



(12) 发明专利申请

(10) 申请公布号 CN 101790715 A

(43) 申请公布日 2010.07.28

(21) 申请号 200980000013.6

(51) Int. Cl.

(22) 申请日 2009.03.03

G06F 3/048 (2006.01)

(30) 优先权数据

G06F 17/30 (2006.01)

12/042, 299 2008.03.04 US

(85) PCT申请进入国家阶段日

2009.04.02

(86) PCT申请的申请数据

PCT/US2009/035874 2009.03.03

(87) PCT申请的公布数据

W02009/111469 EN 2009.09.11

(71) 申请人 苹果公司

地址 美国加利福尼亚

(72) 发明人 R·威廉姆森 G·D·博尔辛加
T·奥默尼克(74) 专利代理机构 中国国际贸易促进委员会专
利商标事务所 11038

代理人 李玲

权利要求书 1 页 说明书 17 页 附图 6 页

(54) 发明名称

触摸事件模型编程接口

(57) 摘要

本发明涉及触摸事件模型编程接口。可以从触摸敏感设备中获取一个或多个触摸输入信号。根据这些触摸输入信号，可以使用触摸事件模型来确定触摸和 / 或手势事件。这些触摸和手势事件可以与那些从触摸敏感设备上显示的 web 页面的不同区域产生的触摸输入信号相关联。通过编程接口，可以提供对至少一个触摸或手势事件的访问。

应用	212
web 页面 SDK	210
基础	208
核心基础	205
OS API	206
核心 OS	204
一个或多个驱动器	202
硬件	200

1. 一种 web 浏览器中的方法,包括 :

接收与手势事件相关联的旋转值;以及

基于所述旋转值而在该 web 浏览器中动态旋转与所述手势事件相关联的 web 页面的单元,其中所述旋转值是以度数为单位的相对增量。

2. 根据权利要求 1 所述的方法,其中所述手势事件包括两个或更多个触摸事件。

3. 一种 web 浏览器中的方法,包括 :

接收与手势事件相关联的缩放值;以及

基于所述缩放值而在该 web 浏览器中动态调整与所述手势事件相关联的 web 页面的单元的大小,其中所述缩放值是以文档像素为单位的相对增量。

4. 根据权利要求 3 所述的方法,其中所述手势事件与两个或更多个触摸事件相关联。

5. 一种 web 浏览器中的方法,包括 :

接收触摸列表,所述触摸列表包括用于标识 web 页面上的一个或多个触摸的触摸事件数据,其中所述触摸事件数据包括触摸标识符以及至少一组触摸位置坐标,其中所述触摸列表还包括用于涉及与每个触摸相关联的触摸事件目标的触摸事件数据,其中所述至少一组触摸位置坐标包括客户机坐标、页面坐标和屏幕坐标中的至少一种。

6. 根据权利要求 5 所述的方法,其中所述触摸事件数据标识一个或多个有改变的触摸。

7. 一种其上存储有指令的计算机可读介质,所述指令在由处理器执行时,使处理器执行包括如下的操作 :

接收与手势事件相关联的旋转值,其中所述手势事件与 web 页面相关联;以及

基于所述旋转值而在 web 浏览器中动态旋转与所述手势事件相关联的 web 页面的单元,其中所述旋转值是以度数为单位的相对增量。

8. 根据权利要求 7 所述的计算机可读介质,其中所述手势事件包括两个或更多个触摸事件。

9. 一种其上存储有指令的计算机可读介质,所述指令在由处理器执行时,使处理器执行包括如下的操作 :

接收与手势事件相关联的缩放值,其中所述手势事件与 web 页面相关联;以及

基于所述缩放值而在该 web 浏览器中动态调整与所述手势事件相关联的 web 页面的单元的大小,其中所述缩放值是以文档像素为单位的相对增量。

10. 根据权利要求 9 所述的计算机可读介质,其中所述手势事件与两个或更多个触摸事件相关联。

11. 一种其上存储有指令的计算机可读介质,所述指令在由处理器执行时,使处理器执行包括如下的操作 :

接收触摸列表,所述触摸列表包括用于标识 web 页面上的一个或多个触摸的触摸事件数据,其中所述触摸事件数据包括触摸标识符以及至少一组触摸位置坐标,其中所述触摸列表还包括用于涉及与每个触摸相关联的触摸事件目标的触摸事件数据,其中所述至少一组触摸位置坐标包括客户机坐标、页面坐标和屏幕坐标中的至少一种。

12. 根据权利要求 11 所述的计算机可读介质,其中所述触摸事件数据标识一个或多个有改变的触摸。

触摸事件模型编程接口

技术领域

[0001] 本主题主要涉及 web 浏览 (web browsing, 网页浏览) 服务。

背景技术

[0002] web 页面 (web page, 网页) 是用标记语言创建的, 该标记语言提供了这样一种手段, 即用于描述文档中基于文本的信息的结构并为该文本增补交互表单、嵌入图像及其他对象。一种流行的标记语言是超文本标记语言 (HTML), 该语言是用被尖括号 (angle bracket) 包围的标记的形式编写的。HTML 可以描述 web 页面的外观和语义, 并且可以包括嵌入式脚本语言代码 (例如**JavaScript®**), 该代码可以影响 web 浏览器及其他 HTML 处理器的行为。**JavaScript®** 为开发人员提供了在 web 页面中添加鼠标事件处理机 (handler) 或事件监听器 (listener) 的能力。这些鼠标事件处理机可以被指定到 web 页面中的特定区域, 并且可以被配置成接收这些区域中的鼠标事件, 例如鼠标释放 (mouse up) 事件或鼠标按下 (mouse down) 事件。

[0003] 相比之下, 对使用触摸敏感设备导航的 web 页面来说, 这些 web 页面通常需要对由用户使用一个或多个手指触摸 web 页面以及做出手势所产生的触摸事件做出响应。常规的鼠标事件处理机不能正确解释这些触摸事件。由此, 这些触摸事件需要一种不同的触摸事件模型来正确解释触摸事件并且允许开发人员充分利用触摸敏感显示器或设备的能力。

发明内容

[0004] 可以从触摸敏感设备中获取一个或多个触摸输入信号。根据这些触摸输入信号, 可以使用触摸事件模型来确定触摸和 / 或手势事件。这些触摸和手势事件可以与那些从触摸敏感设备上显示的 web 页面的不同区域产生的触摸输入信号相关联。通过编程接口, 可以提供对至少一个触摸或手势事件的访问。

[0005] 在某些实施方式中, 一种在 web 浏览器中的方法, 包括: 接收与手势事件相关联的旋转值; 以及根据该旋转值而在 web 浏览器中动态旋转与该手势事件相关联的 web 页面单元, 其中该旋转值是一个以度数为单位的相对增量。该手势事件可以包括两个或多个触摸事件。

[0006] 在某些实施方式中, 一种在 web 浏览器中的方法, 包括: 接收与手势事件相关联的缩放值; 基于该缩放值而在 web 浏览器中动态调整与手势事件相关联的 web 页面单元的大小, 其中该缩放值是以文档像素为单位的相对增量。该手势事件可以与两个或多个触摸事件相关联。

[0007] 在某些实施方式中, 一种在 web 浏览器中的方法, 包括: 接收触摸列表, 该触摸列表包括用于标识 web 页面上的一个或多个触摸的触摸数据, 其中该触摸数据包括触摸标识符以及至少一组触摸位置坐标, 其中该触摸列表还包括用于涉及与每个触摸相关联的触摸事件目标的数据, 其中所述至少一组触摸位置坐标包括一组客户机坐标, 一组页面坐标和一组屏幕坐标。该触摸数据可以标识一个或多个有改变的触摸。

[0008] 在这里还公开了涉及系统、方法和计算机可读介质的其他实施方式。

附图说明

- [0009] 图 1 示出的是例示 web 页面文档。
- [0010] 图 2 示出的是例示的具有多点触摸能力的设备的处理堆栈。
- [0011] 图 3 是用于处理触摸事件的例示处理的流程图。
- [0012] 图 4 示出的是例示的具有多点触摸能力的设备。
- [0013] 图 5 是用于图 4 中具有多点触摸能力的设备的例示网络操作环境的框图。
- [0014] 图 6 是图 4 中具有多点触摸能力的设备的例示实施方式的框图。

具体实施方式

[0015] 例示的 web 页面结构和 DOM

[0016] 图 1A 显示的是可以在浏览器上显示的例示 web 页面 100。该浏览器可以主存在便携式设备上,诸如图 4 中具有多点触摸能力的设备 400 上。在 web 页面 100 上可以显示一个或多个单元,即单元 102(“单元 1”)、单元 104(“单元 2”)以及单元 106(“单元 3”)。这些单元 102、104、106 可以与用户可选的 web 页面 100 中的各区域相对应,并且在这里还可以提供附加功能作为选择结果。举例来说,这些单元可以对应于 web 页面 100 上的按钮。此外,这些单元还可以嵌套,以使一个单元包含另一个单元。例如,单元 104 包含了单元 108。在所显示的示例中,举例来说,单元 108 是一个嵌套在单元 104 内部的擦除器控制(scrubber control),而单元 104 则例如可以是媒体播放器的用户界面。

[0017] 在某些实施方式中,用户可以使用手指而不是鼠标以结合 web 页面 100 上的单元执行各种功能。例如,用户可以使用图 4 所示的触摸敏感显示器 402 来触摸 web 页面 100 的单元。在一个示例中,用户可以通过用一个或多个手指触摸该单元和 / 或通过做出像轻扫(swipe)、合拢(pinch)或旋转(rotate)运动之类的手势来选择某个单元。为了识别触摸输入信号,web 页面 100 的某些区域可以与触摸事件处理机相关联。如将要参考图 1B 所描述的那样,这种处理可以用 DOM 以及嵌入式脚本语言来实现。

[0018] 图 1B 是与 web 页面 100 相关联的例示性 DOM 150。DOM 150 提供了 web 页面 100 的结构表示,并且将 web 页面内容描述成是一组可以被脚本语言(例如**JavaScript®**)解释的对象。在某些实施方式中,DOM 150 通过将 web 页面 100 中的单元 102、104、106、108 映射到树的各独立节点来提供对 web 页面结构的访问。例如,单元 102 对应于节点 154。单元 104 对应于节点 156。单元 106 对应于节点 158。单元 108 对应于节点 160。根节点 152 对应于整个 web 页面 100。

[0019] 在某些实施方式中,通过将 DOM 150 中的相应节点与触摸事件处理机相关联,可以将 web 页面 100 中的一个或多个单元 102、104、106、108 与一个或多个相应的触摸事件处理机相关联。触摸事件处理机可以被插入到 web 页面 100 的 HTML 标签中,并且该触摸事件处理机可以在例如用户在 web 页面 100 上的某个单元内部进行触摸或做出手势时运行脚本语言来执行动作。举例来说,**JavaScript®**可以与 DOM 150 一起工作,以便将动作附着于不同的触摸事件。

[0020] 在某些实施方式中,一个或多个单元 102、104、106、108 可以接收由事件处理机或

监听器检测到的触摸输入。如参考图 2 所描述的那样,该触摸输入可以由触摸事件模型检测并处理成触摸事件,其中该触摸事件模型可以在软件堆栈的一个或多个层中实施。触摸事件可以由 web 页面 100 进一步处理。触摸事件可以采用与触摸敏感设备产生的原始触摸输入信号相比更易于在应用中使用的格式(例如属性)。举例来说,每一个触摸事件都可以包括一组当前正在发生的触摸所在的坐标。

[0021] web 页面 100 中的每一个单元及其关联事件处理机都可以接收、处理和操作触摸事件。举个例子,如果驱动器 202(图 2)感测到与单元 102 相关联的触摸点 110 或是与单元 104 相关联的触摸点 112,那么与单元 102 或 104 相关联的事件处理机就可以接收指示该单元已被触摸的独立的触摸事件,并且可以可选地将触摸事件发送到 web 页面 100 以供进一步处理。在某些实施方式中,如果被触摸的 web 页面 100 的区域不与事件处理机相对应,那么所述输入可由应用层 214 中的浏览器处理,而非 web 页面 100 来处理。

[0022] 在某些实施方式中,在 DOM 150 中可以逐手指逐节点地检测触摸事件。例如,用户可在基本上相同的时间在触摸点 110 和触摸点 112 处触摸该触摸敏感显示器 402,并且触摸事件模型可以检测到两个独立的触摸事件。由于 DOM 150 中的每一个节点 102 和 104 都与独立的触摸事件处理机相关联,因此,可以为触摸点 110 和触摸点 112 检测到独立触摸事件。

[0023] 在某些实施方式中,触摸事件可以作为 EventTarget(事件目标)而被递送到 web 页面 100。触摸事件的某些示例可以包括触摸开始(touchstart),触摸移动(touchmove),触摸结束(touchend)以及触摸取消(touchcancel)。此外,其他触摸事件也是可能的。触摸开始是当用户首次将手指放在触摸敏感显示器 402 上且位于 web 页面 100 中与事件处理机相关联的一区域内时检测到的触摸事件。当用户在 web 页面 100 上四处移动其手指时,则可以检测到一个或多个触摸移动事件。当用户将其手指抬离 web 页面 100 时,则会检测到触摸结束事件。当系统中断常规事件处理时,则可检测到触摸取消。例如,触摸取消事件可以在为防止无意触摸而锁定触摸敏感显示器 402 时发生。

[0024] 在某些实施方式中,手势事件还可以通过组合两个或多个触摸事件而被检测。与触摸事件相似,手势事件(GestureEvent)同样可以作为事件目标(EventTarget)而被递送到 web 页面 100。手势事件的某些示例可以是手势开始(gesturestart),手势改变(gesturechange)和手势结束(gestureend)。手势事件可以包括缩放和 / 或旋转信息。旋转信息可以包括旋转值,该旋转值是一个以度数为单位的相对增量。web 页面 100 上的单元可以根据该旋转值而被动态旋转。缩放信息可以包括一个缩放值,该缩放值是以文档像素为单位的相对增量。对与手势事件相关联的 web 页面 100 上的单元来说,其大小可以根据该缩放值而被动态调整。此外,其他手势事件也是可能的。

[0025] 在某些实施方式中,可以接收一个包含了用于标识 web 页面 100 上的一个或多个触摸的触摸事件数据的触摸列表。触摸事件数据可以包括触摸标识符以及至少一组触摸位置坐标。此外,该触摸列表还可以包括涉及与每一个触摸相关联的触摸事件目标的触摸事件数据。在某些实施方式中,这样一个触摸位置坐标组可以包括客户机坐标、页面坐标和屏幕坐标。在某些实施方式中,触摸事件数据可以标识一个或多个有改变的触摸。

[0026] 在某些实施方式中,GestureEvent 可以在 TouchEvent 之前被发送到 web 页面 100。举个例子,如果用户将手指放在触摸点 110 和触摸点 112 上,然后使用这些手指在触

摸敏感显示器上做出顺时针或逆时针旋转手势,那么触摸事件模型将会检测到这些多个触摸事件,并且会将这些触摸事件组合成一个手势事件。然后,该手势事件可以被发送到 web 页面 100,其后跟随的是经组合形成该手势事件的各触摸事件。这样一来,开发人员可以访问手势事件以及该手势事件中的各个单独的触摸事件,由此就能在开发 web 应用时,为开发人员提供更大的灵活性。

[0027] 在某些实施方式中,触摸事件是依照如下顺序接收的:触摸开始事件、一个或多个触摸移动事件、以及触摸结束或触摸取消事件。通过使用图 1A 的示例,当用户接触触摸点 110 时,与单元 102 相关联的第一触摸事件处理机将会检测到第一触摸开始事件。当用户接触触摸点 112 时,与单元 104 相关联的第二触摸事件处理机将会检测到第二触摸开始事件。当用户旋转其手指而没有抬起其手指时,第一和第二触摸事件处理机将会检测到触摸移动事件,并且该触摸移动事件可以被触摸事件模型解释成是旋转手势事件。当用户结束旋转并且将其手指抬离 web 页面 100 时,第一和第二触摸事件处理机将会检测到触摸结束事件。所有或某些触摸事件可以通过触摸事件应用编程接口 (API) 而对开发人员可用。触摸 API 可以作为软件开发工具包 (SDK) 或是作为应用的一部分 (例如作为浏览器工具包的一部分) 而对开发人员可用。该触摸事件 API 可以依靠其他服务、框架和操作系统来执行其各种功能。如参考图 2 所述,在触摸事件与那些可以插入文档用以在应用中定义事件动作的属性相关联的情况下,这些服务、框架和操作系统可以是软件或处理堆栈的一部分。

[0028] 例示的 IDL

[0029] 现在将用接口描述语言 (IDL) 描述例示的触摸事件模型。IDL 的功能和数据结构可以由 web 设计人员或应用开发人员通过 API 来访问。对触摸事件和 / 或手势事件的访问可以与那些可插入标记语言文档 (例如 HTML, XML) 用以在应用中定义事件动作的属性相关联。例如,这些属性可以插入到 HTML 文档的一个或多个 HTML 标签中,用以产生在触摸敏感显示器 402 上显示的 web 页面。该事件动作可以包括运行一个嵌入式脚本 (例如 **JavaScript®**)。

```
[0030] interface[  
[0031]     Conditional = TOUCH_EVENTS,  
[0032]     GenerateConstructor  
[0033] ]TouchEvent:UIEvent {  
[0034]     void initTouchEvent(in AtomicString type,  
[0035]                           in boolean canBubble,  
[0036]                           in boolean cancelable,  
[0037]                           in DOMWindow view,  
[0038]                           in long detail,  
[0039]                           in long screenX,  
[0040]                           in long screenY,  
[0041]                           in long clientX,  
[0042]                           in long clientY,  
[0043]                           in boolean ctrlKey,  
[0044]                           in boolean altKey,
```

```
[0045]           in boolean shiftKey,
[0046]           in boolean metaKey,
[0047]           in TouchList touches,
[0048]           in TouchList targetTouches,
[0049]           in TouchList changedTouches,
[0050]           in long scale,
[0051]           in long rotation) ;
[0052]   readonly attribute TouchList touches ;// 所有触摸
[0053]   readonly attribute TouchList targetTouches ;// 该TouchEvent Target(触
摸事件目标) 中的所有触摸
[0054]   readonly attribute TouchList changedTouches ;// 当前事件中有改变的所
有触摸
[0055]   readonly attribute long scale ;
[0056]   readonly attribute long rotation ;
[0057]   readonly attribute boolean ctrlKey ;
[0058]   readonly attribute boolean shiftKey ;
[0059]   readonly attribute boolean altKey ;
[0060]   readonly attribute boolean metaKey ;
[0061] } ;
[0062] interface[
[0063]   Conditional = TOUCH_EVENTS,
[0064] ]Touch{
[0065]   readonly attribute EventTarget target ;
[0066]   readonly attribute long identifier ;
[0067]   readonly attribute long clientX ;
[0068]   readonly attribute long clientY ;
[0069]   readonly attribute long pageX ;
[0070]   readonly attribute long pageY ;
[0071]   readonly attribute long screenX ;
[0072]   readonly attribute long screenY ;
[0073] } ;
[0074] interface[
[0075]   Conditional = TOUCH_EVENTS,
[0076]   HasIndexGetter,
[0077] ]TouchList{
[0078]   readonly attribute unsigned long length ;
[0079]   Touch item(in unsigned long index) ;
[0080] } ;
[0081] interface[
```

```
[0082]     Conditional = TOUCH_EVENTS,
[0083]     GenerateConstructor
[0084] ]GestureEvent:UIEvent{
[0085]     void initGestureEvent(in AtomicString type,
[0086]                           in boolean canBubble,
[0087]                           in boolean cancelable,
[0088]                           in DOMWindow view,
[0089]                           in long detail,
[0090]                           in long screenX,
[0091]                           in long screenY,
[0092]                           in long clientX,
[0093]                           in long clientY,
[0094]                           in boolean ctrlKey,
[0095]                           in boolean altKey,
[0096]                           in boolean shiftKey,
[0097]                           in boolean metaKey,
[0098]                           in EventTarget target,
[0099]                           in long scale,
[0100]                           in long rotation) ;
[0101]     readonly attribute EventTarget target ;
[0102]     readonly attribute long scale ;
[0103]     readonly attribute long rotation ;
[0104]     readonly attribute boolean ctrlKey ;
[0105]     readonly attribute boolean shiftKey ;
[0106]     readonly attribute boolean altKey ;
[0107]     readonly attribute boolean metaKey ;
[0108] } ;
[0109] In Document.idl:
[0110] Touch    createTouch(in EventTarget target,
[0111]                      in long identifier,
[0112]                      in long clientX,
[0113]                      in long clientY,
[0114]                      in long pageX,
[0115]                      in long pageY,
[0116]                      in long screenX,
[0117]                      in long screenY)
[0118]     raises(DOMException) ;
[0119] [Custom]TouchList    createTouchList()
[0120]     raises(DOMException) ;
```

[0121] 以下是通过使用如上的例示 IDL 来处理触摸事件的 HTML 代码片段的示例。举例来说,以下 HTML 显示的是用 HTML 代码添加到单元中的触摸事件监听器 TouchStart 和 GestureStart :

[0122] this.element.addEventListener(' touchstart' ,function(e){return self.onTouchStart(e)), false) ;
[0123] this.element.addEventListener(' gesturestart ' , function(e){return self.onGestureStart(e)), false) ;
[0124] 与以上 IDL 相对应的 HTML 代码可以如下：
[0125] <!DOCTYPE html PUBLIC" -//W3C//DTD HTML 4.01 Transitional//EN"
[0126] " http://www.w3.org/TR/html4/loose.dtd" >
[0127] <html lang = " en" >
[0128] <head>
[0129] <meta http-equiv = " Content-Type" content = " text/html ;charset = utf-8" >
[0130] <meta name = " viewport" content = " initial-scale = 1.0" />
[0131] <title>Transform Gestures</title>
[0132] <style type = " text/css" media = " screen" >
[0133] .box{
[0134] position:absolute ;
[0135] height:150px ;
[0136] width:150px ;
[0137] background-color:blue ;
[0138] }
[0139] .box:active {
[0140] background-color:red ;
[0141] }
[0142] body{
[0143] margin:0px ;
[0144] }
[0145] #container{
[0146] position:absolute ;
[0147] width:100% ;
[0148] height:100% ;
[0149] }
[0150] #main-box2{
[0151] top:10px ;
[0152] left:155px ;
[0153] background:red ;
[0154] z-index:1 ;

```
[0155]      }
[0156]      </style>
[0157]      <script type = " text/javascript" charset = " utf-8" >
[0158]          var trackedObjectCount = 0 ;
[0159]          function Box(inElement)
[0160]          {
[0161]              var self = this ;
[0162]              this.element = inElement ;
[0163]              this.scale = 1.0 ;
[0164]              this.rotation = 0 ;
[0165]              this.position = ' 0,0' ;
[0166]              this.element.addEventListener( ' touchstart ' , function(e) {return
self.onTouchStart(e)} , false) ;
[0167]              this.element.addEventListener( ' gesturestart ' , function(e) {return
self.onGestureStart(e)} , false) ;
[0168]          }
[0169]      Box.prototype = {
[0170]          // 位置串是无单位的“x, y”
[0171]          get position()
[0172]          {
[0173]              return this._position ;
[0174]          },
[0175]          set position(pos)
[0176]          {
[0177]              this._position = pos ;
[0178]              var components = pos.split( ' ' )
[0179]              var x = components[0] ;
[0180]              var y = components[1] ;
[0181]              const kUseTransform = true ;
[0182]              if(kUseTransform) {
[0183]                  this.element.style.webkitTransform = ' rotate( ' +this.
rotation+' deg)scale(' +this.scale+'
[0184]                  translate(' +x+' px, ' +y+' px)' ;
[0185]              }
[0186]              else{
[0187]                  this.element.style.left = x+' px' ;
[0188]                  this.element.style.top = y+' px' ;
[0189]              }
[0190]          },

```

```
[0191]     getx()
[0192]     {
[0193]         return parseInt(this._position.split(' , ' )[0]) ;
[0194]     },
[0195]     set x(inX)
[0196]     {
[0197]         var comps = this._position.split(' , ' );
[0198]         comps[0] = inX ;
[0199]         this.position = comps.join(' , ' );
[0200]     },
[0201]     gety()
[0202]     {
[0203]         return parseInt(this._position.split(' , ' )[1]) ;
[0204]     },
[0205]     set y(inY)
[0206]     {
[0207]         var comps = this._position.split(' , ' );
[0208]         comps[1] = inY ;
[0209]         this.position = comps.join(' , ' );
[0210]     },
[0211]     filterEvent:function(e)
[0212]     {
[0213]         // 防止浏览器执行其默认事件（滚动，缩放）
[0214]         e.preventDefault() ;
[0215]         // 在文档级添加事件监听器，由此接收关于其他单元的手势改变事件
[0216]         return(e.target == this.element) ;
[0217]     },
[0218]     onTouchStart:function(e)
[0219]     {
[0220]         if( ! this.filterEvent(e))
[0221]             return false ;
[0222]         // 当第一手指落至这一单元时开始追踪
[0223]         if(e.targetTouches.length != 1)
[0224]             return false ;
[0225]         this.startX = e.targetTouches[0].clientX ;
[0226]         this.startY = e.targetTouches[0].clientY ;
[0227]         var self = this ;
[0228]         if( ! (" touchMoveHandler" in this)){
[0229]             this.touchMoveHandler = function(e) {return self.onTouchMove(e)}
```

```
[0230]     this.touchEndHandler = function(e) {return self.onTouchEnd(e)}
[0231] }
[0232]     document.addEventListener('touchmove', this.touchMoveHandler,
false);
[0233]     document.addEventListener('touchend', this.touchEndHandler,
false);
[0234]     trackedObjectCount++;
[0235]     return false;
[0236] },
[0237] onTouchMove:function(e)
[0238] {
[0239]     if( ! this.filterEvent(e))
[0240]         return false;
[0241]     // 当在这一单元上落有多个触摸时(这是一个手势)不追踪运动
[0242]     if(e.targetTouches.length != 1)
[0243]         return false;
[0244]     var leftDelta = e.targetTouches[0].clientX-this.startX;
[0245]     var topDelta = e.targetTouches[0].clientY-this.startY;
[0246]     var newLeft = (this.x)+leftDelta;
[0247]     var newTop = (this.y)+topDelta;
[0248]     this.position = newLeft+', '+newTop;
[0249]     this.startX = e.targetTouches[0].clientX;
[0250]     this.startY = e.targetTouches[0].clientY;
[0251]     return false;
[0252] },
[0253] onTouchEnd:function(e)
[0254] {
[0255]     if( ! this.filterEvent(e))
[0256]         return false;
[0257]     // 当最后一根手指从这一单元上移开时,停止追踪
[0258]     if(e.targetTouches.length > 0)
[0259]         return false;
[0260]     document.removeEventListener('touchmove', this.touchMoveHandler,
false);
[0261]     document.removeEventListener('touchend', this.touchEndHandler,
false);
[0262]     trackedObjectCount--;
[0263]     return false;
[0264] },
```

```
[0265]     onGestureStart:function(e)
[0266]     {
[0267]         if( ! this.filterEvent(e))
[0268]             return false;
[0269]         var self = this;
[0270]         if( ! ("gestureChangeHandler" in this)){
[0271]             this.gestureChangeHandler = function(e){return self.
onGestureChange(e)}
[0272]             this.gestureEndHandler = function(e){return self.onGestureEnd(e)}
[0273]         }
[0274]         document.addEventListener('gesturechange',this.
gestureChangeHandler,true);
[0275]         document.addEventListener('gestureend',this.gestureEndHandler,
true);
[0276]         return false;
[0277]     },
[0278]     onGestureChange:function(e)
[0279]     {
[0280]         if( ! this.filterEvent(e))
[0281]             return false;
[0282]         // 在追踪一个对象时,只解释手势。否则,解释原始触摸事件
[0283]         // 移动所追踪的对象
[0284]         if(trackedObjectCount == 1){
[0285]             this.scale+ = e.scaling * 0.01;
[0286]             this.rotation+ = e.rotation/2;
[0287]             this.position = this.position;
[0288]         }
[0289]         return false;
[0290]     },
[0291]     onGestureEnd:function(e)
[0292]     {
[0293]         if( ! this.filterEvent(e))
[0294]             return false;
[0295]         document.removeEventListener('gesturechange',this.
gestureChangeHandler,true);
[0296]         document.removeEventListener('gestureend',this.
gestureEndHandler,true);
[0297]         return false;
[0298]     },
```

```
[0299]      }
[0300]      function loaded()
[0301]      {
[0302]          new Box(document.getElementById(' main-box' )) ;
[0303]          new Box(document.getElementById(' main-box2' )) ;
[0304]      }
[0305]      window.addEventListener(' load' , loaded, true) ;
[0306]      </script>
[0307]      </head>
[0308]      <body>
[0309]          <div id = " container" >
[0310]              <div id = " main-box" class = " box" ></div>
[0311]              <div id = " main-box2" class = " box" ></div>
[0312]          </div>
[0313]      </body>
[0314]  </html>
```

[0315] 用于多点触摸设备的例示处理堆栈

[0316] 图 2 是例示的具有多点触摸能力的设备的处理堆栈的图示。如上所述的触摸事件模型可以在处理堆栈及堆栈中的用户各类资源的一个或多个区域中实施。硬件 200 的层可以包括各种硬件接口元件,例如触摸敏感或启用的设备或是触摸敏感显示器。该触摸敏感设备可以包括显示器以及用于同时感测多个触摸的面板。该硬件层 200 还可以包括用于检测触摸敏感显示器或设备的定向(例如纵向,横向)的加速度计。由此,用于指示定向的信号可由触摸事件模型用来缩放 web 页面以进行最优显示。

[0317] 驱动器层 202 中的一个或多个驱动器可以与硬件 200 进行通信。例如,这些驱动器可以接收和处理由硬件层 200 中的触摸敏感显示器或设备产生的触摸输入信号。核心操作系统(OS)204 可以与一个或多个驱动器进行通信。核心 OS 204 可以处理从一个或多个驱动器接收的原始输入数据。在某些实施例中,这些驱动器可以被认为是核心 OS 204 的一部分。

[0318] 一组 OS 应用编程接口(API)206 可以与核心 OS 204 进行通信。这些 API 可以是一组通常与操作系统包含在一起的 API(例如 Linux 或 UNIX API)。其中一组核心基础 API 208 可以使用 OS API 206,而一组基础 API 210 则可以使用核心基础 API 208。

[0319] Web 页面软件开发工具包(SDK)210 可以包括一组被设计成供在设备上运行的应用使用的 API。而触摸事件 API 则例如可被包含在 Web 页面 SDK 210 中。Web 页面 SDK 210 的 API 可以利用基础 API 208。Web 页面 SDK 210 例如可以包括由 APPLE Inc. ® 提供的 WebKIT(Web 工具包)。Web 页面 SDK 210 可以作为 API 提供,或者也可以通过应用,例如,通过诸如由 APPLE Inc. ® 提供的 **SAFARI®** 之类的浏览器来访问。

[0320] 在设备上运行的应用 214 可以利用 Web 页面 SDK 210 的 API 来创建 web 页面。Web 页面 SDK 210 的 API 则又可以与各下层单元进行通信,由此最终与触摸敏感显示器或设备以及各种其他用户接口硬件进行通信。虽然每一层都可以利用其下方的层,但这并不总是

需要的。例如在某些实施例中，应用 214 可以不定期地与 OS API 206 通信。

[0321] 例示的触摸事件处理

[0322] 图 3 是通过 API 来提供对触摸和 / 或手势事件的访问的处理 300 的流程图。处理 300 可以通过获取一个或多个触摸输入信号而开始 (302)。这些触摸输入信号可以从触摸敏感显示器或设备获取。通过使用触摸事件模型，可以根据触摸输入信号来确定触摸事件和 / 或手势 (304)。这些触摸事件可以与显示在触摸敏感显示器或设备上的 web 页面的各区域相关联。例如，触摸敏感显示器可以是移动电话上的显示器，并且触摸敏感设备可以是笔记本计算机上的触摸敏感板。对触摸事件和 / 或手势事件的访问可以通过编程接口来提供 (306)。例如，对在上文中参考图 2 描述的 HTML 片段来说，该片段可以由 web 开发人员插入 HTML 文档，以便为开发人员提供对触摸和 / 或手势事件的访问。触摸事件和 / 或手势事件还可以由 HTML 文档中的代码进一步处理，以便启动事件动作 (306)。

[0323] 移动设备综述

[0324] 图 4 是例示的具有多点触摸能力的设备 400 的框图。在某些实施方式中，具有多点触摸能力的设备 400 包括触摸敏感显示器 402。触摸敏感显示器 402 可以实施液晶显示器 (LCD) 技术，发光聚合物显示器 (LPD) 技术或是某些其他显示器技术。触摸敏感显示器 402 可以对触觉 (haptic) 和 / 或与用户的触知 (tactile) 接触敏感。

[0325] 在某些实施方式中，触摸敏感显示器 402 可以包括多点触摸敏感显示器 402。举例来说，触摸敏感显示器 402 可以处理多个同时的触摸点，这其中包括处理与每一个触摸点的压力、角度和 / 或位置相关的数据。此类处理使用多个手指、和弦 (chording) 及其他交互来方便手势和交互。此外，也可以使用其他触摸敏感显示器技术，例如使用指示笔或其他指示设备来进行接触的显示器。关于多点触摸敏感显示器技术的某些示例在美国专利 No. 6, 323, 846、No. 6, 570, 557、No. 6, 677, 932 以及美国专利公开 2002/0015024A1 中有所描述，这其中的每一份文献全都在这里全部引入作为参考。在某些实施方式中，具有多点触摸能力的设备 400 可以在触摸敏感显示器 402 上显示一个或多个图形用户界面，以便为用户提供对各种系统对象的访问以及向用户传达信息。

[0326] 例示的具有多点触摸能力的设备的功能

[0327] 在某些实施方式中，具有多点触摸能力的设备 400 可以实施多种设备的功能，例如电话设备、电子邮件设备、网络数据通信设备、Wi-Fi 基站设备以及媒体处理设备。在某些实施方式中，具有多点触摸能力的设备 400 可以包括用于显示 web 页面（例如 web 页面 100）的 web 浏览器 404。触摸敏感显示器 402 可以接收在 web 页面 100 上产生的触摸输入信号，并且如上所述的触摸模型可以用于根据触摸输入信号来确定触摸和 / 或手势事件。在某些实施方式中，具有多点触摸能力的设备 400 可以实施网络分布功能。在某些实施方式中，在具有多点触摸能力的设备 400 接近用户耳朵时，可以锁操作 (lockdown) 触摸敏感显示器 402。这种锁操作将会导致产生如参考图 1B 所描述的触摸取消事件。

[0328] 在某些实施方式中，如指向箭头 474 所示，加速度计 472 可以用于检测具有多点触摸能力的设备 400 的移动。相应地，显示对象和 / 或媒体可以根据检测到的定向，例如，纵向或横向来呈现。在某些实施方式中，具有多点触摸能力的设备 400 可以包括用于支持位置确定能力的电路和传感器，其中举例来说，该能力可以是由全球定位系统 (GPS) 或其他定位系统（例如使用 Wi-Fi 接入点的系统、电视信号、蜂窝网格、统一资源定位符 (URL)）提

供的。在某些实施方式中，定位系统（例如 GPS 接收机）既可以集成到具有多点触摸能力的设备 400 中，也可以作为能通过接口来与具有多点触摸能力的设备 400 耦合的独立设备来提供，以便提供对基于位置的服务的访问。这种具有多点触摸能力的设备 400 还可以包括一个或多个无线通信子系统。

[0329] 在某些实施方式中，可以包括诸如通用串行总线 (USB) 端口或对接端口 (docking port) 之类的端口设备，或是某些其他有线端口连接。端口设备，例如可用于与其他计算设备建立有线连接，所述其他计算设备例如可以是其他具有多点触摸能力的设备 400、网络接入设备、个人计算机、打印机或是能够接收和 / 或发射数据的其他处理设备。在某些实施方式中，端口设备允许具有多点触摸能力的设备 400 使用一种或多种协议来与主机设备同步，诸如可以使用 TCP/IP、HTTP、UDP 以及其他任何已知协议。

[0330] 网络操作环境

[0331] 图 5 是用于图 4 中具有多点触摸能力的设备 400 的例示网络操作环境 600 的框图。举例来说，图 4 中具有多点触摸能力的设备 400 可以在数据通信中经一个或多个有线和 / 或无线网络 510 进行通信。例如，无线网络 512 可以是例如蜂窝网络，它可以通过使用网关 516 来与诸如因特网之类的广域网 (WAN) 514 进行通信。同样，接入点 518 可以是诸如 802.11g 无线接入点，它可以提供对广域网 514 的通信访问。在某些实施方式中，语音和数据通信可以经由无线网络 512 和接入点 518 来建立。例如，具有多点触摸能力的设备 400a 可以经由无线网络 512、网关 516 以及广域网 514（例如使用 TCP/IP 或 UDP 协议）来发出和接收电话呼叫（例如使用 VoIP 协议），发送和接收电子邮件消息（例如使用 POP3 协议），以及检索电子文档和 / 或流，例如 web 页面、照片和视频。同样，具有多点触摸能力的设备 400b 可以经由接入点 518 和广域网 514 来发出和接收电话呼叫，发送和接收电子邮件消息，以及检索电子文档。在某些实施方式中，具有多点触摸能力的设备 400 可以使用一条或多条电缆与接入点 518 物理连接，并且接入点 518 可以是个人计算机。在这种配置中，具有多点触摸能力的设备 400 可以被称为“带缆 (tethered)”设备。

[0332] 具有多点触摸能力的设备 400a 和 400b 还可以借助其他手段来建立通信。例如，具有多点触摸能力的设备 400a 可以经由无线网络 512 而与其他无线设备，例如与其他具有多点触摸能力的设备 400、蜂窝电话等等进行通信。同样，具有多点触摸能力的设备 400a 和 400b 可以通过使用一个或多个通信子系统来建立诸如个人局域网之类的点对点通信 520，其中所述通信子系统可以是图 4 所示的 Bluetooth™ 通信设备 488。此外，也可以实施其他的通信协议和拓扑结构。

[0333] 举例来说，具有多点触摸能力的设备 400 可以经由一个或多个有线和 / 或无线网络 510 而与网络资源 530 进行通信。例如，该网络资源可以如参考图 1 ~ 2 所述，是用于递送那些能够经由触摸模型而被触摸的 web 页面的 web 服务器。

[0334] 还可以提供包括软件更新服务的其他服务，其中所述软件更新服务自动确定是否存在用于具有多点触摸能力的设备 400 上的软件的软件更新，随后则将该软件更新下载到具有多点触摸能力的设备 400，在设备 400 上软件可以被手动或自动解包和 / 或安装。

[0335] 例示的移动设备架构

[0336] 图 6 是图 4 中具有多点触摸能力的设备 400 的例示实施方式的框图 600。具有多点触摸能力的设备 400 可以包括存储器接口 602、一个或多个数据处理器、图像处理器和 /

或中央处理单元 604, 以及外围接口 606。存储器接口 602、一个或多个处理器 604 和 / 或外围接口 606 既可以是分立元件, 也可以集成在一个或多个集成电路中。在具有多点触摸能力的设备 400 中, 各种元件可以通过一条或多条通信总线或信号线来耦合。

[0337] 传感器、设备和子系统可以耦合到外围接口 606, 以便帮助实现多种功能。例如, 运动传感器 610、光传感器 612 和接近度传感器 614 可以耦合到外围接口 606, 以方便有关图 4 所述的定向、照明和接近度功能。其他传感器 616 同样可以与外围接口 606 相连, 例如定位系统 (例如 GPS 接收机)、温度传感器、生物测定传感器或其他感测设备, 由此可以帮助实施相关功能。

[0338] 相机子系统 620 和光学传感器 622 可以用于方便诸如记录照片和视频剪辑的相机功能的实现, 其中所述相机子系统和光学传感器例如可以是电荷耦合器件 (CCD) 或互补金属氧化物半导体 (CMOS) 光学传感器。

[0339] 可以通过一个或多个无线通信子系统 624 来帮助实现通信功能, 其中所述无线通信子系统可以包括射频接收机和发射机和 / 或光 (例如红外) 接收机和发射机。通信子系统 624 的特定设计和实施方式可以取决于具有多点触摸能力的设备 400 打算在经其工作的一个或多个通信网络。例如, 具有多点触摸能力的设备 400 可以包括被设计成经 GSM 网络、GPRS 网络、EDGE 网络、Wi-Fi 或 WiMax 网络以及 Bluebooth™ 网络上工作的通信子系统 624。特别地, 无线通信子系统 624 可以包括主机协议, 以使设备 500 可被配置成用于其他无线设备的基站。

[0340] 音频子系统 626 可以与扬声器 628 以及麦克风 630 相耦合, 以便帮助实施启用语音的功能, 例如语音识别、语音复制、数字记录和电话功能。

[0341] I/O 子系统 640 可以包括触摸屏控制器 642 和 / 或一个或多个其他输入控制器 644。触摸屏控制器 642 可以耦合到触摸屏 646。举例来说, 该触摸屏 646 和触摸屏控制器 642 可以使用多种触摸感测技术中的任何一种来检测与之进行的接触和移动或是暂停, 其中感测技术包括但不限于电容性、电阻性、红外和表面声波技术, 以及其他接近度传感器阵列或其他用于确定与触摸屏 646 的一个或多个接触点的部件。

[0342] 一个或多个其他输入控制器 644 可以耦合到其他输入 / 控制设备 648, 例如一个或多个按钮、摇杆开关、拇指旋轮、红外端口、USB 端口、和 / 或指示笔之类的指针设备。所述一个或多个按钮 (未显示) 可以包括用于控制扬声器 628 和 / 或麦克风 630 音量的向上 / 向下按钮。

[0343] 在一个实施方式中, 可以通过按住按钮持续第一持续时间来解除触摸屏 646 的锁定; 并且如果按住按钮持续第二持续时间, 其中第二持续时间长于第一持续时间, 那么可以接通或切断具有多点触摸能力的设备 400 的电源。用户能够定制一个或多个按钮的功能。此外, 举例来说, 触摸屏 646 也可以用于实施虚拟或软按钮和 / 或数字键盘或键盘。

[0344] 在某些实施方式中, 具有多点触摸能力的设备 400 可以呈现已记录的音频和 / 或视频文件, 例如 MP3、AAC 和 MPEG 文件。在某些实施方式中, 具有多点触摸能力的设备 400 可以包括 MP3 播放器的功能, 例如 iPod™。由此, 具有多点触摸能力的设备 400 可以包括一个与 iPod 兼容的 32 引脚连接器。此外还可以使用其他的输入 / 输出和控制设备。

[0345] 存储器接口 602 可以与存储器 650 相耦合。该存储器 650 可以包括高速随机存取存储器和 / 或非易失性存储器, 例如一个或多个磁盘存储设备, 一个或多个光学存储设备,

和 / 或闪存存储器（例如 NAND, NOR）。存储器 650 可以存储操作系统 652，例如 Darwin、RTXC、LINUX、UNIX、OS X、WINDOWS，或是 VxWorks 之类的嵌入式操作系统。该操作系统 652 可以包括用于处理基本系统服务以及执行依赖于硬件的任务的指令。

[0346] 存储器 650 还可以存储通信指令 654，以便帮助实施与一个或多个附加设备、一个或多个计算机和 / 或一个或多个服务器的通信。该存储器 650 可以包括用于帮助实施图形用户界面处理的图形用户界面指令 656；用于帮助实施与传感器相关的处理和功能的传感器处理指令 658；用于帮助实施与电话相关的处理和功能的电话指令 660；用于帮助实施与电子消息收发相关的处理和功能的电子消息收发指令 662；用于帮助实施与 web 浏览相关的处理和功能的 web 浏览指令 664；用于帮助实施与媒体处理相关的处理和功能的媒体处理指令 666；用于帮助实施与 GPS 和导航相关的处理和功能的 GPS/ 导航指令 668；用于帮助实施与相机相关的处理和功能的相机指令 670；和 / 或其他那些用于帮助实施如参考图 1 ~ 5 描述的处理和功能的其他消息收发指令 672。

[0347] 以上标识的每个指令和应用都可以与用于执行上述一个或多个功能的一组指令相对应。这些指令没有必要实现为独立的软件程序、过程或模块。存储器 650 可以包括附加的指令或更少的指令。此外，具有多点触摸能力的设备 400 的各种功能可以在硬件和 / 或软件中实施，包括在一个或多个信号处理和 / 或专用集成电路中实施。

[0348] 所描述的特征既可以在数字电子电路中实施，也可以在计算机硬件、固件、软件或其组合中实施。这些特征可以在计算机程序产品中实施，其中该产品有形包含在机器可读存储设备或传播信号之类的信息载体中，以便由可编程处理器加以执行；此外，方法步骤可以由可编程处理器来执行，其中该处理器将会执行指令程序，以便通过对输入数据执行操作并产生输出来执行所描述的实施方式的功能。

[0349] 有利的是，所描述的特征可以在一个或多个计算机程序中实施，其中所述计算机程序可以在可编程系统上运行，该可编程系统包括至少一个可编程处理器，所述至少一个可编程处理器被耦合以从数据存储系统、至少一个输入设备以及至少一个输出设备中接收数据和指令并向其传送数据和指令。计算机程序是一组可以直接或间接在计算机中使用，以便执行某些活动或是带来某种结果的指令。计算机程序可以用包括编译或解释性语言在内任何形式的编程语言来编写（例如 Objective-C、Java），此外，它也可以采用任何形式来部署，包括独立程序或是模块、元件、子例程，或是适合在计算环境中使用的其他单元。

[0350] 作为例示，适于执行指令程序的处理器包括任何类型的计算机中的通用和专用微处理器，以及唯一处理器或是多处理器或核之一。通常，处理器会接收来自只读存储器、随机存取存储器或是这两者的指令和数据。计算机的基本部件是用于执行指令的处理器以及一个或多个用于存储指令和数据的存储器。通常，计算机还包括一个或多个用于存储数据文件的大容量存储设备或者以可操作的方式耦合成与至所述一个或多个用于存储数据文件的大容量存储设备进行通信；此类设备包括磁盘，例如内部硬盘和可移除盘；磁光盘；以及光盘。适合有形包含计算机程序指令和数据的存储设备包括任何形式的非易失性存储器，其示例包括：半导体存储器设备，如 EPROM、EEPROM 以及闪存设备；磁盘，例如内部硬盘和可移除盘；磁光盘；以及 CD-ROM 和 DVD-ROM 盘。处理器和存储器可以由 ASIC（专用集成电路）补充或可被引入其内。

[0351] 为了提供与用户的交互，这些特征可以在计算机上实施，其中该计算机具有

CRT(阴极射线管)或LCD(液晶显示器)监视器之类的显示设备用于向用户显示信息，并且还具有键盘以及鼠标或轨迹球之类的指示设备以供用户向计算机提供输入。

[0352] 这些特征可以在计算机系统中实施，其中该计算机系统包括诸如数据服务器之类的后端元件，或者包括诸如应用服务器或因特网服务器之类的中间件元件，或者可以包括诸如具有图形用户界面或因特网浏览器的客户计算机之类的前端元件或其任意组合。系统元件可以借助诸如通信网络之类的任何形式的数字数据通信介质来连接。通信网络的示例包括LAN、WAN以及构成因特网的计算机和网络。

[0353] 计算机系统可以包括客户机和服务器。客户机和服务器通常彼此远离，并且通常是通过网络来交互的。客户机与服务器的关系是借助运行在相应计算机上并且彼此具有客户机-服务器关系的计算机程序来产生的。

[0354] 在这里已经描述了多种实施方式。但是应该理解，各种修改都是可行的。例如，通过组合、删除、修改或补充一个或多个实施方式的要素，就可以构成更多的实施方式。另举一例，图中描述的逻辑流程无需按照显示的特定次序或顺序来实现预期结果。此外，可以提供其他步骤，也可以从所描述的流程中删除步骤，并且可以向所描述的系统添加或从中移除其他元件。因此，其他实施方式处于所附权利要求的范围内。

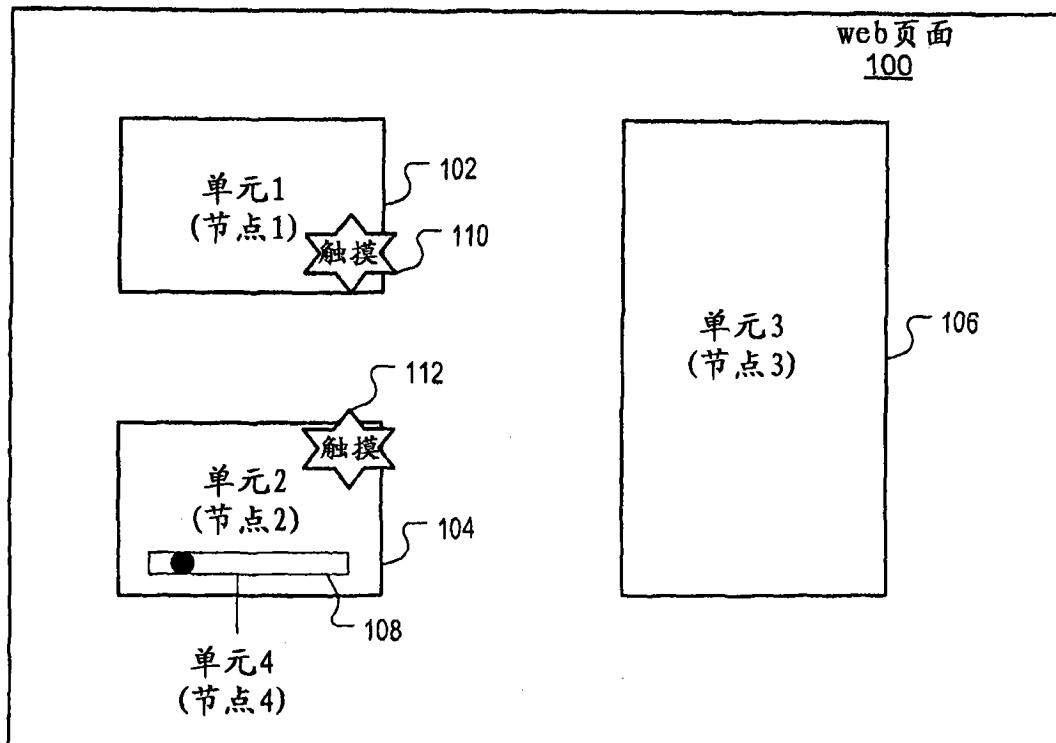


图 1A

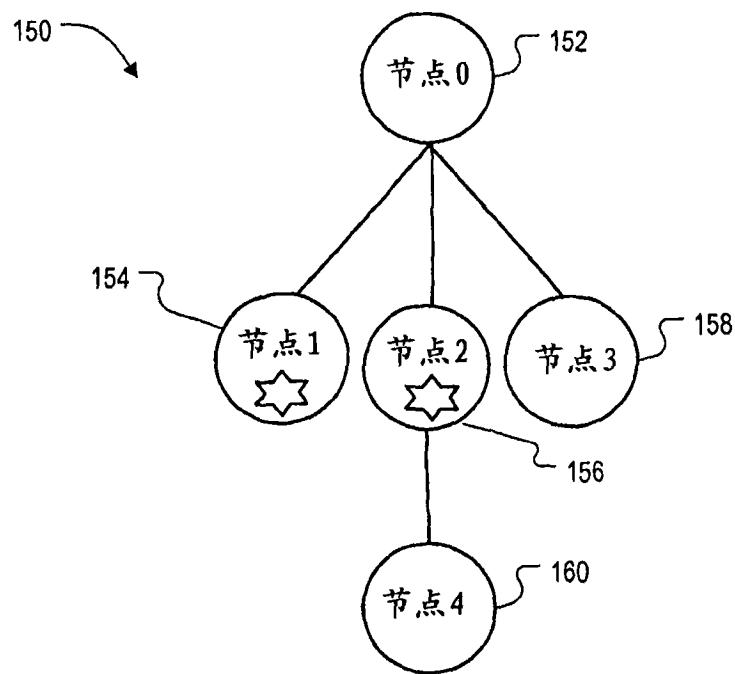


图 1B

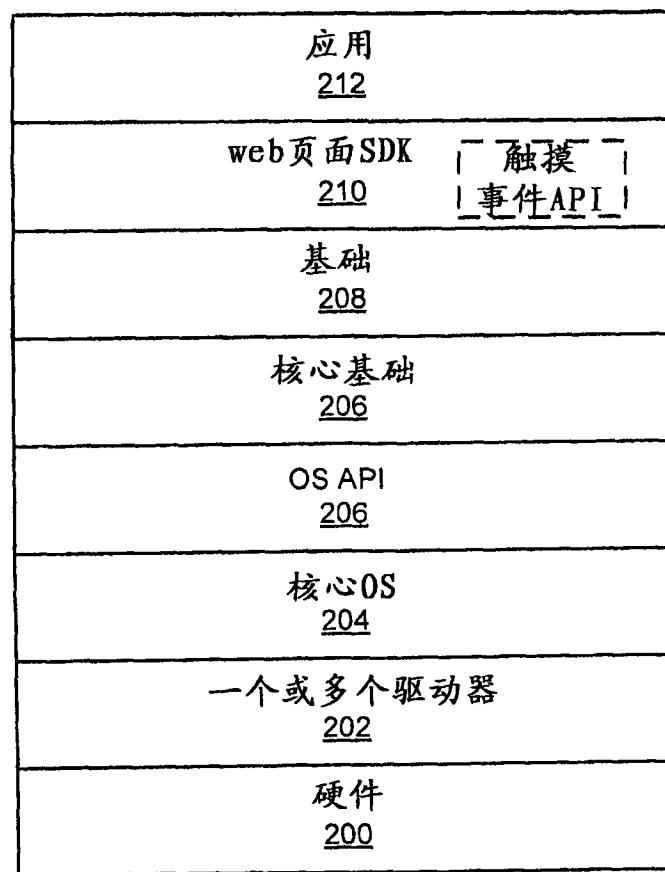


图 2

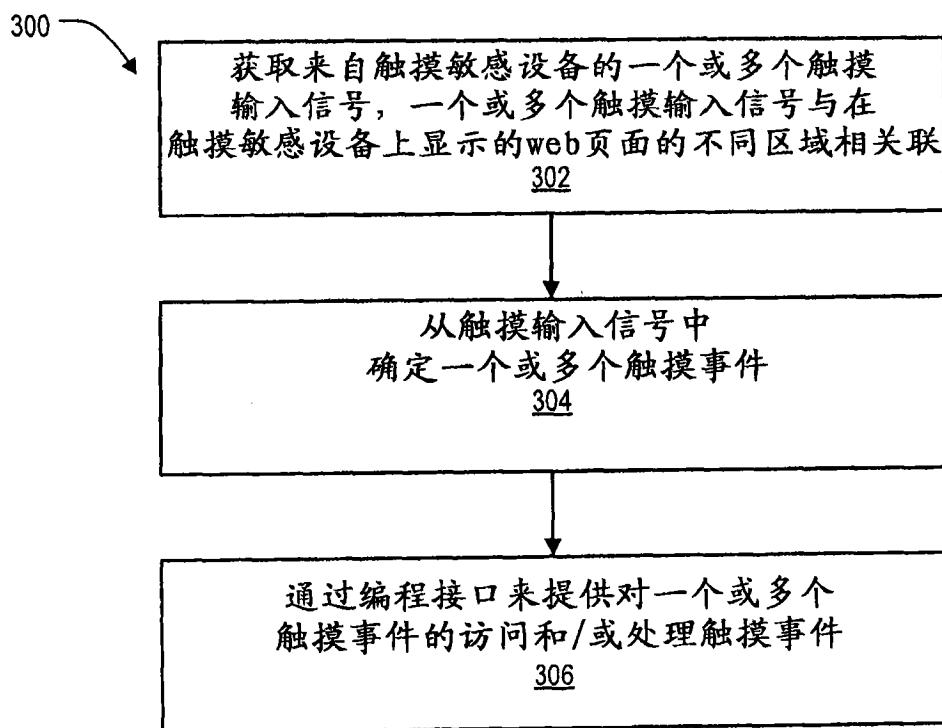


图 3

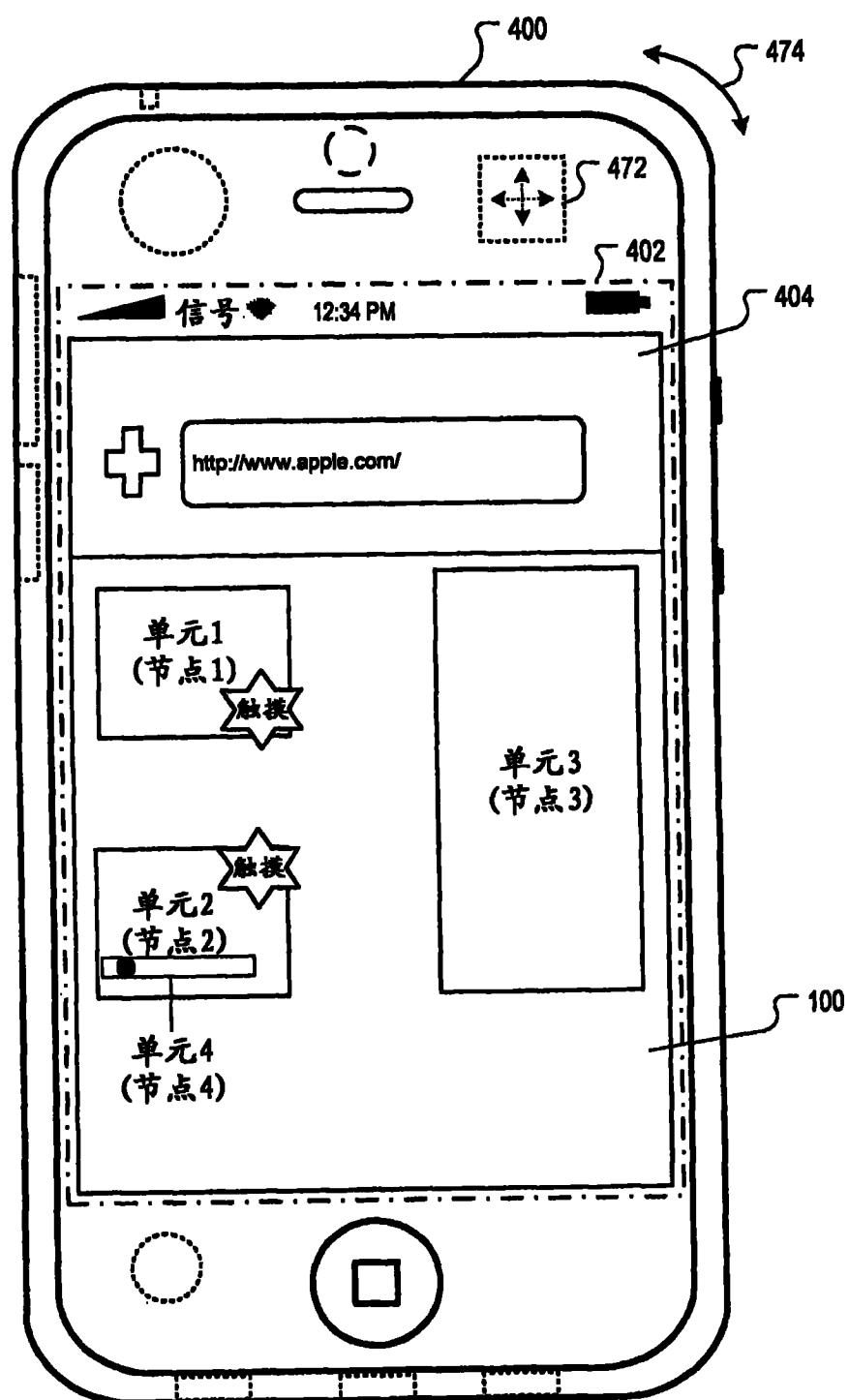


图 4

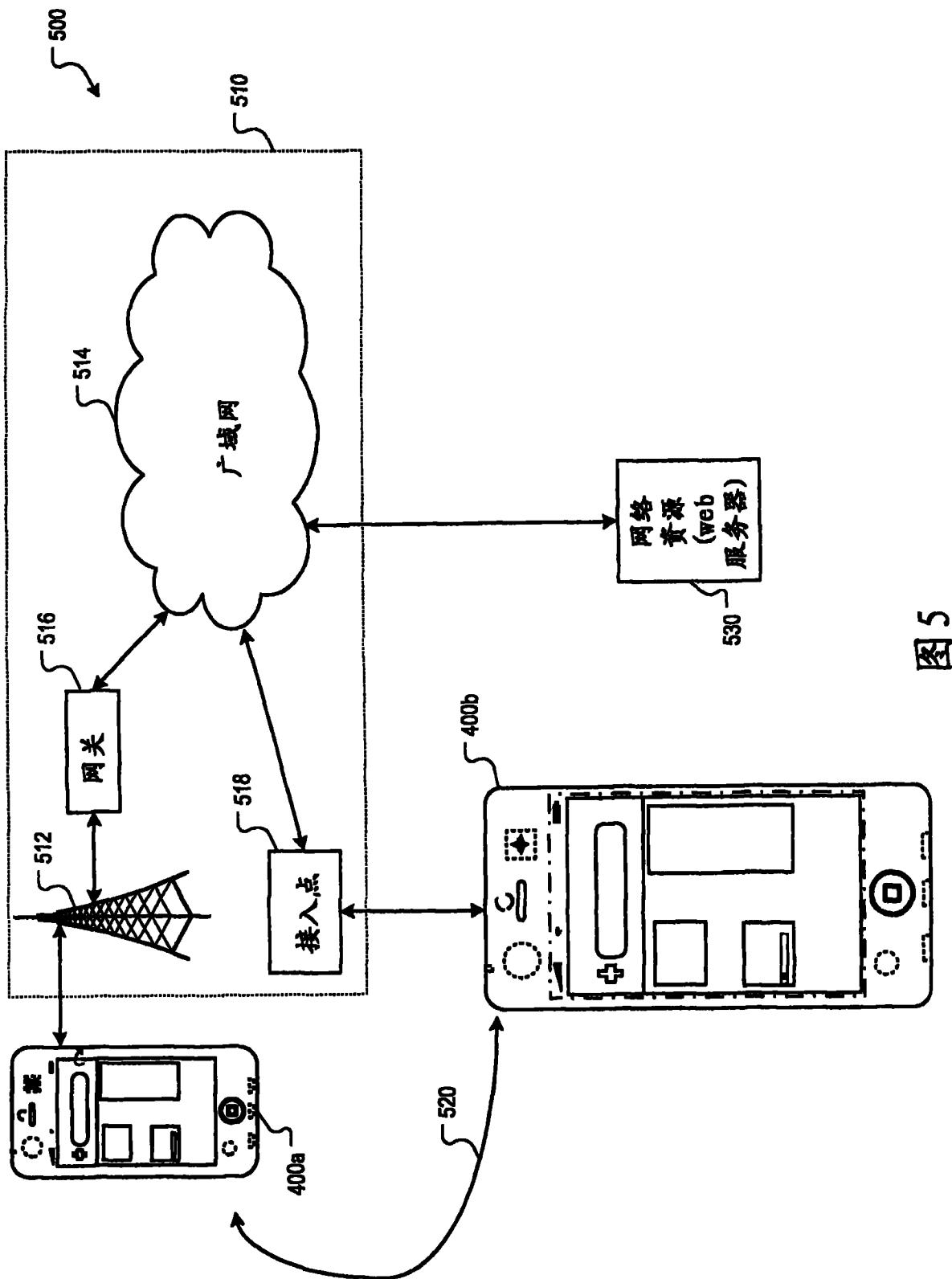


图 5

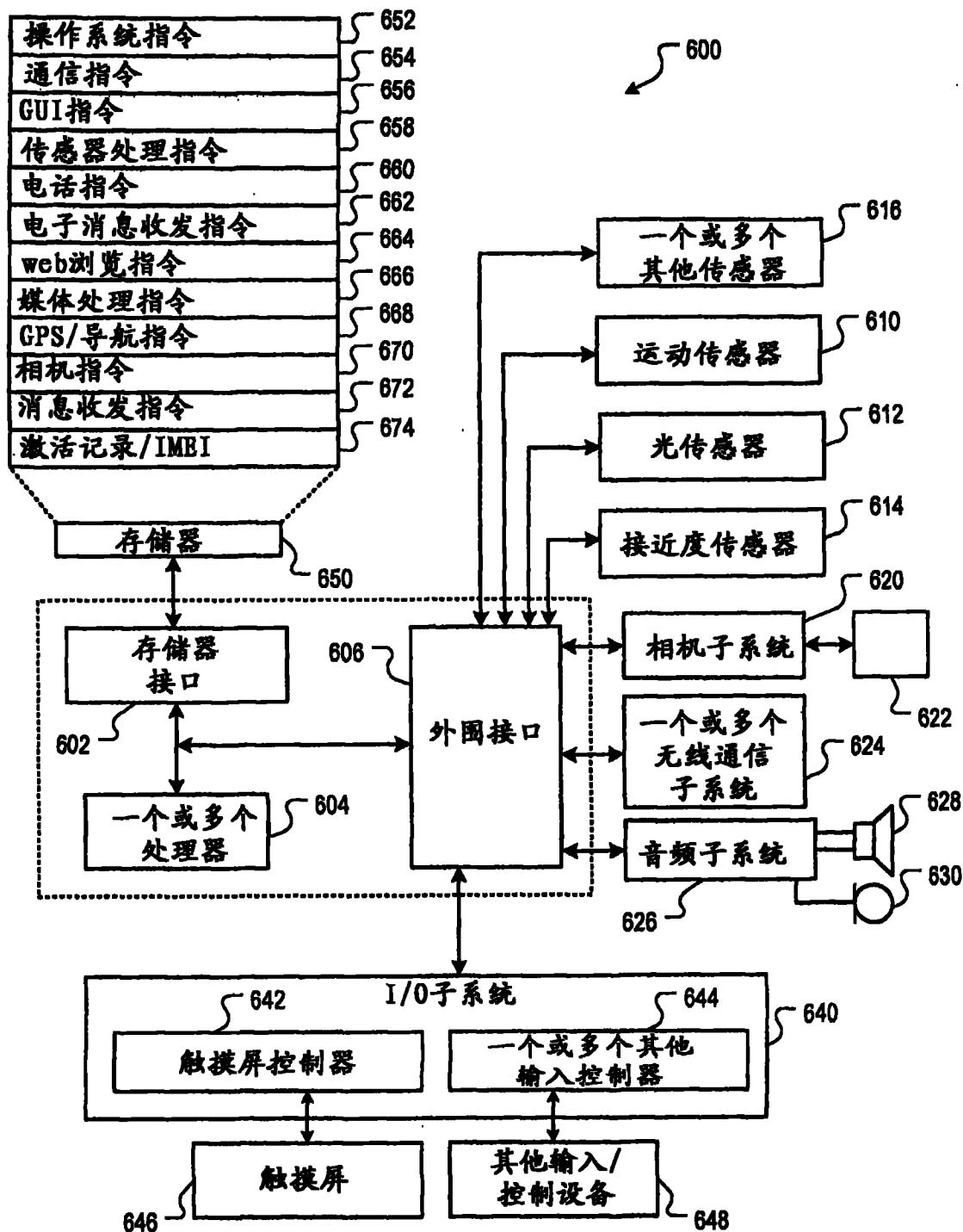


图 6