(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 102591659 A
(43)申请公布日 2012.07.18

(21)申请号 201110449881.1

(22)申请日 2011.12.28

(71)申请人 中标软件有限公司
地址 200030 上海市番禺路 1028 号 1006-1010 室

(72)发明人 兰雨晴 郭建兴 郭峰 李斌 夏颖

(74)专利代理机构 北京汇智英财专利代理事务所 11301
代理人 陈践实

(51)Int.Cl.
G06F 9/44(2006.01)
G06F 9/45(2006.01)

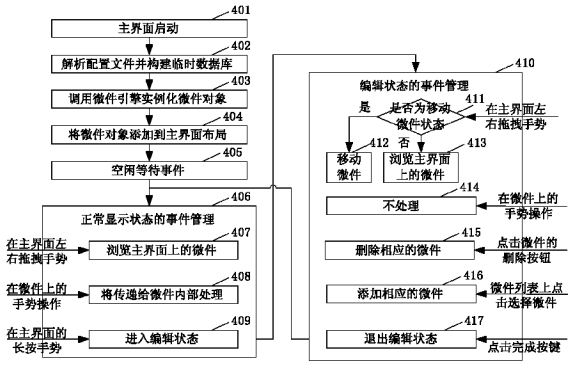
权利要求书 2 页 说明书 6 页 附图 3 页

(54)发明名称

一种移动终端主界面上的微件实现方法及其管理方法

(57)摘要

本发明提供了一种移动终端主界面上的微件实现方法及其管理方法。所述方法包括：主界面上微件的实现方法；主界面通过微件 UI 管理模块对微件的管理方法。微件实现时，编写完微件模块的功能代码，然后编译为一个动态库文件；主界面上添加微件时，微件引擎加载动态库文件，将其中微件模块的类实例化并封装为微件 UI 管理模块能够识别的微件对象。所述微件与主界面采用相同的 UI 框架，通过编译后加载运行的方式，提高了执行效率，在信号传递与处理和手势操作事件上能够做到无缝衔接，且支持多点触控等手势操作。微件的管理功能通过构建临时数据库的方式，提高了对微件进行添加删除和移动位置的管理操作的处理速度。



CN 102591659 A

1. 一种移动终端主界面上的微件实现方法，其特征在于，包括步骤：

（一）微件模块与主界面程序采用相同的 UI 图形控件库实现，编写微件模块的功能代码；

（二）将微件模块编译为一个动态库文件；

（三）用户在主界面上添加微件，微件引擎加载相应的微件模块动态库文件；

（四）将动态库文件中的微件模块通过微件引擎，将微件模块的类实例化为微件模块对象，并进一步封装为为微件 UI 管理模块能够识别的微件对象。

2. 根据权利要求 1 所述的移动终端主界面上的微件实现方法，其特征在于，所述微件模块的功能代码是用 Qt 编写的实现微件功能的类。

3. 根据权利要求 1 所述的移动终端主界面上的微件实现方法，其特征在于，步骤一中：用户编写完成微件要实现的功能后，将其编译为一个动态库文件，这样每一个微件就是一个动态库文件，用户根据个人需要在主界面上添加不同的微件，微件引擎只加载需要的动态库文件。

4. 根据权利要求 1 所述的移动终端主界面上的微件实现方法，其特征在于，上述步骤三中的具体方法为：

（1）初始化微件运行环境，为微件功能对象构造一个保存缓存数据的数据结构；

（2）根据微件模块动态库文件路径，加载并打开微件模块的动态库文件，并将其转为一个对象；

（3）通过打开的动态库文件对象，调用微件模块提供的创建微件模块对象接口，此接口的返回值即为微件模块对象；

（4）将获取的微件模块对象与微件 UI 管理模块需要的一些环境变量等信息进一步封装为微件对象，作为“创建微件对象接口”的返回值传给微件 UI 管理模块。

5. 根据权利要求 4 所述的移动终端主界面上的微件实现方法，其特征在于，前述的步骤（1）中，所述缓存数据为微件运行数据缓存的数据，数据结构以指针形式作为创建微件模块对象接口的参数传给微件模块对象使用。

6. 根据权利要求 4 所述的移动终端主界面上的微件实现方法，其特征在于，前述的步骤（2）中，使用 Qt 的 QPluginLoader 加载并打开微件模块的动态库文件，并将其转为一个对象。

7. 一种如权利要求 1-6 任一项所述移动终端主界面上的微件实现方法的微件管理方法，其特征在于，移动终端系统中配置有：微件引擎模块、配置文件的读写模块、临时数据库管理模块、微件列表创建与更新模块和主界面上微件 UI 管理模块；所述主界面上的微件信息通过临时数据库进行管理，具体步骤为：

主界面初始化时解析配置文件，构建临时数据库；

当在主界面上对微件进行添加删除和移动位置的管理操作时，通过微件 Ui 管理模块提供的接口更新临时数据库；

在退出主界面时再将修改的微件列表及其位置信息回写到配置文件中。

8. 根据权利要求 7 所述的微件管理方法，其特征在于：所述微件的位置信息保存在配置文件中，保存的信息包括需要在主界面中加载的微件列表及其对应的位置信息，其中微件列表为编译出的微件模块动态库文件的路径组成的列表，然后每个文件路径后面是微件

在主界面上的位置序列号。

9. 根据权利要求 7 所述的微件管理方法，其特征在于：所述配置文件的读写模块，用于在系统初始化时读取配置文件，提供给临时数据库管理模块初始化临时数据库；并在退出主界面时，从临时数据库取微件信息更新到配置文件。

一种移动终端主界面上的微件实现方法及其管理方法

技术领域

[0001] 本发明属于移动终端技术领域，尤其涉及移动终端主界面上微件实现方法及其管理的方法。

背景技术

[0002] 微件(Widget)是一种基于互联网的小应用，结合网络向用户提供天气，新闻，备忘等功能。微件基于 Web 技术实现，具有小巧轻便、易于开发、开发周期短、功能完整等优点，现在已扩展到各种 PC 桌面和移动终端上。

[0003] 移动微件是将微件理念移植到移动终端，以微件的形式将一些简单实用的功能部署到移动终端上的应用。移动微件不仅可以独立于浏览器运行，还能有效地利用终端屏幕，让用户更加快速、直接、方便地访问移动互联网，它给移动终端用户带来了良好的呈现方式和互联网体验。随着移动微件的不断发展，微件的应用不仅局限于网络应用，还包括大量的用户经常使用的小功能，用户不需要打开功能完整的应用程序就可以方便快捷地完成自己想做的工作。

[0004] 一个完整的微件运行环境包括操作系统 / 硬件平台层、微件引擎层、微件应用层。微件引擎处于微件运行系统的核心部位，向下微件引擎可以运行在操作系统 / 硬件平台上，向上微件引擎为微件应用提供运行环境。微件应用层主要负责将微件引擎解析生成的微件应用显示在主界面上，并通过微件管理模块对微件应用进行管理。

[0005] 在实现本发明的过程中，发明人发现现有的微件生成与管理过程中存在以下缺点：

(1) 微件一般是基于 HTML、JavaScript、CSS 或 XML 等计算机语言实现，相应的微件引擎是基于 Webkit 封装的解析程序。由于采用这种解析方式的运行效率较低，这类微件实现的功能比较单一，提供的交互操作也很简单，如果用户需要实现功能相对完善且能够提供友好交互的微件时，这种方式就有些力不从心了。

[0006] (2) 在微件的管理上，各终端根据其操作系统的技术框架提供适合各自技术要求和界面风格的管理方法，大多采用 XML 对微件配置参数及运行状态进行管理，这样需要管理模块实时对 XML 的管理文件进行解析与修改，效率不高，且相对比较复杂。

[0007] (3) 目前各移动终端操作系统对微件的大小、形状以及放置位置不做什么要求，在方便用户调整的同时，也导致了界面凌乱等问题。

发明内容

[0008] 为了解决上述技术问题，本发明提供了一种移动终端主界面上的微件实现方法，包括步骤：

(一) 微件模块与主界面程序采用相同的 UI 图形控件库实现，编写微件模块的功能代码；

(二) 将微件模块编译为一个动态库文件；

(三)用户在主界面上添加微件,微件引擎加载相应的微件模块动态库文件;

(四)通过微件引擎将动态库文件中的微件模块的类实例化为微件模块对象并封装为微件 UI 管理模块能够识别的微件对象。

[0009] 其中,所述微件模块是指实现微件的类,是用 Qt 编写的;

所述微件模块对象是将微件模块的类实例化的对象,微件应用在运行状态下内部的核心对象;

所述微件对象是指将微件模块对象经过进一步封装的对象,是微件应用的运行态。

[0010] 步骤一中:用户编写完成微件要实现的功能后,将其编译为一个动态库文件,这样每一个微件就是一个动态库文件,用户根据个人需要在主界面上添加不同的微件,微件引擎只加载需要的动态库文件。

[0011] 上述步骤(三)中的具体方法为:

(1)初始化微件运行环境,为微件功能对象构造一个保存缓存数据的数据结构;

(2)根据微件模块动态库文件路径,加载并打开微件模块的动态库文件,并将其转为一个动态库文件对象;

(3)通过打开的动态库文件对象,调用微件模块提供的创建微件模块对象接口,此接口的返回值即为微件模块对象;

(4)将获取的微件模块对象与微件 UI 管理模块需要的一些环境变量等信息进一步封装为微件对象,作为“创建微件对象接口”的返回值传给微件 UI 管理模块。

[0012] 前述的步骤(1)中,所述缓存数据为微件运行数据缓存结构,数据结构以指针形式作为创建微件模块对象接口的参数传给微件模块对象使用。

[0013] 前述的步骤(2)中,使用 Qt 的 QPluginLoader 加载并打开微件模块的动态库文件,并将其转为一个动态库文件对象。

[0014] 一种如前所述移动终端主界面上的微件实现方法的微件管理方法,移动终端系统中配置有:微件引擎模块、配置文件的读写模块、临时数据库管理模块、微件列表创建与更新模块和主界面上微件 UI 管理模块;所述主界面上的微件信息通过临时数据库进行管理,具体步骤为:主界面初始化时解析配置文件,构建临时数据库;当在主界面上对微件进行添加删除和移动位置的管理操作时,通过微件 UI 管理模块提供的接口更新临时数据库;在退出主界面时再将修改的微件列表及其位置信息回写到配置文件中。

[0015] 所述微件的位置信息保存在配置文件中,保存的信息包括需要在主界面中加载的微件列表及其对应的位置信息,其中微件列表为编译出的微件动态库模块文件的路径组成的列表,然后每个文件路径后面是微件在主界面上的位置序列号。

[0016] 所述配置文件的读写模块,用于在系统初始化时读取配置文件,提供给临时数据库管理模块初始化临时数据库;并在退出主界面时,从临时数据库取微件信息更新到配置文件。

[0017] 本发明的有益效果为:微件与主界面都采用 UI 图形控件库 Qt,通过编译后加载运行的方式,与传统的使用解释语言写的微件相比,提高了执行效率,能够使用相同图形控件库的接口进行开发,在信号传递与处理和手势操作事件的处理上能够做到无缝衔接,且支持多点触控等手势操作;微件的管理功能通过构建临时数据库的方式,提高了对微件进行添加删除和移动位置的管理操作的处理速度;对微件的大小、背景和位置进行了规范,从而

使得移动终端主界面上微件布局更加整齐美观，添加、删除和移动微件时的动画处理进一步增强了用户体验。

附图说明

[0018] 图 1 为微件引擎加载微件的调用关系示意图；

图 2 为本发明实施例提供的主界面正常显示模式的微件示意图；

图 3 为本发明实施例提供的主界面进入编辑状态的微件示意图；

图 4 为本发明实施例提供的主界面上微件初始化过程的流程图；

图 5 为本发明实施例提供的创建微件列表微件的流程图；

图 6 为本发明实施例提供的移动终端主界面上微件管理系统的结构框图。

具体实施方式

[0019] 为了使本发明的目的、技术方案及有益效果更加清楚明白，以下结合附图及实施例，对本发明进行进一步详细说明。应当理解，此处所描述的具体实施例仅用以解释本发明，并不用于限定本发明。

[0020] 在本发明实施例中，微件引擎加载微件运行的方法是通过接口调用，将微件模块实例化并封装为一个微件对象提供给微件 UI 管理模块放置到主界面中。微件模块与主界面程序采用相同的 UI 图形控件库 Qt 实现，开发人员用 C++ 开发语言编写微件模块的功能代码，每个微件模块的代码就是一个类，然后将其编译为一个动态库文件。这样每一个微件模块就是一个动态库文件，用户根据个人需要在主界面上添加不同的微件，微件引擎加载相应的微件模块动态库文件，然后将动态库文件中的微件模块通过微件引擎，将微件模块的类实例化并封装为微件 UI 管理模块能够识别的微件对象。

[0021] 微件引擎加载微件的调用关系如图 1 所示，图中涉及三个模块：微件 UI 管理模块（101）、微件引擎模块（102）和微件模块（103）。其中微件模块是根据用户需要，开发人员开发的各个微件实现的类。

[0022] 微件模块需要实现：（1）为用户实现微件的功能，即微件功能模块（111），这是各个微件的核心功能，是微件模块的一部分微件功能模块就是用 Qt 编写的实现微件功能的类；（2）提供创建微件模块对象接口（106），通过该接口将微件功能模块的类实例化为微件模块对象，并将此对象作为接口返回值，这是每个微件模块的调用接口，格式要求是统一的，供微件引擎模块（102）调用。

[0023] 微件引擎模块（102），是实现微件引擎的类，也是采用 Qt 编写的。微件引擎模块（102）提供创建微件对象接口（105）供微件 UI 管理模块（101）调用，该模块主要负责初始化微件运行环境，加载微件模块的动态库文件，通过微件模块提供的接口提取微件模块对象，并封装为微件对象。具体的创建微件对象的流程如下：

（1）初始化微件运行环境（107），需要为微件模块对象构造一个保存缓存数据的数据结构——微件运行数据缓存结构，数据结构以指针形式作为创建微件模块对象接口（106）的参数传给微件模块对象使用。

[0024] （2）根据微件模块动态库文件路径，使用 Qt 的 QPluginLoader 加载并打开微件模块的动态库文件（108），并将其转为一个对象。

[0025] (3)通过打开的动态库文件对象,调用微件模块提供的创建微件模块对象接口(109),此接口的返回值即为微件模块对象。

[0026] (4)将获取的微件模块对象与微件 UI 管理模块需要的一些环境变量等信息进一步封装为微件对象,作为“创建微件对象接口”的返回值传给微件 UI 管理模块(110)。

[0027] 当主界面的微件 UI 管理模块(101)需要添加一个微件到主界面时,通过模块内的添加微件接口 (104) 将微件模块动态库文件路径作为参数传递给微件引擎模块(102)的创建微件对象接口(107)。微件引擎模块初始化微件运行环境,根据微件模块动态库文件路径加载微件模块动态库文件,并调用微件模块的创建微件模块对象接口(106)获取微件模块对象,然后微件引擎将微件模块对象封装为微件对象作为添加微件接口(104)的返回值返回给微件 UI 管理模块。

[0028] 本发明中微件与主界面都采用 UI 图形控件库 Qt,通过编译后加载运行的方式,与传统的使用解释语言写的微件相比,提高了执行效率,能够使用图形支持库的接口进行开发,在信号传递与处理和手势操作事件的处理上能够做到无缝衔接,且支持多点触控等手势操作 ;微件的管理功能通过构建临时数据库的方式,提高了对微件进行添加删除和移动位置的管理操作的处理速度 ;对微件的大小、背景和位置进行了规范,从而使得移动终端主界面上微件布局更加整齐美观,添加、删除和移动微件时的动画处理进一步增强了用户体验。

[0029] 为了实现移动终端主界面上微件 UI 管理模块对微件的管理,分为两个步骤,一是微件在主界面上的显示与控制方式 ;二是主界面进入编辑状态后对微件进行管理的方法。图 2 是本发明实施例的主界面正常显示模式下的微件示意图,图 3 是本发明实施例的主界面进入编辑状态后的微件示意图,此时可以对微件进行添加、删除和移动位置的管理操作。

[0030] 图 4 示出了本发明实施例提供的主界面上微件管理方法的实现流程图,其详细步骤如下所述 :

在步骤 401 中,系统启动进入主界面初始化程序,包括 UI 界面的绘制、信号连接和事件管理初始化等等。

[0031] 主界面初始化时解析配置文件,构建临时数据库,当在主界面上对微件进行添加删除和移动位置的管理操作时,通过微件 UI 管理模块提供的接口更新临时数据库,在退出主界面时再将修改的微件列表及其位置信息回写到配置文件中。

[0032] 在步骤 402 中,微件 UI 管理模块解析配置文件,并构建临时数据库。在本发明实施例中,根据配置文件生成的临时数据库是实现微件管理的基础,而每个微件模块就是一个动态库文件,配置文件保存微件模块动态库文件的路径和在主界面上的位置信息。微件 UI 管理模块从临时数据库中按照位置顺序依次调用微件引擎创建微件对象(403),创建微件对象的过程如图 1 所示,然后将创建的微件对象按照配置文件中相应的位置信息添加到主界面布局中(404),此时主界面的初始化工作完成,用户可以进行操作了,主程序即进入了空闲等待事件的状态(405)。

[0033] 在本发明实施例中,主程序的事件管理主要是指接受用户手势操作的事件,包括两种状态,一个是在主界面初始化后的正常显示状态的事件管理(406),另一个是进入编辑状态的事件管理(410)。下面分别进行说明 :

在正常显示状态下的事件管理(406),主要处理三类手势操作 :

(1)在主界面上左右拖拽手势,用于浏览主界面上的微件(407),用户向左拖拽主界面,则主界面向左滑动,右侧的微件显示在屏幕上;用户向右拖拽的界面控制亦如此。

[0034] (2)在微件上的手势操作,如手指点击、上下拖拽、多点触摸等等手势操作事件全部传递给微件内部进行处理,微件根据功能需要,可以截获此事件完成相应的功能(408)。

[0035] (3)在主界面上的长按手势,进入编辑状态(409)。进入编辑状态后,从界面上与正常显示状态有明显区别:作为本发明的一个优选实施例,各微件横向间隙扩大,横向间隙设为 21 个像素,但其并不用以限制本发明;并在每个微件右上角显示一个删除按钮用于删除微件;各微件不再响应手势操作;在主界面上显示一个“完成”按钮供用户退出编辑状态;在主界面最右侧添加一个微件列表微件,其外观与其他微件相同。

[0036] 在编辑状态下的事件管理(410),主要包括以下六类事件:

(1)在主界面左右拖拽手势,需要先判断当前是否为移动微件状态(411),如果是,则开始移动微件(412),如果不是,则浏览主界面上的微件(413),与步骤 407 相同。在移动微件状态下,手指左右拖动微件,微件跟随手指左右移动,插入到前后的微件之间,微件移动到目标位置,手指放开即拖拽手势完成时,退出移动微件状态。

[0037] (2)在微件上的手势操作,如手指点击、上下拖拽、多点触摸等等手势操作事件,不进行处理(414)。作为本发明的一个优选实施例,在微件上层覆盖一半透明模板用于截取手势事件,但其并不用以限制本发明。

[0038] (3)点击微件上的删除按钮,删除相应的微件(415)。作为本发明的一个优选实施例,微件上的删除按钮全部放置在微件背景的右上角位置,所有微件删除按钮水平高度一致,但其并不用以限制本发明。删除微件时,以渐进缩小和透明化的动画让微件慢慢消失;最后在临时数据库中删除此微件,及时更新其他微件的位置信息,并在微件列表微件中添加此微件名称。

[0039] (4)在微件列表微件上点击列表上的微件名称,添加相应的微件(416)。点击微件列表中要添加的微件名称;然后主界面的微件 UI 管理模块调用微件引擎创建微件对象,然后将微件用从左到右渐进挤出的动画方式插在微件列表微件的左侧;最后将刚添加的微件模块动态库文件路径和在主界面中的位置信息添加到临时数据库中,并在微件列表微件中删除此微件名称。

[0040] (5)在微件上长按手势,进入微件移动状态(417),此时手指不得离开屏幕,左右拖拽微件来移动微件,手指一旦离开屏幕,则退出微件移动状态。

[0041] (6)点击“完成”按钮,则退出编辑状态,返回到正常显示状态(406)。

[0042] 作为在本发明实施例中,创建微件列表微件的流程图参照图 5 所示。在步骤 501 中,初始化微件列表微件的界面,主要是搭建微件 UI 框架,顶部是一个标签说明;下方是一个可上下拖拽浏览列表的拖拽层,上面放置一个各微件名称列表。然后,在存放微件模块动态库文件的目录中依次读取各动态库文件(502),并解析出各微件名称形成微件名称列表(503)。

[0043] 然后根据临时数据库整理列表,只保留数据库中没有的微件名称,具体方法如下:

首先提取微件名称列表的一项(504),判断其对应的微件模块动态库文件是否在临时数据库中(505),如果是,则从微件名称列表中删除此微件名称(506);如果不是,则判断是

否已到微件名称列表末尾(507),如果不是,则重复步骤 504 至 507 ;如果是,则表示完成了整理列表。最后在步骤 508 中,将保留的微件名称列表以控件形式布局在微件中。

[0044] 图 6 示出了本发明实施例提供的移动终端主界面上微件管理系统的结构框图。为了便于说明,仅示出了与本发明相关的部分。所述微件管理系统包括配置文件的读写模块 601、临时数据库管理模块 602、微件引擎模块 603、微件列表创建与更新模块 604、微件 UI 管理模块 605。

[0045] 配置文件的读写模块 601,用于在系统初始化时读取配置文件,提供给临时数据库管理模块 602 初始化临时数据库 ;并在退出主界面时,从临时数据库取微件信息更新到配置文件。

[0046] 临时数据库管理模块 602,用于根据解析的配置文件构建临时数据库,在添加、删除微件和移动微件位置时更新临时数据库。

[0047] 微件引擎模块 603,用于在主界面初始化以及添加微件时,根据微件模块动态库文件路径,创建微件对象,并返回给主界面 UI 管理模块显示在主界面上。

[0048] 微件列表创建与更新模块 604,首先读取所有的微件模块动态库文件,然后去掉临时数据库中已添加在主界面的微件,就是可添加的微件列表,然后将其显示在微件列表微件上 ;在添加微件时,将微件名称从列表删除 ;在删除微件时,将微件名称添加到列表中。

[0049] 微件 UI 管理模块 605,负责微件对象的显示,响应用户的手势操作,进入和退出编辑状态,以及添加、删除和移动微件时的动画处理等功能。

[0050] 以上所述仅为本发明的较佳实施例而已,并不用以限制本发明,凡在本发明的精神和原则之内所做的任何修改、等同替换和改进等,均应包含在本发明的保护范围之内。

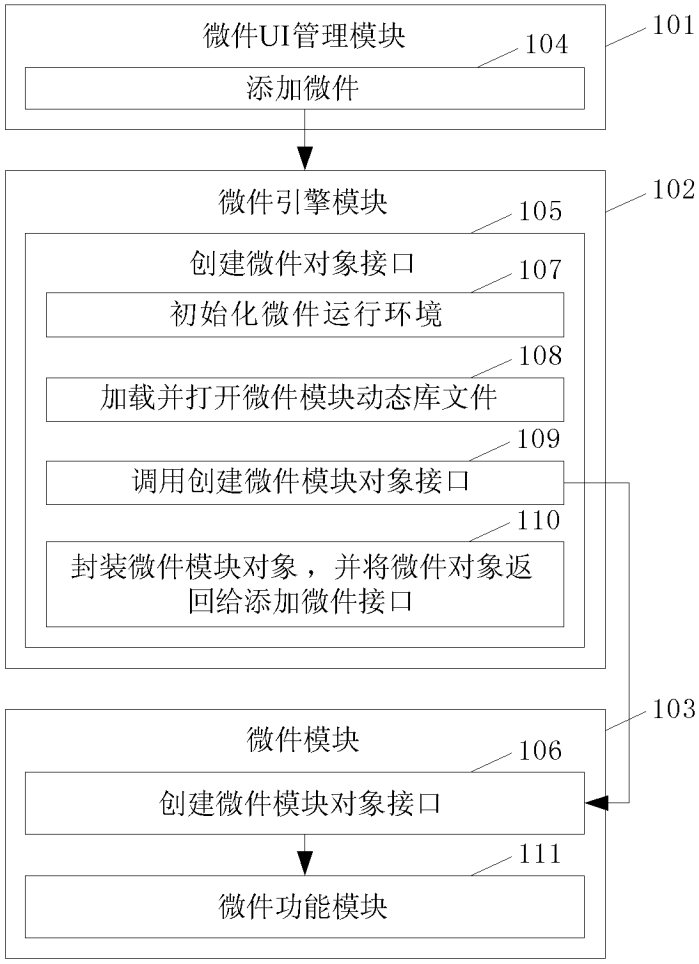

微件UI管理模块
104
添加微件
101
微件引擎模块
105
创建微件对象接口
107
初始化微件运行环境
108
加载并打开微件模块动态库文件
109
调用创建微件模块对象接口
110
封装微件模块对象，并将微件对象返回给添加微件接口
102
微件模块
106
创建微件模块对象接口
111
微件功能模块
103

图 1

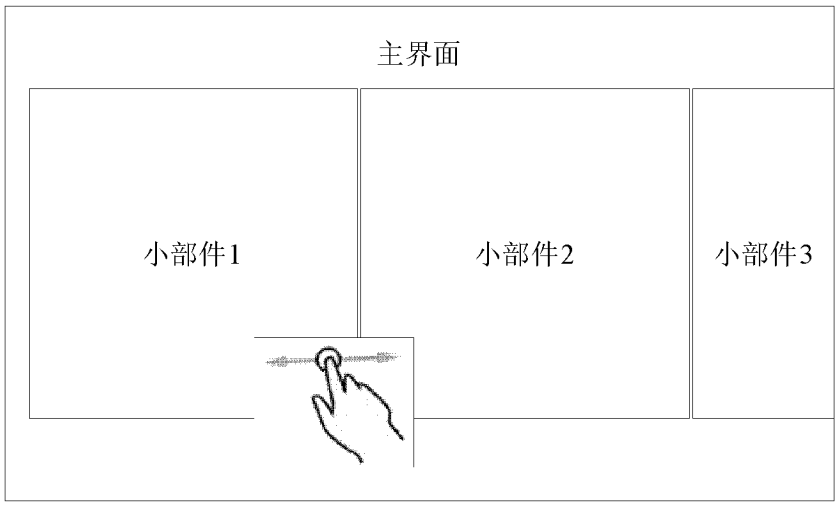

主界面
小部件1
小部件2
小部件3

图 2

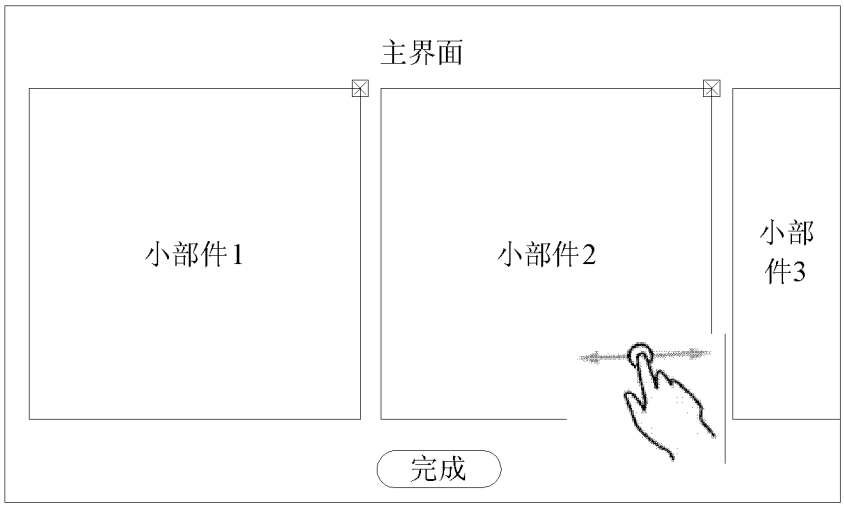

主界面
小部件1
小部件2
小部件3
完成

图 3

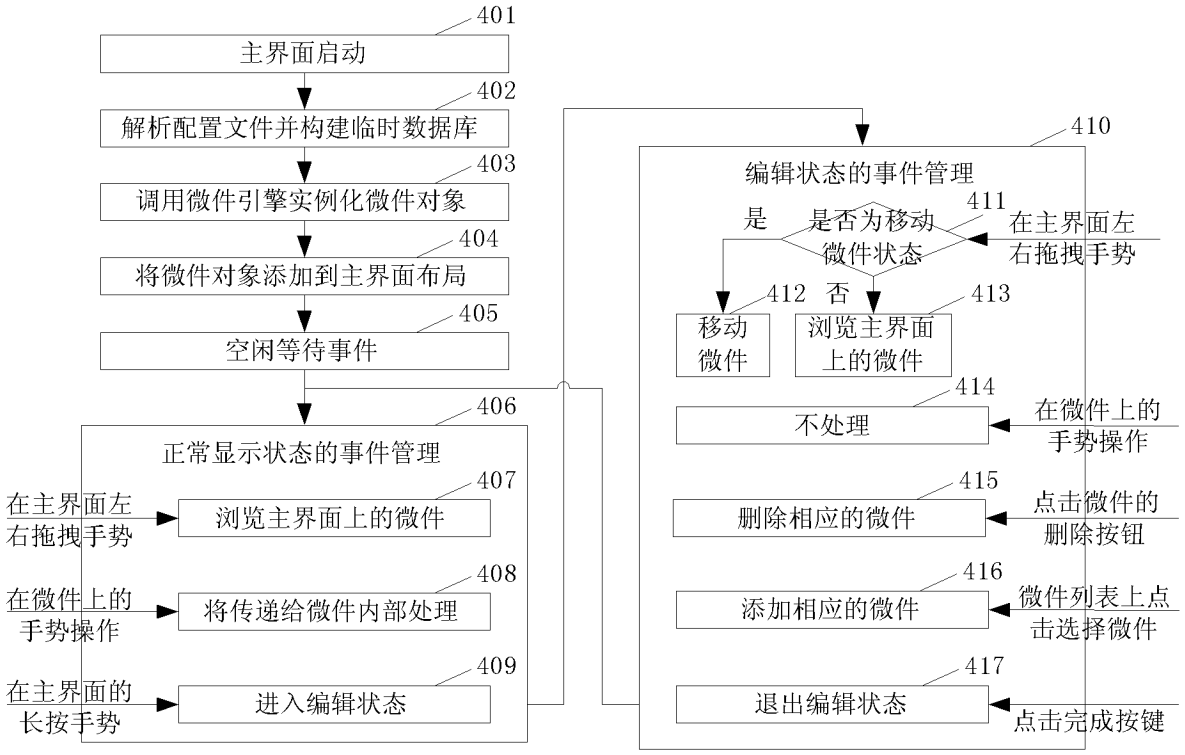

401
主界面启动
402
解析配置文件并构建临时数据库
403
调用微件引擎实例化微件对象
404
将微件对象添加到主界面布局
405
空闲等待事件
406
正常显示状态的事件管理
在主界面左右拖拽手势
407
浏览主界面上的微件
在微件上的手势操作
408
将传递给微件内部处理
在主界面的长按手势
409
进入编辑状态
410
编辑状态的事件管理
是
是否为移动微件状态
411
在主界面左右拖拽手势
412
否
413
移动微件
浏览主界面上的微件
414
不处理
在微件上的手势操作
415
删除相应的微件
点击微件的删除按钮
416
添加相应的微件
微件列表上点击选择微件
417
退出编辑状态
点击完成按键

图 4

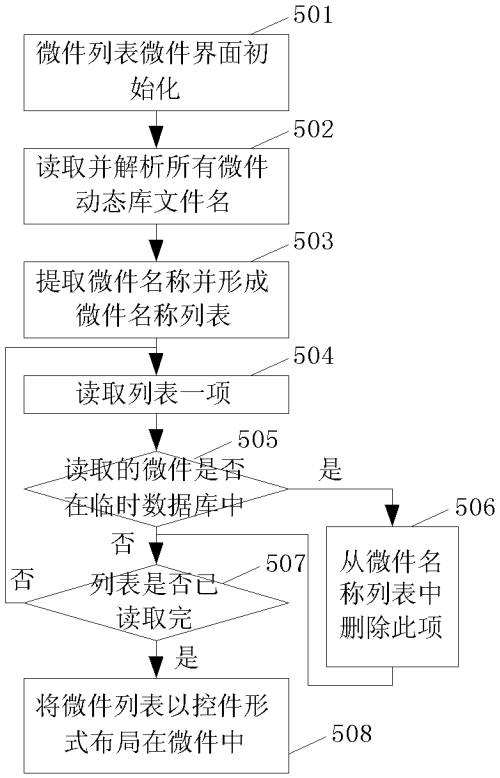

501
微件列表微件界面初始化
502
读取并解析所有微件动态库文件名
503
提取微件名称并形成微件名称列表
504
读取列表一项
505
读取的微件是否在临时数据库中
是
506
从微件名称列表中删除此项
否
507
列表是否已读取完
否
是
将微件列表以控件形式布局在微件中
508

图 5

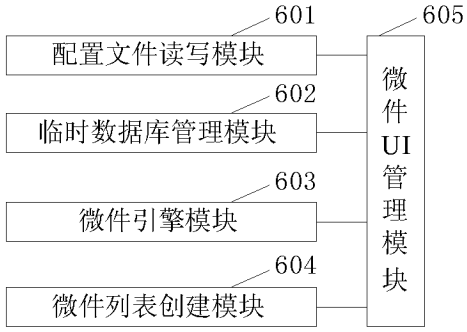

601
配置文件读写模块
602
临时数据库管理模块
603
微件引擎模块
604
微件列表创建模块
605
微件UI管理模块

图 6