(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104025002 A
(43)申请公布日 2014.09.03

(21)申请号 201380004779.8
(22)申请日 2013.01.05
(30)优先权数据
13/343786 2012.01.05 US
(85)PCT国际申请进入国家阶段日
2014.07.04
(86)PCT国际申请的申请数据
PCT/US2013/020416 2013.01.05
(87)PCT国际申请的公布数据
WO2013/103915 EN 2013.07.11
(71)申请人 微软公司
地址 美国华盛顿州
(72)发明人 B.S. 列维 M.I. 沃尔利
C.D. 萨里恩 R.J. 贾雷特 黄子欣
(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 李舒 汪扬
(51)Int.Cl.
G06F 3/048(2013.01)
G06F 3/14(2006.01)
G06F 9/44(2006.01)

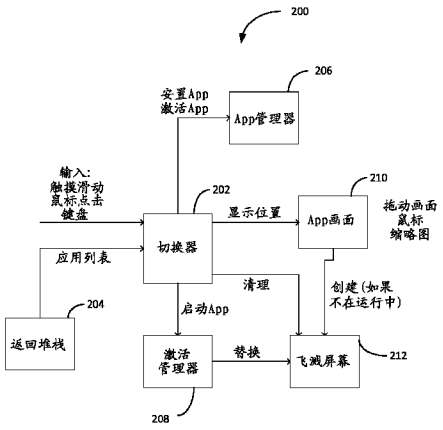
权利要求书2页 说明书8页 附图11页

(54)发明名称

被终止的应用在返回堆栈内的保持

(57)摘要

所要求保护的主题提供用于系统和/或方法,所述系统和/或方法用于影响在计算环境中在返回堆栈上的应用的保持以及应用的重新启动,所述应用在该计算环境中已经被杀死和/或被终止。返回堆栈包括近来被计算环境或者被用户经由计算环境使用过或者以别的方式调用过的应用的列表。在一个实施方案中,计算环境包括图形用户界面(GUI),其中用户可以做出有意的手势以与应用交互,包括重新启动被杀死或者以别的方式被终止的应用。依据打算进行这样的重新启动的用户命令,计算环境可以向用户呈现与被杀死的应用相关联的临时UI屏幕,并且依据用来执行被杀死的应用的提交信号而呈现更完整的运转屏幕。



CN 104025002 A

1. 一种用于在计算环境中在应用列表中保持被杀死的应用的方法，所述计算环境包括一个或者多个处理器、计算机可读存储器、输入和输出组件、用户界面(UI)以及用于管理应用的加载、启动、执行的操作系统，所述 UI 提供对于用户可用的应用的表示，所述方法的步骤包括：

更新应用列表中一组应用的列表，所述应用在所述计算环境中对于用户是可用的；

当所述应用之一从执行状态转变成被杀死状态时，在应用列表中保持被杀死的应用；

依据来自用户的、表明从应用列表中选择所述被杀死应用的命令的信号，创建与所述被杀死的应用相关联的临时 UI 屏幕；

在至少 UI 的一部分上呈现该临时 UI 屏幕，所述临时 UI 屏幕能够被用户在所述 UI 上拖动；以及

一旦所述应用依据来自所述计算环境的提交信号而执行，就将该临时 UI 屏幕替换为与所述应用相关联的常规 UI 屏幕。

2. 权利要求 1 的方法，其中所述应用列表包括一群最近使用过的应用的列表，所述群包括：当前正运行的应用以及被杀死的应用。

3. 权利要求 1 或者 2 的方法，其中所述来自用户的信号包括一群信号中的一个，所述群包括：在触敏屏幕上的手势、鼠标手势、键盘手势、话音激活信号、眼睛追踪信号。

4. 前述权利要求的任一项的方法，其中与所述被杀死的应用相关联的所述临时 UI 屏幕包括如下的屏幕，即：该屏幕包括的图形内容比在所述应用执行时与所述应用相关联的 UI 屏幕少。

5. 前述权利要求的任一项的方法，其中与所述被杀死的应用相关联的所述临时 UI 屏幕包括如下的屏幕，即：该屏幕包括的功能性比在所述应用执行时与所述应用相关联的 UI 屏幕少。

6. 一种用于用户的计算环境，所述计算环境还包括：

处理器；

计算机可读存储器，所述计算机可读存储器能够存储应用，所述应用能由所述处理器执行；

输入和输出组件；

多个应用，所述应用响应于来自所述用户的输入命令；

图形用户界面(GUI)，用于向所述用户呈现视觉信息；

操作系统，用于管理所述应用的执行和其他潜在状态；以及

其中所述操作系统进一步包括：

切换器组件，所述切换器组件能够接受来自所述用户的输入信号，以切换到近来使用过的并且当前处于被杀死状态的应用；

返回堆栈组件，所述返回堆栈组件包括被所述计算环境近来使用过的应用的列表；以及

飞溅屏幕组件，所述飞溅屏幕组件能够创建临时 UI 屏幕，该临时 UI 屏幕能在所述 GUI 上对于被杀死的应用再现，对于该被杀死的应用，所述用户已经给出有意的手势，表明重新启动在所述返回堆栈组件中的所述被杀死应用的命令，并且进一步地，其中在重新启动之前所述临时 UI 屏幕能够由所述用户在所述 GUI 上移动。

7. 权利要求 7 的计算环境，其中所述计算环境还包括触敏显示屏，并且所述用户可以在所述触敏显示屏上给出滑动手势，表明重新启动所述被杀死的应用的命令。

8. 权利要求 7 或者 8 的计算环境，其中所述滑动手势进一步包括触摸手势，以使得所述触摸手势的结束条件向所述计算环境表明用来重新启动所述被杀死的应用的提交信号。

9. 前述权利要求的任一项的计算环境，其中所述临时 UI 屏幕包括如下的屏幕，即：该屏幕包括的图形内容比在所述应用执行时与所述应用相关联的 UI 屏幕少。

10. 前述权利要求的任一项的计算环境，其中与所述被杀死的应用相关联的所述临时 UI 屏幕包括如下的屏幕，即：该屏幕包括的功能性比在所述应用执行时与所述应用相关联的 UI 屏幕少。

被终止的应用在返回堆栈内的保持

背景技术

[0001] 操作系统(OS)通常负责硬件资源的管理以及为计算系统中的应用软件提供公共服务。操作系统影响可执行应用在给定的计算环境和/或平台上的加载、启动和执行。在应用的运行过程期间,应用可能终止,或以别的方式通过各种各样的机制被杀死——机制例如是用户输入、定时的事件、死锁、或其他有计划或无计划的方式。

[0002] 在如今的计算平台和/或环境中,多个应用(其处于某种执行状态中)由操作系统跟踪——并且用户可以通常经由图形用户界面(GUI)与它们交互。用户能够——经由击键和/或其他有意的(deliberate)动作——查看并了解什么以及哪些应用正在有效地运行。此外,用户能够——经由这样的击键和/或其他有意的动作——在这样的应用之间切换,特别是在虚拟化的、分区的和/或多核的计算环境中。

发明内容

[0003] 为了提供对本文所描述的某些方面的基本理解,在下面给出本创新的简化概要。本概要并不是所要求保护的主题的广泛概述。它既不打算标识所要求保护的主题的关键性的或决定性的元素,也不打算描绘本创新主题的范围。其唯一的目的是以简化的形式给出所要求保护的主题的某些概念,以作为之后给出的更详细描述的序言。

[0004] 本申请的某些实施方案提供用于系统和/或方法,所述系统和/或方法用于影响在计算环境中在返回堆栈(backstack)上和/或更一般性地在应用列表(“app 列表”)上的应用的保持,以及应用的重新启动,这些应用在该计算环境中已经被杀死(kill)或者终止。返回堆栈包括近来被计算环境或者被用户经由计算环境使用过或者以别的方式调用过的应用的列表。对于本申请的目的来说,当不要求用户可访问的应用是诸如最近使用过(MRU)那样的任何特定顺序时,术语“app 列表”可满足术语“返回堆栈”的需要。

[0005] 在一个实施方案中,计算环境包括图形用户界面(GUI),其中用户可以做出有意的手势来与应用交互,包括切换和/或重新启动被杀死或以别的方式被终止的应用。依据打算进行这样的切换和/或重新启动的用户命令,计算环境可以向用户呈现与被杀死的应用相关联的临时 UI 屏幕,并且依据用来执行被杀死的应用的提交(commit)信号而呈现更完整的运转屏幕。

[0006] 在一个实施方案中,提交信号和/或命令可以由用户作为第二手势给出。作为第二手势,这样的提交信号可以给予用于计算环境的额外控制点——因为用户的第一手势可能表明用所杀死的应用“切换”当前正在运行的应用。由于用户可能最终决定不“重新启动”所杀死的应用,所以来自用户的第二提交手势向计算环境表明重新启动所杀死的应用。

[0007] 当结合本申请内给出的附图来阅读时,本系统的其他特征和方面在下面的详细说明中给出。

附图说明

[0008] 在参考的附图中图示了示范性的实施方案。旨在使本文所公开的实施方案和附图

被看作说明性的而非限制性的。

[0009] 图 1 描绘当前用户可用的各种各样的计算环境。

[0010] 图 2 是系统及其组件的方框图的一种实施方案，所述系统及其组件可以影响被杀死和 / 或被终止的应用在返回堆栈上的保持，以用于随后的用户交互。

[0011] 图 3 是用于在计算环境中操控和 / 或处理被杀死和 / 或被终止的应用的流程图的一种实施方案。

[0012] 图 4 是用于在计算环境中操控和 / 或处理被杀死和 / 或被终止的应用的时序 / 过程流程图的一种实施方案。

[0013] 图 5A 至 5F 显示一系列的 GUI 屏幕，该 GUI 屏幕描绘用户可以如何经历从返回堆栈中重新启动被杀死和 / 或以别的方式被终止的应用。

[0014] 图 6A 至 6C 显示另外一系列的 GUI 屏幕，该 GUI 屏幕描绘用户可以如何经历从返回堆栈中重新启动被杀死和 / 或以别的方式被终止的应用。

具体实施方式

[0015] 当在本文中被使用时，术语“组件”、“系统”、“界面”等打算指计算机相关的实体，其或为硬件、或为软件（例如，在执行中的）和 / 或为固件。例如，组件可以是在处理器上运行的进程、处理器、对象、可执行文件、程序和 / 或计算机。作为举例说明，在服务器上运行的应用和服务器两者都可以是组件。一个或多个组件可以驻留在进程之内，并且组件可以定位于一台计算机上和 / 或分布在两台或者更多台计算机之间。

[0016] 参考附图来描述所要求保护的主题，其中贯穿全文，同样的参考标号被用来指同样的元素。在下面的描述中，出于解释的目的，为了提供对本创新主题的透彻理解，阐明了许多特定的细节。然而，明显的是，可以实践所要求保护的主题而无需这些特定的细节。在其他实例中，为了便于描述本创新主题，以方框图的形式显示众所周知的结构和设备。

[0017] 介绍

在计算环境的正常操作过程中，典型的用户（或者用户们）可在任何给定的时间有数个运行中和 / 或执行中的应用（替换地称为“app”或“apps”）。这些应用可以是经由用户界面（UI），且更一般地是经由 GUI，而被控制和 / 或可控制的。应用可以是能作为一组图标、图块等而表示给用户的。此外，这些应用和它们的表示可以经由多个输入 / 输出设备或者机制被控制和 / 或可控制，所述输入 / 输出设备或者机制例如是 :触摸屏界面、鼠标、键盘、话音激活、眼睛追踪和 / 或在本领域已知的任何其他手段或机制。计算环境通常在计算环境的操作系统中管理这样的应用的执行和状态。

[0018] 图 1 显示典型的计算环境。用户 102 可以与诸如在群 104 中所描绘的多个计算平台交互。这样的计算平台可以包括 PC、膝上型电脑、智能设备、智能电话、终端或者任何其他已知的可与各种计算组件通信的物理设备。这些设备的每一种都可以经由多个软件和 / 或固件层 106 被直接或间接地控制。这样的层可以进一步包括 UI 或者 GUI 108 和 / 或操作系统 110。无论是由用户直接或间接地控制或可控制，应用通常都由操作系统管理。

[0019] 这些应用可以在“前台”（即，用户可察觉的和 / 或可以同步地和 / 或实时地运行）或者在“后台”（即，用户不容易或者不能明显察觉的和 / 或可以异步地和 / 或非实时地运行）中运行。这些应用——不论它们正运行或处在什么执行状态中（例如，正被用户启动、正

加载、正执行、暂停、正被终止、已终止等）——可以经由一组击键和／或任何其他有意的动作（例如，在触敏屏幕上的多个触摸）而在任何给定的时间为用户所知和／或是用户可访问的。

[0020] 这样的应用的这样的列举和／或显示可以被存储或以别的方式保持在由操作系统所使用的、被称为“返回堆栈”的数据结构中。对于用户可以如何与返回堆栈交互的仅仅一个示例，击键—ALT-TAB 或者 WINDOWS-TAB——在 Microsoft Windows® 操作系统（版本 7 或者其他更早版本）中通常为用户在屏幕上产生视觉显示，表示用户可能想与之交互的所有活动的应用，并且常常按最近使用过（MRU）的顺序。

[0021] 在当前的操作系统中，曾经在运行中和／或执行中、而现在可能已经被杀死（由用户或者操作系统）或以别的方式被终止的应用，可能在返回堆栈中不能被容易地访问或者不被作为用户可与之交互的选项来呈现。

[0022] 为了本申请的目的，术语“被杀死”和“被终止”可作为应用的状态而被可互换地使用。两个术语都指已经到达其生存期结束的应用。通常，这意味着应用不再能够被调度来接收 CPU 时间。一般而言，可能有至少两种方式到达这种状态。第一，应用可能从其最后的指令返回（即，到达停止（halt）状态，其中没有后续的代码要执行）——例如，当可执行的文件从其“主”函数返回时。第二种方式是让操作系统“杀死”应用。这通常可以由进程自身以外的某些事情触发，例如，用户命令终止、应用意外地崩溃或者结束等。

[0023] 现在将描述系统和方法的数个实施方案，其保持被杀死和／或以别的方式被终止的、用户可以与之交互的应用。在数个实施方案中，组件被描述用于这样的保持——以及这样的应用的表示可能经由 GUI 呈现给用户以用于交互——例如用于重新启动这样的应用。

[0024] 被杀死的应用

返回堆栈为用户提供一致且快速的切换机制。令人期望的是，在返回堆栈中添加用于操控被杀死的应用的附加组件或者机制不应该减损此总体的用户体验。

[0025] 由于多种原因，应用可以当处于返回堆栈中时被终止或杀死。例如，可以发生正常的终止，例如，在计算平台上的低存储器条件可能终止近来所使用的 apps。此外，应用可能崩溃或被挂起 -- 例如，在屏幕上、在返回堆栈中或者在切换至该应用时。

[0026] 在用户账户间的切换也可能终止已暂停的 apps 以便释放存储器。当切换用户账户时，系统可能暂停、然后终止 apps（或者只是终止在返回堆栈中的 apps）以便释放存储器用于新的活动的用户会话。当用户切换回原始用户会话时，原始用户在其返回堆栈中曾拥有的所有应用通常将消失。

[0027] 应用被杀死的其他原因包括：（1）apps 想要在切换时终止，因为它们处在不良状态；（2）为了额外的隐私（例如，金融应用等），apps 想要在切换时终止；（3）在运行时环境中的场所改变或者其他改变，其可能需要杀死应用以便让改变得以安装。

[0028] 在设计用于操控和／或管理被杀死的应用的鲁棒的系统时，可能合意的是——当应用被杀死而同时对用户仍可见时——所杀死和／或崩溃的应用将是返回堆栈中的下一个 app。此外，就不毁坏或者以别的方式破坏返回堆栈或其内容而论，在应用处于返回堆栈中时应用的杀死应该是得体的。

[0029] 用户体验示例

现在将描述根据本申请构成的系统的某些实施方案可以如何操控返回堆栈中的被杀

死的应用。对于第一个示例，用户可以去到“开始”处来启动某些喜爱的 apps ：例如，天气、IE 和股票。一旦用户已经启动这些 apps－－不是回到“开始”处在它们之间切换－－用户就可以使用返回堆栈来快速地重新回到它们。用户可以间或看到用于她近来使用过的apps之一的“临时”UI（即，快速且暗淡（dirty）的飞溅屏幕（splash screen）或者部分飞溅屏幕）以表明它还没准备好运行。然而，当用户把它留在屏幕上时，应用快速启动并且返回到用户先前所在之处。

[0030] 在另一个例子中，用户可能经历应用的屏上（on-screen）崩溃。例如，在本申请的一个实施例中假设用户正在阅读当前的新闻－－即，可能用户正在阅读报纸的app。用户可能选择新的文章来读，这可能导致该app 崩溃，因此给用户呈现墙纸屏幕。用户快速地意识到 app 已经崩溃，并且可能从触敏屏幕的左边缘滑动（swipe）（或者某个其他有意的手势），这恢复报纸的 app 并且启动它。因此，在本实施方案中，用户不需要回到启动器屏幕去重新回到已崩溃的 apps。而是，本实施方案可以允许用户利用单一的手势重新回到已崩溃的apps。

[0031] 操纵和安排已终止的 apps

在本系统的一个实施方案中，被杀死和 / 或被终止的应用可以被用户操纵和 / 或安排，就像对于运行中的程序可以做的那样。就这一点而言，被杀死和 / 或被终止的应用可以响应于各种用户命令，诸如是 ：（1）“切换的”；（2）“拍快照（snap）的”；（3）“丢弃的”和 / 或（4）“剪除（prune）的”。将意识到，其他用户命令也许是可能的，并且这样的其他或者不同的命令被预期为在本申请的范围之内。

[0032] 出于解释的目的，用户可以“切换”应用——也就是说，一个正在前台中运行并且经由 UI 呈现给用户的应用，可以在前台中被另外一个应用取代，并且经由 UI 呈现给用户。用户可以给应用“拍快照”，以使得被“拍快照”的应用现在与另外一个应用共享前台和对用户的呈现。用户可以“丢弃”应用，以使得所“丢弃”的应用可以不再经由 UI 呈现给用户，但是所丢弃的应用可能仍然在执行（除非被终止）并且在返回堆栈中保持。用户可以“剪除”应用，以使得所“剪除”的应用被终止。如果所剪除的应用正在运行中，那么剪除将导致该应用被终止并且从返回堆栈中移除。如果所剪除的应用已经被杀死，那么剪除将导致该应用被从返回堆栈中移除。

[0033] 这些用户命令的每一个可以以任何已知的方式递送至计算环境，所述方式例如包括经由在触敏显示屏上所做出的手势。例如，切换和 / 或拍快照命令可以利用滑动手势完成，例如，从屏幕的左边或者右边滑动。丢弃命令可以通过触摸 UI 的表示应用的部分并且将应用“扔”回到返回堆栈中而完成。剪除命令可以通过触摸 UI 的呈现应用的部分并且将应用“扔”至某个位置（例如屏幕的底部）而完成。当然其他手势是可能的，并且只要合适的手势为用户所知且为了适当的用户体验起见而一致就足够了。

[0034] 在其他实施方案中，按照本申请构成的系统相对于被杀死的应用可以具有以下属性 ：（1）如果 app 在处于返回堆栈中时被终止，则 app 将保留在返回堆栈中（并且，在一个实施方案中，处在返回堆栈中与该 app 运行时相同的位置）；（2）被杀死的 apps 可利用包括临时画面（temporary visual）的飞溅屏幕来表示 ；（3）被杀死的 app 可以像其他任何 app （无论在执行中或者在任何其他合适的状态）一样被切换、拍快照、丢弃和 / 或剪除（或者例如经由 UI 给予应用的任何其他可能的用户命令）；（4）被杀死的 app 可依据给定的事件被激活，

例如，当切换超时发生时。在每个实例中，可能是这种情况：计算环境呈现被杀死的 app 的临时表示以使得该临时表示可以像运行中的应用一样被操纵和安排。在这样的实例中，临时表示可在重新启动被杀死的 app 之前被安置为例如在桌面（或者其他地方）上的 app 窗口。

[0035] 在此外的其他实施方案中，按照本申请构成的系统相对于崩溃的或者挂起的应用可以具有以下属性：（1）如果屏上 app 崩溃，则用户可被带去期望的后台，并且崩溃的 app 将是返回堆栈中的下一个事物——用其飞溅屏幕来表示；（2）如果 app 在经由返回堆栈切换回它时崩溃或者挂起，则用户可被带去期望的后台，并且该挂起 / 崩溃的 app 将是返回堆栈中的下一个事物。

[0036] 在此外的其他实施方案中，按照本申请构成的系统相对于应用的屏上表示可以具有以下属性：（1）当用户切换到已经被杀死或者终止而同时处于返回堆栈中的 app 时，该 app 将利用临时画面来表示，该临时画面在 app 被带到屏上时创建；（2）如果所杀死的 app 在屏上被激活，则飞溅屏幕临时画面可被切换（hand off），以使得过渡平稳发生；（3）如果用户切换离开被杀死的 app，则可以使飞溅屏幕临时画面消失。

[0037] 已经描述了某些实施方案和它们的一些方面和属性，现在将公开某些其他实施方案。图 2 描绘了系统（200）的一部分，系统 200 包括能够保持被杀死和 / 或以别的方式终止的应用以便向用户呈现和随后重新启动并执行的组件。系统 200 包括切换器组件 202，其接受多个用户命令作为输入，用户命令例如是触摸滑动、鼠标点击、键盘敲击等。可以做出用户命令以观看多个状态中的应用的列表，多个状态例如是执行中、被杀死、被终止等。该列表可以包括任何由返回堆栈组件 204 给出的合适的数据结构（AppList）。

[0038] 切换器组件 202 管理切换会话的生存期。当会话开始时，切换器 202 可以创建当前返回堆栈的快照。额外的切换可以使用户进一步移动通过该堆栈并且延长会话。当用户与 app 交互时或者在切换之间的短的超时之后，会话结束。切换器组件 202 与 App 管理器 206 及激活管理器 208 通信，它们帮助管理应用的激活和启动。

[0039] 为了支持切换至非运行中的应用，切换器可以预先创建用于当前未运行的应用的飞溅屏幕。该窗口可以由 App 画面 210 和飞溅屏幕 212 来创建——特别是，如果没有用于被杀死应用的当前运行中的窗口和 / 或 UI 屏幕的话。当用户提交了至非运行中的 app 的切换时，切换器可把该窗口切换至窗口激活上下文（context）。当重新启动 app 时，激活代码可能重新使用现有的窗口，而不是创建新的飞溅屏幕。在切换模式的结尾，如果用户不启动与临时画面相关联的 app，则切换器将使该窗口消失。

[0040] 在切换时，系统可能创建一对控件来影响切换：App 画面和放落反馈（Drop Feedback）。在一个实施方案中，切换器一次可以使用放落反馈控件的仅仅一个实例。此外，可能没有由切换器所创建的 App 画面的多于两个的实例，并且通常一次仅有一个。

[0041] 图 3 描绘对于系统的一个实施方案的一个高级别流程图，该系统具有如上所述的 GUI 和操作系统。在 300 处开始，系统可以允许由有意的用户动作驱动应用——例如，在 302，从触敏屏幕的边缘拖动。在 304，系统将试图从返回堆栈中得到应用，并且在 306，检测该应用是否当前正运行。如果不是，那么在 308，系统可以创建用于该应用的飞溅屏幕——并且在 310，无论是这样还是那样（即，无论应用在运行中与否），系统都可以得到对于该应用的缩略图。

[0042] 在312，系统可以安置缩略图——利用或者不利用用户的输入。此外，在314，系统可以得到输入——例如，有关应用的UI屏幕可以驻留于屏幕的何处。在步骤316，系统可询问或以别的方式监视应用是否正在被拖动——并且如果是的话，系统可以在步骤312重复。

[0043] 一旦应用不再被拖动，那么在318，系统可以把前台UI和/或处理切换到该应用。在步骤320，系统可以等待切换输入或者提交。在一个实施方案中，“提交”命令和/或信号指的是对用户是否已经完成切换的确定。例如，这可以是系统对用户想要与当前app交互有较高把握的点。如果当前的app由飞溅屏幕来表示，则这是app将被重新启动的时候。

[0044] 其他实施方案可以选择对于提交的不同触发。例如，提交可以在以下情况中应用：(1)一旦飞溅屏幕被放置于屏幕上时；(2)在任意的超时时刻；(3)在与临时画面交互的时间或者(4)依据与提交相关联的任何用户输入。应当意识到，其他实现可以使用触发的组合，或者其他这样的合适的触发。要满足的是：对于无论何种触发或者命令与“提交”相关联，计算环境应该延迟启动直到用户意图明确。由于启动可能在CPU以及盘使用方面是昂贵的，所以避免不必需的启动会是令人期望的。如果在322处没有提交，则系统将继续至步骤304。然而，如果在步骤322处有提交，那么在324，系统可以检查应用是否正在运行。如果是，那么系统在328处完成。否则，系统将启动应用，并且在326，利用其通常的运转飞溅屏幕替换临时UI。

[0045] 图4描绘一个高级别时序图，其描绘了在切换器组件、飞溅屏幕组件和激活管理器组件之间的处理流程。正如所显示的，在检测到应用拖动后，如果需要(特别是如果应用当前不在执行中)，则切换器组件可以发送信号至飞溅屏幕组件以创建临时UI。此外，切换器组件可以发送用于UI屏幕的位置和显示的信号至飞溅屏幕组件。

[0046] 如果切换器组件检测到提交信号，那么切换器组件可以发送重新启动应用信号至激活管理器组件，并且作为回报，激活管理器组件可以发送交换信号至飞溅屏幕组件。交换信号可以影响临时画面的替换，临时画面用实际应用替换。一旦应用能够表示它自身，就不需要临时画面，并且其可以消失以便释放资源。

[0047] 替换地，如果切换器组件检测到取消拖动(CancelDrag)信号，则切换器组件随后可以发送清理(CleanUp)信号至飞溅屏幕组件。在这种情况下，如果用户选择不切换至临时画面，那么不再需要临时画面并且可以释放其资源。

[0048] 用户界面示例

图5A至5F显示一系列的GUI屏幕，其描绘了用户可能如何经历从返回堆栈中重新启动被杀死的和/或以别的方式被终止的应用。假设用户处于某个计算环境中的正在运行的应用中，并且正在运行第一程序502(比方说，图5A中所描绘的Shutter Web和它的相关联的UI)。再假设计算环境包括活动的触敏屏幕和GUI(例如，带有用于运行中的应用#1的触敏命令516，如所示的)，用户可试图通过有意的手势——比如在图5B中所描绘的从触敏屏幕的左边缘滑动的动作或者手势——从返回堆栈中访问被杀死的应用。

[0049] 临时UI屏幕(表示被杀死的应用)在504处变得可见，带有在点506处的用户的手指支点(digit hold)。经由手指支点，用户可以把临时UI屏幕移动至屏幕上的新位置(例如，移动至右边，如在图5B中所描绘的)。在UI屏幕上做出(或以别的方式与之相关联)的这样的触摸手势也可以用信号通知提交信号——从用户至计算环境——用户期望应用开始执行。在某些实施方案中，触摸手势的结束位置可将信号告知给计算环境，以便或是开始

执行(例如,如果用户的手指支点减慢和/或在触敏显示屏的某个中间点上停止)或是不开始执行(例如,如果用户的数字并未充分地减慢下来和/或用户继续滑动手势以向UI屏幕告知将其移到触敏显示屏表面远处的虚拟“速度”)。

[0050] 在其他实施方案中,“放落画面”的方式可以与用户意图相关联。例如,画面被放落的屏幕的不同区域可与不同的行为相关联。在某些实施方案中,在靠近边缘处放落画面可能表明app不应该被带到屏幕上。在其他地方放落可能表明app应该被带到屏幕上。放落的大概位置可以确定app应该占用屏幕的什么部分。

[0051] 在一个实施方案中,临时UI屏幕可能包括关于被杀死的应用是什么的某些标记——例如,像“天气”之类的临时的(以及或许敷衍的)图标508(或者任何其他合适的App#2)。临时UI屏幕和/或图标可以是一个屏幕,其具有的图形信息和/或功能性比应用正执行或以别的方式运转时经常性地与之相关联的UI屏幕要少。操作系统可能通过用户拖动临时UI屏幕的距离和/或位置而不同地(在用户看来)处理被杀死的应用的呈现。如在图5D中所见的,用户已拖动临时UI屏幕到某个位置(例如,经过整个屏幕的中间部分510)——这可向操作系统表明:用户打算让很快要重新启动的被杀死的应用用掉整个屏幕。

[0052] 在这个示例中,图5E描绘了重新启动的被杀死的应用将占用整个屏幕——因为临时UI屏幕512已经用掉整个屏幕区域。在图5F中,操作系统现在已经成功地启动天气应用(或者任何其他合适的App#2)并且现在正在执行中,正如用其完全运转的正常UI屏幕514所描绘的那样。

[0053] 图6A至6C显示另一系列的GUI屏幕,其描绘用户可能如何经历从返回堆栈中重新启动被杀死和/或以别的方式被终止的应用。图6A再次显示用户的屏幕,其具有当前正在一部分屏幕中执行的一个应用602。在屏幕左手部分,有一个区域用于将要被执行并且将向用户呈现UI屏幕的另一应用。被杀死的应用604(及其相关联的临时UI)正在被拖动至图6B中该左手部分内的位置606,令用户的手指支点被描绘为圆形元素。图6C描绘了系统已经重新启动被杀死的应用并且当前正在其部分屏幕内执行,以及其完整的UI屏幕在608处运转。

[0054] 应当意识到,其他实施方案可允许多于两个apps可以在任何给定时间出现在屏幕上。这样的其他实施方案可选择其他规则来用于在屏幕上布置apps。例如,这些实施方案可牵涉到或可不牵涉到重叠的窗口或者对于所显示apps的数目或那些apps的尺寸的限制。满足本申请的目的的是:安置规则影响在启动app之前安置app的某种方式。

[0055] 上面所描述的内容包括本创新主题的示例。当然不可能为了描述所要求保护的主题而描述组件或方法的每个可想到的组合,但是本领域技术人员可以认识到,本创新主题的许多进一步的组合和排列是可能的。因此,所要求保护的主题旨在包含所有这样的落入所附权利要求的精神和范围之内的变更、修改和变化。

[0056] 除非另外地指明,否则特别地且对于由上述的组件、设备、电路、系统等执行的各种功能而言,被使用来描述这样的组件的术语(包括对“装置”的引用)旨在对应于任何执行所描述组件的所规定功能的组件(例如,功能上的等同),即便其在结构上不等同于所公开的结构,该公开的结构执行在所要求保护主题的本文所例示的示范性方面的功能。就这一点而言,也将认识到,本创新包括系统以及计算机可读介质,所述计算机可读介质具有计算

机可执行指令，用于执行所要求保护的主题的各种方法的动作和 / 或事件。

[0057] 此外，虽然本创新主题的特定特征可能已相对于若干实现中的仅仅一个实现被公开，但这样的特征可与其他实现的一个或者多个其他特征相组合，这对于任何给定的或者特定的应用来说是令人期望的和有利的。此外，就术语“包含”(“includes”和“including”)及其变体在详细说明或者权利要求书中被使用而言，这些术语以与术语“包括”(“ comprising”)类似的方式被规定为包含性的。

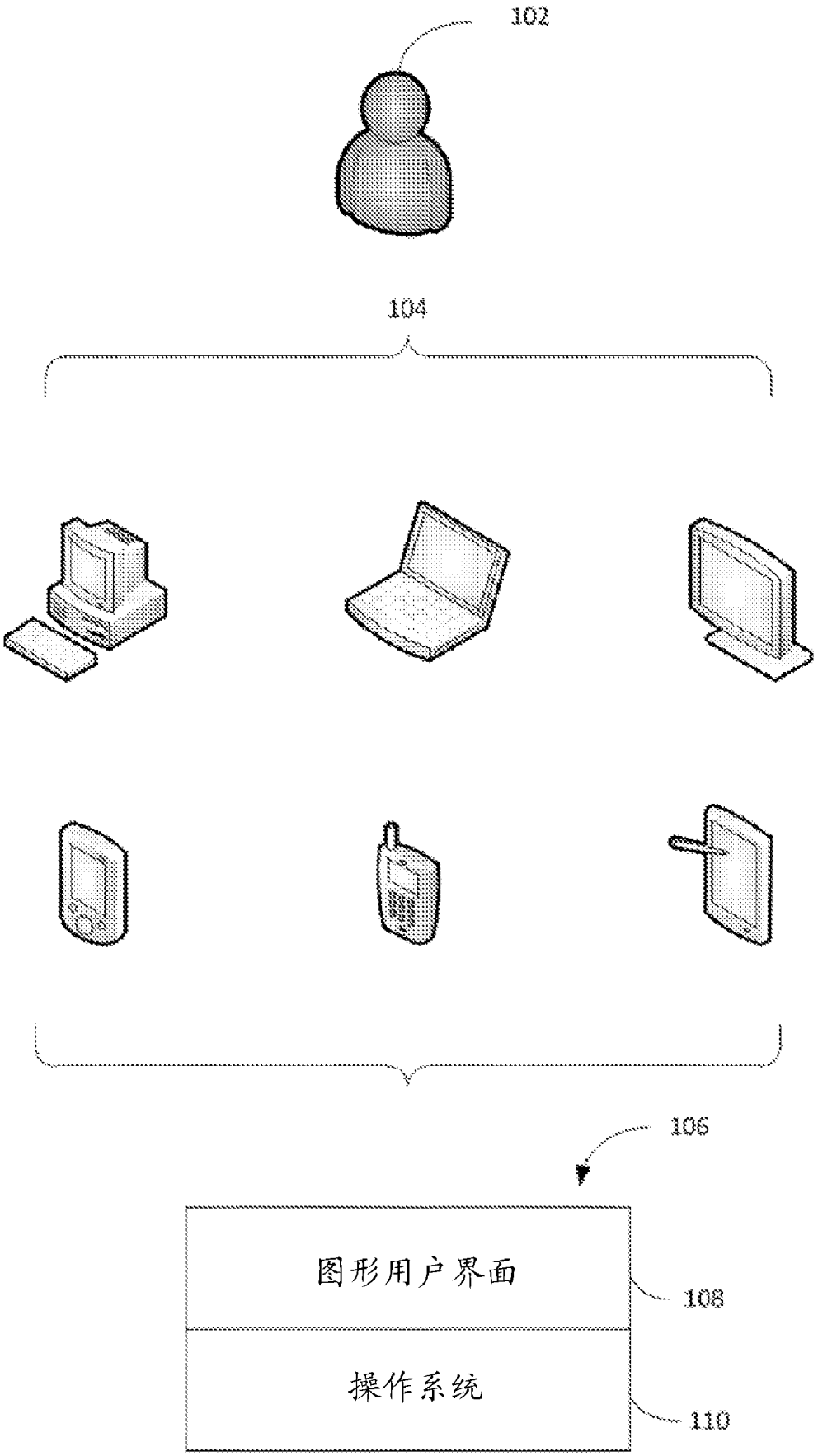


图 1

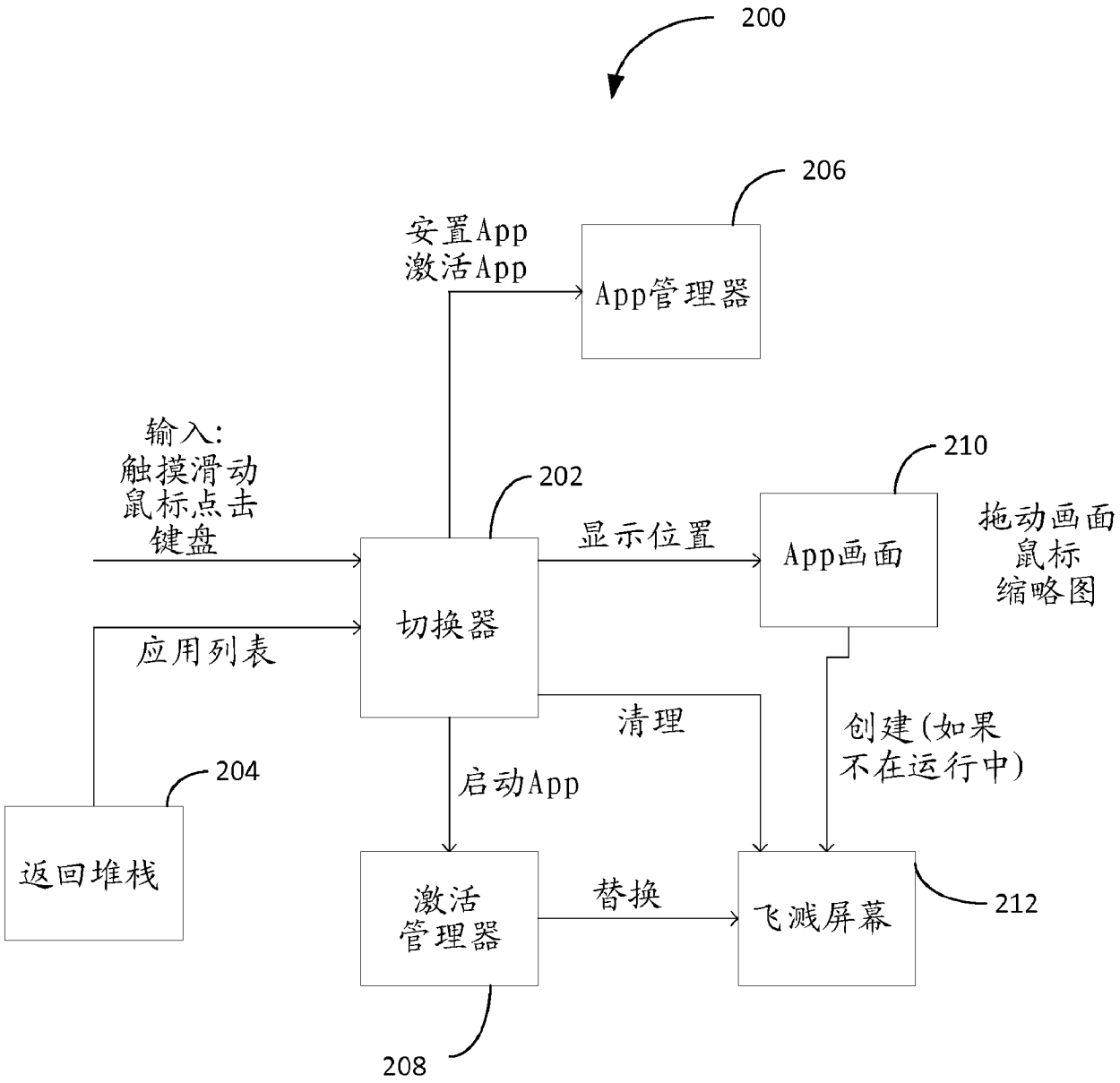

200
206
安置App
激活App
App管理器
输入:
触摸滑动
鼠标点击
键盘
202
显示位置
210
App画面
拖动画面
鼠标
缩略图
切换器
应用列表
清理
创建(如果
不在运行中)
204
启动App
返回堆栈
激活
管理器
替换
飞溅屏幕
212
208

图 2

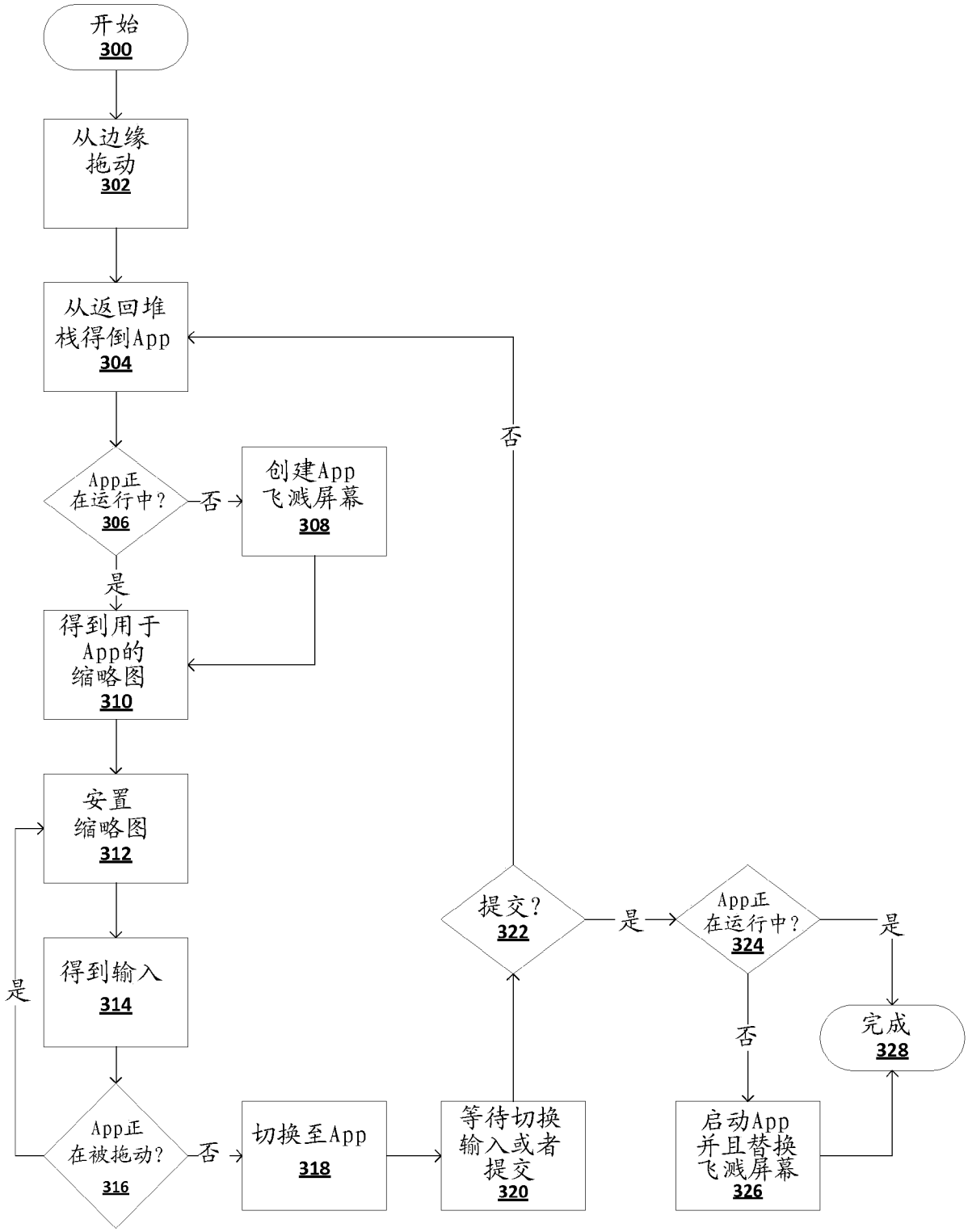

开始
300
从边缘拖动
302
从返回堆栈得倒App
304
App正在运行中?
306
否
创建App飞溅屏幕
308
是
得到用于App的缩略图
310
安置缩略图
312
得到输入
314
App正在被拖动?
316
否
切换至App
318
等待切换输入或者提交
320
提交?
322
否
是
App正在运行中?
324
是
否
启动App并且替换飞溅屏幕
326
完成
328
是

图 3

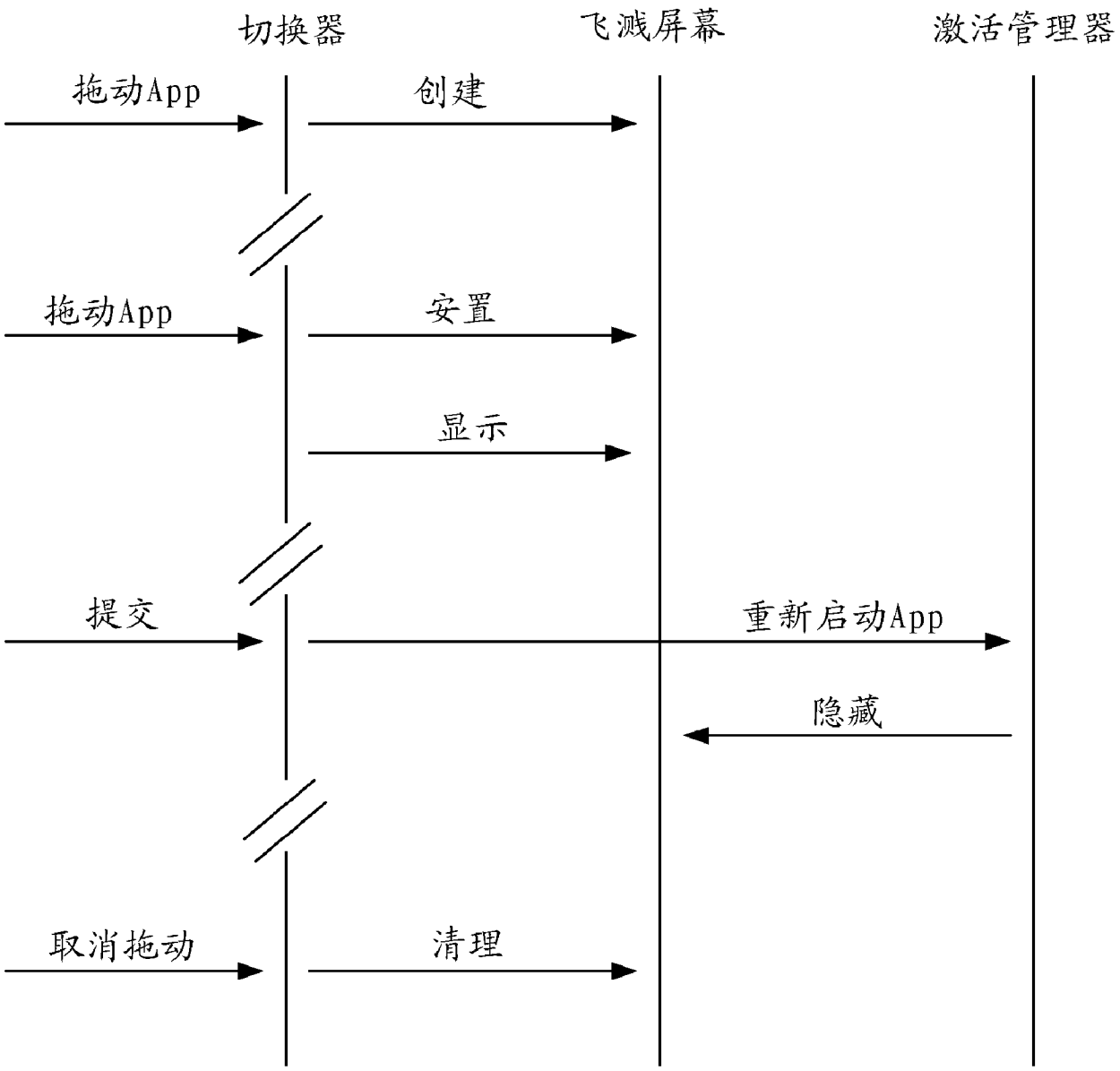


图 4

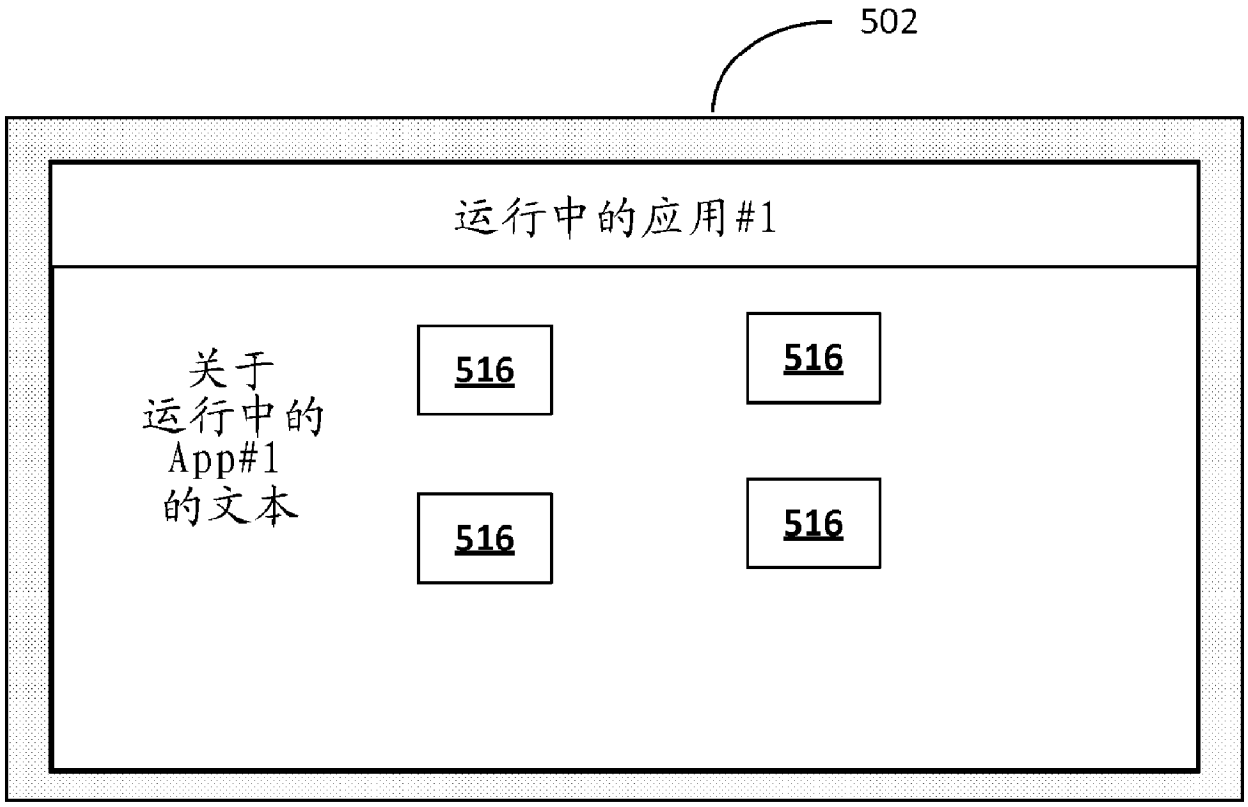


图 5A

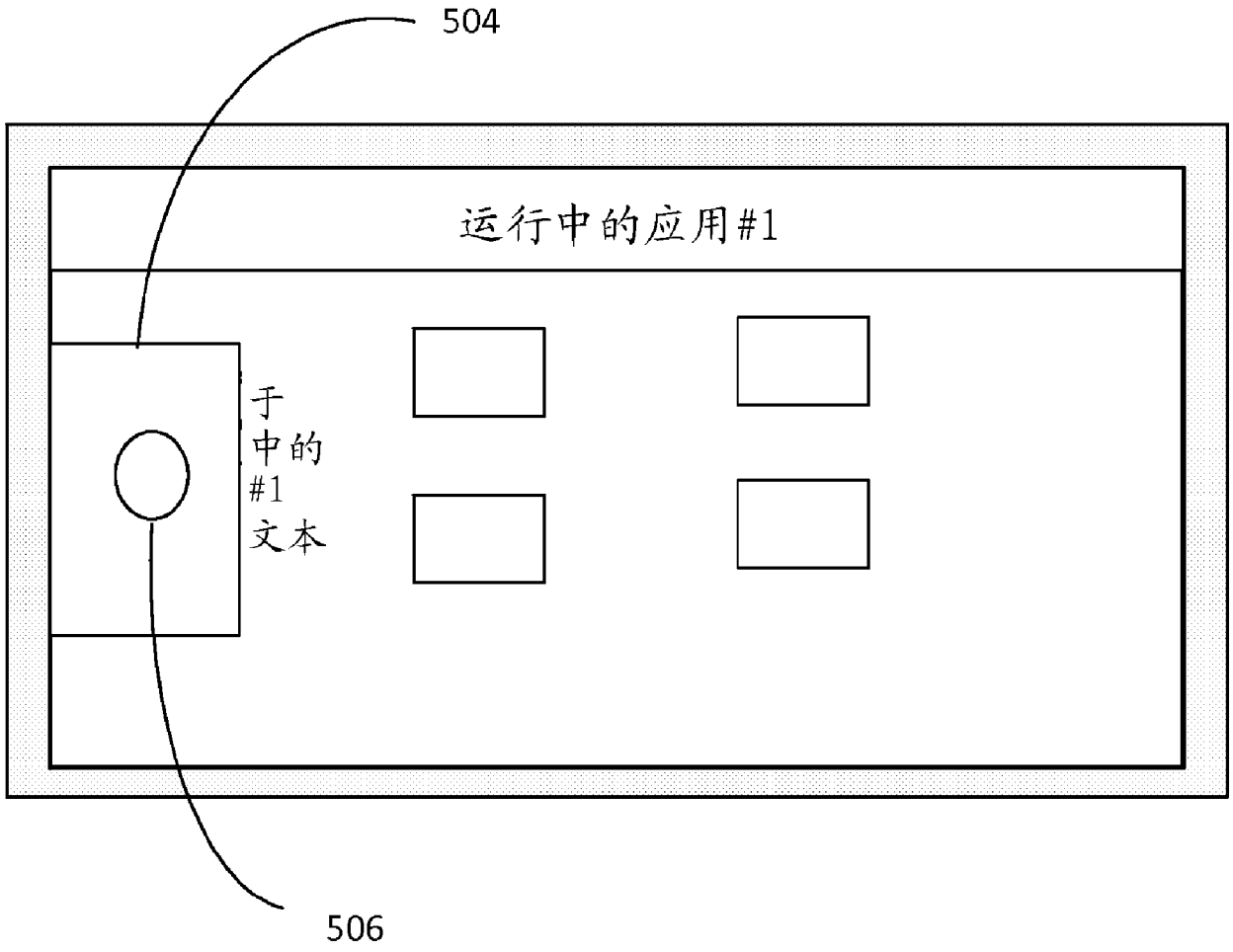


图 5B

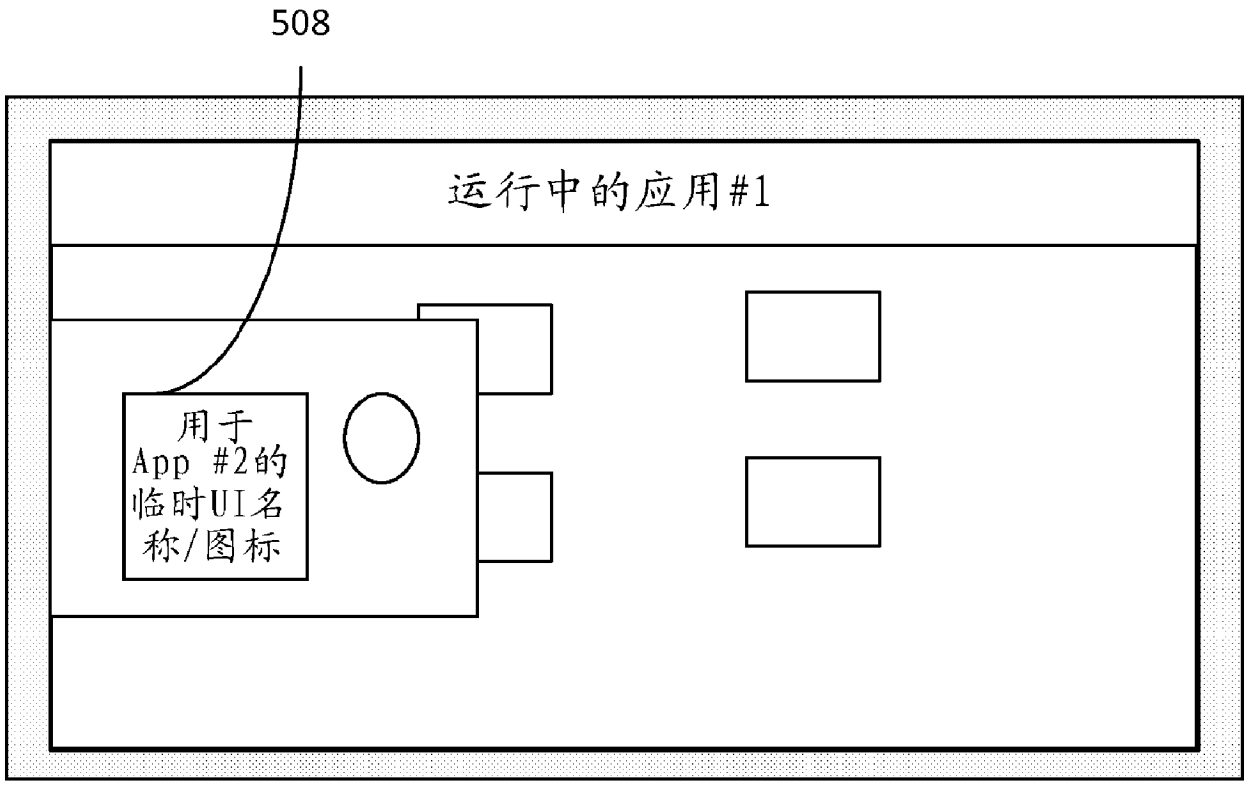

508
运行中的应用#1
用于App #2的临时UI名称/图标

图 5C

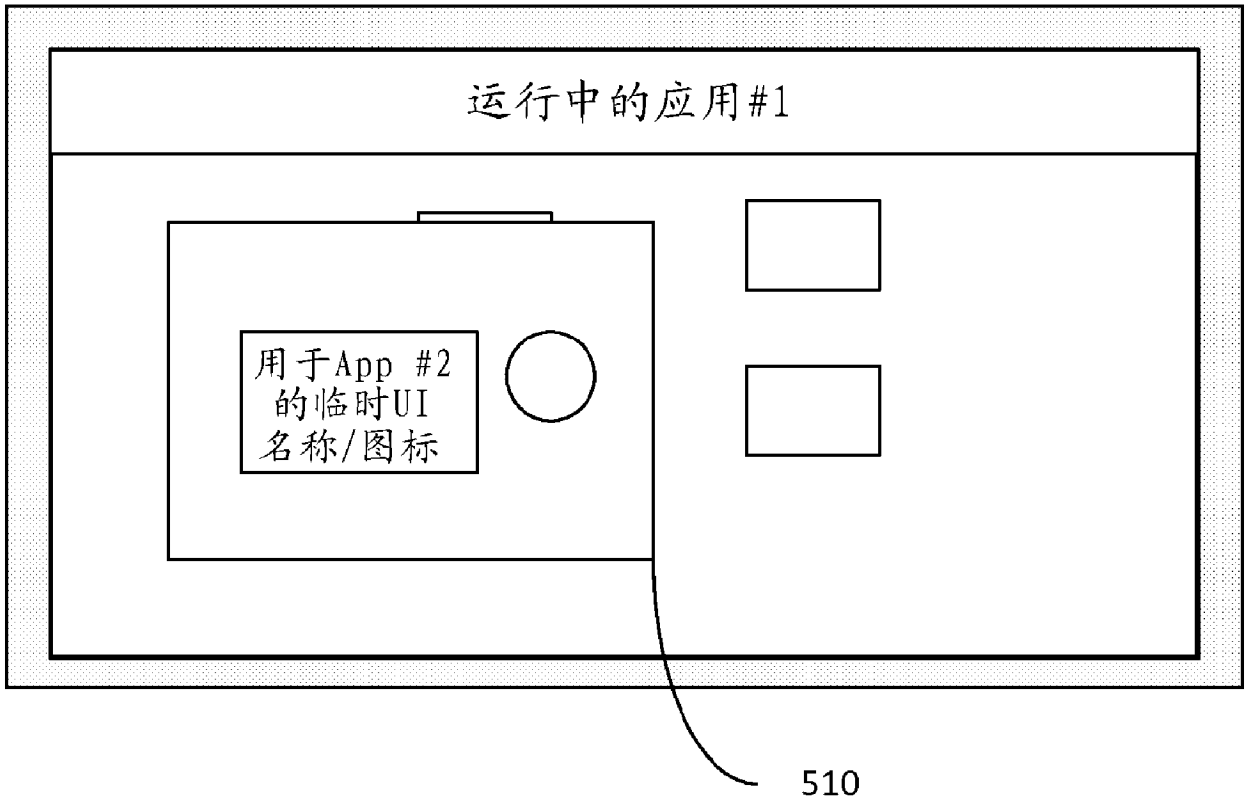

运行中的应用#1
用于App #2的临时UI名称/图标
510

图 5D

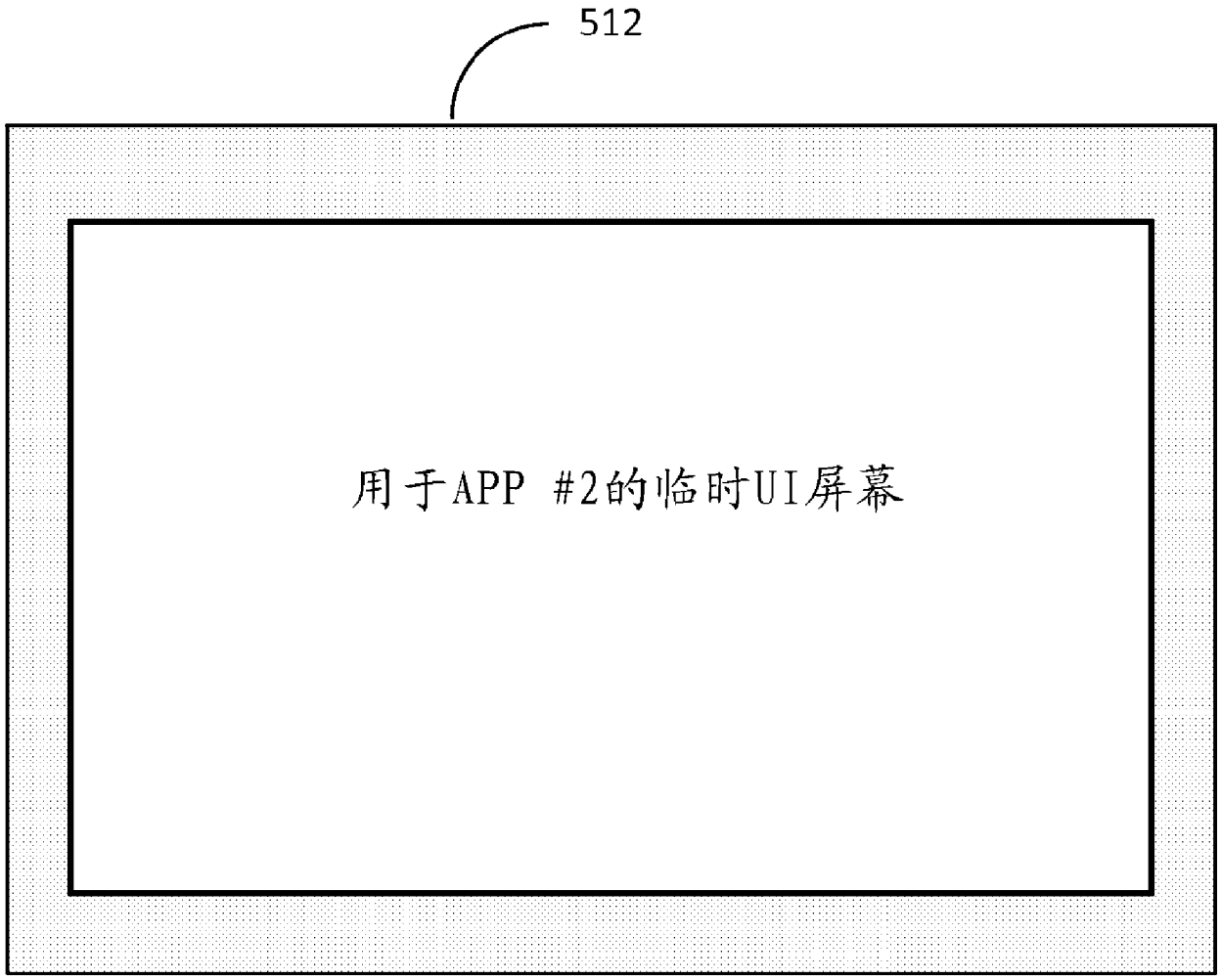


图 5E

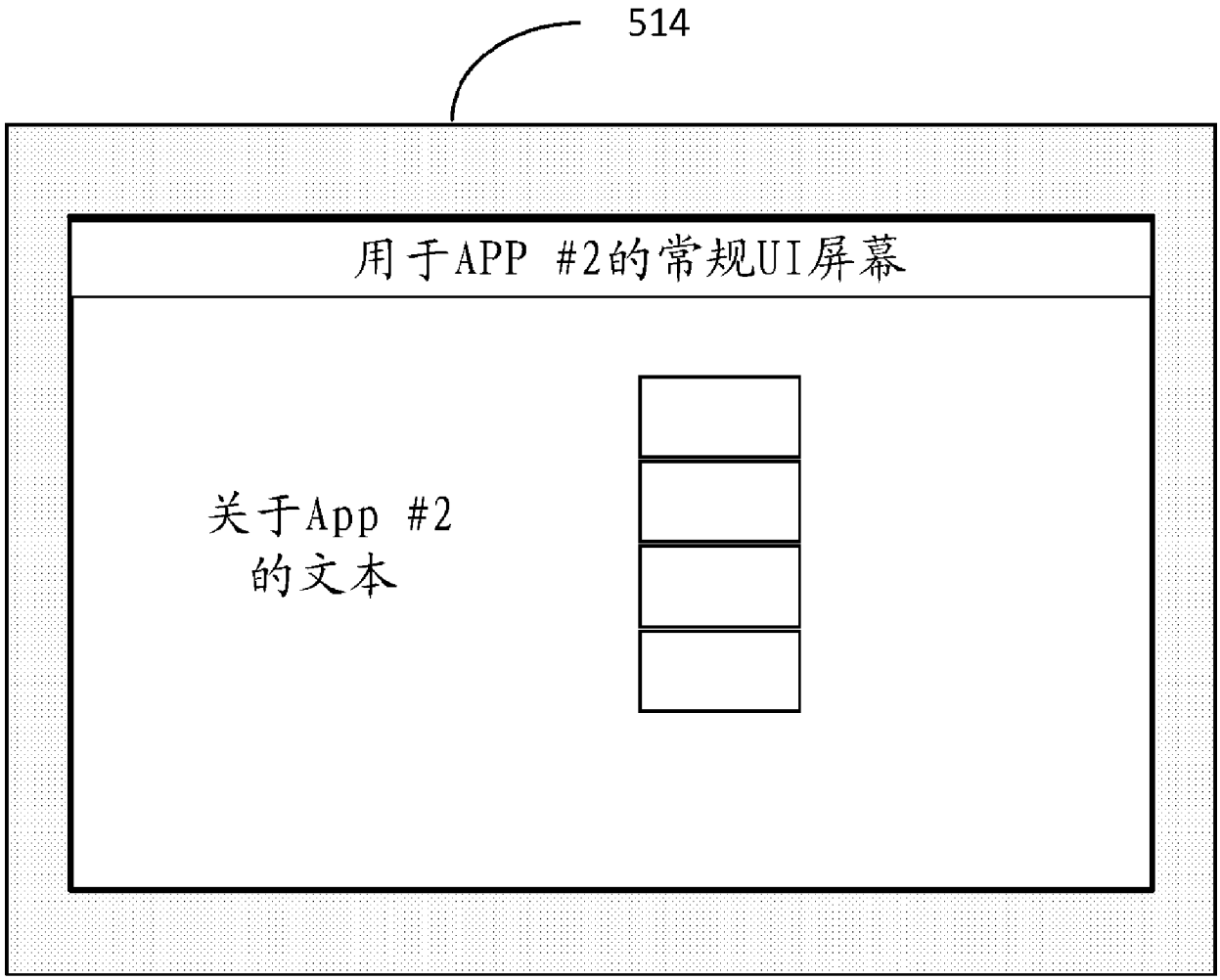


图 5F

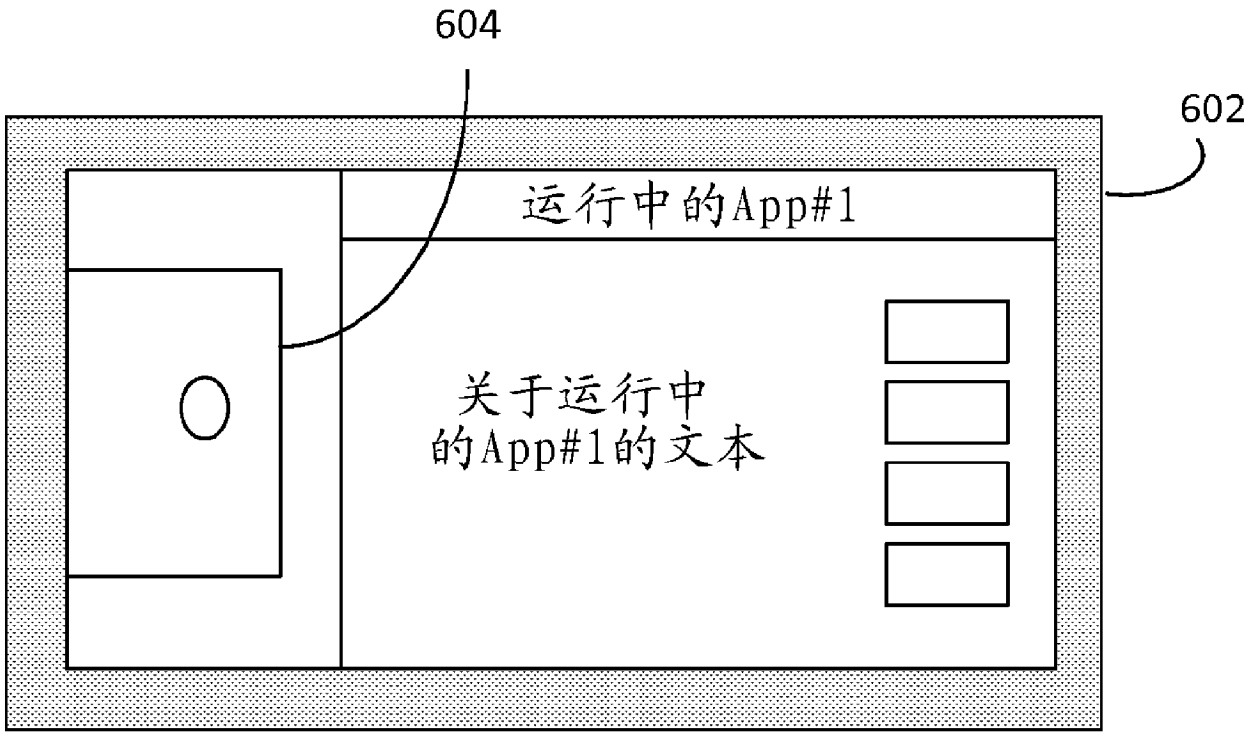

604
602
运行中的App#1
关于运行中
的App#1的文本

图 6A

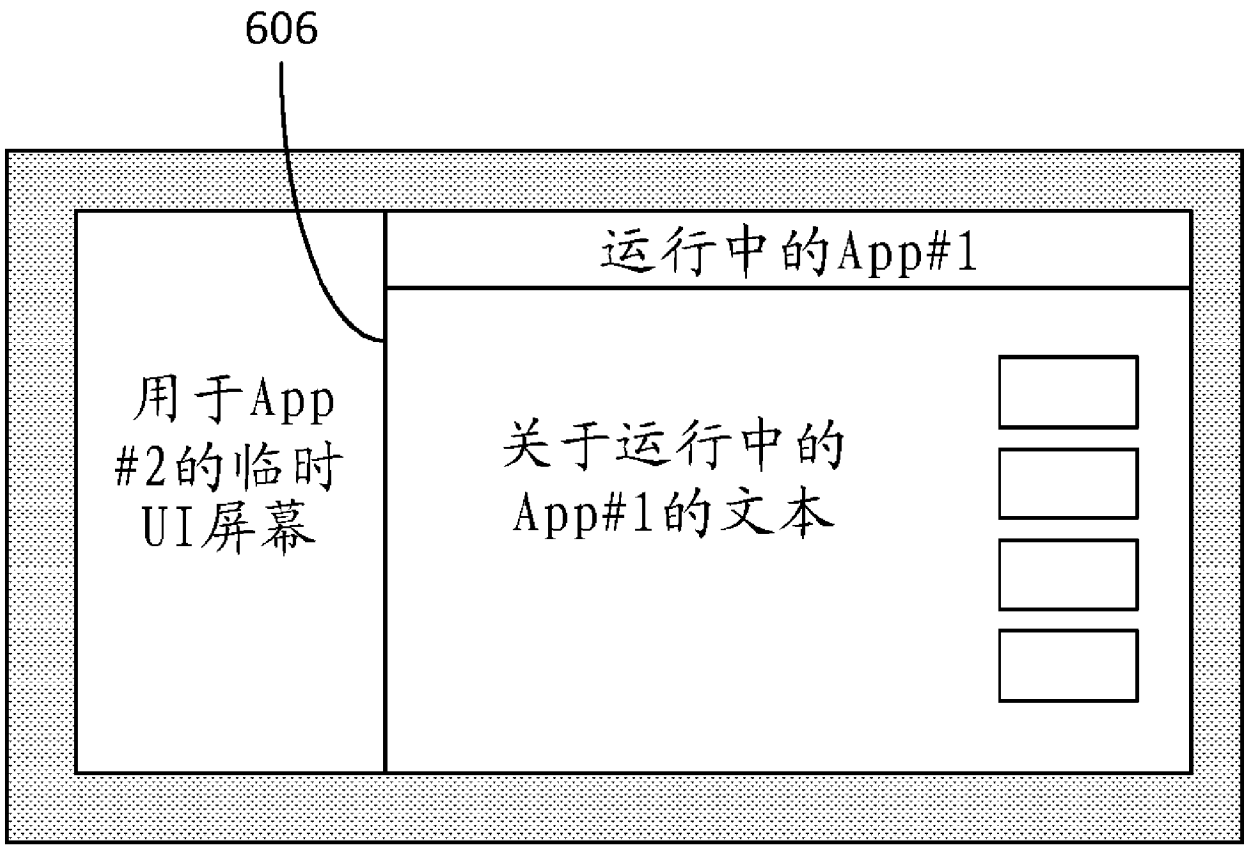

606
运行中的App#1
用于App
#2的临时
UI屏幕
关于运行中的
App#1的文本

图 6B

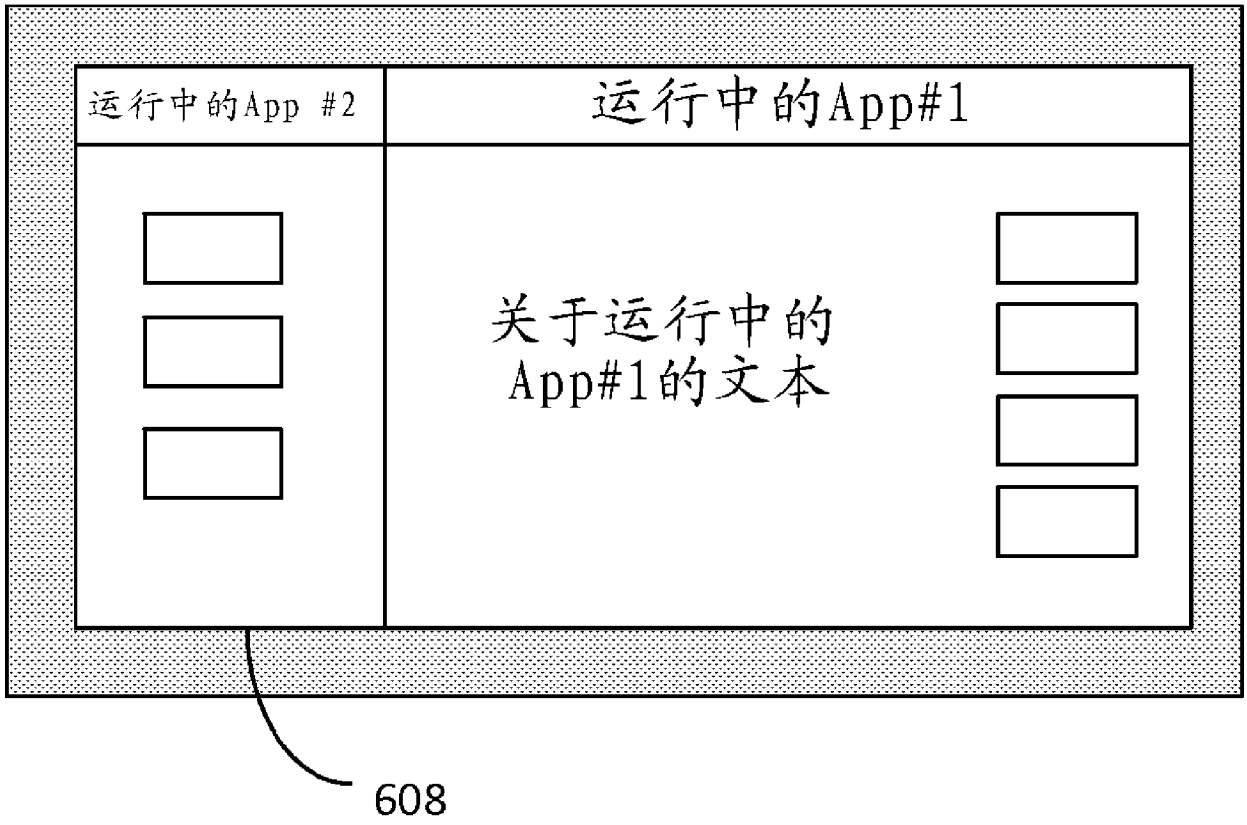


图 6C