(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 103761044 B
(45)授权公告日 2018.10.16

(21)申请号 201410028534.5

(22)申请日 2009.03.03

(65)同一申请的已公布的文献号
申请公布号 CN 103761044 A

(43)申请公布日 2014.04.30

(30)优先权数据
12/042,299 2008.03.04 US

(62)分案原申请数据
200980000013.6 2009.03.03

(73)专利权人 苹果公司
地址 美国加利福尼亚州

(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 北京市金杜律师事务所 11256
代理人 王茂华

(51)Int.Cl.
G06F 3/0488(2013.01)
G06F 17/30(2006.01)

(56)对比文件
WO 2006/128248 A1,2006.12.07,说明书第2页第16行至第30行、第9页第25行至第10页第8行,图2-8.
US 2008/0028327 A1,2008.01.31,说明书第47段、第49段、第50段、第52段.
US 2006/0161871 A1,2006.07.20,全文.
CN 1841284 A,2006.10.04,全文.
WO 02/08881 A2,2002.01.31,说明书第12页第25-26行、第12页第31-33行、第13页第1行至第14页第2行、第14页第4-9行、第16页第4-17行,图2a、2b、2c、3a、3b、4a、5a、5b、9.

审查员 陈鸣

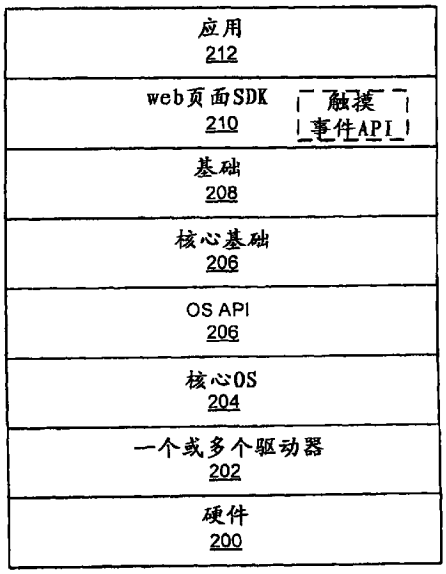
权利要求书3页 说明书16页 附图5页

(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。



CN 103761044 B

1.一种用于在触摸敏感设备上进行web浏览的方法,所述方法包括:

接收触摸事件,所述触摸事件包括多个不同触摸列表,所述多个不同触摸列表包括:

第一触摸列表,用于与所述触摸事件的目标相关联的并且同时检测到的所有触摸,

第二触摸列表,用于同时检测到的所述触摸中的所有改变的触摸,以及

第三触摸列表,用于在所述触摸敏感设备上同时检测到的所有触摸;以及

处理所述触摸事件。

2.根据权利要求1所述的方法,包括:根据所述第一触摸列表中的一个或多个触摸处理所述触摸事件。

3.根据权利要求1-2任一项所述的方法,其中所述多个不同触摸列表中的至少一个触摸列表包括用以标识web页面上的一个或多个触摸的触摸事件数据,所述触摸事件数据包括触摸标识符和至少一组触摸位置坐标,所述触摸列表还包括与每个触摸相关联的触摸事件目标,并且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。

4.根据权利要求3所述的方法,还包括:通过组合两个或更多触摸事件来生成手势事件。

5.根据权利要求4所述的方法,其中所述手势事件包括以下之一:缩放信息和旋转信息,所述缩放信息包括缩放值,并且所述旋转信息包括旋转值。

6.一种用于在触摸敏感设备上进行web浏览的设备,所述设备包括:

用于接收触摸事件的装置,所述触摸事件包括多个不同触摸列表,所述多个不同触摸列表包括:

第一触摸列表,用于与所述触摸事件的目标相关联的并且同时检测到的所有触摸,

第二触摸列表,用于同时检测到的所述触摸中的所有改变的触摸,以及

第三触摸列表,用于在所述触摸敏感设备上同时检测到的所有触摸;以及

用于处理所述触摸事件的装置。

7.根据权利要求6所述的设备,包括:用于根据所述第一触摸列表中的一个或多个触摸处理所述触摸事件的装置。

8.根据权利要求6-7任一项所述的设备,其中所述多个不同触摸列表中的至少一个触摸列表包括用以标识web页面上的一个或多个触摸的触摸事件数据,所述触摸事件数据包括触摸标识符和至少一组触摸位置坐标,所述触摸列表还包括与每个触摸相关联的触摸事件目标,并且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。

9.根据权利要求8所述的设备,包括:用于通过组合两个或更多触摸事件来生成手势事件的装置。

10.根据权利要求9所述的设备,其中所述手势事件包括以下之一:缩放信息和旋转信息,所述缩放信息包括缩放值,并且所述旋转信息包括旋转值。

11.一种用于在触摸敏感设备上进行web浏览的方法,所述方法包括:

提供触摸事件,所述触摸事件包括多个不同触摸列表,所述多个不同触摸列表包括:

第一触摸列表,用于与所述触摸事件的目标相关联的并且同时检测到的所有触摸,

第二触摸列表,用于同时检测到的所述触摸中的所有改变的触摸,以及

第三触摸列表，用于在所述触摸敏感设备上同时检测到的所有触摸；以及

利用包括超文本标记语言HTML文档的web页面中的代码来处理所述触摸事件；以及

生成所述web页面的显示。

12.根据权利要求11所述的方法，包括：接收旋转值和/或缩放值。

13.根据权利要求11-12任一项所述的方法，其中所述多个不同触摸列表中的至少一个触摸列表包括用以标识web页面上的一个或多个触摸的触摸事件数据，所述触摸事件数据包括触摸标识符和至少一组触摸位置坐标，所述触摸列表还包括与每个触摸相关联的触摸事件目标，并且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。

14.根据权利要求13所述的方法，还包括：通过组合两个或更多触摸事件来生成手势事件。

15.根据权利要求14所述的方法，其中所述手势事件包括以下之一：缩放信息和旋转信息，所述缩放信息包括缩放值，并且所述旋转信息包括旋转值。

16.一种用于在触摸敏感设备上进行web浏览的设备，所述设备包括：

用于提供触摸事件的装置，所述触摸事件包括多个不同触摸列表，所述多个不同触摸列表包括：

第一触摸列表，用于与所述触摸事件的目标相关联的并且同时检测到的所有触摸，

第二触摸列表，用于同时检测到的所述触摸中的所有改变的触摸，以及

第三触摸列表，用于在所述触摸敏感设备上同时检测到的所有触摸；以及

用于利用包括超文本标记语言HTML文档的web页面中的代码来处理所述触摸事件的装置；以及

用于生成所述web页面的显示的装置。

17.根据权利要求16所述的设备，包括：接收旋转值和/或缩放值。

18.根据权利要求16-17任一项所述的设备，其中所述多个不同触摸列表中的至少一个触摸列表包括用以标识web页面上的一个或多个触摸的触摸事件数据，所述触摸事件数据包括触摸标识符和至少一组触摸位置坐标，所述触摸列表还包括与每个触摸相关联的触摸事件目标，并且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。

19.根据权利要求18所述的设备，包括：用于通过组合两个或更多触摸事件来生成手势事件的装置。

20.根据权利要求19所述的设备，其中所述手势事件包括以下之一：缩放信息和旋转信息，所述缩放信息包括缩放值，并且所述旋转信息包括旋转值。

21.一种用于在触摸敏感设备上进行web浏览的方法，所述方法包括：

提供用于一个或多个触摸事件的接口，所述接口被配置为传递多个不同触摸列表，所述多个不同触摸列表包括：

第一触摸列表，用于与所述触摸事件的目标相关联的并且同时检测到的所有触摸，

第二触摸列表，用于同时检测到的所述触摸中的所有改变的触摸，以及

第三触摸列表，用于在所述触摸敏感设备上同时检测到的所有触摸；以及

将所述多个不同触摸列表传递给web页面以便处理。

22.根据权利要求21所述的方法，其中所述web页面包括用于处理所述一个或多个触摸事件的一个或多个指令。

23.根据权利要求21-22任一项所述的方法，其中所述web页面包括用于根据所述第一触摸列表中的一个或多个触摸来处理所述一个或多个触摸事件的一个或多个指令。

24.根据权利要求21-22任一项所述的方法，其中所述多个不同触摸列表中的至少一个触摸列表包括用以标识web页面上的一个或多个触摸的触摸事件数据，所述触摸事件数据包括触摸标识符和至少一组触摸位置坐标，所述触摸列表还包括与每个触摸相关联的触摸事件目标，并且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。

25.根据权利要求24所述的方法，还包括：通过组合两个或更多触摸事件来生成手势事件。

26.一种用于在触摸敏感设备上进行web浏览的设备，所述设备包括：

用于提供用于一个或多个触摸事件的接口的装置，所述接口被配置为传递多个不同触摸列表，所述多个不同触摸列表包括：

第一触摸列表，用于与所述触摸事件的目标相关联的并且同时检测到的所有触摸，

第二触摸列表，用于同时检测到的所述触摸中的所有改变的触摸，以及

第三触摸列表，用于在所述触摸敏感设备上同时检测到的所有触摸；以及

用于将所述多个不同触摸列表传递给web页面以便处理的装置。

27.根据权利要求26所述的设备，其中所述web页面包括用于处理所述一个或多个触摸事件的一个或多个指令。

28.根据权利要求26-27任一项所述的设备，其中所述web页面包括用于根据所述第一触摸列表中的一个或多个触摸来处理所述一个或多个触摸事件的一个或多个指令。

29.根据权利要求26-27任一项所述的设备，其中所述多个不同触摸列表中的至少一个触摸列表包括用以标识web页面上的一个或多个触摸的触摸事件数据，所述触摸事件数据包括触摸标识符和至少一组触摸位置坐标，所述触摸列表还包括与每个触摸相关联的触摸事件目标，并且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。

30.根据权利要求29所述的设备，包括：用于通过组合两个或更多触摸事件来生成手势事件的装置。

触摸事件模型编程接口

[0001] 相关申请

[0002] 本申请是国际申请日为2009年03月03日、国际申请号为PCT/US2009/035874、进入中国国家阶段日期为2009年04月02日、国家申请号为200980000013.6的发明专利申请的分案申请。

技术领域

[0003] 本主题主要涉及web浏览(web browsing,网页浏览)服务。

背景技术

[0004] 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。

[0005] 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

[0006] 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。

[0007] 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。

[0008] 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。

[0009] 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。

[0010] 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

[0011] 图1示出的是例示web页面文档。

[0012] 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。

[0013] 图3是用于处理触摸事件的例示处理的流程图。

[0014] 图4示出的是例示的具有多点触摸能力的设备。

[0015] 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。

[0016] 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

[0017] 例示的web页面结构和DOM

[0018] 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。

[0019] 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。

[0020] 图1B是与web页面100相关联的例示性DOM150。DOM150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，DOM150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。

[0021] 在某些实施方式中，通过将DOM150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DOM150一起工作，以便将动作附着于不同的触摸事件。

[0022] 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。

[0023] 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。

[0024] 在某些实施方式中，在DOM150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DOM150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。

[0025] 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。

[0026] 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。

[0027] 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。

[0028] 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。

[0029] 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。

[0030] 例示的IDL

[0031] 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®)。

[0032]
```
interface [
  Conditional=TOUCH_EVENTS,
  GenerateConstructor
] TouchEvent : UIEvent {

  void initTouchEvent(in AtomicString type,
                      in boolean canBubble,
                      in boolean cancelable,
                      in DOMWindow view,
                      in long detail,
                      in long screenX,
                      in long screenY,
                      in long clientX,
                      in long clientY,
```

[0033]

```
                in boolean ctrlKey,
                in boolean altKey,
                in boolean shiftKey,
                in boolean metaKey,
                in TouchList touches,
                in TouchList targetTouches,
                in TouchList changedTouches,
                in long scale,
                in long rotation);

    readonly attribute TouchList touches;        //所有触摸
    readonly attribute TouchList targetTouches;  //该 TouchEvent Target（触摸事件目标）中的所有触摸
    readonly attribute TouchList changedTouches; //当前事件中有改变的所有触摸

    readonly attribute long scale;
    readonly attribute long rotation;
    readonly attribute boolean ctrlKey;
    readonly attribute boolean shiftKey;
    readonly attribute boolean altKey;
    readonly attribute boolean metaKey;
};

interface [
    Conditional=TOUCH_EVENTS,
] Touch {
    readonly attribute EventTarget target;

    readonly attribute long identifier;

    readonly attribute long clientX;
    readonly attribute long clientY;
    readonly attribute long pageX;
    readonly attribute long pageY;
    readonly attribute long screenX;
    readonly attribute long screenY;
};

interface [
    Conditional=TOUCH_EVENTS,
    HasIndexGetter,
] TouchList {
    readonly attribute unsigned long  length;
    Touch         item(in unsigned long index);
};

interface [
    Conditional=TOUCH_EVENTS,
    GenerateConstructor
] GestureEvent : UIEvent {
    void initGestureEvent( in AtomicString type,
                           in boolean canBubble,
                           in boolean cancelable,
```

```
                    in DOMWindow view,
                    in long detail,
                    in long screenX,
                    in long screenY,
                    in long clientX,
                    in long clientY,
                    in boolean ctrlKey,
                    in boolean altKey,
                    in boolean shiftKey,
                    in boolean metaKey,
                    in EventTarget target,
                    in long scale,
                    in long rotation);

    readonly attribute EventTarget target;

    readonly attribute long scale;
    readonly attribute long rotation;
```

[0034]

```
    readonly attribute boolean ctrlKey;
    readonly attribute boolean shiftKey;
    readonly attribute boolean altKey;
    readonly attribute boolean metaKey;
};

In Document.idl:

Touch          createTouch(in EventTarget target,
                    in long identifier,
                    in long clientX,
                    in long clientY,
                    in long pageX,
                    in long pageY,
                    in long screenX,
                    in long screenY)
  raises (DOMException);
[Custom] TouchList      createTouchList()
  raises (DOMException);
```

[0035] 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：

[0036] this.element.addEventListener('touchstart',function(e){return self.onTouchStart(e)},false)；

[0037] this.element.addEventListener('gesturestart',function(e){return self.onGestureStart(e)},false扎

[0038] 与以上IDL相对应的HTML代码可以如下：

[0039]

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
      <meta name="viewport" content="initial-scale=1.0" />
      <title>Transform Gestures</title>
      <style type="text/css" media="screen">
        .box {
          position: absolute;
          height: 150px;
          width: 150px;
          background-color: blue;
        }

        .box:active {
          background-color: red;
        }

        body {
          margin: 0px;
        }

        #container {
          position: absolute;
          width: 100%;
          height: 100%;
        }

        #main-box2 {
          top: 10px;
          left: 155px;
          background: red;
          z-index: 1;
        }

      </style>
      <script type="text/javascript" charset="utf-8">

        var trackedObjectCount = 0;

        function Box(inElement)
        {
```

[0040]

```
        var self = this;

        this.element = inElement;

        this.scale = 1.0;
        this.rotation = 0;
        this.position = '0,0';

        this.element.addEventListener('touchstart', function(e) { return self.onTouchStart(e) }, false);
        this.element.addEventListener('gesturestart', function(e) { return self.onGestureStart(e) }, false);
    }

    Box.prototype = {
        //位置串是无单位的"x, y"
        get position()
        {
            return this._position;
        },

        set position(pos)
        {
            this._position = pos;

            var components = pos.split(',')
            var x = components[0];
            var y = components[1];

            const kUseTransform = true;
            if (kUseTransform) {
                this.element.style.webkitTransform = 'rotate(' + this.rotation + 'deg) scale(' + this.scale + ')
translate(' + x + 'px, ' + y + 'px)';
            }
            else {
                this.element.style.left = x + 'px';
                this.element.style.top = y + 'px';
            }
        },

        get x()
        {
            return parseInt(this._position.split(',')[0]);
        },

        set x(inX)
        {
            var comps = this._position.split(',');
            comps[0] = inX;
            this.position = comps.join(',');
        },

        get y()
        {
```

```
        return parseInt(this._position.split(',')[1]);
    },

    set y(inY)
    {
        var comps = this._position.split(',');
        comps[1] = inY;
        this.position = comps.join(',');
    },

    filterEvent: function(e)
    {
        // 防止浏览器执行其默认事件（滚动，缩放）
        e.preventDefault();

        // 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
        return (e.target == this.element);
    },

    onTouchStart: function(e)
    {
        if (!this.filterEvent(e))
            return false;
```

[0041]

```
        // 当第一手指落至这一单元时开始追踪
        if (e.targetTouches.length != 1)
            return false;

        this.startX = e.targetTouches[0].clientX;
        this.startY = e.targetTouches[0].clientY;

        var self = this;
        if (!("touchMoveHandler" in this)) {
            this.touchMoveHandler = function(e) { return self.onTouchMove(e) }
            this.touchEndHandler = function(e) { return self.onTouchEnd(e) }
        }

        document.addEventListener('touchmove', this.touchMoveHandler, false);
        document.addEventListener('touchend', this.touchEndHandler, false);

        trackedObjectCount++;

        return false;
    },

    onTouchMove: function(e)
    {
        if (!this.filterEvent(e))
            return false;
```

[0042] //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

```
    if (e.targetTouches.length != 1)
      return false;

    var leftDelta = e.targetTouches[0].clientX - this.startX;
    var topDelta = e.targetTouches[0].clientY - this.startY;

    var newLeft = (this.x) + leftDelta;
    var newTop = (this.y) + topDelta;

    this.position = newLeft + ',' + newTop;

    this.startX = e.targetTouches[0].clientX;
    this.startY = e.targetTouches[0].clientY;

    return false;
  },

  onTouchEnd: function(e)
  {
    if (!this.filterEvent(e))
      return false;

    // 当最后一根手指从这一单元上移开时，停止追踪
    if (e.targetTouches.length > 0)
      return false;
```

[0043]

```
    document.removeEventListener('touchmove', this.touchMoveHandler, false);
    document.removeEventListener('touchend', this.touchEndHandler, false);

    trackedObjectCount--;

    return false;
  },

  onGestureStart: function(e)
  {
    if (!this.filterEvent(e))
      return false;

    var self = this;
    if (!("gestureChangeHandler" in this)) {
      this.gestureChangeHandler = function(e) { return self.onGestureChange(e) }
      this.gestureEndHandler = function(e) { return self.onGestureEnd(e) }
    }

    document.addEventListener('gesturechange', this.gestureChangeHandler, true);
    document.addEventListener('gestureend', this.gestureEndHandler, true);

    return false;
  },

  onGestureChange: function(e)
```

[0044]

```
            {
              if (!this.filterEvent(e))
                return false;

              //在追踪一个对象时，只解释手势。否则，解释原始触摸事件
              //移动所追踪的对象
              if (trackedObjectCount == 1) {
                this.scale += e.scaling * 0.01;
                this.rotation += e.rotation / 2;
                this.position = this.position;
              }

              return false;
            },

            onGestureEnd: function(e)
            {
              if (!this.filterEvent(e))
                return false;

              document.removeEventListener('gesturechange', this.gestureChangeHandler, true);
              document.removeEventListener('gestureend', this.gestureEndHandler, true);

              return false;
            },
          }

          function loaded()
          {
            new Box(document.getElementById('main-box'));
            new Box(document.getElementById('main-box2'));
          }
          window.addEventListener('load', loaded, true);
        </script>
      </head>
    <body>
      <div id="container">
        <div id="main-box" class="box"></div>
        <div id="main-box2" class="box"></div>
      </div>
    </body>
</html>
```

[0045] 用于多点触摸设备的例示处理堆栈

[0046] 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。

[0047] 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OS204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OS204的一部分。

[0048] 一组OS应用编程接口(API)206可以与核心OS204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API208可以使用OS API206,而一组基础API210则可以使用核心基础API208。

[0049] 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SDK210中。web页面SDK210的API可以利用基础API208。Web页面SDK210例如可以包括由APPLE Inc.®提供的Web KIT(Web工具包)。Web页面SDK210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。

[0050] 在设备上运行的应用214可以利用Web页面SDK210的API来创建web页面。web页面SDK210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 API206通信。

[0051] 例示的触摸事件处理

[0052] 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)。

[0053] 移动设备综述

[0054] 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。

[0055] 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。

[0056] 例示的具有多点触摸能力的设备的功能

[0057] 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。

[0058] 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。

[0059] 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。

[0060] 网络操作环境

[0061] 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。

[0062] 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。

[0063] 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。

[0064] 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。

[0065] 例示的移动设备架构

[0066] 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。

[0067] 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的功能。

[0068] 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。

[0069] 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。

[0070] 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。

[0071] 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局限于电容性、电阻性、红外和表面声波技术，以及其他的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。

[0072] 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。

[0073] 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。

[0074] 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。

[0075] 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。

[0076] 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。

[0077] 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。

[0078] 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。

[0079] 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。

[0080] 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二者的指令和数据。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。

[0081] 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。

[0082] 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。

[0083] 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。

[0084] 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。

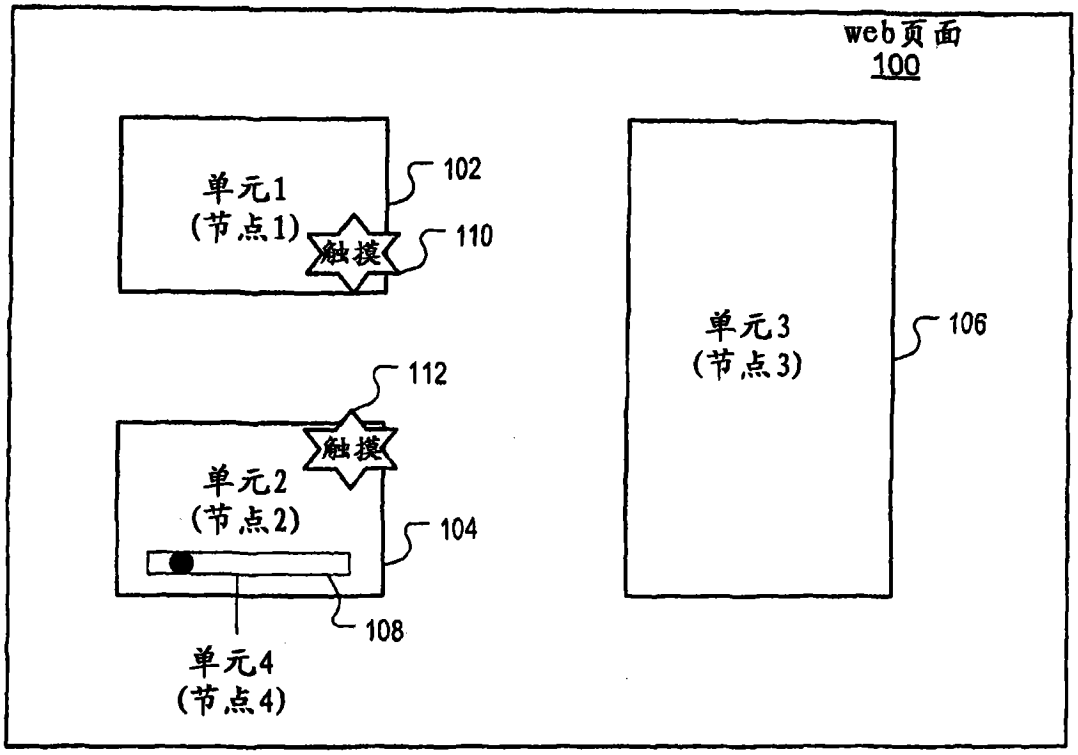

web页面
100
单元1
(节点1)
触摸
102
110
单元3
(节点3)
106
112
触摸
单元2
(节点2)
104
108
单元4
(节点4)

图1A

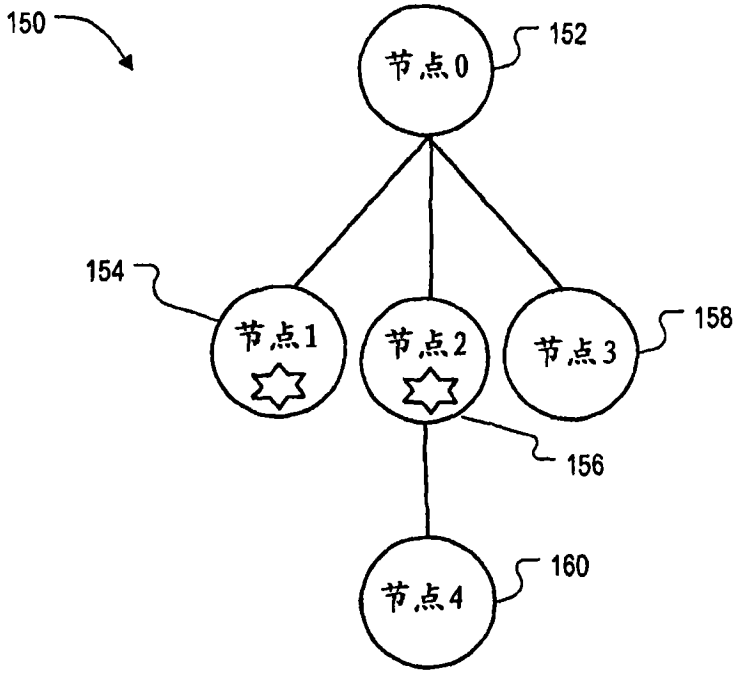

150
节点0
152
154
节点1
节点2
节点3
158
156
节点4
160

图1B

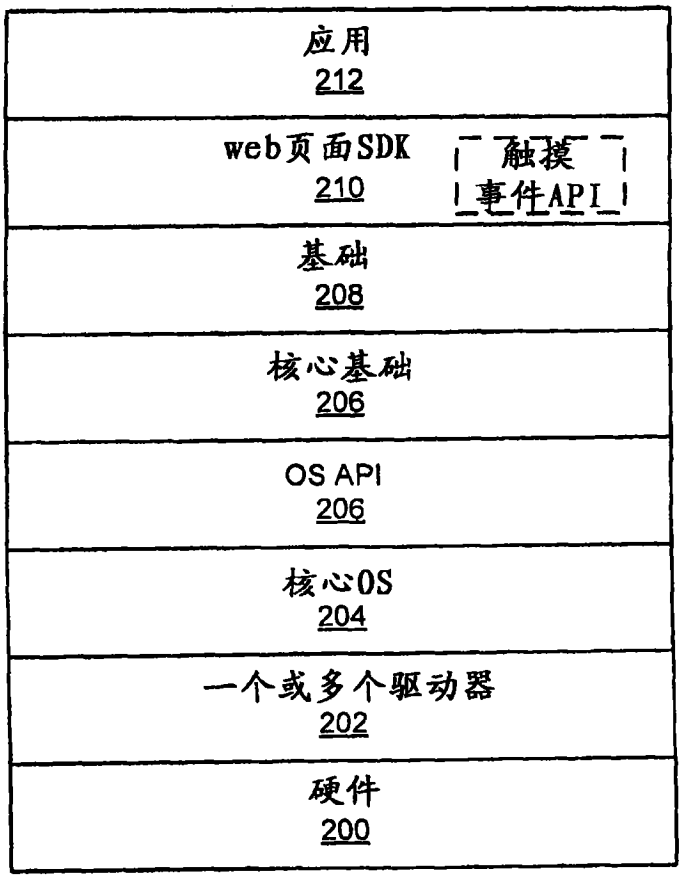

应用
212
web页面SDK
210
触摸
事件API
基础
208
核心基础
206
OS API
206
核心OS
204
一个或多个驱动器
202
硬件
200

图2

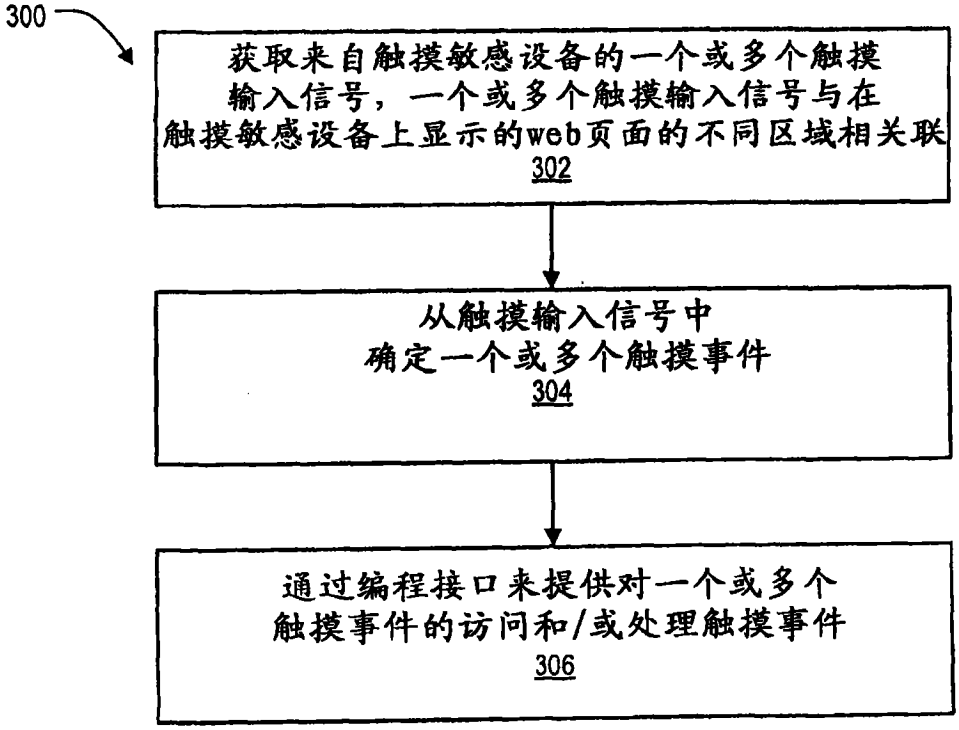

300
获取来自触摸敏感设备的一个或多个触摸输入信号，一个或多个触摸输入信号与在触摸敏感设备上显示的web页面的不同区域相关联
302
从触摸输入信号中确定一个或多个触摸事件
304
通过编程接口来提供对一个或多个触摸事件的访问和/或处理触摸事件
306

图3

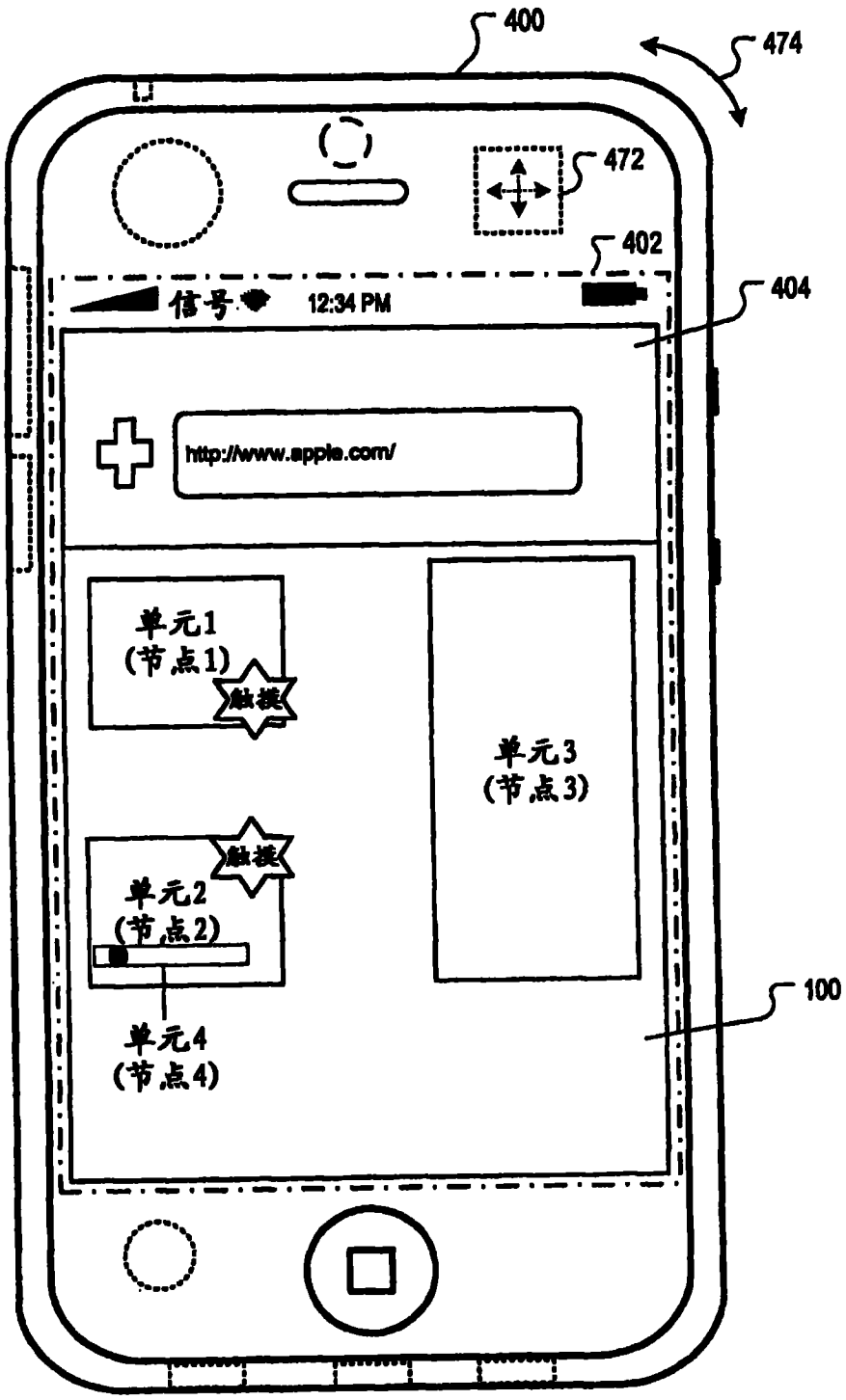

400
474
472
402
404
信号
12:34 PM
http://www.apple.com/
单元1
(节点1)
触摸
单元3
(节点3)
触摸
单元2
(节点2)
单元4
(节点4)
100

图4

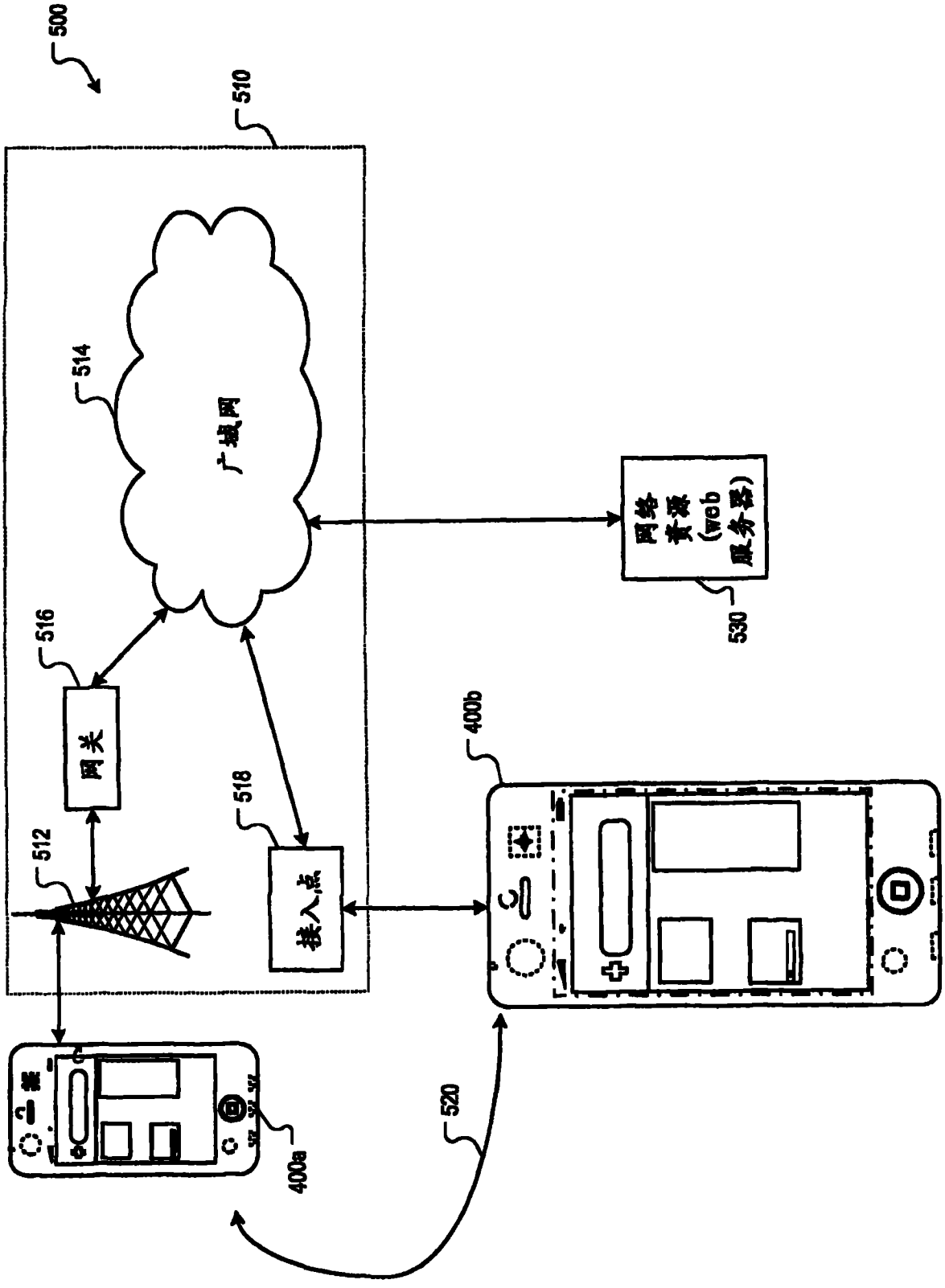

500
510
514
广域网
网络资源(web服务器)
530
516
网关
518
接入点
512
400b
520
400a

图5

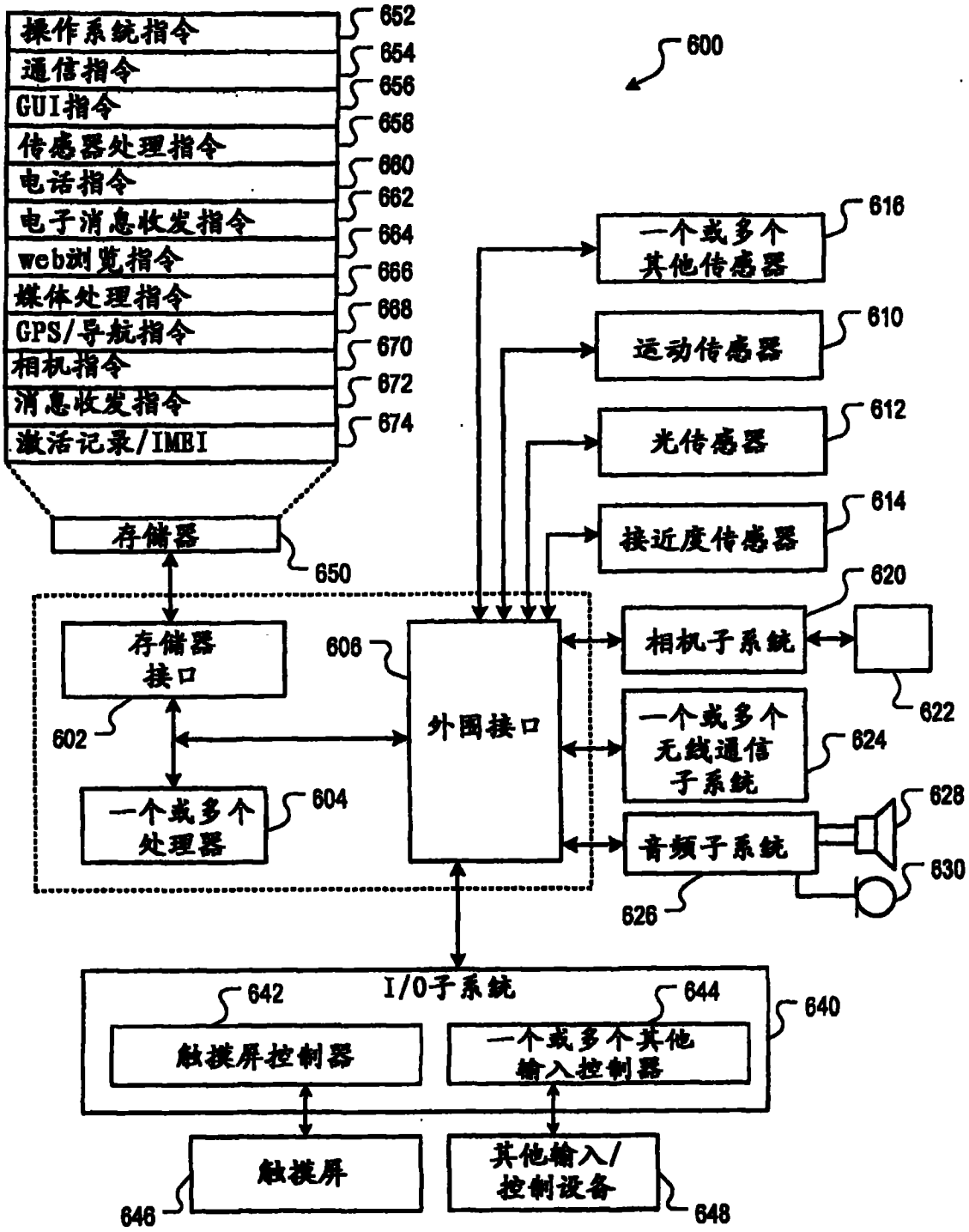

操作系统指令
通信指令
GUI指令
传感器处理指令
电话指令
电子消息收发指令
web浏览指令
媒体处理指令
GPS/导航指令
相机指令
消息收发指令
激活记录/IMEI
652
654
656
658
660
662
664
666
668
670
672
674
600
存储器
650
一个或多个其他传感器
616
运动传感器
610
光传感器
612
接近度传感器
614
相机子系统
620
622
存储器接口
602
外围接口
606
一个或多个处理器
604
一个或多个无线通信子系统
624
音频子系统
626
628
630
I/O子系统
640
642
触摸屏控制器
644
一个或多个其他输入控制器
触摸屏
646
其他输入/控制设备
648

图6