图 3B 和 3C 的独占性例外列表 326 和 366 所述。如上在图 5C 的讨论中所述，事件识别器的独占性例外列表可用于允许事件识别器继续事件识别，即使当构成它们各自的事件定义的子事件序列重叠时也是如此。因而，在一些实施例中，事件例外列表包括其对应的事件定义具有重复子事件的事件 632，诸如图 5C 的单击 / 双击事件例子。

[0144] 另一方面，事件定义可定义用户输入操作 634。

[0145] 在一些实施例中，一个或多个事件识别器可适合于把子事件序列的每个子事件的传递推迟到事件被识别之后。

[0146] 方法 600 检测 636 一个或多个子事件的序列，在一些实施例中，一个或多个子事件的序列可包括原始的触摸事件 638。原始的触摸事件可包括但不限于触摸敏感表面上的基于触摸的姿态的基本组成，诸如与初始手指或铁笔向下触摸相关的数据，与在触摸敏感表面内开始移动多根手指或铁笔相关的数据，两根手指沿相反方向的移动，从触摸敏感表面提起铁笔，等等。

[0147] 一个或多个子事件的序列中的子事件可包括多种形式，包括但不限于按键按下，按键按下并保持，按键按下释放，按钮按下，按钮按下并保持，按钮按下释放，操纵杆移动，鼠标移动，鼠标按钮按下，鼠标按钮释放，铁笔触摸，铁笔移动，铁笔释放，口头指令，检测的眼睛运动，生物特征输入，和检测的用户生理变化，等等。

[0148] 方法 600 识别 640 视图分层结构的视图之一作为选中视图。选中视图确定视图分层结构中的哪些视图是主动涉及视图。图 3A 中描述了一个例子，其中主动涉及视图 306 包括搜索结果面板 304 和地图视图 305，因为触摸子事件 301 接触与地图视图 305 相关的区域。

[0149] 在一些实施例中，视图分层结构内的第一个主动涉及视图可被配置成 642 阻止向与所述第一个主动涉及视图相关的事件识别器传递相应子事件。这种行为能够实现上面关于图 3B 和 3C 讨论的跳过属性（分别为 330 和 370）。当为事件识别器设定了跳过属性时，对于与视图分层结构中的其它主动涉及视图相关的事件识别器，仍然进行相应子事件的传递。

[0150] 另一方面，视图分层结构内的第一个主动涉及视图可被配置成 644 阻止向与所述第一个主动涉及视图相关的事件识别器传递相应子事件，除非所述第一个主动涉及视图是选中视图。这种行为能够实现上面关于图 3B 和 3C 讨论的有条件跳过属性（分别为 332 和 372）。

[0151] 在一些实施例中，视图分层结构内的第二个主动涉及视图被配置成 646 阻止向与第二个主动涉及视图相关的事件识别器和向与第二个主动涉及视图的先辈相关的事件识别器传递相应子事件。这种行为能够实现上面关于图 3B 和 3C 讨论的停止属性（分别为

328 和 368)。

[0152] 方法 600 把相应子事件传递给 648 视图分层结构内的每个主动涉及视图的事件识别器。在一些实施例中，视图分层结构中的主动涉及视图的事件识别器处理相应子事件，之后处理子事件序列中的下一个子事件。另一方面，视图分层结构内的主动涉及视图的事件识别器在处理相应子事件的同时，做出它们的子事件识别判定。

[0153] 在一些实施例中，视图分层结构中的主动涉及视图的事件识别器可同时处理一个或多个子事件的序列 650 ；另一方面，视图分层结构中的主动涉及视图的事件识别器可并行处理一个或多个子事件的序列。

[0154] 在一些实施例中，一个或多个事件识别器可适合于推迟传递 652 子事件序列的一个或多个子事件，直到在所述事件识别器识别了该事件之后为止。这种行为反映延迟的事件。例如，考虑在对其来说可能存在多种轻击姿态的视图中的单击姿态。在这种情况下，轻击事件变成“轻击 + 延迟”识别器。本质上，当事件识别器实现这种行为时，该事件识别器将延迟事件识别，直到肯定子事件的序列确实对应于其事件定义为止。当接受方视图不能恰当地对取消事件作出响应时，这种行为是适当的。在一些实施例中，事件识别器将延迟更新相对于其相应的主动涉及视图的其事件识别状态，直到事件识别器确定子事件的序列不对应于其事件定义为止。如上关于图 3B 和 3C 所述，设置了延迟触摸开始标记 328、368，延迟触摸结束标记 330、370，和触摸取消标记 332、372，以使子事件传递技术以及事件识别器和视图状态信息更新适应特定需要。

[0155] 为了便于解释，参考具体实施例说明了上述描述。但是，上面的说明性讨论并非意图是详尽的，也并非意图把本发明局限于公开的具体形式。鉴于上面的教导，许多修改和变化都是可能的。为了最佳地解释本发明的原理及其实际应用，选择和说明了上述实施例，以使本领域的技术人员能够最好地利用本发明，以及具有适合于预期的特殊用途的各种修改的各个实施例。

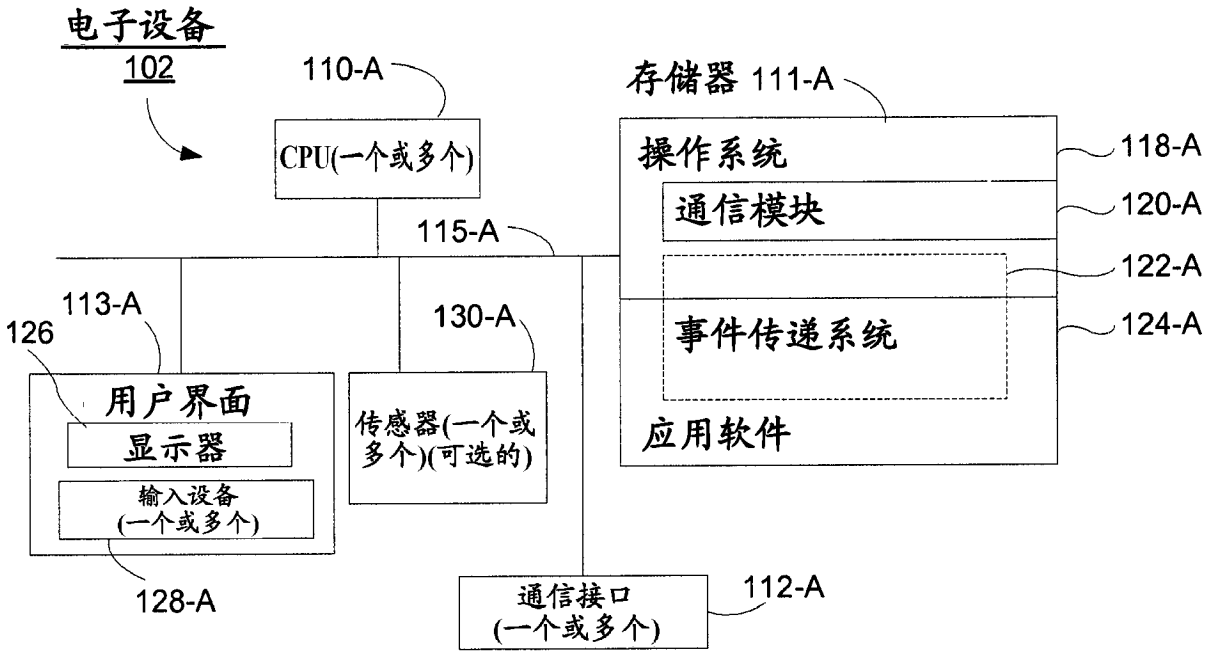

电子设备
102
110-A
CPU(一个或多个)
存储器 111-A
操作系统
118-A
通信模块
120-A
122-A
事件传递系统
124-A
应用软件
115-A
113-A
126
130-A
用户界面
显示器
输入设备
(一个或多个)
传感器(一个或多个)(可选的)
128-A
通信接口
(一个或多个)
112-A

图 1A

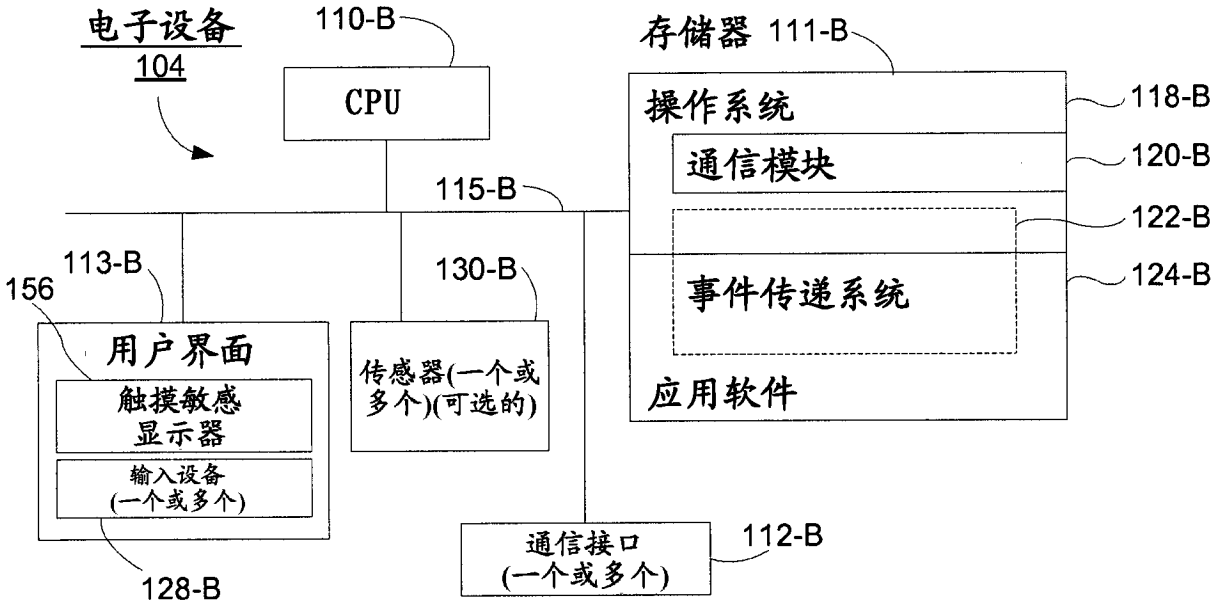

电子设备
104
110-B
CPU
存储器 111-B
操作系统
118-B
通信模块
120-B
122-B
事件传递系统
124-B
应用软件
115-B
113-B
156
130-B
用户界面
触摸敏感
显示器
输入设备
(一个或多个)
传感器(一个或多个)(可选的)
128-B
通信接口
(一个或多个)
112-B

图 1B

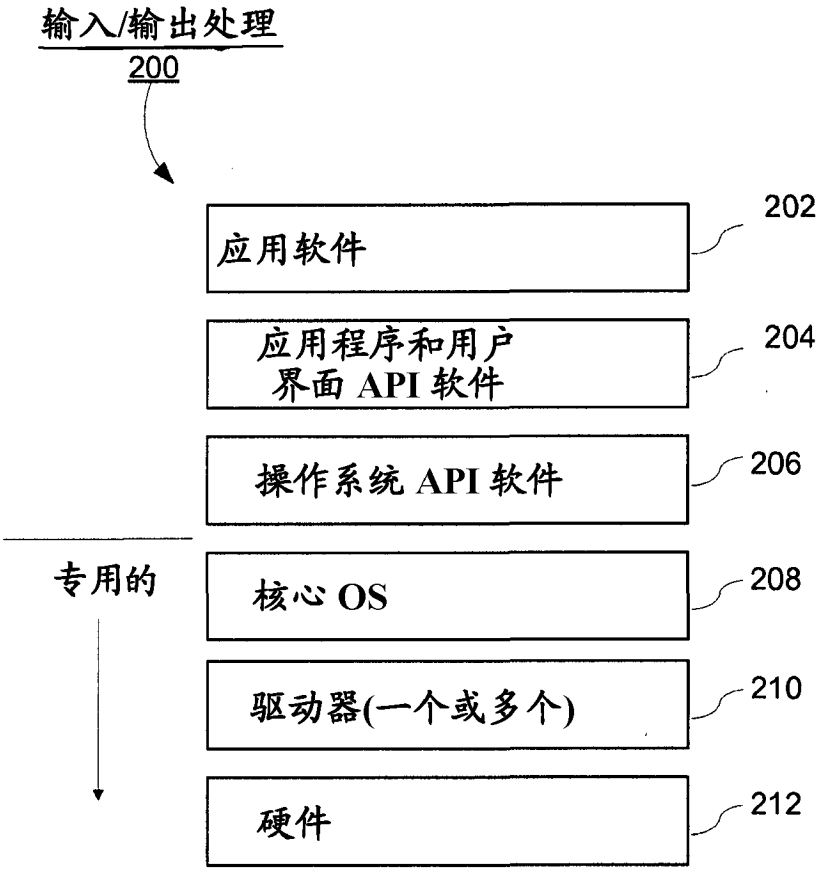

输入/输出处理
200
应用软件
202
应用程序和用户
界面 API 软件
204
操作系统 API 软件
206
专用的
核心 OS
208
驱动器(一个或多个)
210
硬件
212

图 2

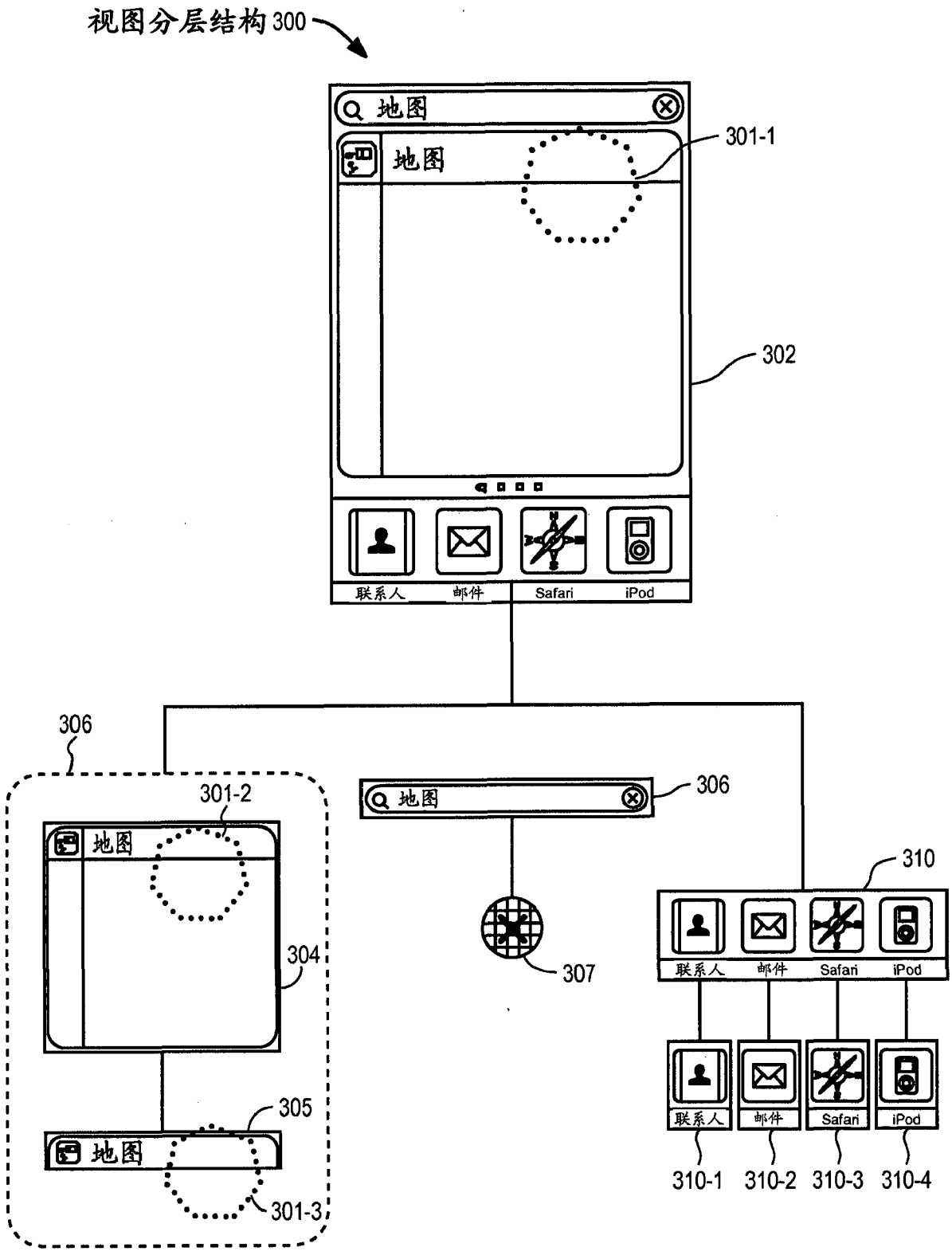

视图分层结构 300
地图
地图
301-1
302
联系人
邮件
Safari
iPod
306
301-2
地图
304
地图
305
301-3
306
307
310
联系人
邮件
Safari
iPod
联系人
邮件
Safari
iPod
310-1
310-2
310-3
310-4

图 3A

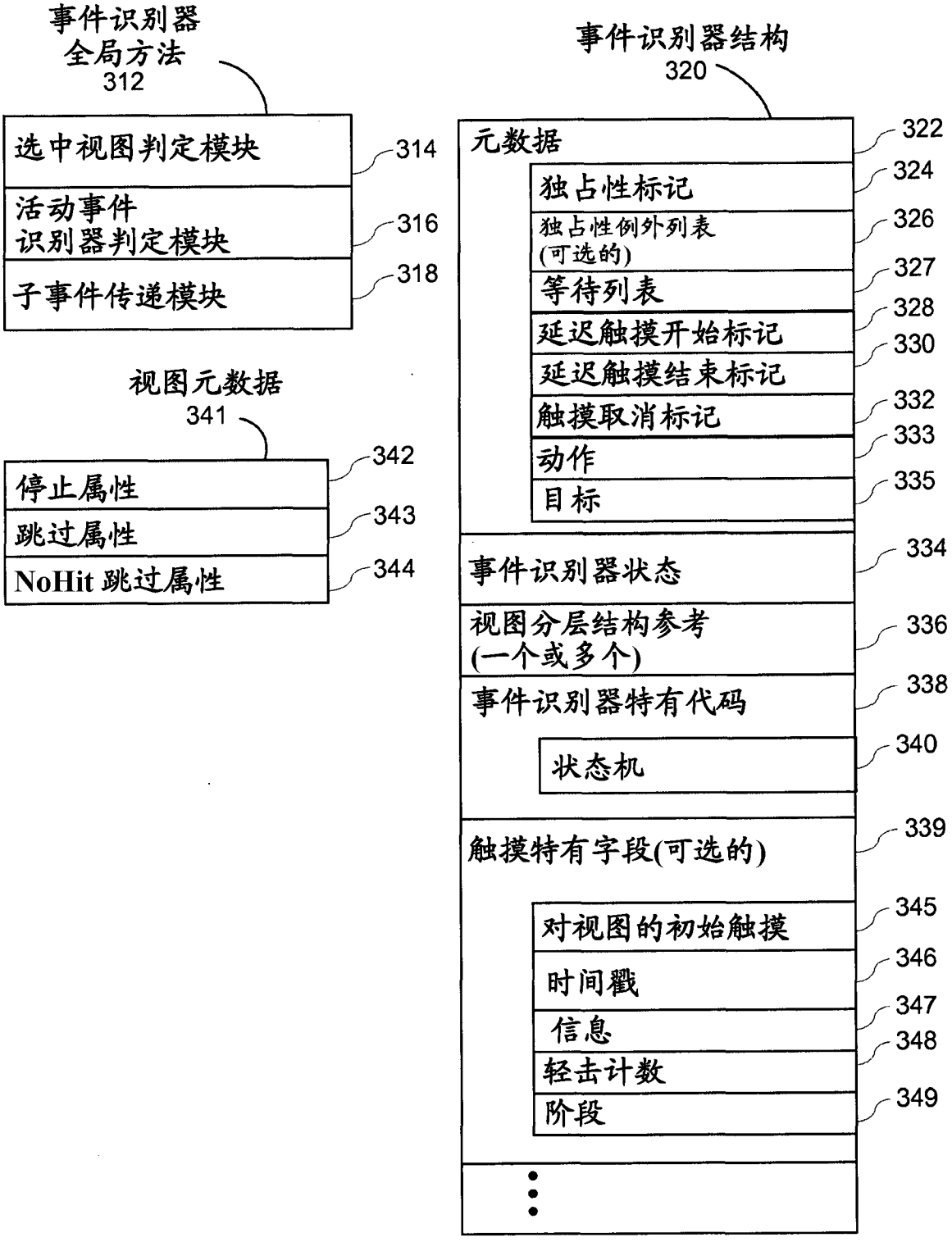

事件识别器
全局方法
312
选中视图判定模块
314
活动事件
识别器判定模块
316
子事件传递模块
318
视图元数据
341
停止属性
342
跳过属性
343
NoHit 跳过属性
344
事件识别器结构
320
元数据
322
独占性标记
324
独占性例外列表
(可选的)
326
等待列表
327
延迟触摸开始标记
328
延迟触摸结束标记
330
触摸取消标记
332
动作
333
目标
335
事件识别器状态
334
视图分层结构参考
(一个或多个)
336
事件识别器特有代码
338
状态机
340
触摸特有字段(可选的)
339
对视图的初始触摸
345
时间戳
346
信息
347
轻击计数
348
阶段
349

图 3B

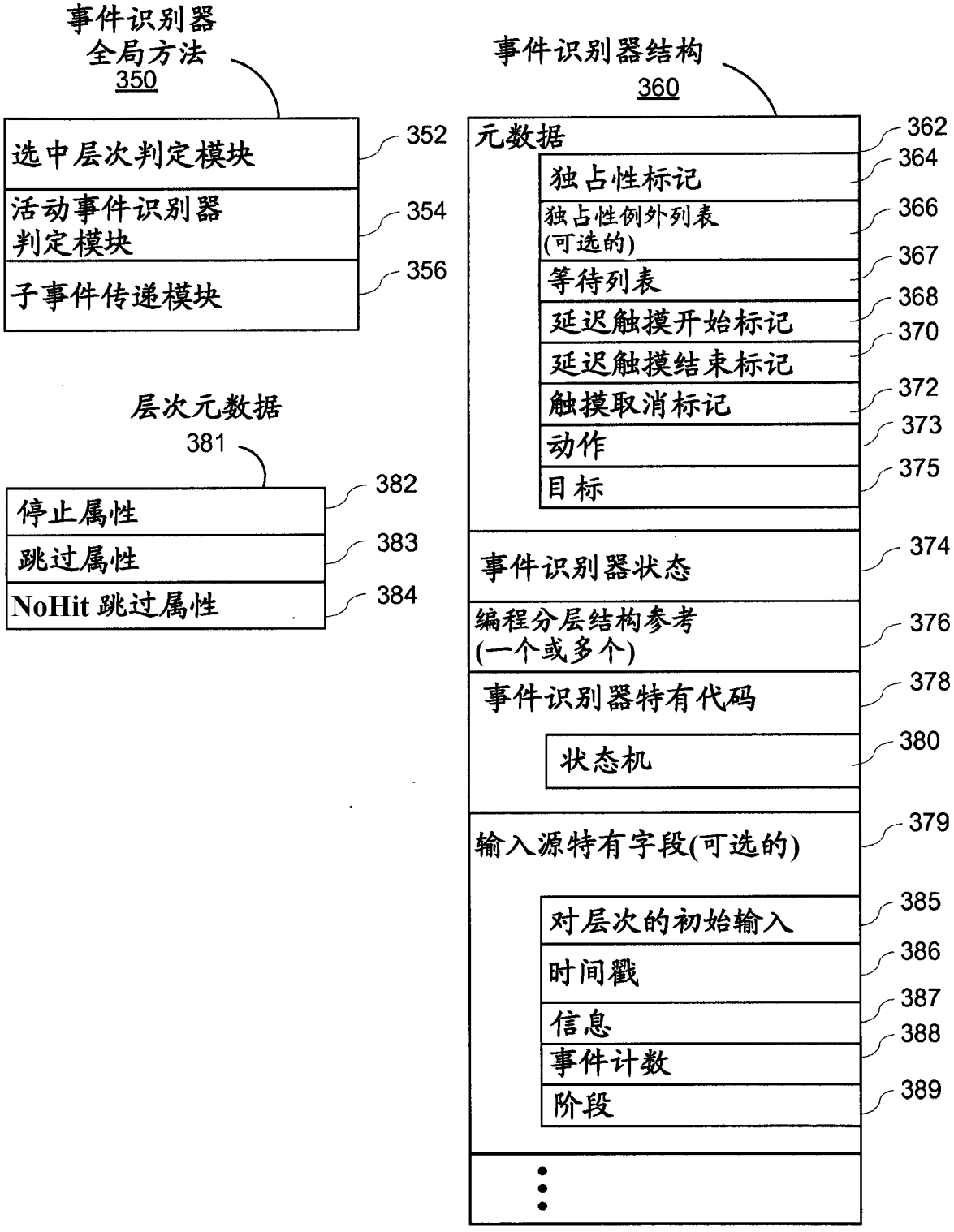

事件识别器
全局方法
350
选中层次判定模块
352
活动事件识别器
判定模块
354
子事件传递模块
356
层次元数据
381
停止属性
382
跳过属性
383
NoHit 跳过属性
384
事件识别器结构
360
元数据
362
独占性标记
364
独占性例外列表
(可选的)
366
等待列表
367
延迟触摸开始标记
368
延迟触摸结束标记
370
触摸取消标记
372
动作
373
目标
375
事件识别器状态
374
编程分层结构参考
(一个或多个)
376
事件识别器特有代码
378
状态机
380
输入源特有字段(可选的)
379
对层次的初始输入
385
时间戳
386
信息
387
事件计数
388
阶段
389

图 3C

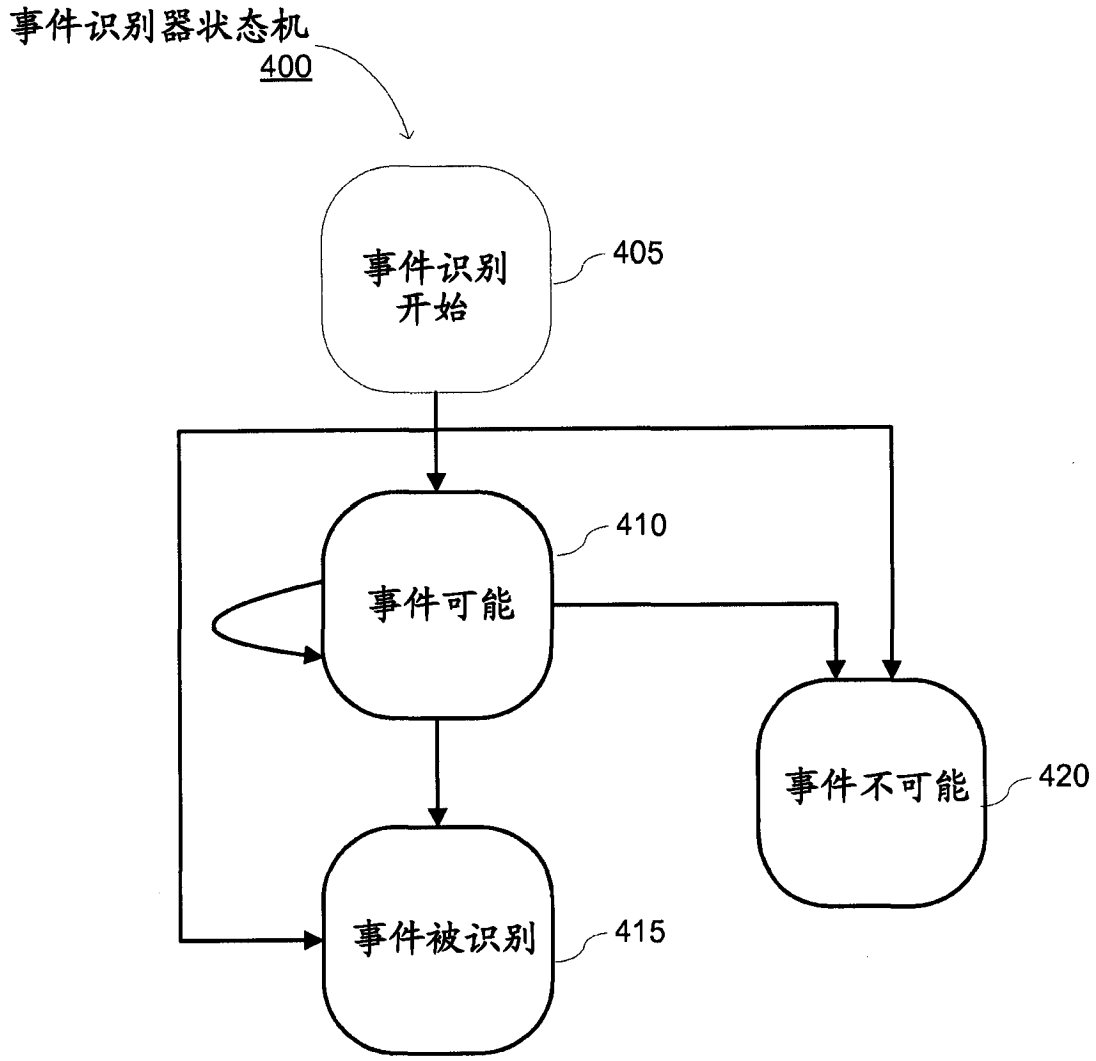


图 4A

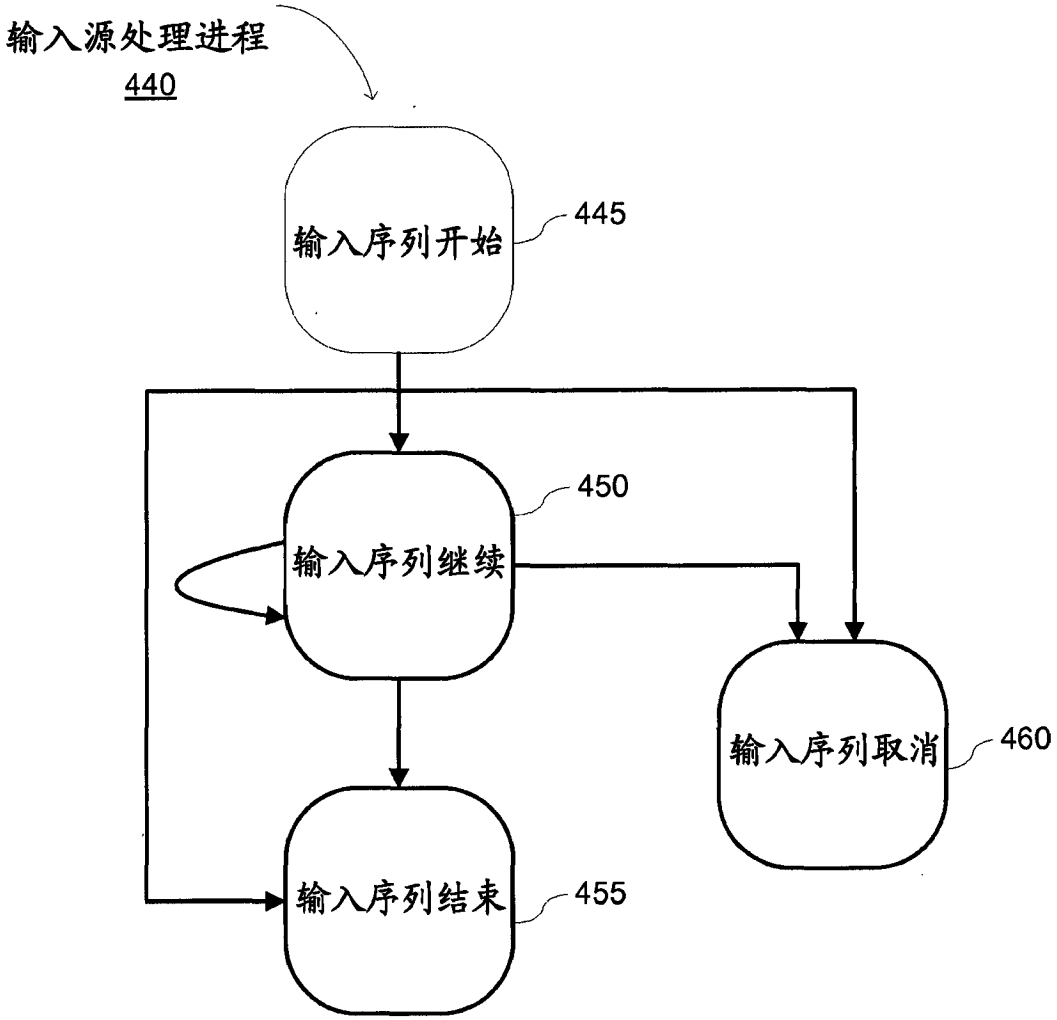


图 4B

子事件序列 465

- 检测手指向下 465-1
- 测量延迟 465-2
- 检测至少 N 个像素的手指垂直轻扫 465-3
- 检测手指抬起 465-4

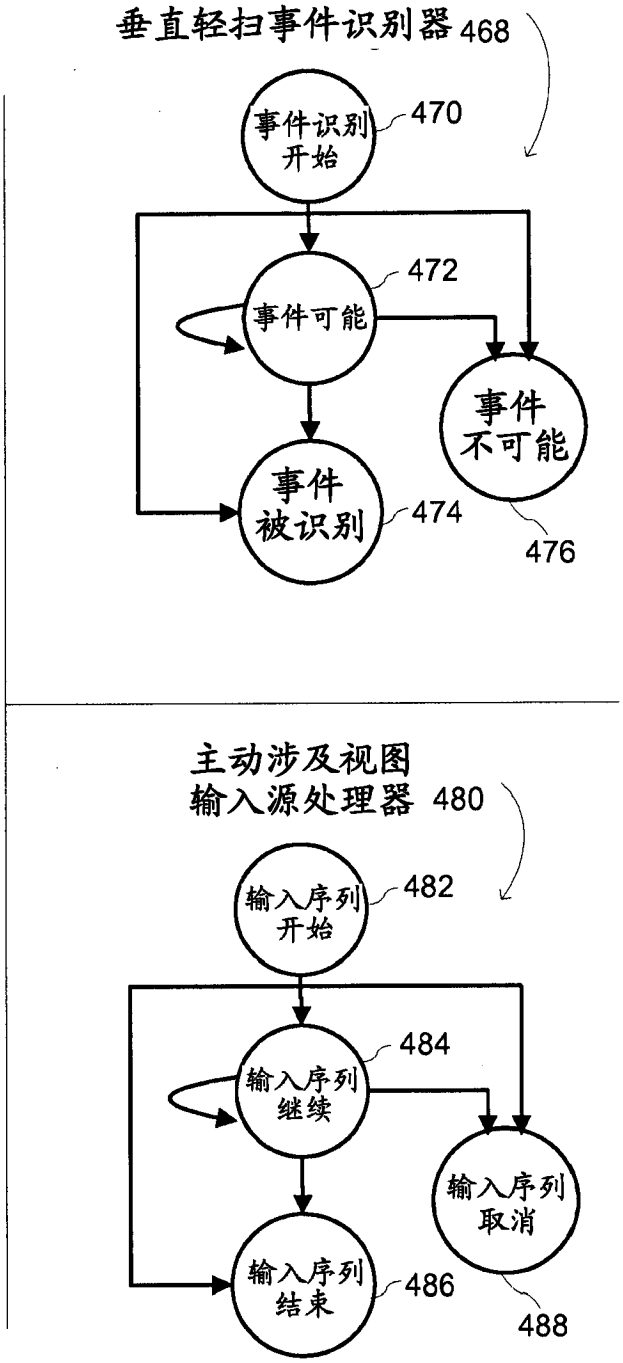


图 4C

子事件序列 520

- 检测手指向下 521-1
- 测量延迟 521-2
- 检测手指抬起 521-3

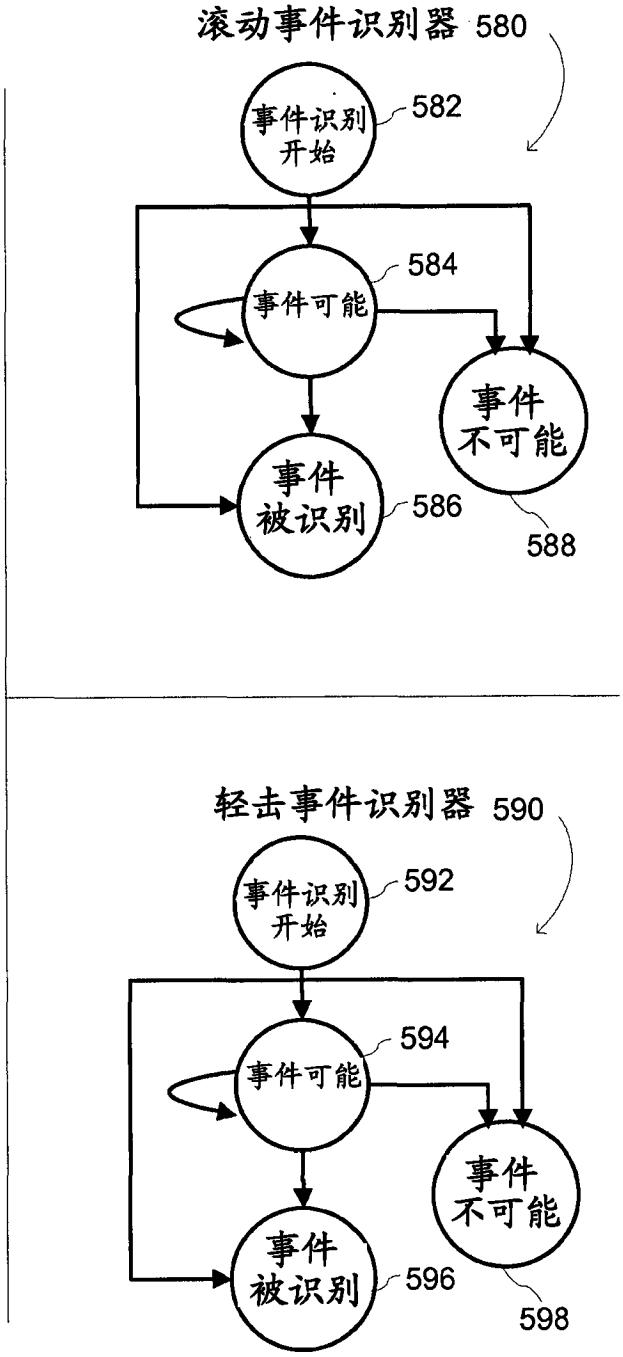


图 5A

子事件序列 530

- 检测手指向下 531-1
- 测量延迟 531-2
- 检测手指移动 531-3
- 检测手指抬起 531-4

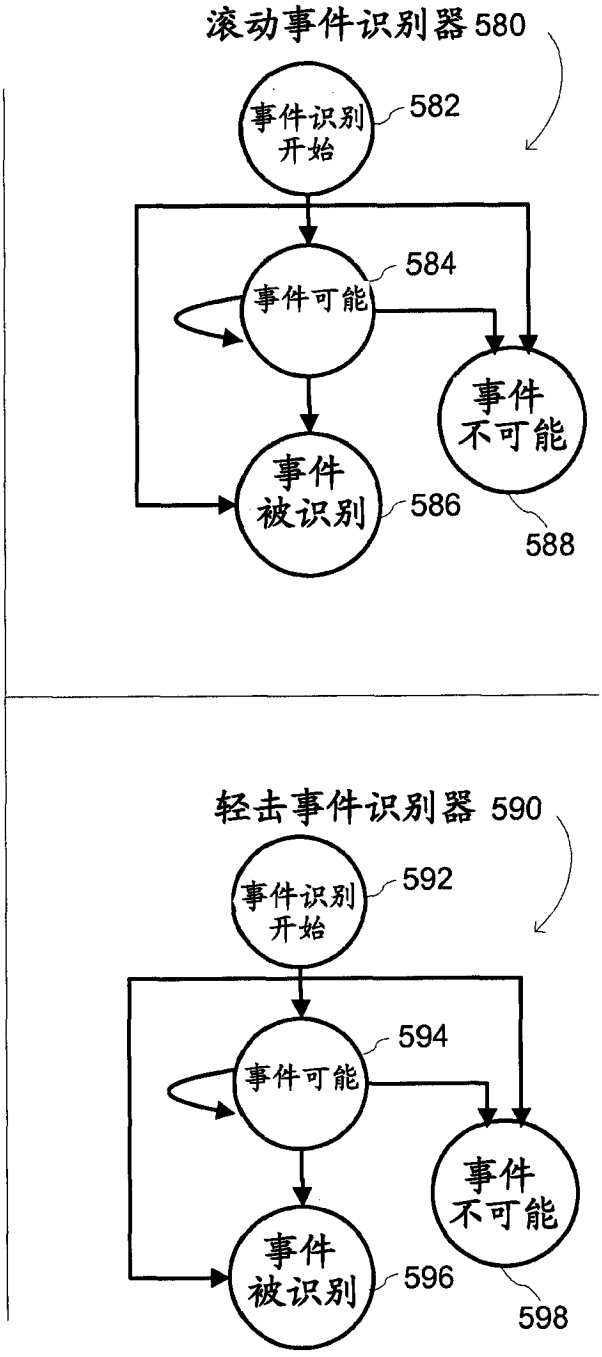


图 5B

子事件序列 540

- 检测手指向下 541-1
- 测量延迟 541-2
- 检测手指抬起 541-3
- 测量延迟 541-4
- 检测手指向下 541-5
- 测量延迟 541-6
- 检测手指抬起 541-7

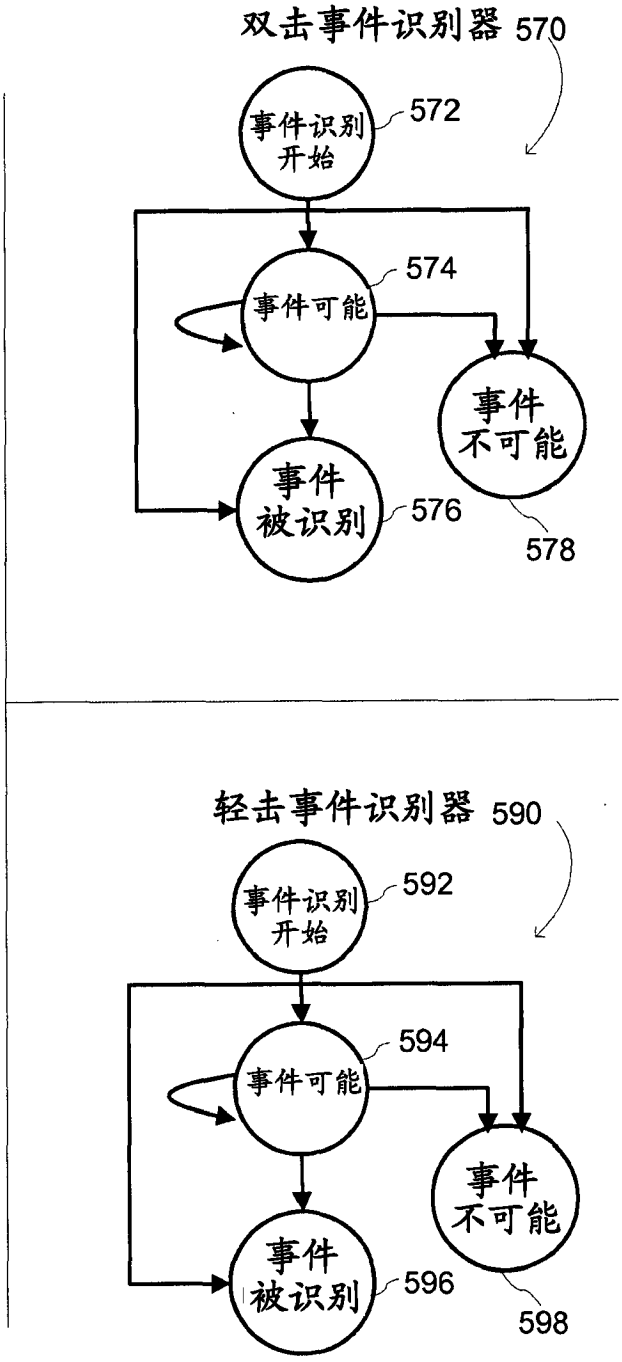


图 5C

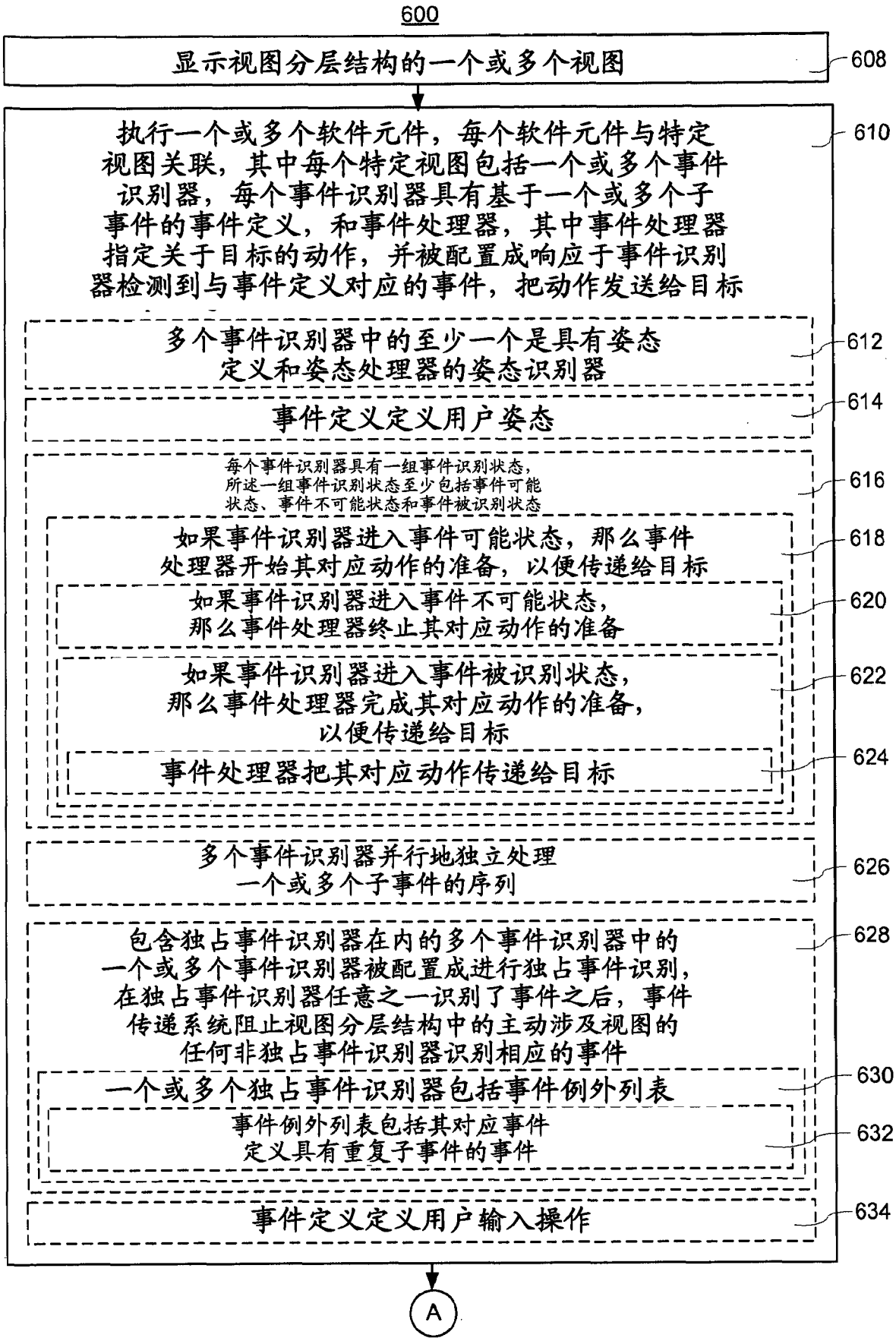

600
显示视图分层结构的一个或多个视图
608
执行一个或多个软件元件，每个软件元件与特定视图关联，其中每个特定视图包括一个或多个事件识别器，每个事件识别器具有基于一个或多个子事件的事件定义，和事件处理器，其中事件处理器指定关于目标的动作，并被配置成响应于事件识别器检测到与事件定义对应的事件，把动作发送给目标
610
多个事件识别器中的至少一个是具有姿态定义和姿态处理器的姿态识别器
612
事件定义定义用户姿态
614
每个事件识别器具有一组事件识别状态，所述一组事件识别状态至少包括事件可能状态、事件不可能状态和事件被识别状态
616
如果事件识别器进入事件可能状态，那么事件处理器开始其对应动作的准备，以便传递给目标
618
如果事件识别器进入事件不可能状态，那么事件处理器终止其对应动作的准备
620
如果事件识别器进入事件被识别状态，那么事件处理器完成其对应动作的准备，以便传递给目标
622
事件处理器把其对应动作传递给目标
624
多个事件识别器并行地独立处理一个或多个子事件的序列
626
包含独占事件识别器在内的多个事件识别器中的一个或多个事件识别器被配置成进行独占事件识别，在独占事件识别器任意之一识别了事件之后，事件传递系统阻止视图分层结构中的主动涉及视图的任何非独占事件识别器识别相应的事件
628
一个或多个独占事件识别器包括事件例外列表
630
事件例外列表包括其对应事件定义具有重复子事件的事件
632
事件定义定义用户输入操作
634
A

图 6A

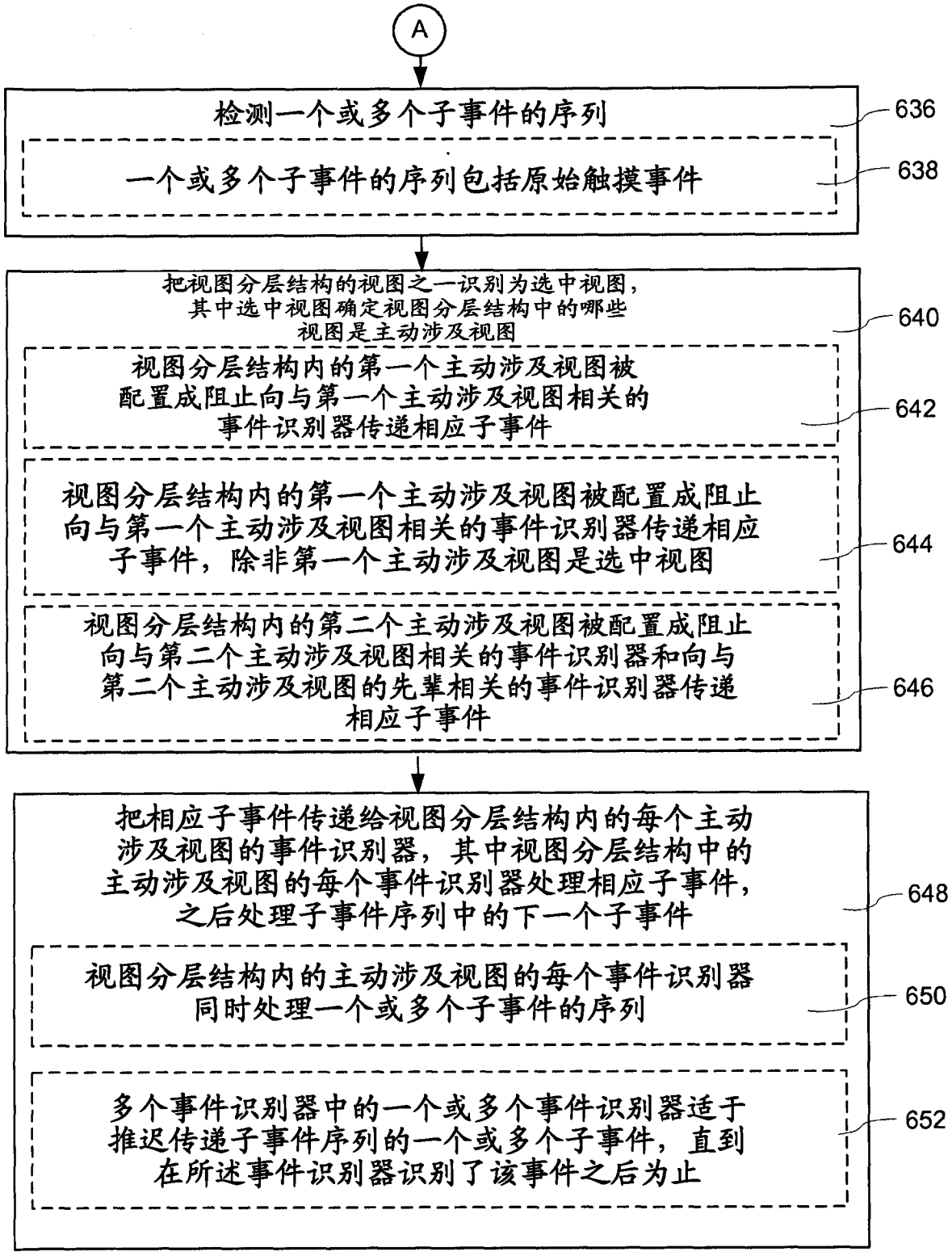

A
检测一个或多个子事件的序列
636
一个或多个子事件的序列包括原始触摸事件
638
把视图分层结构的视图之一识别为选中视图，其中选中视图确定视图分层结构中的哪些视图是主动涉及视图
640
视图分层结构内的第一个主动涉及视图被配置成阻止向与第一个主动涉及视图相关的事件识别器传递相应子事件
642
视图分层结构内的第一个主动涉及视图被配置成阻止向与第一个主动涉及视图相关的事件识别器传递相应子事件，除非第一个主动涉及视图是选中视图
644
视图分层结构内的第二个主动涉及视图被配置成阻止向与第二个主动涉及视图相关的事件识别器和向与第二个主动涉及视图的先辈相关的事件识别器传递相应子事件
646
把相应子事件传递给视图分层结构内的每个主动涉及视图的事件识别器，其中视图分层结构中的主动涉及视图的每个事件识别器处理相应子事件，之后处理子事件序列中的下一个子事件
648
视图分层结构内的主动涉及视图的每个事件识别器同时处理一个或多个子事件的序列
650
多个事件识别器中的一个或多个事件识别器适于推迟传递子事件序列的一个或多个子事件，直到在所述事件识别器识别了该事件之后为止
652

图 6B