

[19] 中华人民共和国国家知识产权局



[12] 发明专利申请公布说明书

[21] 申请号 200910118596.4

[51] Int. Cl.

G06F 3/048 (2006.01)

G06F 3/041 (2006.01)

H04M 1/247 (2006.01)

[43] 公开日 2009 年 9 月 9 日

[11] 公开号 CN 101526880A

[22] 申请日 2009.3.4

[21] 申请号 200910118596.4

[30] 优先权

[32] 2008. 3. 4 [33] US [31] 12/042,318

[71] 申请人 苹果公司

地址 美国加利福尼亚

[72] 发明人 J·C·比弗 A·普拉齐

[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

代理人 李 玲

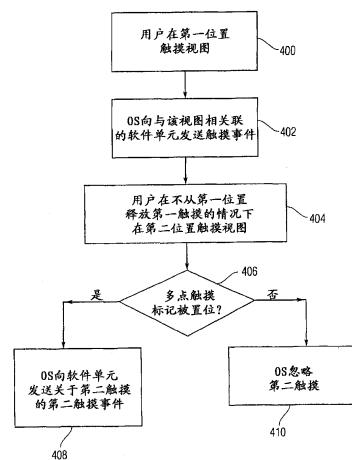
权利要求书 14 页 说明书 17 页 附图 6 页

[54] 发明名称

触摸事件模型

[57] 摘要

本发明涉及触摸事件模型。本发明的实施例涉及的是为应用级软件来定义触摸事件的方法、软件、设备和 API。此外，某些实施例旨在为在多点触摸使能设备中运行的应用简化对单个和多个触摸事件的识别。为了简化对单个和多个触摸事件的识别，特定窗口内部的每一个视图都可以被配置成多点触摸视图或是单点触摸视图。此外，每一个视图都可以被配置成独占或是非独占视图。根据视图的配置，可以忽略或是识别在该视图和其他视图中的触摸事件。被忽略的触摸就无需被发送到应用。通过有选择地忽略触摸，就能让那些没有利用先进的多点触摸特征的较为简单的软件单元与更为复杂的软件单元在相同设备上同时运行。



1. 一种用于在多点触摸设备上处理触摸事件的方法，包括：

显示一个或多个视图；

执行一个或多个软件单元，每一个软件单元各自与一个特定视图相关联；

将多点触摸标记或独占触摸标记与每一个视图相关联；

在所述一个或多个视图上接收一个或多个触摸；以及

根据多点触摸标记和独占触摸标记的值，有选择地向与其上接收到触摸的所述一个或多个视图相关联的一个或多个软件单元发送一个或多个触摸事件，每一个触摸事件描述所接收到的触摸。

2. 根据权利要求 1 所述的方法，还包括：

如果多点触摸标记与特定视图相关联，则允许向与其他视图相关联的软件单元发送与在该特定视图上接收的触摸事件同时发生的其他触摸事件。

3. 根据权利要求 1 所述的方法，其中如果多点触摸标记与特定视图相关联，则所述多点触摸标记表明是否允许与该特定视图相关联的软件单元处理位于该视图中的多个同时发生的触摸。

4. 根据权利要求 1 所述的方法，其中当在具有被断言的独占触摸标记的视图上接收触摸时，该独占触摸标记防止将触摸事件发送到与具有所断言的独占触摸标记的视图以外的视图相关联的软件单元。

5. 根据权利要求 1 所述的方法，其中多点触摸设备是移动电话。

6. 根据权利要求 1 所述的方法，其中多点触摸设备是数字媒体播放器。

7. 根据权利要求 1 所述的方法，包括：

将多点触摸标记与第一视图相关联；

在第一视图上接收第一触摸，该第一视图是所述一个或多个视图中的一个；

向第一软件单元发送描述第一触摸的触摸事件，该第一软件单元

是所述一个或多个软件单元中的一个并且与第一视图相关联；

确定所述与第一视图相关联的多点触摸标记是否表明第一视图是多点触摸视图；以及

如果第一视图不是多点触摸视图，则阻止描述位于第一视图中的任何其他触摸的所有触摸事件，直至不再接收到第一触摸。

8. 根据权利要求 7 所述的方法，还包括：

将独占触摸标记与所述一个或多个视图中的每一个相关联；

确定所述与第一视图相关联的独占触摸标记是否表明第一视图是独占触摸视图；以及

如果第一视图是独占触摸视图，则阻止描述位于除第一视图之外的任何视图中的任何其他触摸的所有触摸事件，直至不再接收到第一触摸。

9. 根据权利要求 8 所述的方法，其中第一视图不是独占触摸视图，该方法还包括：

在多点触摸面板上接收第二触摸，该第二触摸位于第二视图并且与第二软件单元相关联；

确定所述与第二视图相关联的独占触摸标记是否表明第二视图是独占触摸视图；以及

如果第二视图是独占触摸视图，则防止向第二软件单元发送与第二触摸相关联的触摸事件，直至不再接收到第一触摸。

10. 根据权利要求 9 所述的方法，还包括：

如果第二视图不是独占触摸视图，则向第二软件单元发送描述第二触摸的触摸事件。

11. 一种用于在多点触摸设备上识别一个或多个触摸事件的方法，包括：

定义一个或多个视图；

将独占触摸或多点触摸标记指定给每一个视图；以及

根据每一个视图的独占触摸或多点触摸标记，接受在每一个视图中检测到的一个或多个触摸事件。

12. 一种包含被配置成在多点触摸设备上执行的多个指令的计算机可读介质，该指令被配置成使多点触摸设备：

显示一个或多个视图；

执行一个或多个软件单元，每一个软件单元各自与一个特定视图相关联；

将多点触摸标记或独占触摸标记与每一个视图相关联；

在所述一个或多个视图上接收一个或多个触摸；以及

根据多点触摸标记和独占触摸标记的值，有选择地向与其上接收到触摸的视图相关联的一个或多个软件单元发送一个或多个触摸事件，每一个触摸事件描述所接收到的触摸。

13. 根据权利要求 12 所述的计算机可读介质，其中该指令还被配置成使多点触摸设备：

如果多点触摸标记与特定视图相关联，则允许向与其他视图相关联的软件单元发送与在该特定视图上接收的触摸事件同时发生的其他触摸事件。

14. 根据权利要求 12 所述的计算机可读介质，其中如果多点触摸标记与特定视图相关联，则所述多点触摸标记表明是否允许与该特定视图相关联的软件单元处理位于该视图中的多个同时发生的触摸。

15. 根据权利要求 12 所述的计算机可读介质，其中当在具有被断言的独占触摸标记的视图上接收触摸时，该独占触摸标记防止将触摸事件发送到与具有所断言的独占触摸标记的视图以外的视图相关联的软件单元。

16. 根据权利要求 12 所述的计算机可读介质，其中多点触摸设备是移动电话。

17. 根据权利要求 12 所述的计算机可读介质，其中多点触摸设备是数字媒体播放器。

18. 根据权利要求 12 所述的计算机可读介质，其中该指令还被配置成使多点触摸设备：

将多点触摸标记与第一视图相关联；

在第一视图上接收第一触摸，该第一视图是所述一个或多个视图中的一个；

向第一软件单元发送描述第一触摸的触摸事件，该第一软件单元是所述一个或多个软件单元中的一个并且与第一视图相关联；

确定所述与第一视图相关联的多点触摸标记是否表明第一视图是多点触摸视图；以及

如果第一视图不是多点触摸视图，则阻止描述位于第一视图中的任何其他触摸的所有触摸事件，直至不再接收到第一触摸。

19. 根据权利要求 18 所述的计算机可读介质，其中该指令还被配置成使多点触摸设备：

将独占触摸标记与所述一个或多个视图中的每一个相关联；

确定所述与第一视图相关联的独占触摸标记是否表明第一视图是独占触摸视图；以及

如果第一视图是独占触摸视图，则阻止描述位于除第一视图之外的任何视图中的任何其他触摸的所有触摸事件，直至不再接收到第一触摸。

20. 根据权利要求 19 所述的计算机可读介质，其中第一视图不是独占触摸视图，并且该指令还被配置成使多点触摸设备：

在多点触摸面板上接收第二触摸，该第二触摸位于第二视图并且与第二软件单元相关联；

确定所述与第二视图相关联的独占触摸标记是否表明第二视图是独占触摸视图；以及

如果第二视图是独占触摸视图，则防止向第二软件单元发送与第二触摸相关联的触摸事件，直至不再接收到第一触摸。

21. 根据权利要求 20 所述的计算机可读介质，其中该指令还被配置成使多点触摸设备：

如果第二视图不是独占触摸视图，则向第二软件单元发送描述第二触摸的触摸事件。

22. 一种包含被配置成在多点触摸设备上执行的多个指令的计算

机可读介质，该指令被配置成使多点触摸设备：

    定义一个或多个视图；

    将独占触摸或多点触摸标记指定给每一个视图；以及

    根据每一个视图的独占触摸或多点触摸标记，接受在每一个视图中检测到的一个或多个触摸事件。

23. 一种用于在多点设备上识别点事件的方法，包括：

    显示一个或多个视图；

    执行一个或多个软件单元，每一个软件单元各自与一个特定视图相关联；

    将多点标记或独占点标记与每一个视图相关联；

    在所述一个或多个视图上接收一个或多个点输入；以及

    根据多点标记和独占点标记的值，有选择地向与其上接收到点输入的视图相关联的一个或多个软件单元发送一个或多个点事件，每一个点事件描述所接收到的点输入。

24. 一种用于在多点设备上识别一个或多个点事件的方法，包括：

    定义一个或多个视图；

    将独占点或多点标记指定给每一个视图；以及

    根据每一个视图的独占点或多点标记，接受在每一个视图中检测到的一个或多个点事件。

25. 一种包含被配置成在多点设备上执行的多个指令的计算机可读介质，该指令被配置成使多点设备：

    显示一个或多个视图；

    执行一个或多个软件单元，每一个软件单元各自与一个特定视图相关联；

    将多点标记或独占点标记与每一个视图相关联；

    在所述一个或多个视图上接收一个或多个点输入；以及

    根据多点标记和独占点标记的值，有选择地向与其上接收到点输入的视图相关联的一个或多个软件单元发送一个或多个点事件，每一个点事件都描述所接收到的点输入。

26. 一种包含被配置成在多点设备上执行的多个指令的计算机可读介质，该指令被配置成使多点设备：

    定义一个或多个视图；

    将独占点或多点标记指定给每一个视图；以及

    根据每一个视图的独占点或多点标记，接受在每一个视图中检测到的一个或多个点事件。

27. 一种多点触摸使能移动电话，该移动电话包括计算机可读介质，该计算机可读介质包含被配置成在该移动电话上执行的多个指令，该指令被配置成使该移动电话：

    显示一个或多个视图；

    执行一个或多个软件单元，每一个软件单元各自与一个特定视图相关联；

    将多点触摸标记或独占触摸标记与每一个视图相关联；

    在所述一个或多个视图上接收一个或多个触摸；以及

    根据多点触摸标记和独占触摸标记的值，有选择地向与其上接收到触摸的视图相关联的一个或多个软件单元发送一个或多个触摸事件，每一个触摸事件描述所接收到的触摸。

28. 一种多点触摸使能数字媒体播放器，该数字媒体播放器包括计算机可读介质，该计算机可读介质包含被配置成在该数字媒体播放器上执行的多个指令，该指令被配置成使该数字媒体播放器：

    显示一个或多个视图；

    执行一个或多个软件单元，每一个软件单元各自与一个特定视图相关联；

    将多点触摸标记或独占触摸标记与每一个视图相关联；

    在所述一个或多个视图上接收一个或多个触摸；以及

    根据多点触摸标记和独占触摸标记的值，有选择地向与其上接收到触摸的视图相关联的一个或多个软件单元发送一个或多个触摸事件，每一个触摸事件描述所接收到的触摸。

29. 一种包含触摸数据结构的计算机可读介质，所述触摸数据结

构定义在多点触摸面板上已经接收或正在接收的单个触摸在特定时间的状态，该触摸数据结构包括：

阶段字段，用于定义触摸在该特定时间的阶段；

视图字段，用于表明接收到或正在接收触摸的视图；以及

位置字段，用于表明当前正在接收触摸的位置。

30. 根据权利要求 29 所述的计算机可读介质，还包括：用于表明在多点触摸面板上正在或已经被同时接收的两个或更多个触摸在特定时间的状态的触摸事件数据结构，该触摸事件数据结构包括多个如权利要求 29 所述的触摸数据结构，每一个数据结构都与所述两个或更多个触摸中的一个相关联。

31. 根据权利要求 29 所述的计算机可读介质，其中该触摸数据结构还包括：

时间字段，用于表明所述特定时间；

窗口字段，用于表明接收到或正在接收触摸的窗口；

先前位置字段，用于表明先前接收到触摸的位置；以及

视图首次触摸字段，用于表明该触摸是接收到或正在接收该触摸的视图首次接收的触摸。

32. 根据权利要求 29 所述的计算机可读介质，其中所述触摸数据结构还包括信息字段，用于表明当前触摸是否包含轻扫运动，并且如果包含的话，则该信息字段表明该运动所依照的大体方向。

33. 根据权利要求 29 所述的计算机可读介质，其中所述阶段字段保持用于表明该触摸是在所述特定时间已经启动的新触摸的值。

34. 根据权利要求 29 所述的计算机可读介质，其中所述阶段字段保持用于表明该触摸已经从先前位置移动到新位置的值。

35. 根据权利要求 29 所述的计算机可读介质，其中所述阶段字段保持用于表明该触摸从发布关于该触摸的先前数据结构时起即保持固定的值。

36. 根据权利要求 29 所述的计算机可读介质，还包括一个轻击计数字段，用于表明在触摸位置顺序执行的短击的次数。

37. 根据权利要求 29 所述的计算机可读介质，其中所述阶段字段保持用于表明触摸已经结束的字段。

38. 根据权利要求 29 所述的计算机可读介质，其中所述阶段字段保持用于表明该触摸已经被多点触摸面板或是包含多点触摸面板的设备中的另一个部件取消的值。

39. 一种包含多点触摸面板和计算机可读介质的设备，其中该计算机可读介质包括触摸数据结构，该触摸数据结构定义在多点触摸面板上已经接收或正在接收的单个触摸在特定时间的状态，该触摸数据结构包括：

阶段字段，用于定义触摸在该特定时间的阶段；

视图字段，用于表明接收到或正在接收触摸的视图；以及

位置字段，用于表明当前正在接收触摸的位置。

40. 一种用于操作多点触摸使能设备的方法，包括：

执行代表视图的软件单元，该软件单元在显示器上显示该视图的视觉表示；

在多点触摸面板上检测触摸；以及

产生用于定义触摸在特定时间的状态的数据结构，该数据结构包括：

阶段字段，用于定义触摸在该特定时间的阶段；

视图字段，用于表明接收到或正在接收触摸的视图；以及

位置字段，用于表明当前正在接收触摸的位置。

41. 一种在具有包含显示器和多点触摸面板的设备以及与用户界面软件交互的应用软件的环境中，通过应用编程接口 API 来执行操作的方法，所述方法包括：

在多点触摸面板上检测触摸；

产生用于定义触摸在特定时间的状态的数据结构，该数据结构包括：

阶段字段，用于定义触摸在该特定时间的阶段；

视图字段，用于表明接收到或正在接收触摸的视图；以及

位置字段，用于表明当前正在接收触摸的位置；以及由用户界面软件将该数据结构发送到应用软件。

42. 一种用于在多点触摸设备上处理触摸事件的装置，包括：

用于显示一个或多个视图的部件；

用于执行一个或多个软件单元的部件，每一个软件单元各自与一个特定视图相关联；

用于将多点触摸标记或独占触摸标记与每一个视图相关联的部件；

用于在所述一个或多个视图上接收一个或多个触摸的部件；以及

用于根据多点触摸标记和独占触摸标记的值，有选择地向与其上接收到触摸的所述一个或多个视图相关联的一个或多个软件单元发送一个或多个触摸事件的部件，每一个触摸事件描述所接收到的触摸。

43. 根据权利要求 42 所述的装置，还包括：

用于如果多点触摸标记与特定视图相关联，则允许向与其他视图相关联的软件单元发送与在该特定视图上接收的触摸事件同时发生的其他触摸事件的部件。

44. 根据权利要求 42 所述的装置，其中如果多点触摸标记与特定视图相关联，则所述多点触摸标记表明是否允许与该特定视图相关联的软件单元处理位于该视图中的多个同时发生的触摸。

45. 根据权利要求 42 所述的装置，其中当在具有被断言的独占触摸标记的视图上接收触摸时，该独占触摸标记防止将触摸事件发送到与具有所断言的独占触摸标记的视图以外的视图相关联的软件单元。

46. 根据权利要求 42 所述的装置，其中多点触摸设备是移动电话。

47. 根据权利要求 42 所述的装置，其中多点触摸设备是数字媒体播放器。

48. 根据权利要求 42 所述的装置，包括：

用于将多点触摸标记与第一视图相关联的部件；

用于在第一视图上接收第一触摸的部件，该第一视图是所述一个或多个视图中的一个；

用于向第一软件单元发送描述第一触摸的触摸事件的部件，该第一软件单元是所述一个或多个软件单元中的一个并且与第一视图相关联；

用于确定所述与第一视图相关联的多点触摸标记是否表明第一视图是多点触摸视图的部件；以及

用于如果第一视图不是多点触摸视图，则阻止描述位于第一视图中的任何其他触摸的所有触摸事件，直至不再接收到第一触摸的部件。

49. 根据权利要求 48 所述的装置，还包括：

用于将独占触摸标记与所述一个或多个视图中的每一个相关联的部件；

用于确定所述与第一视图相关联的独占触摸标记是否表明第一视图是独占触摸视图的部件；以及

用于如果第一视图是独占触摸视图，则阻止描述位于除第一视图之外的任何视图中的任何其他触摸的所有触摸事件，直至不再接收到第一触摸的部件。

50. 根据权利要求 49 所述的装置，其中第一视图不是独占触摸视图，该装置还包括：

用于在多点触摸面板上接收第二触摸的部件，该第二触摸位于第二视图并且与第二软件单元相关联；

用于确定所述与第二视图相关联的独占触摸标记是否表明第二视图是独占触摸视图的部件；以及

用于如果第二视图是独占触摸视图，则防止向第二软件单元发送与第二触摸相关联的触摸事件，直至不再接收到第一触摸的部件。

51. 根据权利要求 50 所述的装置，还包括：

用于如果第二视图不是独占触摸视图，则向第二软件单元发送描述第二触摸的触摸事件的部件。

52. 一种用于在多点触摸设备上识别一个或多个触摸事件的装

置，包括：

用于定义一个或多个视图的部件；  
用于将独占触摸或多点触摸标记指定给每一个视图的部件；以及  
用于根据每一个视图的独占触摸或多点触摸标记，接受在每一个视图中检测到的一个或多个触摸事件的部件。

53. 一种用于在多点设备上识别点事件的装置，包括：

用于显示一个或多个视图的部件；  
用于执行一个或多个软件单元的部件，每一个软件单元各自与一个特定视图相关联；  
用于将多点标记或独占点标记与每一个视图相关联的部件；  
用于在所述一个或多个视图上接收一个或多个点输入的部件；以及

用于根据多点标记和独占点标记的值，有选择地向与其上接收到点输入的视图相关联的一个或多个软件单元发送一个或多个点事件的部件，每一个点事件描述所接收到的点输入。

54. 一种用于在多点设备上识别一个或多个点事件的装置，包括：

用于定义一个或多个视图的部件；  
用于将独占点或多点标记指定给每一个视图的部件；以及  
用于根据每一个视图的独占点或多点标记，接受在每一个视图中检测到的一个或多个点事件的部件。

55. 一种用于操作多点触摸使能设备的装置，包括：

用于执行代表视图的软件单元的部件，该软件单元在显示器上显示该视图的视觉表示；  
用于在多点触摸面板上检测触摸的部件；以及  
用于产生用于定义触摸在特定时间的状态的数据结构的部件，该数据结构包括：

阶段字段，用于定义触摸在该特定时间的阶段；

视图字段，用于表明接收到或正在接收触摸的视图；以及  
位置字段，用于表明当前正在接收触摸的位置。

56. 一种在具有包含显示器和多点触摸面板的设备以及与用户界面软件交互的应用软件的环境中，通过应用编程接口 API 来执行操作的装置，所述装置包括：

用于在多点触摸面板上检测触摸的部件；

用于产生用于定义触摸在特定时间的状态的数据结构的部件，该数据结构包括：

阶段字段，用于定义触摸在该特定时间的阶段；

视图字段，用于表明接收到或正在接收触摸的视图；以及

位置字段，用于表明当前正在接收触摸的位置；以及

用于由用户界面软件将该数据结构发送到应用软件的部件。

57. 一种多点触摸使能移动电话，该移动电话包括存储器，该存储器包含被配置成在该移动电话上执行的多个模块，所述模块包括：

被配置成使移动电话显示一个或多个视图的模块；

被配置成使移动电话执行一个或多个软件单元的模块，每一个软件单元各自与一个特定视图相关联；

被配置成使移动电话将多点触摸标记或独占触摸标记与每一个视图相关联的模块；

被配置成使移动电话在所述一个或多个视图上接收一个或多个触摸的模块；以及

被配置成使移动电话根据多点触摸标记和独占触摸标记的值，有选择地向与其上接收到触摸的视图相关联的一个或多个软件单元发送一个或多个触摸事件的模块，每一个触摸事件描述所接收到的触摸。

58. 一种多点触摸使能数字媒体播放器，该数字媒体播放器包括存储器，该存储器包含被配置成在该数字媒体播放器上执行的多个模块，所述模块包括：

被配置成使数字媒体播放器显示一个或多个视图的模块；

被配置成使数字媒体播放器执行一个或多个软件单元的模块，每一个软件单元各自与一个特定视图相关联；

被配置成使数字媒体播放器将多点触摸标记或独占触摸标记与

每一个视图相关联的模块；

被配置成使数字媒体播放器在所述一个或多个视图上接收一个或多个触摸的模块；以及

被配置成使数字媒体播放器根据多点触摸标记和独占触摸标记的值，有选择地向与其上接收到触摸的视图相关联的一个或多个软件单元发送一个或多个触摸事件的模块，每一个触摸事件描述所接收到的触摸。

59. 一种制造包含触摸数据结构的计算机可读介质的方法，所述触摸数据结构定义在多点触摸面板上已经接收或正在接收的单个触摸在特定时间的状态，所述方法包括：

使所述触摸数据结构包括阶段字段，用于定义触摸在该特定时间的阶段；

使所述触摸数据结构包括视图字段，用于表明接收到或正在接收触摸的视图；以及

使所述触摸数据结构包括位置字段，用于表明当前正在接收触摸的位置。

60. 根据权利要求 59 所述的方法，还包括：用于表明在多点触摸面板上正在或已经被同时接收的两个或更多个触摸在特定时间的状态的触摸事件数据结构，该触摸事件数据结构包括多个如权利要求 59 所述的触摸数据结构，每一个数据结构都与所述两个或更多个触摸中的一个相关联。

61. 根据权利要求 59 所述的方法，还包括：

使所述触摸数据结构包括时间字段，用于表明所述特定时间；

使所述触摸数据结构包括窗口字段，用于表明接收到或正在接收触摸的窗口；

使所述触摸数据结构包括先前位置字段，用于表明先前接收到触摸的位置；以及

使所述触摸数据结构包括视图首次触摸字段，用于表明该触摸是接收到或正在接收该触摸的视图首次接收的触摸。

62. 根据权利要求 59 所述的方法，还包括：

使所述触摸数据结构包括信息字段，用于表明当前触摸是否包含轻扫运动，并且如果包含的话，则该信息字段表明该运动所依照的大体方向。

63. 根据权利要求 59 所述的方法，其中所述阶段字段保持用于表明该触摸是在所述特定时间已经启动的新触摸的值。

64. 根据权利要求 59 所述的方法，其中所述阶段字段保持用于表明该触摸已经从先前位置移动到新位置的值。

65. 根据权利要求 59 所述的方法，其中所述阶段字段保持用于表明该触摸从发布关于该触摸的先前数据结构时起即保持固定的值。

66. 根据权利要求 59 所述的方法，还包括：

使所述触摸数据结构包括一个轻击计数字段，用于表明在触摸位置顺序执行的短击的次数。

67. 根据权利要求 59 所述的方法，其中所述阶段字段保持用于表明触摸已经结束的字段。

68. 根据权利要求 59 所述的方法，其中所述阶段字段保持用于表明该触摸已经被多点触摸面板或是包含多点触摸面板的设备中的另一个部件取消的值。

69. 一种制造包含多点触摸面板和计算机可读介质的方法，其中该计算机可读介质包括触摸数据结构，该触摸数据结构定义在多点触摸面板上已经接收或正在接收的单个触摸在特定时间的状态，所述方法包括：

使所述触摸数据结构包括阶段字段，用于定义触摸在该特定时间的阶段；

使所述触摸数据结构包括视图字段，用于表明接收到或正在接收触摸的视图；以及

使所述触摸数据结构包括位置字段，用于表明当前正在接收触摸的位置。

## 触摸事件模型

### 技术领域

本申请一般涉及多点和多点触摸使能设备（multi-point and multi-touch enabled device），尤其涉及在多点和多点触摸使能设备中对单个和多个点击及触摸事件进行识别。

### 背景技术

在本领域中，多点触摸使能设备（或称多重触摸使能设备，multi-touch enabled device）是已知的。多点触摸使能设备是一种能够同时感测多个触摸的设备。于是，多点触摸使能设备可以例如感测在多点触摸面板（multi-touch panel）上的两个不同位置同时发生的由两根手指按压面板所引起的两个触摸事件。在 2007 年 1 月 3 日提交的名为“PROXIMITY AND MULTI-TOUCH SENSOR DETECTION AND DEMODULATION”的美国专利申请 No.11/649,998 中论述了多点触摸使能设备的示例，由此该申请在这里全部引入作为参考。而多点使能设备（multi-point enabled device）限定的是一组更广泛的设备，这其中包括多点触摸使能设备以及诸如在上述美国专利申请 11/649,998 中讨论的多接近度传感器设备（multi-proximity sensor device）的类似设备。

虽然对于多点触摸使能界面（multi-touch enabled interface）的益处是已知的，但是这些设备却会带来某些界面设计上的挑战。现有的界面设计已经约定性地假定采用的是每次规定单个位置的单点指示用户输入设备（single pointing user input device）。其示例包括鼠标或触摸板。

更具体地，很多现有的图形用户界面（GUI）系统提供的是在其中将显示器的各部分与各单独的软件单元相关联的用户界面。由此举

例来说，显示器的多个部分可以与一窗口相关联，并且该窗口可以与特定软件应用和/或进程相关联。鼠标可以用于与窗口以及关联于该窗口的应用或进程进行交互。然后，鼠标指针可以移动到另一个窗口，以便与另一个应用或进程进行交互。由于所使用的只是单点指示设备，因此，每次只能发生与单个窗口以及应用或进程进行交互。

假设在任一时间都与窗口进行单独交互可以极大简化用户界面设计。在窗口内运行的应用和/或进程可以在检测到的与特定窗口的交互是接收到的唯一输入这一假设下工作。由此，应用和/或进程自己不需要关心发生在该窗口以外的显示器其他部分的其他用户交互的可能性。此外，窗口还可以被额外地划分成不同的单元，其中每一个单元都与该窗口某一具体部分相关联。每一个单元都可以由单独的软件单元（例如软件对象）来实现。同样，每一个软件对象可以处理那些发生在与之关联的区域内部的交互，而其自身则无需要关心那些可能在其他地方同时发生的交互。

另一方面，如果使用的是多点触摸界面，那么在显示器的不同部分有可能同时发生两个或更多触摸事件。这样就很难将显示器分成不同的部分，并且很难实现与每一个部分相关联的不同的独立软件单元进程交互。此外，即使将显示器分成了不同部分，但在单个部分中有可能发生多个触摸事件。由此，单个应用、进程或其他软件单元有可能需要处理多个同时发生的触摸事件。但是，如果每一个应用、进程或其他软件单元都需要考虑多个触摸交互，那么对在多点触摸使能设备上运行的软件来说，其总成本和复杂度将会过高。更具体地说，每一个应用都有可能需要处理大量引入的触摸数据。这样就可能会在看似功能简单的应用中需要很高的复杂度，并且可能会让对多点触摸使能设备进行编程变得困难和昂贵。此外，采用单点指示设备的现有软件很难被转换成或端接到（port to）可以在多点或多点触摸使能设备上工作的版本。

## 发明内容

本发明的实施例涉及的是为应用级软件来定义触摸事件的方法、软件、设备和 API。此外，某些实施例旨在为在多点触摸使能设备中运行的应用简化对单个和多个触摸事件的识别。为了简化对单个和多个触摸事件的识别，特定窗口内的每一个视图都可以被配置成多点触摸视图或是单点触摸视图。此外，每一个视图都可以被配置成独占（exclusive）或是非独占（non-exclusive）视图。根据视图的配置，可以忽略或是识别在该视图和其他视图中的触摸事件。被忽略的触摸就无需被发送到应用。通过有选择地忽略触摸，就能让那些没有利用先进的多点触摸特征的较为简单的应用或软件单元与更为复杂的应用或软件单元在相同设备上（甚至在同一时间）执行。

#### 附图说明

图 1 是根据本发明一个实施例的示例性的具有多点触摸能力的设备（multi-touch capable device）的输入/输出处理堆栈的图示。

图 2A 是根据本发明一个实施例的示例性的多点触摸使能设备的图示。

图 2B 是根据本发明一个实施例的另一个示例性的多点触摸使能设备的图示。

图 3 是根据本发明一个实施例的示例性多点触摸显示器的图示。

图 4 是显示根据本发明一个实施例的多点触摸标记的操作的示性方法的流程图。

图 5A 和 5B 是显示根据本发明一个实施例的独占触摸标记的操作的示性方法的流程图。

#### 具体实施方式

在以下关于优选实施例的描述中，将会对构成本申请的一部分的附图进行参考，在这些附图中借助例证显示了可以实现本发明的具体实施例。应该理解的是，可以利用其他实施例，并且可以做出结构上的改变而不脱离本发明优选实施例的范围。

本申请涉及一种用于在多点和多点触摸使能设备中运行的用户界面应用简化对单个和多个触摸事件的识别的触摸事件模型。为了简化对单个和多个触摸事件的识别，特定窗口内的每一个视图都可以被配置成多点触摸视图或是单点触摸视图。此外，每一个视图都可以被配置成独占或是非独占视图。根据视图的配置，可以忽略或是识别该视图以及其他视图中的触摸事件。

虽然在这里是依照特定的具有多点触摸能力的设备来描述和例证本发明的实施例的，但是应该理解，本发明的实施例并不局限于这类设备，而是可以广泛应用于任何具有多点触摸能力的设备。此外，本发明的实施例并不局限于多点触摸设备，而是还包含了多点设备，诸如在上述美国申请 No.11/649,998 中论述的多接近度传感器设备。

某些实施例涉及 API。一般来说，API 是计算机系统为了支持对来自软件操作的服务的请求而提供的源代码接口。API 是依照可以在系统构建时被解释或编译的程序语言来规定的，而不是依照关于如何将数据布置在存储器中的显性低级描述来规定的。提供 API 功能的软件被称为 API 的实现。诸如计算机系统、电子设备、便携式设备和手持式设备之类的各种设备都具有软件应用。上述设备在软件应用与用户界面软件之间对接，以便为该设备的用户提供特定的特征和操作。

本发明的至少某些实施例可以包括处于用户界面软件与软件应用相交互的环境中的一个或多个 API。各种函数调用 (call) 或消息经由用户界面软件与软件应用之间的 API 来传送。函数调用或消息的传送可以包括函数调用或消息的发布、启动、启用 (invoke) 或接收。示例性的 API 可以包括触摸事件信息的发送。此外，API 还可以实现具有参数、变量或指针的函数。API 可以接收所公开的参数或其他参数组合。除了所公开的 API 之外，其他 API 也可以单独或组合执行与所公开的 API 相似的功能。

图 1 是根据本发明某些实施例的示例性的具有多点触摸能力的设备的输入/输出处理堆栈的图示。硬件 100 可以设置在多点触摸使能设备的基准级 (base level) 上。它可以包括各种硬件接口元件，例如

多点触摸使能面板 101 和/或加速度计 102。多点触摸面板可以包括显示器以及同时感测多个触摸的面板。这种面板的示例在上述申请 11/649,998 中有更为详细的论述。加速度计可以是一种用于感测多点触摸使能设备的加速度的硬件设备。它可以用于感测设备何时移动，正如何移动，是否跌落等等。此外，也可以包含其他的硬件接口设备，例如陀螺仪、扬声器、按钮、红外（IR）传感器等等（未示出）。

与硬件 100 通信的可以是一个或一组驱动器 103。该驱动器可以接收和处理从硬件接收的输入数据。核心操作系统（OS）104 可以与所述一个或多个驱动器进行通信。核心 OS 可以处理从所述一个或多个驱动器接收的原始输入数据。在某些实施例中，可以将该驱动器考虑成核心 OS 的一部分。

一组 OS 应用编程接口（API）105 可以与核心 OS 进行通信。这些 API 可以是通常与操作系统（例如 Linux 或 UNIX API）包含在一起的一组 API。用户界面 API 106（UI API）可以包括一组被设计成供在设备上运行的应用所使用的 API。这些 UI API 可以使用 OS API。在设备上运行的应用 107 可以利用 UI API 中的 API 与用户通信。该 UI API 则又可以与较低级单元进行通信，最终与多点触摸面板 101 以及各种其他用户界面硬件进行通信。虽然每一层都可以利用其下方的层，但是这并不总是必需的。例如，在某些实施例中，应用 107 可以不定期地与 OS API 105 进行通信，API 105 和 106 可以包括相应的应用编程接口组及其相应的实现。例如，UI API 106 还可以包括用于实现 UI API 的用户界面（UI）软件。

图 2A 和 2B 是根据本发明某些实施例的两种示例性的多点触摸使能设备的图示。图 2A 显示的是示例性设备 200。该设备 200 可以包括 CPU 201 以及通过总线 204 连接的存储器 202。该总线还可以连接到多点触摸显示器 203。多点触摸显示器可以包括多点触摸面板和显示器。通过组合多点触摸面板和显示器，可以形成多点触摸显示器 203。该多点触摸显示器可以对应于图 1 硬件层 100 内的多点触摸面板 101。CPU 可以用于执行存储在存储器中的软件。该 CPU 执行的

软件可以包括图 1 的层 103~109。由此，软件可以包括驱动器、OS、各种 API 和应用。

图 2B 显示的是备选设备 210。设备 210 可以与设备 200 相类似。但是，设备 210 可以包括单独的多点触摸面板 (212) 和显示器 (211)，用以代替设备 200 的单个单元。由此，对设备 210 来说，就无需通过触摸显示器来与多点触摸面板进行交互。设备 210 可以是例如配备了多点触摸追踪板的膝上型计算机（多点触摸面板充当跟踪板）。

图 2A 和 2B 的多点触摸面板和/或显示器还可以利用其他传感器技术，诸如上述美国申请 No.11/649,998 中讨论的接近度感测。通常，多点面板 (multi-point panel) 和/或显示器可用于图 2A 和 2B 的设备。多点面板和/或显示器可以具备各种类型的传感器技术的特征。例如，它可以具备以下特征：纯多点触摸技术（由此得到多点触摸面板和/或显示器）、多接近度感测技术、这两种技术的组合或是其它类型的多点技术。

图 2A 和 2B 的设备可以包括各种不同类型的多点触摸使能设备。例如，它们可以包括移动电话、便携式视频游戏控制台、电子音乐播放器、电子书、PDA、电子组织器 (organizer)、电子邮件设备、膝上型或其他个人计算机、信息亭计算机 (kiosk computer)、自动售货机等等。

图 3 是示意性多点触摸显示器 300 的图示。该多点触摸显示器可以是图 2A 的显示器 203 或图 2B 的显示器 211。该显示器可以显示由在并入了显示器 (例如图 2A 的设备 200 或图 2B 的设备 210) 的设备中运行的软件所生成的各种用户界面单元 (诸如，图形等)。用户可以与各种用户界面单元进行交互，以便与软件进行交互。当使用图 2A 的设备时，用户可以通过直接在显示器上触摸这些用户界面单元来与之进行交互。当使用图 2B 的设备时，用户可以触摸单独的多点触摸板 212，以便移动和控制显示器 211 上的一个或多个光标，其中该光标被用于与软件进行交互。

在显示器 300 上呈现的用户界面单元可以包括一个或多个视图。

每一个视图都可以代表一个由单独的软件单元处理的图形用户界面单元。单独的软件单元可以包括不同的应用、不同的进程或线程（即使是在相同应用内部）、不同的例程或子例程、不同的对象等等。在某些实施例中，每一个单独的软件单元都可以为其实体的相应部分创建用户界面单元，并且还可以接收和处理该实体部分的触摸输入。该触摸输入可以由结合图 1 所讨论的各层来处理，所述层随后会将经过处理的触摸输入数据发送到软件单元（它可以是应用 109 的一部分）。经过处理的触摸输入数据可以被称为一个或多个触摸事件，并且可以采用一种比由多点触摸面板产生的原始触摸数据更易于处理的格式。例如，每一个触摸事件都可以包括当前正发生触摸处的一组坐标。在某些实施例中，这组坐标可以对应于触摸质心。为了简明起见，以下论述可以通过简单地引用视图本身来引用一个与该视图相关的软件单元。

视图可以是嵌套的。换句话说，一个视图可以包含其他视图。因此，与第一视图相关联的软件单元可以包括或链接到与第一视图内的视图相关联的一个或多个软件单元。某些视图可以与应用相关联，而其他视图可以与高级 OS 单元相关联，例如图形用户界面、窗口管理器等等。

图 3 的示意性显示器显示的是一个音乐浏览应用。该显示器可以包括状态栏（status bar）视图 301，它表明该设备的总体状态。该状态栏视图可以是 OS 的一部分。还可以包含标题视图 302。该标题视图自身可以包括若干个其他视图，例如中心标题视图 310、后退按钮 312 以及前进按钮 311。此外还可以包括表格视图 303。该表格视图 303 可以包括一个或多个表格单元视图，例如表格单元视图 304。可以看出的是，在一个实施例中，表格单元视图可以是曲目标题。另外还可以包括按钮栏视图 305。该按钮栏视图可以包括按钮 306~309。

每一个视图及其关联软件单元都能够接收、处理和操作在所述特定视图上发生的触摸事件。由此举例来说，如果用户触摸了曲目标题视图 304，那么与该视图相关联的软件单元可以接收到一个表明该视

图已被触摸的触摸事件、对该事件进行处理、并做出相应的响应。举个例子，该软件单元可以改变视图的图形表示（也就是突出显示该视图），和/或引起其他动作，诸如播放与所触摸的视图相关联的曲目。

在某些实施例中，触摸事件是在视图层级中最低一级上处理的。由此举例来说，如果用户触摸标题栏 302，那么触摸事件就没有必要直接由关联于该标题栏视图的软件单元来处理，而是可以由与包括在该标题栏视图内且其中发生触摸的视图相关联的软件单元来处理（也就是与视图 310、311 和 312 之一相关联的软件单元）。在某些实施例中，某些更高级视图也可以处理触摸事件。此外，仍然可以向那些不与正被触摸的视图相关联的各种软件单元进行报警，或者这些软件单元可以发现该视图正被触摸。

由于显示器 300 是多点触摸显示器，因此，多个触摸可以在同一时间发生。所述多个触摸既可以在同一视图中发生，也可以在两个或更多个不同的视图中发生。此外，用户还可以执行具有预定意义的手势（例如按下一个或更多手指并移动这些手指）。在 2004 年 7 月 30 日提交的名为“**GESTURES FOR TOUCH SENSITIVE INPUT DEVICES**”的美国专利申请 No.10/903,964 中对多点触摸手势（或称“多重触摸手势”，multi-touch gesture）有着更为详细的论述，并且该申请在这里全部引入作为参考。

视图可以接收始于该视图内的触摸事件。如果用户保持将某个手指按在显示器上，那么该视图随后可以接收到表明持续性触摸的多个触摸事件。如果用户移动已按下的手指，那么该视图可以接收到表明触摸移动的多个触摸事件。如果用户将按下的手指移到视图之外，那么该视图仍旧可以接收与该移动相关联的触摸事件（并且所述手指已经移至的视图无需接收这些触摸事件）。由此，视图可以接收与在始于该视图的手势或移动相关联的事件，即使在该事件在视图之外继续的情况下也是如此。

触摸可以是指以将手指或另一身体部位或物体按在多点触摸面板（或多点触摸显示器，multi-touch display）表面之上为开始并且以

手指或物体从显示器上移开之时为结束的动作。由此，所述触摸既可以包括手指或物体的移动，也可以包括将手指或物体在一时间段内保持在同一位置上。

触摸事件可以由一个或多个 API (及其相应的实施方式) 发送到视图 (或是实现该视图的软件单元)。在以下的附录 A 中提供了用于处理触摸事件的 API 的一个示例。根据附录 A 中的 API，该 API 可以向每一个视图发送包含了一个或多个单触摸数据结构 (或触摸数据结构) 的触摸事件数据结构。每一个触摸事件数据结构都可以定义在特定时刻在该视图上发生的所有触摸的当前状态。在触摸事件数据结构内的相应触摸数据结构可以定义特定时刻的一个或多个相应的单触摸的当前状态。由此，如果在特定时刻在特定视图中发生三个触摸，那么可以向该视图发送一个包含了三个触摸数据结构的触摸事件数据结构，其中这三个触摸数据结构定义了五个触摸状态。在某些实施例中，触摸数据结构即使在与其相关联的触摸不再发生的情况下也可以被发送，由此可以警告视图该触摸已经终止。

如上所述，触摸可以包括不需要同时发生的动作。例如，触摸可以包括移动手指或者在一时间段内将手指保持按压显示器的动作。但是，触摸数据结构定义的是特定时间的触摸状态。这样一来，多个触摸数据结构可以与单个触摸相关联，由此可以定义不同时刻的单个触摸。

每一个触摸数据结构都可以包括各种字段。“首次视图触摸”字段可以表明该触摸数据结构是否定义了对特定视图的首次触摸 (从实现该视图的软件单元被实例化时开始)。“时间戳”字段可以表明该触摸数据结构所涉及的特定时间。

“信息”字段可以用于表明触摸是否为基本手势 (*rudimentary gesture*)。举个例子，“信息”字段可以表明该触摸是否为轻扫 (*swipe*)，如果是的话，则确定该轻扫的方向。轻扫是一种在直线方向上快速拖动一根或多根手指的动作。API 实现可以确定某一触摸是否为轻扫，并且可以将该信息通过“信息”字段传递到应用，由此减轻应用在触摸

为轻扫的情况下必需的某些数据处理负担。

“轻击计数 ( tap count )”字段可以表明在触摸位置处顺序进行了多少次轻击。轻击可以被定义成是在面板上某一特定位置快速按下并抬起手指的动作。如果在面板上的相同位置再次快速连续地按下并释放手指，那么可以发生多个顺序轻击。这样该 API 实现就可以为有关各种应用的轻击进行计数，并且通过“轻击计数”字段来中继该信息。有时，对触摸使能界面来说，同一位置的多次轻击可以被认为是非常有用且易于记忆的命令。于是通过对轻击进行计数，API 同样可以减轻应用的某些数据处理负担。

“阶段”字段可以表明触摸当前所处的特定阶段。该阶段字段可以具有各种值，诸如可以表明触摸数据结构定义了一个尚未被先前触摸数据结构所引用的新触摸的“触摸开始阶段”。“触摸移动阶段”值可以表明所定义的触摸已经从先前触摸数据结构中定义的位置移动。“触摸固定阶段”值可以指示该触摸从生成关于该触摸的最后一个触摸数据结构时起即停留在同一位置。“触摸结束阶段”值可以指示该触摸已经结束（例如用户将其手指抬离多点触摸显示器的表面）。“触摸取消阶段”值可以指示该触摸已由设备取消。已取消的触摸可以是不必由用户结束但却能被设备确定为忽略的触摸。举例来说，设备可以确定触摸是无意中生成的（即将其视为便携式多点触摸使能设备放在口袋中的结果），并由此忽略该触摸。“阶段字段”的每一个值都可以是一个整数。

由此，每一个触摸数据结构可以定义触摸在特定时间的状况如何（例如触摸是否固定、被移动等等），而且还可以定义与该触摸相关联的其他信息（例如位置）。相应地，每一个触摸数据结构都可以定义特定触摸在特定时刻的状态。在触摸事件数据结构中可以添加一个或多个引用同一时间的触摸数据结构，其中该触摸事件数据结构可以定义特定视图在某个时刻正接收的所有触摸的状态（如上所述，某些触摸数据结构也可以引用已经结束并且不再被接收的触摸）。随着时间流逝，多个触摸事件数据结构可以被发送到实现视图的软件，以便

为软件提供用于描述在所述视图上发生的触摸的连续信息。对设备中的一个或多个单元，诸如硬件 100、驱动器 103、核心 OS 104、OS API 105 以及 UI API 来说，它们可以检测多点触摸面板 101 上的触摸，并且可以生成定义这些触摸的各种触摸事件数据结构。

处理多个触摸和多点触摸手势（multi-touch gesture）的能力有可能会增加各种软件单元的复杂度。在某些情况下，这种附加的复杂度可能是实现先进和期望界面特征所必需的。例如，游戏有可能需要处理在不同视图同时发生的多个触摸的能力，这是因为游戏通常需要同时按下多个按钮。但是，某些较为简单的应用和/或视图（及其关联的软件单元）未必需要先进的界面特征。例如，简单的按钮（例如按钮 306）用单个触摸就能令人满意的工作，因而就不需要多点触摸功能。在这些情况下，下层 OS 就可能向与某个旨在仅通过单个触摸（例如按钮）进行操作的视图相关联的软件单元发送不必要的或是过多的触摸数据（例如多点触摸数据）。由于软件单元有可能需要处理该数据，因此它有可能需要具有处理多个触摸的软件单元的所有复杂度的特征，即使在其关联的只是涉及单个触摸的视图的情况下也是如此。这样一来，由于传统上在鼠标界面环境（也就是各种按钮等等）中易于编程的软件单元有可能会在多点触摸环境中变得复杂的多，因此就会提高为设备开发软件的成本。

本发明的实施例旨在根据预定设置通过向各种软件单元有选择地提供触摸数据来解决上述问题。由此，在这里可以为选定的软件单元提供更为简单的界面，同时其他软件单元则可以利用更为复杂的多点触摸输入。

本发明的实施例可以依赖于与一个或多个视图相关联的一个或多个标记，其中每一个标记及其组合都指示了一种用于特定视图的触摸事件处理模式。举例而言，在这里可以使用多点触摸和/或独占触摸标记。多点触摸标记可以指示特定视图是否有能力接收多个同时发生的触摸。独占触摸标记可以指示特定视图在其正接收触摸事件时是否允许其他视图接收触摸事件。

图 4 是显示根据本发明一个实施例的多点触摸标记操作的流程图。在步骤 400，用户可以在视图内的第一位置触摸该视图。可以假设当接收到步骤 400 的触摸时，在多点触摸显示器上不存在其他触摸。在步骤 402，OS 可以向关联于该触摸位置的软件单元发送定义了接收到的触摸的触摸事件。

在步骤 404，用户可以在不释放第一触摸的同时（也就是在保持将手指按住第一位置的同时）在第二位置触摸该视图。由此举例来说，用户可以在步骤 400 触摸表格单元视图 304 的右部，并且在步骤 404 中在不从右部释放其手指的情况下触摸该表格单元视图 304 的左部。因此，第二触摸与第一触摸是同时进行的（这样就利用了显示器 300 的多点触摸能力）。

在步骤 406，OS 可以确定用于正被触摸的视图的多点触摸标记是否被置位。如果该多点触摸标记被置位，那么该视图可以是能够处理多个同时进行的触摸的视图。由此，在步骤 408 中可以将关于第二触摸的第二触摸事件发送到与该视图相关联的软件单元。应该指出的是，第一触摸事件的新实例（instance）同样是可以被发送的，以表明第一触摸事件仍在进行（也就是尚未抬起处于第一位置的手指）。如果将处于第一位置的手指以不抬起手指的方式从该位置移开（也就是说，如果在显示器表面上“拖动”手指），那么第一触摸事件的新实例可以规定不同的位置。

另一方面，如果该多点触摸标记未被置位，那么 OS 可以忽略或阻止第二触摸。如果忽略第二触摸，那么有可能导致不向与所触摸的视图相关联的软件单元发送任何与第二触摸相关联的触摸事件。在某些实施例中，OS 可以在必要时向其他软件单元警告该第二触摸。

由此，对那些被编程为每次只处理单个触摸的相对简单的软件单元来说，本发明的实施例可以允许这些软件单元将其多点触摸标记保持为未被断言（unasserted），并且由此确保不会向其发送作为多个同时进行的触摸的一部分的触摸事件。同时，可以处理多个同时进行的触摸的更为复杂的软件单元可以断言其多点触摸标记，并且可以接

收与其关联视图上发生的所有触摸有关的触摸事件。由此就能降低简单软件单元的开发成本，同时还为更为复杂的单元提供先进的多点触摸功能。

图 5A 和 5B 是显示根据本发明一个实施例的独占触摸标记的操作的示意性方法的流程图。在步骤 500，用户可以触摸第一视图。在步骤 502，OS 可以向与第一视图相关联的第一软件单元发送触摸事件。在步骤 504，用户可以在不释放第一触摸的情况下触摸第二视图。

在步骤 506，OS 可以检查用于第一视图的独占触摸标记是否被断言。如果该标记被置位（被断言），则意味着第一视图需要以独占方式接收触摸，并且不会有其他触摸被发送至其他视图。由此，如果独占触摸标记被置位，那么 OS 可以忽略（或阻止）第二触摸，并且不将其发送给任何软件单元。如果独占触摸标记未被置位，那么该处理可以继续执行图 5B 中的步骤 510。

在步骤 510，OS 可以确定用于第二视图的独占视图标记是否被置位。如果该标记被置位，那么第二视图只能接收独占触摸事件。由此，如果存在已经由另一个视图（即第一视图）接收的另一个触摸事件，那么第二视图将不能接收触摸事件，并且 OS 可以忽略第二触摸（步骤 512）。但是，如果用于第二触摸的独占触摸标记未被置位（未被断言），那么 OS 可以将与第二触摸相关联的触摸事件发送到第二视图。更具体地，OS 可以将与第二触摸相关联的触摸事件发送到与第二视图相关联的软件单元（步骤 514）。

由此，该独占触摸标记可以确保被标记为独占的视图在其作为显示器上接收触摸事件的唯一视图时唯一地接收触摸事件。独占触摸标记对于简化在多点触摸使能设备上运行的应用软件而言是非常有用的。在某些状况中，允许多个视图同时接收触摸有可能会产生复杂的冲突和差错。举例而言，如果同时按下删除曲目按钮和播放曲目按钮，那么有可能导致出现错误。为了避免此类冲突就可能需要复杂昂贵的软件。但是，本发明的实施例可以通过提供独占触摸标记来减少对于此类软件的需要，因为提供独占触摸标记可以确保该标记被置位的视

图将仅在其是正接收触摸事件的唯一视图时才会接收触摸事件。作为替换，一个或多个视图可以使其独占触摸标记未被断言，由此允许在这些视图的两个或更多个视图上的多个同时触摸。

在某些实施例中，所述独占标记可以表示对于整个显示器的独占性。由此，当独占标记被置位的视图正在接收触摸事件时，会阻止该显示器中的所有其他视图接收任何触摸事件。在替换实施例中，所述独占标记可以表示较小区域内的独占性，诸如在单个应用或单个窗口内的独占性。例如，当其独占性标记被置位的第一视图正接收触摸事件时，它可以阻止同一窗口中的其他视图接收任何触摸事件，但却不会阻止其他窗口中的视图。

独占触摸标记和多点触摸标记可以被组合。由此，正被显示的一个或多个视图可以各自包括两个标记，即多点触摸标记和独占触摸标记。在某些实施例中，所有被显示的视图都可以包括这两个标记。其中一个标记的值不必依赖于另一个标记的值。在一个示例中，独占和多点触摸标记都被置位的视图可以允许该视图内的多个触摸，但却是以唯一的方式接收触摸（也就是说，在该视图正接收触摸时可以阻止对其他视图的触摸）。两标记都未被断言的视图可以阻止该视图内的多个触摸，但允许该视图内的单个触摸，即使在其他视图中有触摸同时发生的情况下也是如此。多点触摸标记未被断言而独占触摸标记被断言的视图则可以在任何其他视图中没有发生其他触摸时允许该视图内的唯一的单个触摸。多点触摸标记被断言而独占触摸标记未被断言的视图则可以允许为其接收所有触摸。对于两标记都被断言的视图来说，该视图可以在没有发生关于其他视图的其他触摸的同时允许该视图中的多个触摸。

替换实施例可以只具有两标记(以及相关联功能)中一种的特征。由此，某些实施例可以只使用多点触摸标记或是只使用独占触摸标记。在某些实施例中，不同的视图可以使用不同的标记组合。

由图 4、5A 和 5B 中的 OS 执行的各种功能可以改由其他软件执行，诸如各种实用软件 (utility software)。这些功能可以在图 1 的

层 103~108 中的任何一层的软件执行。在替换实施例中，这些功能甚至可以由硬件 100 执行。

在下文中提供了一组例示代码，该代码显示的是根据本发明某些实施例而与视图相关联的例示软件单元的方法。本领域技术人员将会认识到，其他代码同样可以用于实现上述功能。

虽然以上论述集中在多点触摸显示器和面板上，但是本发明并不局限于多点触摸设备，而是可以包括如上所述的各种多点设备（例如，包括多接近度传感器设备）。对多点设备来说，多点和独占点标记都是可以使用的。这些标记可以按与如上所述的多点触摸和独占触摸标记相类似的方式工作。

虽然在这里参考附图并且结合实施例来对本发明进行了全面描述，但是应该指出，各种变化和修改对本领域技术人员来说都是显而易见的。这些变化和修改应该被理解成是包含由所附权利要求定义的本发明的范围以内。

## 附录 A

### 示例性 UI API 代码

```

@interface UIResponder

- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event;

- (void)touchesCancelled;

// 该方法可以被实现以替代各个
touchBegan:/touchMoved:/touchEnded:
// 视图作者希望同时处理所有相关联触摸时的方法
- (void)touchesChangedWithEvent:(UIEvent *)event;

@end

typedef enum {
    UITouchPhaseBegan,           // 只要手指触摸表面
    UITouchPhaseMoved,          // 只要手指在表面移动
    UITouchPhaseStationary,     // 自前一事件开始只要手指正触摸表面
    并且还没有移动时
    UITouchPhaseEnded,          // 只要手指离开表面
    UITouchPhaseCancelled,      // 只要触摸未结束但需要停止追踪时
    (例如, 将设备放到脸上)
} UITouchPhase;

enum {
    UITouchSwipedUp      = 1 << 0, // 如果以某一角度轻扫, 则可以置位
    一个以上的轻扫标记
    UITouchSwipedDown    = 1 << 1, // 这些轻扫的方向与UI取向相关
    (参见UI应用)
    UITouchSwipedLeft    = 1 << 2,
    UITouchSwipedRight   = 1 << 3,
};

typedef unsigned int UITouchInfo;

@interface UITouch : NSObject
{
    BOOL           _firstTouchForView;
    NSTimeInterval _timestamp;
    UITouchPhase   _phase;
    UITouchInfo    _info;
    NSUInteger     _tapCount;

    UIWindow      *_window;
    UIView         *_view;

    CGPoint        _locationInView;
    CGPoint        _previousLocationInView;
}

```

```
}

- (NSTimeInterval)timestamp;
- (UITouchPhase)touchPhase;
- (UITouchInfo)touchInfo;
- (NSUInteger)tapCount; // 在特定时间量、在特定点内
向下触摸

- (UIWindow *)window;
- (UIView *)view;

- (CGPoint)locationInView;
- (CGPoint)previousLocationInView;

@end

@interface UIEvent : NSObject
{
    CFTypeRef           _event;
    NSTimeInterval      _timestamp;
    NSMutableSet        *_touches;
    CFMutableDictionaryRef _keyedTouches;
}

- (NSTimeInterval)timestamp;

- (NSSet *)allTouches;
- (NSSet *)touchesForWindow:(UIWindow *)window;
- (NSSet *)touchesForView:(UIView *)view;

@end .
```

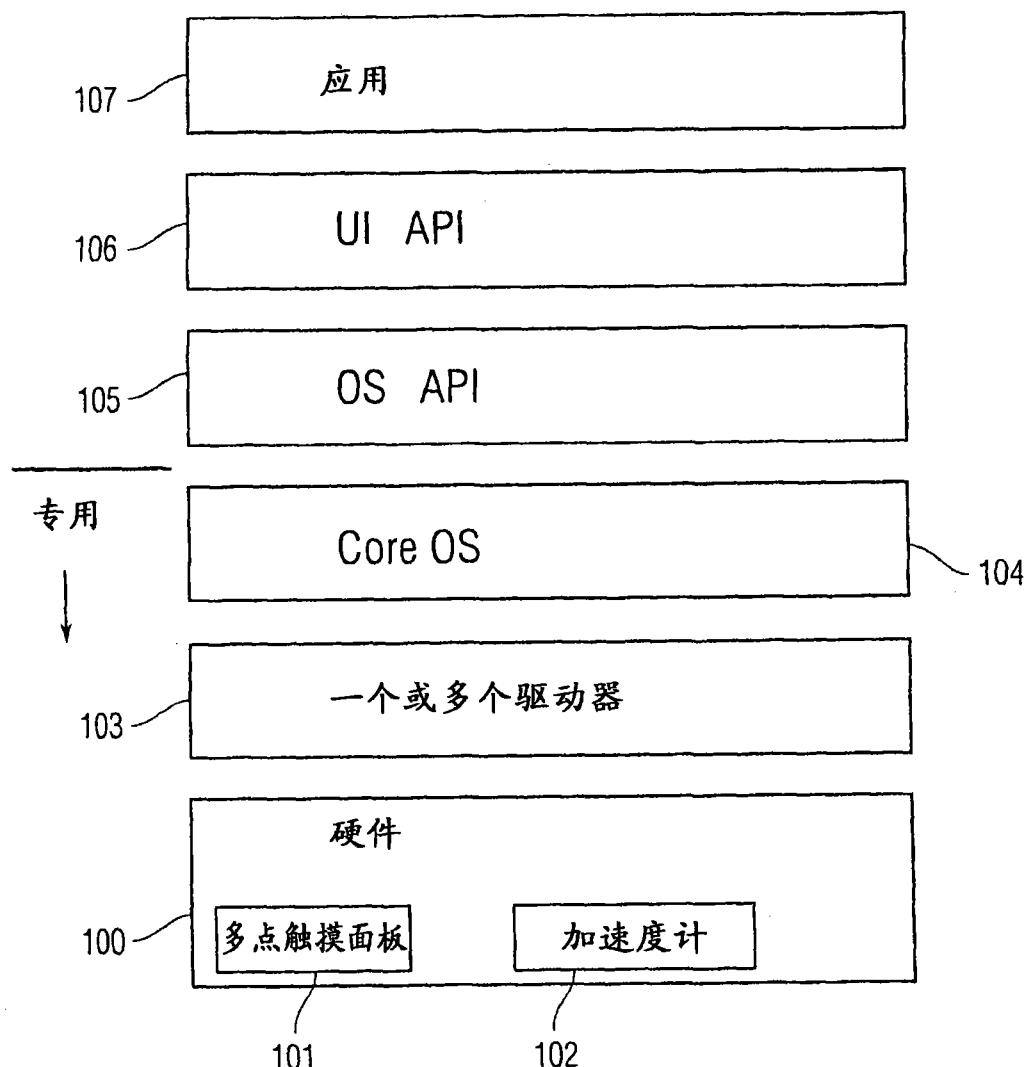


图 1

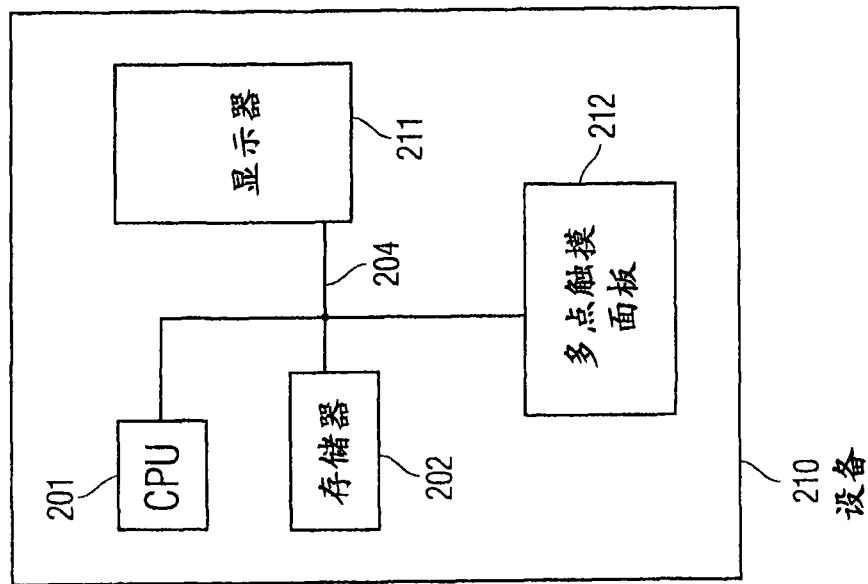


图 2B

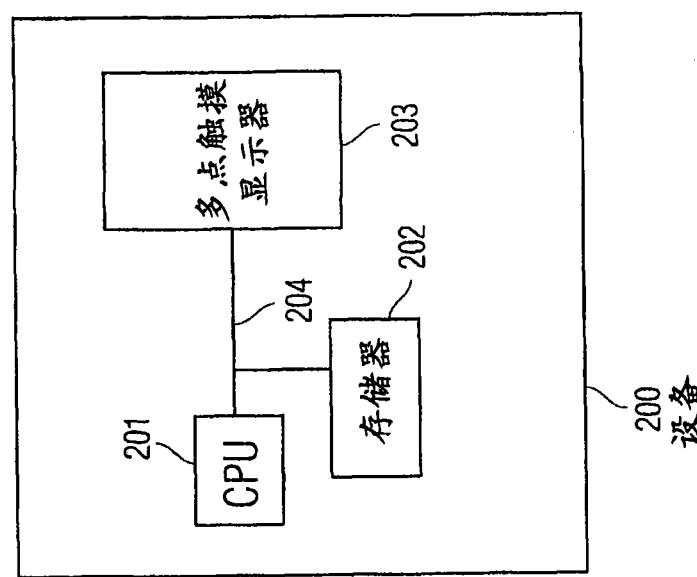


图 2A

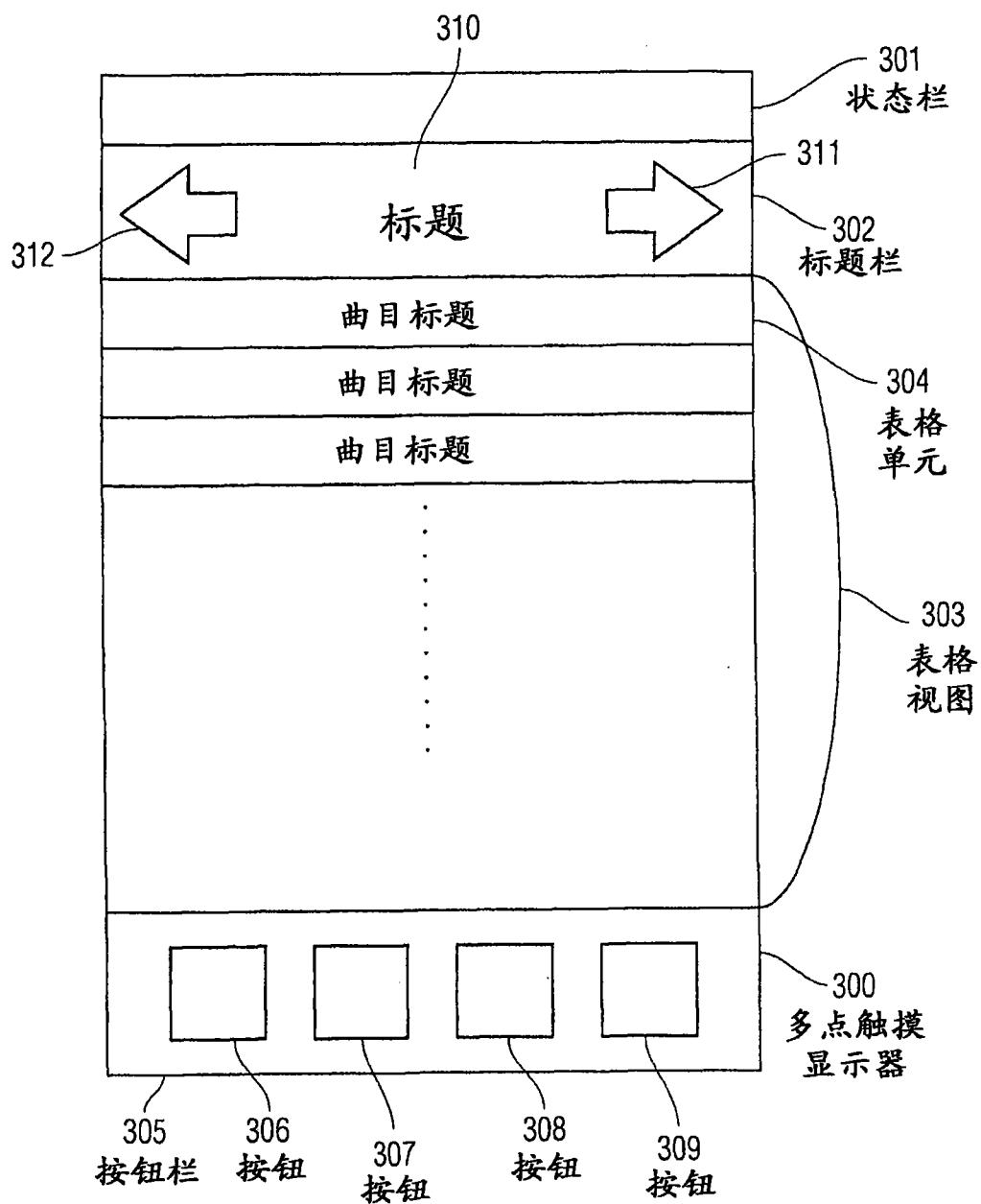


图 3

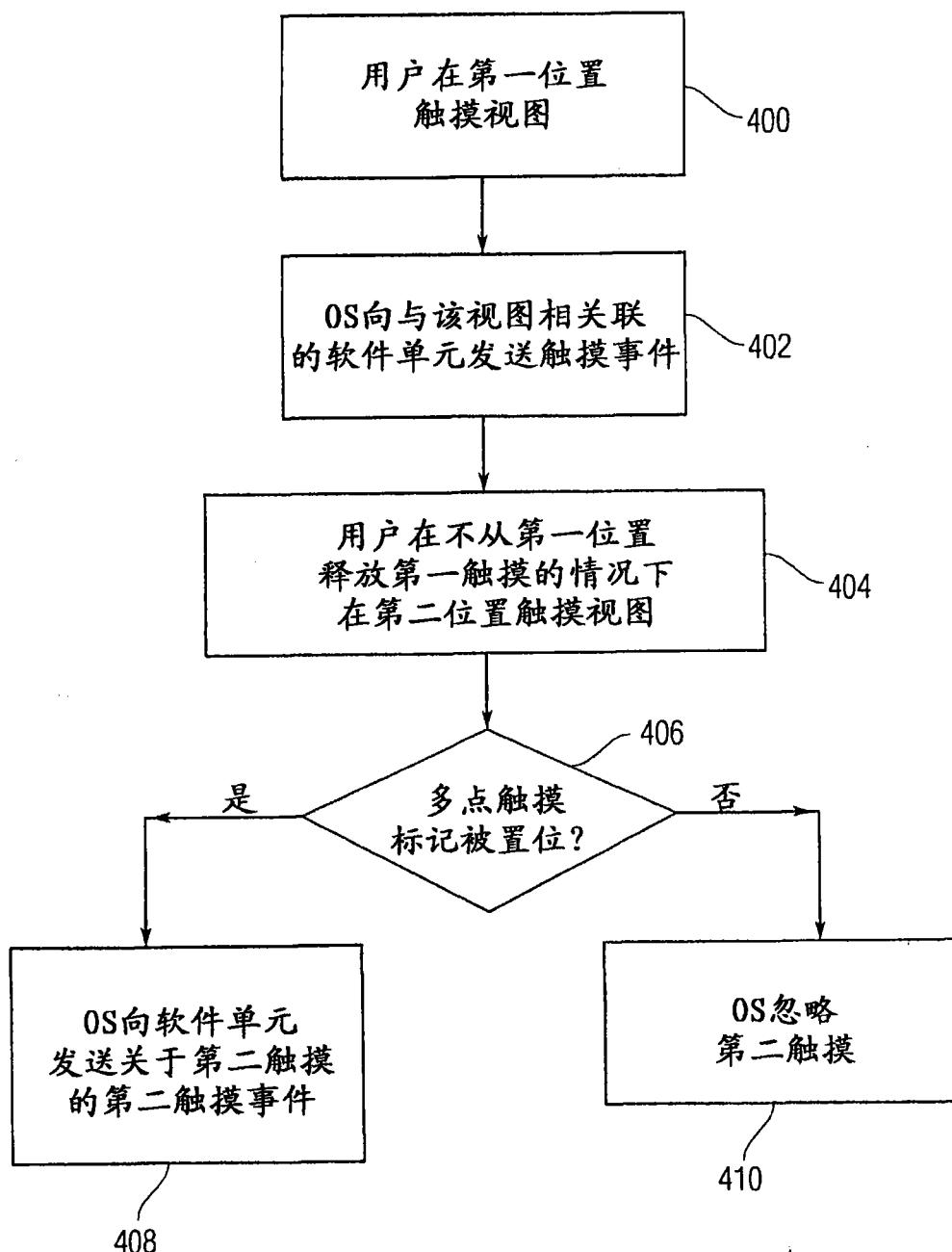


图 4

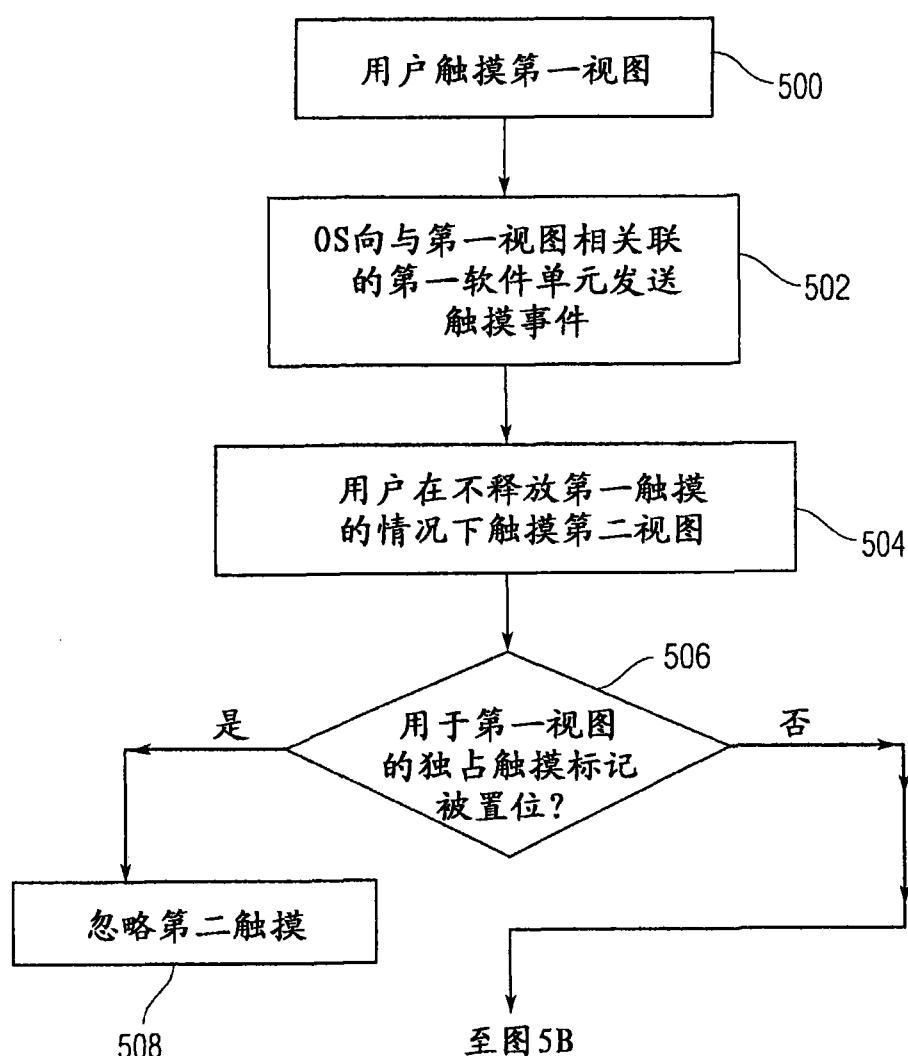


图 5A

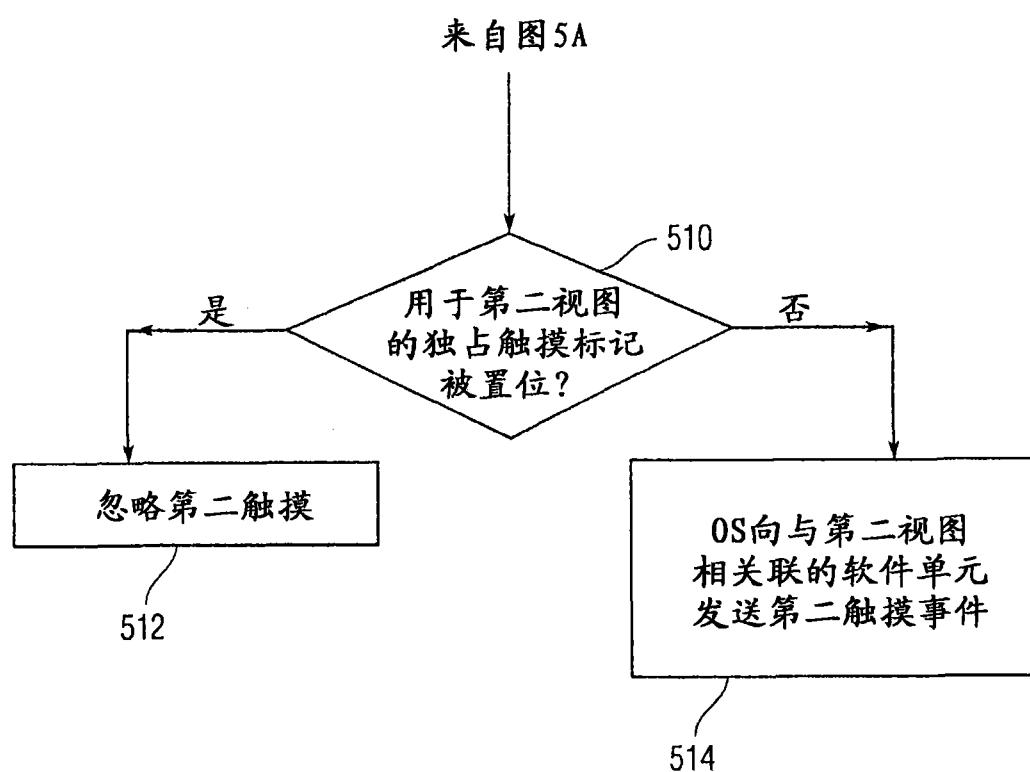


图 5B