(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 113822011 A
(43)申请公布日 2021.12.21

(21)申请号 202010570465.6

(22)申请日 2020.06.19

(71)申请人 华为技术有限公司
地址 518129 广东省深圳市龙岗区坂田华为总部办公楼

(72)发明人 卢亮 刘耀明 彭林

(74)专利代理机构 深圳中一联合知识产权代理有限公司 44414
代理人 左婷兰

(51)Int.Cl.
G06F 40/109(2020.01)

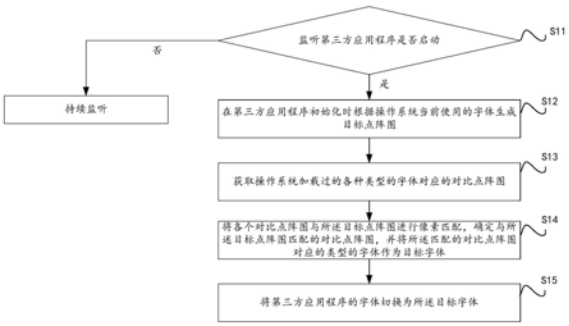
权利要求书2页 说明书23页 附图10页

(54)发明名称

字体切换方法及电子设备

(57)摘要

本申请提供了一种字体切换方法及电子设备，通过点阵图的像素点匹配，找出与操作系统当前使用的字体一致的目标字体，并将目标字体应用到第三方应用程序中，使得第三方应用程序能够准确地跟随操作系统的字体变化，避免出现第三方应用软件使用的字体与操作系统使用的字体不一致的问题。



CN 113822011 A

1.一种字体切换方法,其特征在于,包括:

根据操作系统当前使用的字体生成目标点阵图;

获取操作系统加载过的各种类型的字体对应的对比点阵图;

将各个对比点阵图与所述目标点阵图进行像素匹配,确定与所述目标点阵图匹配的对比点阵图,并将匹配到的对比点阵图对应的类型的字体作为目标字体;

将第三方应用程序的字体切换为所述目标字体。

2.根据权利要求1所述的方法,其特征在于,所述根据操作系统当前使用的字体生成目标点阵图,包括:

创建位图对象和画布对象;

根据操作系统当前使用的字体在所述画布对象中绘制预设字符,并将绘制的结果保存在所述位图对象中,得到目标点阵图。

3.根据权利要求1所述的方法,其特征在于,所述将各个对比点阵图与所述目标点阵图进行像素匹配,确定与所述目标点阵图匹配的对比点阵图,并将匹配到的对比点阵图对应的类型的字体作为目标字体,包括:

将得到的各个对比点阵图分别与目标点阵图进行像素点比对,当匹配到与目标点阵图的像素点分布一致的对比点阵图时,将与目标点阵图的像素点分布一致的对比点阵图作为与目标点阵图匹配的对比点阵图,并将该对比点阵图对应的字体确定为目标字体。

4.根据权利要求1所述的方法,其特征在于,所述目标点阵图与所述对比点阵图均为单色位图,相应地,所述将各个对比点阵图与所述目标点阵图进行像素匹配,确定与所述目标点阵图匹配的对比点阵图,并将匹配到的对比点阵图对应的类型的字体作为目标字体,包括:

分别获取所述目标点阵图的二进制数据和各个所述对比点阵图的二进制数据;

将与目标点阵图的二进制数据匹配一致的对比点阵图的二进制数据对应的对比点阵图确定为与目标点阵图匹配的对比点阵图,并将匹配到的对比点阵图对应的类型的字体作为目标字体。

5.根据权利要求1所述的方法,其特征在于,所述获取操作系统加载过的各种类型的字体对应的对比点阵图包括:

获取所述操作系统加载过的各种类型的字体;

创建所述操作系统加载过的各种类型的字体对应的画布对象和位图对象;

根据所述操作系统加载过的各种类型的字体在所述画布对象中绘制出预设字符,并将绘制的结果保存在各个类型的字体对应的位图对象中。

6.根据权利要求1所述的方法,其特征在于,所述获取操作系统加载过的各种类型的字体对应的对比点阵图,还包括:

获取所述操作系统加载过的所有类型的字体及所述操作系统的默认字体;

根据所述操作系统加载过的所有类型的字体和所述操作系统的默认字体,确定非操作系统默认字体;并生成与所述非操作系统默认字体对应的对比点阵图;

获取与所述非操作系统默认字体对应的对比点阵图。

7.根据权利要求6所述的方法,其特征在于,所述生成与所述非操作系统默认字体对应的对比点阵图包括:

创建与各个非操作系默认字体对应的画布对象和位图对象；

根据所述各个非操作系默认字体在所述画布对象中绘制出预设字符，并将绘制的结果保存在各个非操作系统默认字体对应的位图对象中。

8.根据权利要求1至7任一项所述的方法，其特征在于，在根据操作系统当前使用的字体生成目标点阵图之前，还包括：

监听所述第三方应用程序是否启动。

9.根据权利要求1至8任一项所述的方法，其特征在于，所述操作系统当前使用的字体为用户选择的自定义字体。

10.根据权利要求1至9任一项所述的方法，其特征在于，所述将第三方应用程序的字体切换为所述目标字体，包括：

将所述目标字体的字体文件的存放路径设置给第三方应用程序；

控制所述第三方应用程序根据所述目标字体的字体文件的存放路径获取并加载目标字体文件；

根据加载的目标字体文件对第三方应用程序的显示界面进行渲染，使得所述第三方应用程序的显示界面所使用的字符为所述目标字体文件对应的字符。

11.一种电子设备，其特征在于，包括：

生成单元，用于根据操作系统当前使用的字体生成目标点阵图；

获取单元，用于获取操作系统加载过的各种类型的字体对应的对比点阵图；

匹配单元，用于将各个对比点阵图与所述目标点阵图进行像素匹配，确定与所述目标点阵图匹配的对比点阵图，并将所述匹配的对比点阵图对应的类型的字体作为目标字体；

切换单元，用于将第三方应用程序的字体切换为所述目标字体。

12.一种电子设备，包括存储器、处理器以及存储在所述存储器中并可在所述处理器上运行的计算机程序，其特征在于，所述处理器执行所述计算机程序时实现如权利要求1至10任一项所述的字体切换方法。

13.一种芯片系统，其特征在于，所述芯片系统包括处理器，所述处理器与存储器耦合，所述处理器执行所述存储器中存储的计算机程序，以实现如权利要求1至10任一项所述的字体切换方法。

14.一种计算机可读存储介质，所述计算机可读存储介质存储有计算机程序，其特征在于，所述计算机程序被处理器执行时实现如权利要求1至10任一项所述的字体切换方法。

字体切换方法及电子设备

技术领域

[0001] 本申请涉及电子技术领域,尤其涉及一种字体切换方法及电子设备。

背景技术

[0002] 随着电子科技的快速发展,电子设备(如手机、平板电脑等)通常都具有各种主题包,在主题商店中有各种主题供用户选择,以丰富电子设备的显示画面,尤其是主题中的自定义字体多种多样,用户可以从中选择自己喜欢的自定义字体,并将该自定义字体设置为操作系统主题字体,即将该自定义字体导入到电子设备的操作系统中,使得电子设备的桌面的显示界面的字体显示为该自定义字体。然而,电子设备中安装的第三方应用软件并不能明确当前操作系统使用的自定义字体的存储路径,因此,在操作系统切换字体后,第三方应用软件使用的字体并不能准确地跟随操作系统使用的字体的变换而变换,这将导致出现第三方应用软件与操作系统使用的字体不一致的情况。其中,第三方应用软件是指非操作系统自带的应用程序或软件,通过安装第三方开发的程序可以扩展手机的应用功能,第三方应用软件包括但不限于即时通讯软件、游戏软件、浏览器软件等。

发明内容

[0003] 本申请提供一种字体切换方法及电子设备,解决了现有技术中第三方应用软件使用的字体与操作系统使用的字体不一致的问题。

[0004] 为达到上述目的,本申请采用如下技术方案:

[0005] 第一方面,本申请实施例提供了一种字体切换方法,该方法可以包括:根据操作系统当前使用的字体生成目标点阵图;获取操作系统加载过的各种类型的字体对应的对比点阵图;将各个对比点阵图与所述目标点阵图进行像素匹配,确定与所述目标点阵图匹配的对比点阵图,并将所述匹配的对比点阵图对应的类型的字体作为目标字体;将第三方应用程序的字体切换为所述目标字体。其中,操作系统加载过的各种类型的字体可以包括操作系统加载过的所有不同类型的字体,例如宋体、楷体、黑体等。也可以是操作系统加载过的非操作系统默认字体。字体可以使用Typeface类来进行描述,每个类型的字体都有其对应的详细描述信息,Typeface类中定义有FontID、GlyphID、字体的样式(style)属性,style属性可以为正常、加粗、倾斜或加粗倾斜。FontID是Font的标识,是全局唯一的标识,比如加粗宋体的FontID与正常宋体的FontID不同,宋体的FontID和楷体的FontID也不同。Font为具备样式的字体,可以认为是字体的子集,比如字体为宋体时,Font可以是正常宋体、加粗宋体、倾斜宋体或加粗倾斜宋体。GlyphID为字符在对应字体下字形的标识,通过GlyphID可以获取字符的Glyph。Glyph为字体的字形,每个字符在对应的字体下都有属于自己的字形,在渲染显示字符时,需要使用该字符的字形以及其他显示属性(比如字号、倾斜、黑体、倾斜黑体、颜色等)进行渲染显示。

[0006] 上述目标点阵图为能够体现操作系统当前使用字体的字体特征的点阵图,具体可以是采用该操作系统当前使用字体绘制的一段文字而生成的点阵图,例如位图(bitmap)、

像素图等。上述对比点阵图同样为能够体现操作系统加载过的各种类型的字体的字体特征的点阵图，具体可以是采用操作系统加载过的各种类型的字体绘制的一段文字生成的点阵图。其中，点阵图是指由n多个像素组成的图，由像素阵列的排列来实现其显示效果，在对点阵图进行操作时，具体是对点阵图中的每个像素进行操作，即操作的对象是每个像素，可以改变该像素的色相、饱和度、透明度等属性来改变该点阵图的显示效果，像素点可以进行不同排列和染色来构成不同的图像。需要说明的是，在对比点阵图上绘制的预设字符时，需要使用与绘制目标点阵图时相同字号大小和相同字符内容的预设字符进行绘制。由于生成的位图是在相同字号下生成的，因此，得到的字符位图的尺寸也会是一样的，这样能够有效地避免了归一化处理，提高像素比对时的精准度。

[0007] 本申请实施例提供的字体切换方法，通过点阵图的像素点匹配，找出与操作系统当前使用的字体一致的目标字体，然后将其应用到第三方应用程序中，使得第三方应用程序能够准确地跟随操作系统的字体变化，避免第三方应用软件使用的字体与操作系统使用的字体不一致的问题。

[0008] 在本申请实施例中，上述操作系统当前使用的字体可以是用户选择的自定义字体。用户可以通过在主题商店等应用程序中选择用户喜欢的自定义主题或自定义字体，操作系统会将用户选择的自定义主题中的字体或用户选择的自定义字体设置成操作系统当前使用的字体。

[0009] 在第一方面的一种可能的实现方式中，在根据操作系统当前使用的字体生成目标点阵图之前，还可以实时监听第三方应用程序是否启动。在监听到第三方应用程序启动时再执行生成目标点阵图的步骤。

[0010] 可以通过在进程管理服务中设置一个监听程序(例如hook程序)，以此来检测第三方应用程序是否启动。通过实时监听第三方应用程序是否启动，当第三方应用程序启动时及时地进行字体切换的操作，能够保证用户在使用第三方应用程序时不会程序使用字体不一致的情况。且可以通过设置监听程序来检测第三方应用程序是否启动，能够自动实现字体的切换，提高切换效率。

[0011] 在第一方面的一种可能的实现方式中，上述根据操作系统当前使用的字体生成目标点阵图可以包括：创建位图对象和画布对象；根据操作系统当前使用的字体在所述画布对象中绘制预设字符，并将绘制的结果保存在所述位图对象中，得到目标点阵图。可以将上述字符内容设定为“这是一段测试文字”，将字号设置为48px。可理解的是，上述字符内容和字号大小可以根据需求进行设置，在此不再加以赘述。还可以理解的是，位图的大小会影响匹配效果，位图尺寸越大匹配的精度越高，准确率也越高，但是会导致字形比对效率下降，因此，设置的字号也不是越大越好，为了达到较佳的比对精度和比对效率，经测试发现可以选用字号大小为48px，此时得到的位图尺寸为64*64。

[0012] 通过操作系统原有的API来创建用于生成目标点阵图的位图对象和画布对象，然后基于位图对象进行像素比对，能够有效地保证比对结果的准确性，且只需要调用操作系统原有的API就能实现，因此具备良好的系统兼容性。

[0013] 在第一方面的一种可能的实现方式中，所述获取操作系统加载过的各种类型的字体对应的对比点阵图包括：获取所述操作系统加载过的所有类型的字体，并生成与所述操作系统加载过的所有类型的字体对应的对比点阵图；

[0014] 获取与所述操作系统加载过的各种类型的字体对应的对比点阵图。

[0015] 为了保证能够准确地识别出操作系统当前使用的目标字体，通过获取到操作系统加载过的所有类型的字体，然后生成各自对应的对比点阵图，就能够通过对比点阵图与目标点阵图进行像素点比对找出操作系统当前使用的目标字体。

[0016] 具体地，上述生成与所述操作系统加载过的所有类型的字体对应的对比点阵图包括：创建所述操作系统加载过的所有类型的字体对应的画布对象和位图对象；根据所述操作系统加载过的所有类型的字体在所述画布对象中绘制出预设字符，并将绘制的结果保存在各个类型的字体对应的位图对象中。

[0017] 通过操作系统原有的API来创建用于生成对比点阵图的位图对象和画布对象，然后基于目标点阵图的位图对象和对比点阵图的位图对象进行像素比对，就能够有效地保证比对结果的准确性，且只需要调用操作系统原有的API就能实现，因此具备良好的系统兼容性。

[0018] 在第一方面的一种可能的实现方式中，上述所述获取操作系统加载过的各种类型的字体对应的对比点阵图包括：获取所述操作系统加载过的所有类型的字体及所述操作系统的默认字体；根据所述操作系统加载过的所有类型的字体和所述操作系统的默认字体，确定非操作系统默认字体；并生成与所述非操作系统默认字体对应的对比点阵图；获取与所述非操作系统默认字体对应的对比点阵图。

[0019] 由于用户通过主题商店等应用选择的自定义字体通常都不包含在操作系统的默认字体中，因此，可以获取电子设备的操作系统加载过的非操作系统默认字体，这样能够有效地减少对比点阵图的数量，提高处理效率，减少系统资源的占用。

[0020] 具体地，所述生成与所述非操作系统默认字体对应的对比点阵图包括：创建与所述各个非操作系默认字体对应的画布对象和位图对象；根据所述各个非操作系默认字体在所述画布对象中绘制出预设字符，并将绘制的结果保存在各个非操作系统默认字体对应的位图对象中。

[0021] 通过操作系统原有的API来创建用于生成对比点阵图的位图对象和画布对象，然后基于目标点阵图的位图对象和对比点阵图的位图对象进行像素比对，就能够有效地保证比对结果的准确性，且只需要调用操作系统原有的API就能实现，因此具备良好的系统兼容性。

[0022] 在第一方面的一种可能的实现方式中，上述将各个对比点阵图与所述目标点阵图进行像素匹配，确定与所述目标点阵图匹配的对比点阵图，并将匹配到的对比点阵图对应的类型的字体作为目标字体，包括：

[0023] 将得到的各个对比点阵图分别与目标点阵图进行像素点比对，当匹配到与目标点阵图的像素点分布一致的对比点阵图时，将与目标点阵图的像素点分布一致的对比点阵图作为与目标点阵图匹配的对比点阵图，并将该对比点阵图对应的字体确定为目标字体。通过像素点分布是否一致来确定与目标点阵图匹配的对比点阵图，能够准确地确定出目标字体。

[0024] 在第一方面的一种可能的实现方式中，所述目标点阵图与所述对比点阵图均为单色位图，相应地，所述将各个对比点阵图与所述目标点阵图进行像素匹配，确定与所述目标点阵图匹配的对比点阵图，并将匹配到的对比点阵图对应的类型的字体作为目标字体，包

括：

[0025] 分别获取所述目标点阵图的二进制数据和各个所述对比点阵图的二进制数据；

[0026] 将与目标点阵图的二进制数据匹配一致的对比点阵图的二进制数据对应的对比点阵图确定为与目标点阵图匹配的对比点阵图，并将匹配到的对比点阵图对应的类型的字体作为目标字体。利用二进制数据进行比对能够有效地提升比对效率。

[0027] 在第一方面的一种可能的实现方式中，所述将第三方应用程序的字体切换为所述目标字体，包括：

[0028] 将所述目标字体的字体文件的存放路径设置给第三方应用程序；

[0029] 控制所述第三方应用程序根据所述目标字体的字体文件的存放路径获取并加载目标字体文件；

[0030] 根据加载的目标字体文件对第三方应用程序的显示界面进行渲染，使得所述第三方应用程序的显示界面所使用的字符为所述目标字体文件对应的字符。

[0031] 第二方面，本申请实施例提供了一种电子设备，该电子设备可以包括生成单元、获取单元、匹配单元和切换单元。

[0032] 上述生成单元用于根据操作系统当前使用的字体生成目标点阵图；

[0033] 上述获取单元用于获取操作系统加载过的各种类型的字体对应的对比点阵图；

[0034] 上述匹配单元用于将各个对比点阵图与所述目标点阵图进行像素匹配，确定与所述目标点阵图最匹配的对比点阵图，并将所述最匹配的对比点阵图对应的类型的字体作为目标字体；

[0035] 上述切换单元用于将第三方应用程序的字体切换为所述目标字体。

[0036] 本申请实施例提供的电子设备，同样可以通过模板的像素匹配，找出与操作系统当前使用的字体一致的目标字体，然后将其应用到第三方应用程序中，使得第三方应用程序能够准确地跟随操作系统的字体变化，避免第三方应用软件使用的字体与操作系统使用的字体不一致的问题。

[0037] 在第二方面的一种可能的实现方式中，上述电子设备还可以包括监听单元。

[0038] 上述监听单元用于在根据操作系统当前使用的字体生成目标点阵图之前，实时监听第三方应用程序是否启动。

[0039] 在监听单元监听到第三方应用程序启动时再由第一生成单元执行生成目标点阵图的步骤。

[0040] 通过实时监听第三方应用程序是否启动，当第三方应用程序启动时及时地进行字体切换的操作，能够保证用户在使用第三方应用程序时不会程序使用字体不一致的情况。且可以通过设置监听程序来检测第三方应用程序是否启动，能够自动实现字体的切换，提高切换效率。

[0041] 在第二方面的一种可能的实现方式中，上述生成单元可以包括：第一创建单元和第一绘制单元。

[0042] 上述第一创建单元用于创建位图对象和画布对象；

[0043] 上述第一绘制单元用于根据操作系统当前使用的字体在所述画布对象中绘制预设字符，并将绘制的结果保存在所述位图对象中，得到目标点阵图。

[0044] 通过操作系统原有的API来创建用于生成目标点阵图的位图对象和画布对象，然

后基于位图对象进行像素比对，能够有效地保证比对结果的准确性，且只需要调用操作系统原有的API就能实现，因此具备良好的系统兼容性。

[0045] 在第二方面的一种可能的实现方式中，上述获取单元主要用于：获取所述操作系统加载过的各种类型的字体，并生成与所述操作系统加载过的各种类型的字体对应的对比点阵图；获取与所述操作系统加载过的各种类型的字体对应的对比点阵图。

[0046] 为了准确地识别出操作系统当前使用的目标字体，通过获取到操作系统加载过的各种类型的字体，然后生成各自对应的对比点阵图，就能够通过对比点阵图与目标点阵图进行像素比对找出操作系统当前使用的目标字体。

[0047] 具体地，上述获取单元可以包括第二创建单元和第二绘制单元。

[0048] 上述第二创建单元用于创建所述操作系统加载过的各种类型的字体对应的画布对象和位图对象；

[0049] 第二绘制单元用于根据所述操作系统加载过的各种类型的字体在所述画布对象中绘制出预设字符，并将绘制的结果保存在各个类型的字体对应的位图对象中。

[0050] 通过操作系统原有的API来创建用于生成对比点阵图的位图对象和画布对象，然后基于目标点阵图的位图对象和对比点阵图的位图对象进行像素比对，就能够有效地保证比对结果的准确性，且只需要调用操作系统原有的API就能实现，因此具备良好的系统兼容性。

[0051] 在第二方面的一种可能的实现方式中，上述获取单元主要用于获取所述操作系统加载过的各种类型的字体及所述操作系统的默认字体；根据所述操作系统加载过的各种类型的字体和所述操作系统的默认字体，确定非操作系统默认字体；并生成与所述非操作系统默认字体对应的对比点阵图；获取与所述非操作系统默认字体对应的对比点阵图。

[0052] 由于用户通过主题商店等应用选择的自定义字体通常都不包含在操作系统的默认字体中，因此，上述各种类型的字体还可以是电子设备的操作系统加载过的非操作系统默认字体，这样能够有效地减少对比点阵图的数量，提高处理效率，减少系统资源的占用。

[0053] 具体地，上述获取单元可以包括第三创建单元和第三绘制单元。

[0054] 上述第三创建单元用于创建与所述各个非操作系默认字体对应的画布对象和位图对象。

[0055] 上述第三绘制单元用于根据所述各个非操作系默认字体在画布对象中绘制出预设字符，并将绘制的结果保存在各个非操作系统默认字体对应的位图对象中。

[0056] 通过操作系统原有的API来创建用于生成对比点阵图的位图对象和画布对象，然后基于目标点阵图的位图对象和对比点阵图的位图对象进行像素比对，就能够有效地保证比对结果的准确性，且只需要调用操作系统原有的API就能实现，因此具备良好的系统兼容性。

[0057] 在第二方面的一种可能的实现方式中，上述匹配单元130主要用于将得到的各个对比点阵图分别与目标点阵图进行像素点比对，当匹配到与目标点阵图的像素点分布一致的对比点阵图时，将与目标点阵图的像素点分布一致的对比点阵图作为与目标点阵图匹配的对比点阵图，并将该对比点阵图对应的字体确定为目标字体。

[0058] 在第二方面的一种可能的实现方式中，上述目标点阵图与上述对比点阵图均为单色位图，相应地，上述匹配单元主要用于分别获取所述目标点阵图的二进制数据和各个所

述对比点阵图的二进制数据;将与目标点阵图的二进制数据匹配一致的对比点阵图的二进制数据对应的对比点阵图确定为与目标点阵图匹配的对比点阵图,并将匹配到的对比点阵图对应的类型的字体作为目标字体。

[0059] 在第二方面的一种可能的实现方式中,上述切换单元主要用于将所述目标字体的字体文件的存放路径设置给第三方应用程序;控制所述第三方应用程序根据所述目标字体的字体文件的存放路径获取并加载目标字体文件;根据加载的目标字体文件对第三方应用程序的显示界面进行渲染,使得所述第三方应用程序的显示界面所使用的字符为所述目标字体文件对应的字符。

[0060] 第三方面,本申请实施例提供一种电子设备,包括:存储器和处理器,所述存储器用于存储计算机程序;所述处理器用于在调用所述计算机程序时执行上述第一方面所述的方法。

[0061] 第四方面,本申请实施例提供一种计算机可读存储介质,其上存储有计算机程序,计算机程序被处理器执行时实现上述第一方面所述的方法

[0062] 第五方面,本申请实施例提供一种计算机程序产品,当计算机程序产品在电子设备上运行时,使得电子设备执行上述第一方面所述的方法

[0063] 第六方面,本申请实施例提供一种芯片系统,所述芯片系统包括存储器和处理器,所述处理器执行所述存储器中存储的计算机程序,以实现上述第一方面所述的方法,所述芯片系统可以为单个芯片,或者多个芯片组成的芯片模组。

[0064] 上述第二方面至第六方面的有益效果可以参见上述第一方面中的相关描述,在此不再赘述。

附图说明

[0065] 图1为本申请实施例相关的手机的部分结构架构图;

[0066] 图2为本申请实施例提供的手机的软件系统架构图;

[0067] 图3为本申请实施例提供的手机的一种图像用户界面的示意图;

[0068] 图4为本申请实施例提供的手机的另一种图像用户界面示意图;

[0069] 图5为本申请实施例提供的手机的又一种图像用户界面示意图;

[0070] 图6为本申请实施例提供的手机的再一种图像用户界面示意图;

[0071] 图7为本申请实施例提供的手机的主体商店应用的字体控件的显示界面示意图;

[0072] 图8为现有技术中存在的操作系统工具栏的字体和浏览器网页的字体不一致的示意图;

[0073] 图9为本申请实施例提供一种字体切换方法的示意性流程图;

[0074] 图10为本申请实施例提供的根据操作系统当前使用的字体生成目标点阵图的方法的示意性流程图;

[0075] 图11为本申请实施例提供的目标点阵图和对比点阵图的示意图;

[0076] 图12为本申请实施例提供的生成目标点阵图和生成对比点阵图的示意图

[0077] 图13为本申请实施例提供一种获取对比点阵图的方法的示意性流程图;

[0078] 图14为本申请实施例提供的另一种获取对比点阵图的方法的示意性流程图;

[0079] 图15为本申请实施例提供的一种电子设备的结构示意图。

具体实施方式

[0080] 以下描述中，为了说明而不是为了限定，提出了诸如特定操作系统结构、技术之类的具体细节，以便透彻理解本申请实施例。然而，本领域的技术人员应当清楚，在没有这些具体细节的其它实施例中也可以实现本申请。在其它情况中，省略对众所周知的操作系统、装置、电路以及方法的详细说明，以免不必要的细节妨碍本申请的描述。

[0081] 应当理解，当在本申请说明书和所附权利要求书中使用时，术语“包括”指示所描述特征、整体、步骤、操作、元素和/或组件的存在，但并不排除一个或多个其它特征、整体、步骤、操作、元素、组件和/或其集合的存在或添加。

[0082] 还应当理解，在本申请说明书和所附权利要求书中使用的术语“和/或”是指相关联列出的项中的一个或多个的任何组合以及所有可能组合，并且包括这些组合。

[0083] 如在本申请说明书和所附权利要求书中所使用的那样，术语“如果”可以依据上下文被解释为“当...时”或“一旦”或“响应于确定”或“响应于检测到”。类似地，短语“如果确定”或“如果检测到[所描述条件或事件]”可以依据上下文被解释为意指“一旦确定”或“响应于确定”或“一旦检测到[所描述条件或事件]”或“响应于检测到[所描述条件或事件]”。

[0084] 另外，在本申请说明书和所附权利要求书的描述中，术语“第一”、“第二”、“第三”等仅用于区分描述，而不能理解为指示或暗示相对重要性。

[0085] 在本申请说明书中描述的参考“一个实施例”或“一些实施例”等意味着在本申请的一个或多个实施例中包括结合该实施例描述的特定特征、结构或特点。由此，在本说明书中的不同之处出现的语句“在一个实施例中”、“在一些实施例中”、“在其他一些实施例中”、“在另外一些实施例中”等不是必然都参考相同的实施例，而是意味着“一个或多个但不是所有的实施例”，除非是以其他方式另外特别强调。术语“包括”、“包含”、“具有”及它们的变形都意味着“包括但不限于”，除非是以其他方式另外特别强调。

[0086] 本申请实施例为了解决现有的第三方应用软件使用的字体不能准确跟随电子设备的操作系统使用的字体变换而变换，导致第三方应用软件使用的字体与操作系统使用的字体不一致的问题，通过根据操作系统当前使用的字体创建目标点阵图，然后找到该电子设备使用过的全部字体，并根据这些字体创建对应的对比点阵图，再进一步地从对比点阵图中找到与目标点阵图匹配的目标字体，就能准确地将该目标字体设置给将第三方应用程序，使得第三方应用程序所使用的字体能够与电子设备的操作系统所使用的字体一致，避免第三方应用软件使用的字体与操作系统使用的字体不一致的情况。

[0087] 本申请实施例提供的字体切换方法可以应用于手机、平板电脑、可穿戴设备、车载设备、增强现实(augmented reality,AR)/虚拟现实(virtual reality,VR)设备、笔记本电脑、超级移动个人计算机(ultra-mobile personal computer,UMPC)、上网本、个人数字助理(personal digital assistant,PDA)等电子设备上，本申请实施例对电子设备的具体类型不作任何限制。

[0088] 例如，所述电子设备可以是WLAN中的站点(STAION,ST)，可以是蜂窝电话、无绳电话、会话启动协议(Session InitiationProtocol,SIP)电话、无线本地环路(Wireless Local Loop,WLL)站、个人数字处理(Personal Digital Assistant,PDA)设备、具有无线通信功能的手持设备、计算设备或连接到无线调制解调器的其它处理设备、车载设备、车联网终端、电脑、膝上型计算机、手持式通信设备、手持式计算设备、卫星无线设备、无线调制解

调器卡、电视机顶盒(set top box,STB)、用户驻地设备(customer premise equipment,CPE)和/或用于在无线操作系统上进行通信的其它设备以及下一代通信操作系统,例如,5G网络中的电子设备或者未来演进的公共陆地移动网络(Public Land Mobile Network,PLMN)网络中的电子设备等。

[0089] 作为示例而非限定,当所述电子设备为可穿戴设备时,该可穿戴设备还可以是应用穿戴式技术对日常穿戴进行智能化设计、开发出可以穿戴的设备的总称,如眼镜、手套、手表、服饰及鞋等。可穿戴设备即直接穿在身上,或是整合到用户的衣服或配件的一种便携式设备。可穿戴设备不仅仅是一种硬件设备,更是通过软件支持以及数据交互、云端交互来实现强大的功能。广义穿戴式智能设备包括功能全、尺寸大、可不依赖智能手机实现完整或者部分的功能,如智能手表或智能眼镜等,以及只专注于某一类应用功能,需要和其它设备如智能手机配合使用,如各类进行体征监测的智能手环、智能首饰等。

[0090] 以所述电子设备为手机为例,图1示出了与本实施例相关的手机100的部分结构的示意框图。

[0091] 手机100可以包括处理器110,外部存储器接口120,内部存储器121,通用串行总线(universal serial bus,USB)接口130,充电管理模块140,电源管理模块141,电池142,天线1,天线2,移动通信模块150,无线通信模块160,音频模块170,传感器模块180,按键,马达,指示器,以及用户标识模块(subscriber identification module,SIM)卡接口等。

[0092] 可以理解的是,本发明实施例示意的结构并不构成对手机100的具体限定。在本申请另一些实施例中,手机100可以包括比图示更多或更少的部件,或者组合某些部件,或者拆分某些部件,或者不同的部件布置。图示的部件可以以硬件,软件或软件和硬件的组合实现。

[0093] 处理器110可以包括一个或多个处理单元,例如:处理器110可以包括应用处理器(application processor,AP),调制解调处理器,图形处理器(graphics processing unit,GPU),图像信号处理器(image signal processor,ISP),控制器,视频编解码器,数字信号处理器(digital signal processor,DSP),基带处理器,和/或神经网络处理器(neural-network processing unit,NPU)等。其中,不同的处理单元可以是独立的器件,也可以集成在一个或多个处理器中。

[0094] 控制器可以根据指令操作码和时序信号,产生操作控制信号,完成取指令和执行指令的控制。

[0095] 处理器110中还可以设置存储器,用于存储指令和数据。在一些实施例中,处理器110中的存储器为高速缓冲存储器。该存储器可以保存处理器110刚用过或循环使用的指令或数据。如果处理器110需要再次使用该指令或数据,可从所述存储器中直接调用。避免了重复存取,减少了处理器110的等待时间,因而提高了操作系统的效率。

[0096] 在一些实施例中,处理器110可以包括一个或多个接口。接口可以包括集成电路(inter-integrated circuit,I2C)接口,集成电路内置音频(inter-integrated circuit sound,I2S)接口,脉冲编码调制(pulse code modulation,PCM)接口,通用异步收发传输器(universal asynchronous receiver/transmitter,UART)接口,移动产业处理器接口(mobile industry processor interface,MIPI),通用输入输出(general-purpose input/output,GPIO)接口,用户标识模块(subscriber identity module,SIM)接口,和/或

通用串行总线(universal serial bus,USB)接口等。

[0097] I2C接口是一种双向同步串行总线,包括一根串行数据线(serial data line,SDA)和一根串行时钟线(derail clock line,SCL)。在一些实施例中,处理器110可以包含多组I2C总线。处理器110可以通过不同的I2C总线接口分别耦合触摸传感器180K,充电器,闪光灯,摄像头等。例如:处理器110可以通过I2C接口耦合触摸传感器180K,使处理器110与触摸传感器180K通过I2C总线接口通信,实现手机100的触摸功能。

[0098] MIPI接口可以被用于连接处理器110与显示屏194,摄像头等外围器件。MIPI接口包括显示屏串行接口(display serial interface,DSI)等。在一些实施例中,处理器110和显示屏194通过DSI接口通信,实现手机100的显示功能。

[0099] 可以理解的是,本发明实施例示意的各模块间的接口连接关系,只是示意性说明,并不构成对手机100的结构限定。在本申请另一些实施例中,手机100也可以采用上述实施例中不同的接口连接方式,或多种接口连接方式的组合。

[0100] 手机100通过GPU,显示屏194,以及应用处理器等实现显示功能。GPU为图像处理的微处理器,连接显示屏194和应用处理器。GPU用于执行数学和几何计算,用于图形渲染。处理器110可包括一个或多个GPU,其执行程序指令以生成或改变显示信息。

[0101] 显示屏194用于显示界面,图像,视频等。显示屏194包括显示面板。显示面板可以采用液晶显示屏(liquid crystal display,LCD),有机发光二极管(organic light-emitting diode,OLED),有源矩阵有机发光二极体或主动矩阵有机发光二极体(active-matrix organic light emitting diode的,AMOLED),柔性发光二极管(flex light-emitting diode,FLED),Miniled,MicroLed,Micro-oLed,量子点发光二极管(quantum dot light emitting diodes,QLED)等。在一些实施例中,手机100可以包括1个或N个显示屏194,N为大于1的正整数。

[0102] 外部存储器接口120可以用于连接外部存储卡,例如Micro SD卡,实现扩展手机100的存储能力。外部存储卡通过外部存储器接口120与处理器110通信,实现数据存储功能。例如将音乐,视频等文件保存在外部存储卡中。

[0103] 内部存储器121可以用于存储计算机可执行程序代码,所述可执行程序代码包括指令。内部存储器121可以包括存储程序区和存储数据区。其中,存储程序区可存储系统,至少一个功能所需的应用程序(比如声音播放功能,图像播放功能等)等。存储数据区可存储手机100使用过程中所创建的数据(比如音频数据,电话本等)等。此外,内部存储器121可以包括高速随机存取存储器,还可以包括非易失性存储器,例如至少一个磁盘存储器件,闪存器件,通用闪存存储器(universal flash storage,UFS)等。处理器110通过运行存储在内部存储器121的指令,和/或存储在设置于处理器中的存储器的指令,执行手机100的各种功能应用以及数据处理。在一些实施例中,处理器110通过查找当前进程的内存映射表,根据字体的文件类型,就能够查找到内部存储器中加载过的所有字体,即该手机100使用过的全部字体。需要说明的是,字体的文件是以.ttf后缀结束的文件或者是以.woff后缀结束的文件。其中,以.tff后缀结束的文件是指以TrueType字库标准创建的字体文件,以.woff后缀结束的文件是指采用Web开放字体格式的字体文件。

[0104] 传感器模块180可以包括触摸传感器180K,触摸传感器180K,也称“触控器件”。触摸传感器180K可以设置于显示屏194,由触摸传感器180K与显示屏194组成触摸屏,也称“触

控屏”。触摸传感器180K用于检测作用于其上或附近的触摸操作。触摸传感器可以将检测到的触摸操作传递给应用处理器，以确定触摸事件类型。可以通过显示屏194提供与触摸操作相关的视觉输出。在另一些实施例中，触摸传感器180K也可以设置于手机100的表面，与显示屏194所处的位置不同。

[0105] 手机100的软件操作系统可以采用分层架构，事件驱动架构，微核架构，微服务架构，或云架构。本发明实施例以分层架构的Android系统为例，示例性说明手机100的软件结构。

[0106] 图2是本发明实施例的手机100的软件结构框图。

[0107] 分层架构将软件分成若干个层，每一层都有清晰的角色和分工。层与层之间通过软件接口通信。在一些实施例中，将Android系统分为四层，从上至下分别为应用程序层，应用程序框架层，Andriod运行时(Android runtime)和系统库，以及内核层。

[0108] 应用程序层可以包括一系列应用程序包。

[0109] 如图2所示，应用程序包可以包括相机，图库，日历，通话，地图，导航，WLAN，蓝牙，音乐，视频，短信息、主题商店等应用程序。

[0110] 应用程序框架层(Framework层)为应用程序层的应用程序提供应用编程接口(application programming interface，API)和编程框架。应用程序框架层包括一些预先定义的函数。

[0111] 如图2所示，应用程序框架层可以包括窗口管理器，内容提供器，视图系统，电话管理器，资源管理器，通知管理器等。

[0112] Android Runtime包括核心库和虚拟机。Android runtime负责Andriod系统的调度和管理。

[0113] 核心库包含两部分：一部分是java语言需要调用的功能函数，另一部分是Andriod的核心库。

[0114] 应用程序层和应用程序框架层运行在虚拟机中。虚拟机将应用程序层和应用程序框架层的java文件执行为二进制文件。虚拟机用于执行对象生命周期的管理，堆栈管理，线程管理，安全和异常的管理，以及垃圾回收等功能。

[0115] 系统库可以包括多个功能模块。例如：表面管理器(surface manager)，媒体库(Media Libraries)，三维图形处理库(例如：OpenGL ES)，2D图形引擎(例如：SGL、Skia)等。图形处理库还包含画布、颜色过滤、点、矩形等，可以将他们直接绘制在电子设备的显示界面上。

[0116] 表面管理器用于对显示子系统进行管理，并且为多个应用程序提供了2D和3D图层的融合。

[0117] 媒体库支持多种常用的音频，视频格式回放和录制，以及静态图像文件等。媒体库可以支持多种音视频编码格式，例如：MPEG4，H.264，MP3，AAC，AMR，JPG，PNG等。

[0118] 三维图形处理库用于实现三维图形绘图，图像渲染，合成，和图层处理等。

[0119] 2D图形引擎是2D绘图的绘图引擎，在一些实施例中，上述2D图形引擎可以是绘制和渲染文字的skia。

[0120] 内核层是硬件和软件之间的层。内核层至少包含显示驱动，摄像头驱动，音频驱动，传感器驱动。

[0121] 下面结合字体切换场景，示例性说明手机100软件以及硬件的工作流程。

[0122] 当触摸传感器180K接收到触摸操作，相应的硬件中断被发给内核层。内核层将触摸操作加工成原始输入事件(包括触摸坐标，触摸操作的时间戳等信息)。原始输入事件被存储在内核层。应用程序框架层从内核层获取原始输入事件，识别该输入事件所对应的控件。以该触摸操作是触摸单击操作，该单击操作所对应的控件为浏览器应用图标的控件为例，浏览器应用调用应用框架层的接口，启动浏览器应用，进而通过调用内核层启动显示驱动，通过显示屏194显示画面。

[0123] 以下实施例可以在具有上述硬件结构/软件结构的手机100上实现。以下实施例将以手机100为例，对本申请实施例提供的字体切换方法进行说明。

[0124] 图3示出了如图1、图2所示手机100的一种图形用户界面(Graphical User Interface，GUI)，该GUI可以为该手机的桌面301，当手机检测到用户点击桌面301上的主题商店应用(Application，APP)的图标302的操作后，可以启动主题商店应用。图4所示的另一种GUI，该GUI可以称为主题商店APP的显示界面403，在该显示界面403中的菜单栏404可以包括主题控件、字体控件、壁纸控件、铃声控件等控件，用户可以通过在触摸屏上输入特定的指令使得显示界面403显示字体控件对应的显示界面，以供用户选择喜欢的字体并应用到操作系统中。上述特定的指令可以是按特定方向滑动触摸屏的手势指令，也可以是点击控件的触发指令，还可以是其他输入指令，在此不加以限制。

[0125] 示例性的，如图5所示，当启动主题商店应用时，显示界面403首先显示的是主题控件对应的界面，当用户向左滑动触摸屏时，显示界面403会显示字体控件对应的界面；当用户向右滑动触摸屏时，显示界面403会显示壁纸控件对应的显示界面。示例性的，如图6所示，用户还可以点击菜单栏404中的字体控件，使得显示界面403显示字体控件对应的显示界面。图7示出了字体控件对应的显示界面的示意图，如图7中的(a)所示，该显示界面403可以显示字体选择列表，上述字体列表中可以包括字体1、字体2、字体3、字体4等多个选项，当手机检测到用户点击显示界面403上的字体1的图标后，可以显示该字体1的预览效果以及选择按钮405(如图7中的(b)所示)，当手机检测到用户点击显示界面403上的选择按钮405后，可以显示提示信息“是否使用当前字体”，并提供“取消”和“确认”选项(如图7中的(c)所示)；用户选择“确认”选项后，将用户选择的字体应用到操作系统中。

[0126] 在电子设备的操作系统界面由默认字体切换为用户选择的自定义字体后，可以根据操作系统设置中选择的语言，使用操作系统的控件绘制选择的语言对应的中文、英文等字体，并通过进程加载这些字体相应的字体库，或者字体。例如，在Andriod系统中，可以使用Andriod系统自带的软件开发工具包(Software Development Kit，简称SDK)来绘制用户选择的语言所对应的中文、英文等字体。由于使用的是操作系统中控件进行展现，因此，操作系统中的应用进程在加载字体如.ttf字体文件类型的字体时，在应用进程相应的maps文件中会生成.ttf字体文件信息的记录。通过查询maps文件中的.ttf字体文件信息的记录，就可以准确得出当前操作系统使用过的全部字体，包括自定义字体。

[0127] 以Andriod系统为例，将字体应用到Andriod系统中是通过Andriod系统的2D图形引擎skia来实现的。当Andriod系统需要加载字体时，会启动并初始化zygote，zygote初始化后，preloadclass会加载Typeface类，在Typeface类中调用jni-->skia来加载字体。例如通过preloadClasses()中调用Class.forName("android.graphics.Typeface")来加载

Typeface类,并调用Typeface类的static块。create()函数通过调用相应的native方法就能创建字体对象,并通过init()函数解析字体配置文件,据此加载字体。

[0128] 需要说明是是,Typeface类用于定义字体的字体类,内部包含特定字体的详细描述信息,代表了一种类型的字体,例如宋体、楷体、黑体等。Typeface类中定义有FontID、GlyphID、字体的样式(style)属性,style属性可以为正常、加粗、倾斜或加粗倾斜。FontID是Font的标识,是全局唯一的标识,比如加粗宋体的FontID与正常宋体的FontID不同,宋体的FontID和楷体的FontID也不同。Font为具备样式的字体,可以认为是字体的子集,比如字体为宋体时,Font可以是正常宋体、加粗宋体、倾斜宋体或加粗倾斜宋体。GlyphID为字符在对应字体下字形的标识,通过GlyphID可以获取字符的Glyph。Glyph为字体的字形,每个字符在对应的字体下都有属于自己的字形,在渲染显示字符时,需要使用该字符的字形以及其他显示属性(比如字号、倾斜、黑体、倾斜黑体、颜色等)进行渲染显示。

[0129] Andriod系统的字体的配置文件位于workspace/frameworks/base/data/fonts/文件夹下,分为system_fonts.xml和fallback_fonts.xml两个文件。其中,system_fonts.xml文件包含操作系统默认字体。在加载过程中优先通过从system_fonts.xml文件开始查找对应的字体配置文件,如果在system_fonts.xml文件中没有查找到再进入fallback_fonts.xml文件进行查找。在找到对应的字体配置文件后,将该字体配置文件加载到Preloading classes中,然后通过init()函数解析该配置文件得到字体信息。上述字体信息就包括字体的样式、字体的大小、字形等信息。

[0130] 示例性的,以下是system_fonts.xml的部分文件:

[0131]
```
<family>
  <nameset>
      <name>sans-serif</name>
      <name>arial</name>
      <name>helvetica</name>
  </nameset>
```

[0132]
```
   <fileset>
      <file>Roboto-Regular.ttf</file>
      <file>Roboto-Bold.ttf</file>
      <file>Roboto-Italic.ttf</file>
      <file>Roboto-Boldtalic.ttf</file>
   </fileset>
</family>
```

[0133] 其中,<family></family>之间的内容称为一个字体族,<nameset></nameset>之间的内容为字体的名称,<fileset></fileset>之间的内容为调用的字体。

[0134] 然而此时第三方应用程序(例如浏览器、游戏等应用程序)所使用的字体并不能准确地跟随操作系统字体,现有的第三方应用程序通常会在使用skia库的基础上适配跟随操作系统字体变化功能,然而其适配跟随会存在误用的情况。其中,第三方应用软件是指非操作系统自带的应用程序或软件,通过安装第三方开发的程序可以扩展手机的应用功能,第

三方应用软件包括但不限于即时通讯软件、游戏软件、浏览器软件等。需要说明的是，在本申请实施例中，上述第三方应用程序主要涉及到具有webview组件或其他涉及自绘制的基础引擎的应用程序，例如浏览器、游戏等。在此以第三方应用程序为浏览器为例，在浏览器内核中存在一个字体类库，该字体类库用于存储字体类(Typeface类)，浏览器在加载网页后需要对网页中的字符进行渲染显示时，就需要从字体类库中读取可以支持网页字符显示的字体类(Typeface类)，然后利用读取的字体类对网页进行渲染，现有的第三方应用程序的字体适配跟随的方案主要包括以下两种方案，现以第三方应用程序为浏览器为例进行阐述：

[0135] 方案1：浏览器启动后读取当前内存中已经加载的字体，并去掉操作系统字体目录system/fonts/下的.ttf文件(即将system_fonts.xml文件中的字体去掉)，如果发现有其他字体存在，则将这些字体假设成操作系统当前使用的字体，然后在浏览器中使用这些字体文件的一个字体对应的字体类进行网页的渲染。

[0136] 然而，内存中加载的字体并不一定是操作系统当前使用的字体(例如应用程序主动通过代码createTypeFace触发第三方路径的字体加载)，则有可能导致浏览器使用的字体与操作系统当前使用的字体不一致的情况。且如果内存中有多个非操作系统字体目录(即不在system_fonts.xml文件中)的字体文件，则同样可能存在浏览器使用的字体与操作系统当前使用的字体不一致的情况。示例性的，假如内存中存在字体1.tff和字体2.tff这两个非操作系统字体目录下的字体文件，此时操作系统当前使用的字体为字体1，而根据方案1的跟随机制，则浏览器有可能使用字体1.tff对应的字体类来进行网页渲染，也有可能使用字体2.tff对应的字体类来进行网页渲染。假设操作系统当前使用的字体为字体1，而浏览器采用字体2进行网页渲染，就会出现如图8所示的操作系统工具栏的字体和浏览器网页的字体不一致的情况。

[0137] 方案2：通过推断出目前几个主流厂商存放第三方字体文件(例如自定义字体文件等非操作系统默认字体文件)的目录，并查找该目录下存在相关字体文件，若该目录下存在字体字体文件，则在浏览器网页中优先使用该字体文件对应的字体类进行网页的渲染。

[0138] 然而，由于该目录下存在的字体文件可能是操作系统之前使用过的字体，而非当前使用的字体，即无法确定该目录下存在的字体是否是当前使用的字体，因此也同样会出现浏览器使用的字体与操作系统当前使用的字体不一致的情况。且若操作系统改变存放第三方字体目录，还会导致设置的跟随操作系统字体变化的功能失效的情况。示例性的，假设一开始存放第三方字体的目录下存放着操作系统当前使用的字体1对应的字体文件字体1.tff，在此之后通过代码人为地将字体2对应的字体文件字体2.tff写入到该目录下，此时，当打开浏览器后，浏览器会采用字体2.tff对应的字体类进行网页渲染，同样会出现如图8所示的操作系统工具栏的字体和浏览器网页的字体不一致的情况。

[0139] 由此可知，现有的第三方应用程序使用的字体适配跟随操作系统使用的字体的方案还是存在字体误用的可能，即第三方应用程序使用的字体存在不能准确跟随电子设备的操作系统字体变换而变换，进而导致第三方应用软件使用的字体与操作系统使用的字体不一致的问题，而本申请通过根据操作系统当前使用的字体来创建目标点阵图，然后找到进程中全部加载过的字体(即该操作系统使用过的全部字体)，根据这些字体对应的对比点阵图与目标点阵图进行像素匹配，然后从对比点阵图中找到与目标点阵图匹配的对比点阵

图,并将其对应的字体作为目标字体,再将第三方应用程序的字体切换为目标字体,使得第三方应用程序所使用的字体能够与电子设备的操作系统所使用的字体一致,避免第三方应用软件使用的字体与操作系统使用的字体不一致的情况。

[0140] 请参阅图9,图9是本申请实施例提供的一种字体切换方法的示意性流程图,本实施例中,流程的执行主体为电子设备。如图9所示,本实施例提供的一种字体切换方法包括S11～S15,详述如下:

[0141] S11:监听第三方应用程序是否启动,若是,则执行S12;否则持续监听。

[0142] 在具体应用中,电子设备的操作系统可以在进程管理服务中设置一个监听程序(例如hook程序),以此来检测第三方应用程序是否启动。以Andriod系统为例,可以在ActivityManagerService中设置一个监听程序,通过该监听程序来检测第三方应用程序是否启动。需要说明的是,上述监听程序可以通过不断循环监听Activity栈顶,然后当监听到Activity栈顶中的进程的标识与第三方应用程序的唯一标识一致时,返回第一信息,上述第一信息用于通知操作系统监听到该第三方应用程序正在启动。需要说明的是,ActivityManagerService是Android系统提供的用于管理Activity运行状态的系统进程管理服务,Activity是Andriod系统中一个负责与用户交互的组件,能够实现与用户的交互,可以通过setContentView(View)来显示指定控件以及获取用户对于电子设备的屏幕的操作。还需要说明的是,在监听程序中可以设置多个第三方应用程序,当检测到Activity栈顶中的进程的标识与上述多个第三方应用程序中任一个第三方应用进程的唯一标识一致时,说明操作系统正在启动与该唯一标识对应的第三应用程序。需要说明的是,上述第三方应用程序的唯一标识用于区分各个应用程序,在本实施例中,上述第三方应用程序的唯一标识可以是该第三方应用程序的包名(packageName)。还需要说明的是,上述第三方应用程序的唯一标识还可以是开发者在开发时为应用程序设定的ApplicationId,在此不再加以赘述。

[0143] 在本实施例中,还可以通过触摸屏检测用户是否点击/触摸上述第三方应用程序来监听第三方应用程序是否启动。若触摸屏检测到用户点击/触摸该第三方应用程序,则说明第三方应用程序启动,否则,第三方应用程序未启动。示例性的,上述第三方应用程序的启动方式可以是用户点击该第三方应用程序时启动,电子设备会通过触摸屏实时检测用户是否点击该第三方应用程序,当检测到用户点击该第三方应用程序的情况下,相应的硬件中断被发给内核层。内核层将点击操作加工成原始输入事件(包括点击坐标,点击操作的时间戳等信息)。原始输入事件被存储在内核层。应用程序框架层会从内核层获取原始输入事件,识别该输入事件所对应的控件(即第三方应用程序)。以该触摸操作是触摸单击操作,该单击操作所对应的控件为浏览器应用图标的控件为例,浏览器应用调用应用框架层的接口,启动浏览器应用,进而通过调用内核层启动显示驱动,通过显示屏194显示画面操作系统会启动该第三方应用程序。需要说明的是,上述触摸操作还可以是双击触摸、长按触摸、短按触摸等多种操作,在此不加以限制。

[0144] 在具体应用中,还可以通过判断电子设备的所有进程中是否有该第三方应用程序的进程,若该电子设备的进程中存在该第三应用程序的进程,则说明上述第三方应用程序启动,否则说明上述第三方应用程序未启动。具体的,可以通过ActivityManager获得电子设备的操作系统里正在运行的activities,上述activities包括进程(Process)、应用程序

包、服务(Service)、任务(Task)等信息，然后获取上述第三方应用程序的唯一标识，利用一个增强for循环获取到电子设备的所有进程，然后基于上述第三方应用程序的唯一标识进行查询，查找该电子设备的所有进程中是否存在与该第三方应用程序的唯一标识一致的进程，若存在，则说明上述第三方应用程序启动，否则说明该第三方应用程序未启动。

[0145] S12：在第三方应用程序初始化时根据操作系统当前使用的字体生成目标点阵图。

[0146] 具体的，当电子设备检测到第三方应用程序启动时，第三方应用程序会进行初始化操作，在初始化操作过程中会调用电子设备的操作系统提供的应用程序编程接口(Application Programming Interface，API)来加载操作系统当前使用的自定义字体，并根据该字体创建能够用于像素比对的目标点阵图。需要说明的是，上述目标点阵图为能够体现操作系统当前使用的字体的字体特征的点阵图，具体可以是采用该操作系统当前使用的字体绘制的一段文字而生成的点阵图，例如位图(bitmap)、像素图等。点阵图是指由n多个像素组成的图，由像素阵列的排列来实现其显示效果，在对点阵图进行操作时，具体是对点阵图中的每个像素点进行操作，即操作的对象是每个像素点，可以改变该像素点的色相、饱和度、透明度等属性来改变该点阵图的显示效果，像素点可以进行不同排列和染色来构成不同的图像。

[0147] 以上述第三方应用程序为浏览器为例，在本实施例中，当启动浏览器时，浏览器会进行字体库初始化，在浏览器进行字体库初始化时，操作系统会通过自定义字体文件读取该自定义字体的字体文件流，然后基于该自定义字体的字体文件流生成目标点阵图。其中，字体文件流是指字体的二进制数据组成的数据流。上述浏览器可以是独立的浏览器应用程序，也可以是使用浏览器内核显示网页的应用程序，在此不加以限制。

[0148] 具体的生成字体目标点阵图过程可以参见图10，图10为本实施例提供的根据操作系统当前使用的字体生成目标点阵图的方法的流程示意图，如图10所示，该方法可以包括S21～S22，详述如下：

[0149] S21：创建位图对象和画布对象。

[0150] 在本实施例中，当监听到第三方应用程序启动时，电子设备的操作系统会自动调用创建位图对象的API来创建一个空的位图对象(bitmap)，如通过Bitmapd createBitmap的API创建一个空的bitmap。然后再创建一个画布对象(canvas)，并将位图对象的参数设置到画布对象中，使得在画布对象中绘制的图像能够被保存在该位图对象中。

[0151] S22：根据操作系统当前使用的字体在所述画布对象中绘制预设字符，并将绘制的结果保存在所述位图对象中，得到目标点阵图。

[0152] 在具体应用中，先通过使用操作系统当前使用的字体对应的字体文件流在画布对象中绘制出预设字符，然后将其保存在位图对象中，此时保存得到的位图对象(bitmap)即为使用操作系统当前使用的字体生成的目标点阵图。为了提高自动比对的准确性，需要设定绘制的预设字符的字号大小和字符内容。示例性的，可以将上述字符内容设定为“这是一段测试文字”，将字号设置为48px，通过使用操作系统当前使用的字体对应的字体文件流在画布对象中绘制出字号为48px，字符内容为“这是一段测试文字”的预设字符，并保存在位图对象中，就可以得到如图11中所示的目标点阵图bitmap target。可理解的是，上述字符内容和字号大小可以根据需求进行设置，在此不再加以赘述。还可以理解的是，位图的大小会影响匹配效果，位图尺寸越大匹配的精度越高，准确率也越高，但是会导致字形比对效率

下降，因此，设置的字号也不是越大越好，为了达到较佳的比对精度和比对效率，经测试发现可以选用字号大小为48px，此时得到的位图尺寸为64*64。

[0153] 为了减少目标点阵图占用的内存空间，上述位图对象还可以是单色位图对象，单色位图对象是指仅由黑白两种颜色构成的位图对象，其采用“0”表示黑色像素点，采用“1”表示白色像素点。

[0154] S13：获取操作系统加载过的各种类型的字体对应的对比点阵图。

[0155] 在本实施例中，上述电子设备的操作系统加载过的各种类型的字体可以是操作系统加载过的所有类型的字体(包括操作系统默认字体和非操作系统默认字体)，也可以是电子设备的操作系统加载过的非操作系统默认字体。可理解的是，本申请是为了实现第三方应用程序使用的字体准确跟随操作系统使用的字体，因此该操作系统加载过的各种类型的字体至少包括两种字体，一个是操作系统变换字体前使用的字体，另一个是操作系统变换字体后使用的字体。

[0156] 需要说明的是，可以在操作系统加载字体的时候就创建各个类型的字体对应的对比点阵图，并将预先创建好的对比点阵图存储在预设存储位置，上述预设存储位置可以根据实际需求进行设置，在此不加以限制和赘述。

[0157] 在具体应用中，可以先获取操作系统加载过的各种类型的字体，然后根据各种类型的字体对应的字体文件流生成其对应的对比点阵图。具体的，可以按照上述生成目标点阵图的方式来生成各种类型的字体中各种类型的字体对应的对比点阵图。即根据各种类型的字体对应的字体文件流在画布对象中绘制出预设字符，然后将绘制的结果保存在各种类型的字体对应的位图对象中，得到各种类型的字体对应的对比点阵图。通过调用操作系统原有的API创建多个新的bitmap(数量与各种类型的字体数量一致)和对应的canvas，然后根据各种类型的字体分别在其对应的canvas中绘制预设字符，然后将其保存在bitmap中，得到各种类型的字体对应的对比点阵图。

[0158] 请继续参阅图11，示例性的，假设操作系统加载过的各种类型的字体中包括字体1、字体2、字体3及字体4这四中类型的字体，通过调用操作系统原有的API创建bitmap1、bitmap2、bitmap3和bitmap4这四个位图对象。根据bitmap1创建canvas1，根据bitmap2创建canvas2，根据bitmap3创建canvas3，根据bitmap4创建canvas4。在canvas1中使用字体1对应的字体文件流绘制字号为48px，字符内容为“这是一段测试文字”的预设字符，并将绘制的结果保存在bitmap1中；在canvas2中使用字体2对应的字体文件流绘制字号为48px，字符内容为“这是一段测试文字”的预设字符，并将绘制的结果保存在bitmap2中；在canvas3使用字体3对应的字体文件流绘制字号为48px，字符内容为“这是一段测试文字”的预设字符，将绘制的结果保存在bitmap3中；在canvas4中使用字体4对应的字体文件流绘制字号为48px，字符内容为“这是一段测试文字”的预设字符，并将绘制的结果保存在bitmap4中。此时bitmap1就是字体1对应的对比点阵图，bitmap2就是字体2对应的对比点阵图，bitmap3就是字体3对应的对比点阵图，bitmap4就是字体4对应的对比点阵图。

[0159] 需要说明的是，在对比点阵图上绘制的预设字符时，需要使用与绘制目标点阵图时相同字号大小和相同字符内容的预设字符进行绘制。由于生成的位图是在相同字号下生成的，因此，得到的字符位图的尺寸也会是一样的，这样能够有效地避免了归一化处理，提高像素比对时的精准度。示例性的，假设操作系统当前使用的字体是字体1，请参阅图12，电

子设备可以根据字体1来生成与其对应的目标点阵图bitmap target，然后根据操作系统加载过的字体1、字体2、字体3及字体4生成与字体1、字体2、字体3及字体4对应的对比点阵图bitmap1、bitmap2、bitmap3、bitmap4。

[0160] 还需要说明的是，在本实施例中，电子设备也可以先获取对比点阵图，之后再根据操作系统当前使用的字体生成目标点阵图。还可以在根据操作系统当前使用的字体生成目标点阵图的同时获取对比点阵图。也就是说，步骤S12与步骤S13之间没有严格的时序执行关系，步骤S12可以在步骤S13之前执行，也可以在步骤S13之后执行，还可以与步骤S13同时执行，本实施例对此不做特别限定。

[0161] 图13为本实施例提供一种获取对比点阵图的方法的流程示意图，如图13所示，该方法可以包括S31～S32，详述如下：

[0162] S31：获取操作系统加载过的所有类型的字体，并生成与所述操作系统加载过的所有类型的字体对应的对比点阵图。

[0163] 在具体应用中，电子设备的操作系统中每个应用进程均对应有一个maps文件(/proc/self/maps)，在该maps文件下，存在一些.ttf字体文件类型的字体的引用记录，这些引用记录用于记录引用过的.ttf字体文件，以.ttf为后缀的文件即为当前进程引用过的字体文件，即该操作系统加载过的字体。通过指令“/proc/self/maps|grep ttf”调出maps文件下的.tff文件就能够获取到操作系统加载过的所有字体文件(以.tff为后缀的文件)，进而获取到操作系统加载过的各种类型的字体。可理解的是，操作系统加载过的各种类型的字体中必定存在操作系统当前使用的字体，因此，通过上述操作获取到的字体中必定包括操作系统当前使用的字体。

[0164] 在具体应用中，根据操作系统加载过的各种类型的字体中的各种类型的字体对应的字体文件流在画布对象中绘制出预设字符，然后保存在各个对比字体文件对应的位图对象中，就能够得到各个对比字体文件对应的对比点阵图。

[0165] 示例性的，假设获取到的操作系统加载过的各种类型的字体文件包括字体1、字体2、字体3及字体4，通过调用操作系统原有的API创建bitmap1、bitmap2、bitmap3和bitmap4这四个位图对象。然后根据bitmap1创建canvas1，根据bitmap2创建canvas2，根据bitmap3创建canvas3，根据bitmap4创建canvas4。在canvas1中使用字体1对应的字体文件流绘制字号为48px，字符内容为“这是一段测试文字”的预设字符，并将绘制的结果保存在bitmap1中；在canvas2中使用字体2对应的字体文件流绘制字号为48px，字符内容为“这是一段测试文字”的预设字符，并将绘制的结果保存在bitmap2中；在canvas3使用字体3对应的字体文件流绘制字号为48px，字符内容为“这是一段测试文字”的预设字符，将绘制的结果保存在bitmap3中；在canvas4中使用字体4对应的字体文件流绘制字号为48px，字符内容为“这是一段测试文字”的预设字符，并将绘制的结果保存在bitmap4中。此时bitmap1就是字体1对应的对比点阵图，bitmap2就是字体2对应的对比点阵图，bitmap3就是字体3对应的对比点阵图，bitmap4就是字体4对应的对比点阵图。

[0166] S32：获取与所述操作系统加载过的所有类型的字体对应的对比点阵图。

[0167] 在具体应用中，根据S31生成的对比点阵图可以存储在特定存储位置，通过查找该位置下的文件就能够得到对比点阵图。

[0168] 在一实施例中，由于用户通过主题商店等应用选择的自定义字体通常都不包含在

操作系统的默认字体中，因此，上述操作系统加载过的各种类型的字体还可以是电子设备的操作系统加载过的非操作系统默认字体，这样能够有效地减少对比点阵图的数量，提高处理效率，减少系统资源的占用。下面以上述操作系统加载过的各种类型的字体为电子设备的操作系统加载过的非操作系统默认字体为例，对获取对比点阵图的过程进行说明。

[0169] 图14为本实施例提供的另一种获取对比点阵图的方法的流程示意图，如图14所示，该方法可以包括S41～S45，详述如下：

[0170] S41：获取操作系统加载过的所有类型的字体。

[0171] 同样地，为了确定操作系统加载过的非操作系统默认字体就需要先获取到操作系统加载过的各种类型的字体，然后再将操作系统的默认字体从操作系统加载过的各种类型的字体中删除，剩下的字体就是操作系统加载过的非操作系统默认字体，具体的获取过程与S31类似，在此不再加以赘述。

[0172] S42：获取操作系统的默认字体。

[0173] 在具体应用中，操作系统的默认字体是指操作系统自带的默认字体，例如在电子设备出厂设置时使用的字体。可理解的是，上述操作系统的默认字体可以只包括一个默认字体，也可以包括多个默认字体，可以包括英文默认字体，也可以包括中文默认字体，在此不加以限制。

[0174] 在具体应用中，操作系统自带的默认字体通常有特定的配置文件来保存其相关的描述信息，通过特定的配置文件就能获取到操作系统的默认字体。该特定的配置文件可以基于不同的操作系统来确定，例如，安卓系统的默认字体的配置文件就是位于workspace/frameworks/base/data/fonts/文件夹下的system_fonts.xml文件，因此可以通过获取位于workspace/frameworks/base/data/fonts/文件夹下的system_fonts.xml文件来获取操作系统的默认字体。具体地，可以通过<fileset></fileset>之间的内容来确定字体类型。

[0175] S43：根据所述操作系统加载过的所有类型的字体和所述操作系统的默认字体，确定非操作系统默认字体。

[0176] 在具体应用中，将操作系统的默认字体从操作系统加载过的所有字体中删除掉之后，就可以确定出非操作系统默认字体。

[0177] 示例性，若获取到的操作系统加载过的所有字体包括字体1、字体2、字体3、字体4及字体5这五种字体。获取到的操作系统的默认字体包括字体1和字体3这两种字体，则非操作系统默认字体中包括字体2、字体4及字体5这三种字体。

[0178] S44：生成非操作系统默认字体对应的对比点阵图。

[0179] 在此，根据非操作系默认字体对应的字体文件流在画布对象中绘制出预设字符，然后将绘制的结果保存在各种类型的字体对应的位图对象中，就能够得到各种类型的字体对应的对比点阵图。

[0180] 示例性的，假设获取到的操作系统加载过的所有字体包括字体1、字体2、字体3、字体4及字体5这五种字体。获取到的操作系统的默认字体包括字体1和字体3这两种字体，也就是说非操作系统默认字体包括字体2、字体4及字体5这三种字体，通过调用操作系统原有的API创建bitmap2、bitmap4和bitmap5这三个位图对象。然后根据bitmap2创建canvas2，根据bitmap4创建canvas4，根据bitmap5创建canvas5。在canvas2中使用字体2对应的字体文件流绘制字号为48px，字符内容为“这是一段测试文字”的预设字符，并将绘制的结果保存

在bitmap2中;在canvas4中使用字体4对应的字体文件流绘制字号为48px,字符内容为“这是一段测试文字”的预设字符,并将绘制的结果保存在bitmap4中;在canvas5使用字体5对应的字体文件流绘制字号为48px,字符内容为“这是一段测试文字”的预设字符,将绘制的结果保存在bitmap5中。此时bitmap2就是字体2对应的对比点阵图,bitmap4就是字体4对应的对比点阵图,bitmap5就是字体5对应的对比点阵图。

[0181] S45:获取与所述非操作系统默认字体对应的对比点阵图。

[0182] 同样地,非操作系统默认字体对应的对比点阵图可以存储在特定存储位置,通过查找该位置下的文件就能够得到对比点阵图。

[0183] S14:将各个对比点阵图与所述目标点阵图进行像素匹配,确定与所述目标点阵图匹配的对比点阵图,并将所述匹配的对比点阵图对应的类型的字体作为目标字体。

[0184] 在具体应用中,将得到的各个对比点阵图分别与目标点阵图进行像素点比对,当匹配到与目标点阵图的像素点分布一致的对比点阵图时,将与目标点阵图的像素点分布一致的对比点阵图作为与目标点阵图匹配的对比点阵图,并将该对比点阵图对应的字体确定为目标字体。

[0185] 在具体应用中,上述目标点阵图和上述对比点阵图可以都是单色位图,通过比较各个单色位图中的每一个像素点的值来判断目标点阵图的像素点分布与对比点阵图的像素点分布是否一致。

[0186] 在此,可以通过分别获取目标点阵图的二进制数据和各个对比点阵图的二进制数据,从而匹配出与目标点阵图的二进制数据完全一致的对比点阵图,则该对比点阵图对应的字体就是目标字体对应的字体。

[0187] 在此,也可以基于现有的位图的匹配度算法算出目标点阵图与各个对比点阵图的匹配度,然后根据目标点阵图与各个对比点阵图的匹配度来确定出与目标点阵图匹配的对比点阵图,并将最匹配的对比点阵图对应的字体作为目标字体。上述根据目标点阵图与各个对比点阵图的匹配度来确定出与目标点阵图匹配的对比点阵图可以是通过设定匹配度阈值,当对比点阵图与目标点阵图的匹配度超过匹配度阈值时,则确定该对比点阵图就是目标字体对应的对比点阵图。在本实施例中,上述匹配度阈值可以是100%,这个可以根据实际需求进行设定,在此不加以限定。需要说明的是,位图的匹配度算法比较多,而且比较成熟,这个可以依据实际需要来选择,在此不加以限制。

[0188] S15:将第三方应用程序的字体切换为所述目标字体。

[0189] 在具体应用中,将第三方应用程序使用的字体切换为该目标字体,可以是将该目标字体的字体文件对应的存放路径设置给到第三方应用程序,然后第三方应用程序就能够在启动的初始化过程中完成字体的切换,使得该第三方应用程序使用上述目标字体,即使用与操作系统当前使用的字体一致的字体。目标字体的字体文件对应的存放路径可以通过第三方应用程序读取maps文件中该字体文件的文件信息记录,进而得到该字体文件的存放路径。

[0190] 以上述第三方应用程序为浏览器为例,对字体切换过程进行说明:浏览器在启动时会进行字体库初始化,此时浏览器根据目标字体的字体文件路径获取目标字体文件并加载该自定义字体文件,通过加载的自定义字体文件显示目标字体文件对应的字体,使用目标字体文件中的字体族对浏览器的显示界面进行渲染,进而使得浏览器的显示界面所使用

的字符为该目标字体文件对应的字符。

[0191] 综上，本申请实施例提供的字体切换方法，能够通过像素比对的方法准确地确定出于操作系统当前使用的字体一致的目标字体，然后将该目标字体应用到将第三方应用程序中，就能够使得第三方应用程序所使用的字体能够与电子设备的操作系统所使用的字体一致，避免第三方应用软件使用的字体与操作系统使用的字体不一致的情况。

[0192] 可以理解的是，上述实施例中各步骤的序号的大小并不意味着执行顺序的先后，各过程的执行顺序应以其功能和内在逻辑确定，而不应对本申请实施例的实施过程构成任何限定。

[0193] 基于同一发明构思，作为对上述方法的实现，本申请实施例提供了一种电子设备，该电子设备实施例与前述方法实施例对应，为便于阅读，本实施例不再对前述方法实施例中的细节内容进行逐一赘述，但应当明确，本实施例中的装置能够对应实现前述方法实施例中的全部内容。

[0194] 图15示出了本申请实施例提供的一种电子设备的结构框图，该电子设备100包括的各单元用于执行上述实施例中的各步骤，具体请参阅上述实施例中的相关描述，为了便于说明，仅示出了与本申请实施例相关的部分。请参阅图13，该电子设备100包括生成单元110、获取单元120、匹配单元130和切换单元140。其中：

[0195] 生成单元110用于根据操作系统当前使用的字体生成目标点阵图；

[0196] 获取单元120用于获取操作系统加载过的各种类型的字体对应的对比点阵图；

[0197] 匹配单元130用于将各个对比点阵图与所述目标点阵图进行像素匹配，确定与所述目标点阵图匹配的对比点阵图，并将所述匹配的对比点阵图对应的字体作为目标字体；

[0198] 切换单元140用于将第三方应用程序的字体切换为所述目标字体。

[0199] 在本申请实施例中，上述电子设备还可以包括监听单元。

[0200] 监听单元用于在根据操作系统当前使用的字体生成目标点阵图之前，实时监听第三方应用程序是否启动。

[0201] 在监听单元监听到第三方应用程序启动时再由第一生成单元执行生成目标点阵图的步骤。

[0202] 在一实施例中，上述生成单元110可以包括：第一创建单元和第一绘制单元。

[0203] 上述第一创建单元用于创建位图对象和画布对象；

[0204] 上述第一绘制单元用于根据操作系统当前使用的字体在所述画布对象中绘制预设字符，并将绘制的结果保存在所述位图对象中，得到目标点阵图。

[0205] 在一实施例中，获取单元120主要用于：获取所述操作系统加载过的所有类型的字体，并生成与所述操作系统加载过的所有类型的字体对应的对比点阵图；获取与所述操作系统加载过的所有类型的字体对应的对比点阵图。

[0206] 具体地，上述获取单元120可以包括第二创建单元和第二绘制单元。

[0207] 上述第二创建单元用于创建所述操作系统加载过的所有类型的字体对应的画布对象和位图对象；

[0208] 第二绘制单元用于根据所述操作系统加载过的所有类型的字体对应的字体文件流在画布对象中绘制出预设字符，并将绘制的结果保存在各个类型的字体对应的位图对象中。

[0209] 在一实施例中，上述获取单元120主要用于获取所述操作系统加载过的所有类型的字体及所述操作系统的默认字体；根据所述操作系统加载过的所有类型的字体和所述操作系统的默认字体，确定非操作系统默认字体；并生成与所述非操作系统默认字体对应的对比点阵图；获取与所述非操作系统默认字体对应的对比点阵图。

[0210] 具体地，获取单元120可以包括第三创建单元和第三绘制单元。

[0211] 上述第三创建单元用于创建与所述各个非操作系默认字体对应的画布对象和位图对象。

[0212] 上述第三绘制单元用于根据所述各个非操作系默认字体对应的字体文件流在画布对象中绘制出预设字符，并将绘制的结果保存在各个非操作系统默认字体对应的位图对象中。

[0213] 在一实施例中，上述匹配单元130主要用于将得到的各个对比点阵图分别与目标点阵图进行像素点比对，当匹配到与目标点阵图的像素点分布一致的对比点阵图时，将与目标点阵图的像素点分布一致的对比点阵图作为与目标点阵图匹配的对比点阵图，并将该对比点阵图对应的字体确定为目标字体。

[0214] 在一实施例中，上述目标点阵图与上述对比点阵图均为单色位图，相应地，上述匹配单元主要用于分别获取所述目标点阵图的二进制数据和各个所述对比点阵图的二进制数据；将与目标点阵图的二进制数据匹配一致的对比点阵图的二进制数据对应的对比点阵图确定为与目标点阵图匹配的对比点阵图，并将匹配到的对比点阵图对应的类型的字体作为目标字体。

[0215] 在一实施例中，上述切换单元主要用于将所述目标字体的字体文件的存放路径设置给第三方应用程序；控制所述第三方应用程序根据所述目标字体的字体文件的存放路径获取并加载目标字体文件；根据加载的目标字体文件对第三方应用程序的显示界面进行渲染，使得所述第三方应用程序的显示界面所使用的字符为所述目标字体文件对应的字符。

[0216] 以上可以看出，本申请实施例提供的一种电子设备，同样能够通过根据操作系统当前使用的字体来创建目标点阵图，然后找到进程中全部加载过的字体(即该操作系统使用过的全部字体)，根据这些字体对应的对比点阵图与目标点阵图进行像素匹配，然后从对比点阵图中找到与目标点阵图最匹配的对比点阵图，并将其对应的字体作为目标字体，再将第三方应用程序的字体切换为目标字体，使得第三方应用程序所使用的字体能够与电子设备的操作系统所使用的字体一致，避免第三方应用软件使用的字体与操作系统使用的字体不一致的情况。

[0217] 需要说明的是，上述装置/单元之间的信息交互、执行过程等内容，由于与本申请方法实施例基于同一构思，其具体功能及带来的技术效果，具体可参见方法实施例部分，此处不再赘述。

[0218] 所属领域的技术人员可以清楚地了解到，为了描述的方便和简洁，仅以上述各功能单元、模块的划分进行举例说明，实际应用中，可以根据需要而将上述功能分配由不同的功能单元、模块完成，即将所述装置的内部结构划分成不同的功能单元或模块，以完成以上描述的全部或者部分功能。实施例中的各功能单元、模块可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以两个或两个以上单元集成在一个单元中，上述集成的单元既可以采用硬件的形式实现，也可以采用软件功能单元的形式实现。另外，各功能单

元、模块的具体名称也只是为了便于相互区分，并不用于限制本申请的保护范围。上述系统中单元、模块的具体工作过程，可以参考前述方法实施例中的对应过程，在此不再赘述。

[0219] 本申请实施例还提供了一种计算机可读存储介质，所述计算机可读存储介质存储有计算机程序，所述计算机程序被处理器执行时可实现上述字体切换方法中的步骤。

[0220] 本申请实施例提供了一种计算机程序产品，当计算机程序产品在移动终端上运行时，使得移动终端执行时可实现上述字体切换方法中的步骤。

[0221] 另外，本申请的实施例还提供一种装置，这个装置具体可以是芯片，组件或模块，该装置可包括相连的处理器和存储器；其中，存储器用于存储计算机执行指令，当装置运行时，处理器可执行存储器存储的计算机执行指令，以使芯片执行上述各方法实施例中的字体切换方法。

[0222] 所述集成的单元如果以软件功能单元的形式实现并作为独立的产品销售或使用时，可以存储在一个计算机可读取存储介质中。基于这样的理解，本申请实现上述实施例方法中的全部或部分流程，可以通过计算机程序来指令相关的硬件来完成，所述的计算机程序可存储于一计算机可读存储介质中，该计算机程序在被处理器执行时，可实现上述各个方法实施例的步骤。其中，所述计算机程序包括计算机程序代码，所述计算机程序代码可以为源代码形式、对象代码形式、可执行文件或某些中间形式等。所述计算机可读介质至少可以包括：能够将计算机程序代码携带到拍照装置/电子设备的任何实体或装置、记录介质、计算机存储器、只读存储器(ROM，Read-Only Memory)、随机存取存储器(RAM，Random Access Memory)、电载波信号、电信信号以及软件分发介质。例如U盘、移动硬盘、磁碟或者光盘等。在某些司法管辖区，根据立法和专利实践，计算机可读介质不可以是电载波信号和电信信号。

[0223] 在上述实施例中，对各个实施例的描述都各有侧重，某个实施例中没有详述或记载的部分，可以参见其它实施例的相关描述。

[0224] 本领域普通技术人员可以意识到，结合本文中所公开的实施例描述的各示例的单元及算法步骤，能够以电子硬件、或者计算机软件和电子硬件的结合来实现。这些功能究竟以硬件还是软件方式来执行，取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能，但是这种实现不应认为超出本申请的范围。

[0225] 在本申请所提供的实施例中，应该理解到，所揭露的装置/网络设备和方法，可以通过其它的方式实现。例如，以上所描述的装置/网络设备实施例仅仅是示意性的，例如，所述模块或单元的划分，仅仅为一种逻辑功能划分，实际实现时可以有另外的划分方式，例如多个单元或组件可以结合或者可以集成到另一个系统，或一些特征可以忽略，或不执行。另一点，所显示或讨论的相互之间的耦合或直接耦合或通讯连接可以是通过一些接口，装置或单元的间接耦合或通讯连接，可以是电性，机械或其它的形式。

[0226] 所述作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元显示的部件可以是或者也可以不是物理单元，即可以位于一个地方，或者也可以分别到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

[0227] 最后应说明的是：以上所述，仅为本申请的具体实施方式，但本申请的保护范围并

不局限于此，任何在本申请揭露的技术范围内的变化或替换，都应涵盖在本申请的保护范围之内。因此，本申请的保护范围应以所述权利要求的保护范围为准。

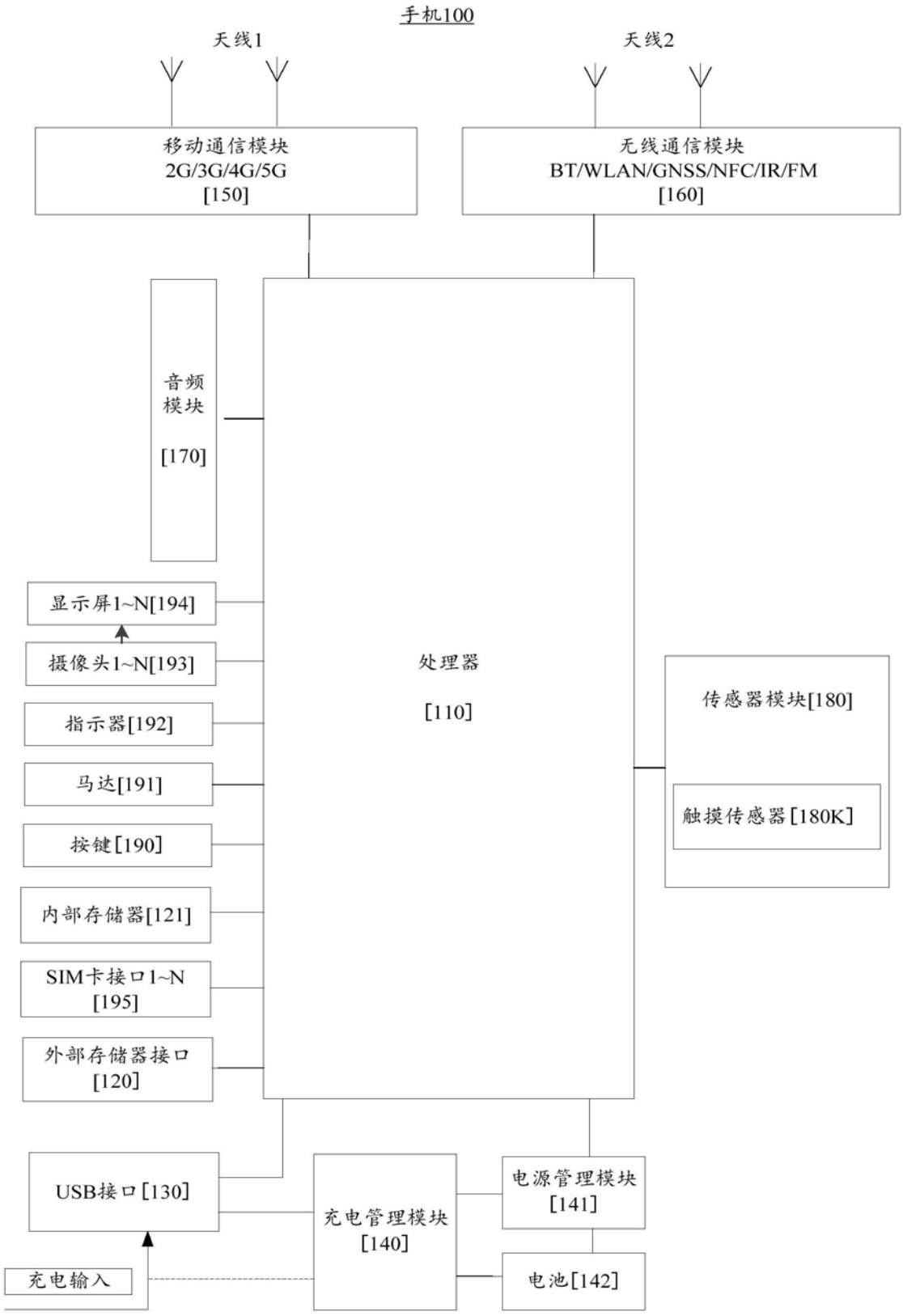

手机100
天线1
天线2
移动通信模块
2G/3G/4G/5G
[150]
无线通信模块
BT/WLAN/GNSS/NFC/IR/FM
[160]
音频
模块
[170]
显示屏1~N[194]
摄像头1~N[193]
指示器[192]
马达[191]
按键[190]
内部存储器[121]
SIM卡接口1~N
[195]
外部存储器接口
[120]
处理器
[110]
传感器模块[180]
触摸传感器[180K]
USB接口[130]
充电管理模块
[140]
电源管理模块
[141]
电池[142]
充电输入

图1

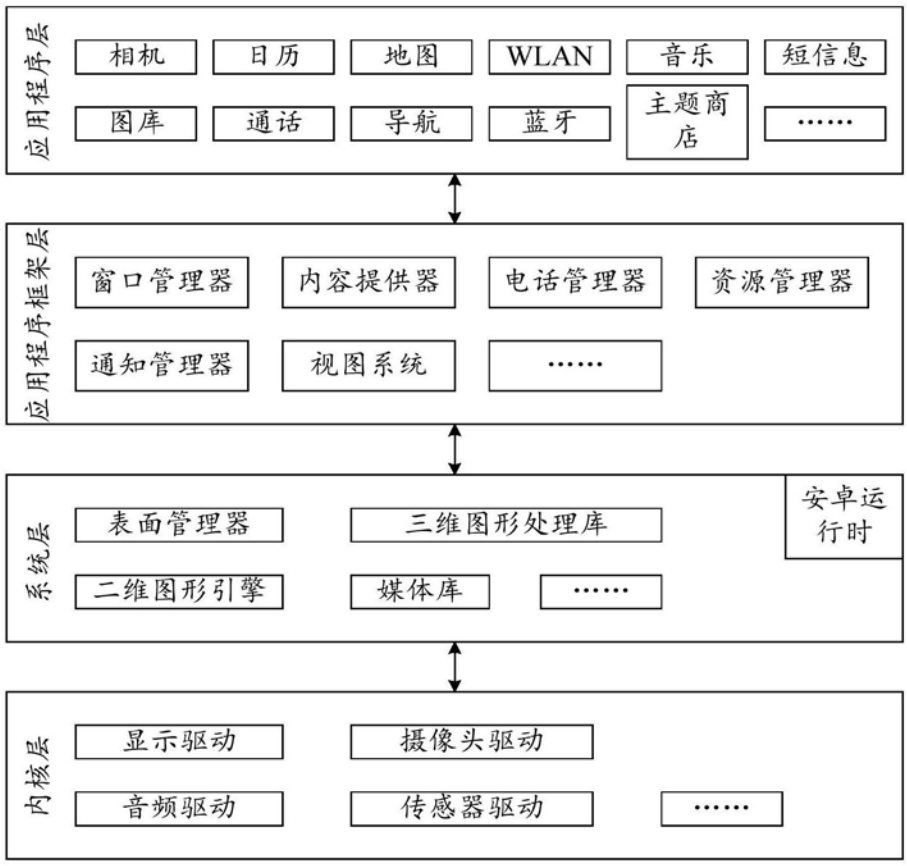

应用程序层
相机
日历
地图
WLAN
音乐
短信息
图库
通话
导航
蓝牙
主题商店
……
应用程序框架层
窗口管理器
内容提供器
电话管理器
资源管理器
通知管理器
视图系统
……
系统层
表面管理器
三维图形处理库
安卓运行时
二维图形引擎
媒体库
……
内核层
显示驱动
摄像头驱动
音频驱动
传感器驱动
……

图2

图3

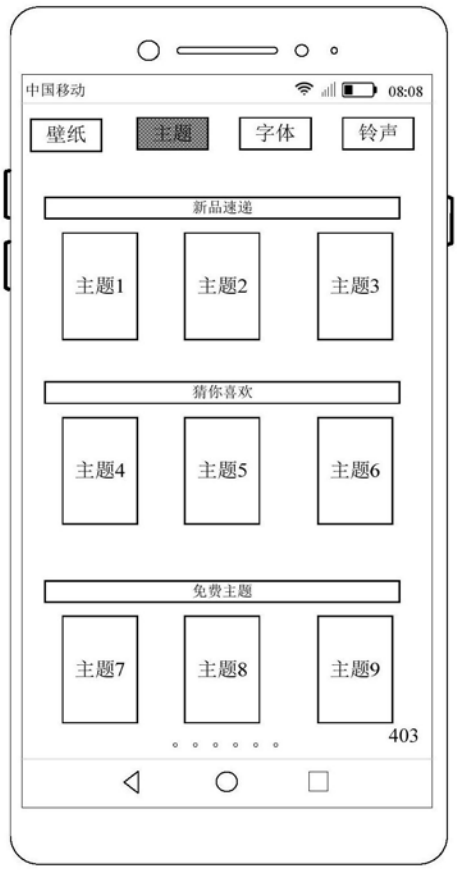

中国移动
08:08
壁纸
主题
字体
铃声
新品速递
主题1
主题2
主题3
猜你喜欢
主题4
主题5
主题6
免费主题
主题7
主题8
主题9
403

图4

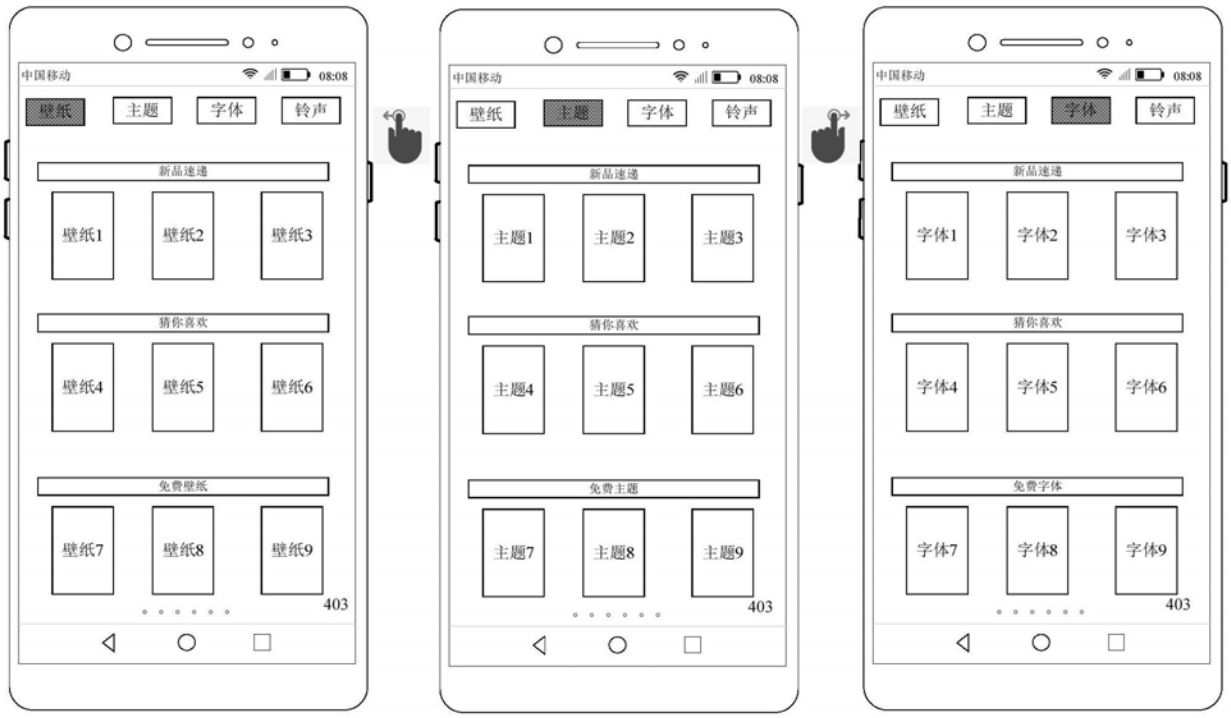

中国移动
08:08
壁纸
主题
字体
铃声
新品速递
壁纸1
壁纸2
壁纸3
猜你喜欢
壁纸4
壁纸5
壁纸6
免费壁纸
壁纸7
壁纸8
壁纸9
403
中国移动
08:08
壁纸
主题
字体
铃声
新品速递
主题1
主题2
主题3
猜你喜欢
主题4
主题5
主题6
免费主题
主题7
主题8
主题9
403
中国移动
08:08
壁纸
主题
字体
铃声
新品速递
字体1
字体2
字体3
猜你喜欢
字体4
字体5
字体6
免费字体
字体7
字体8
字体9
403

图5

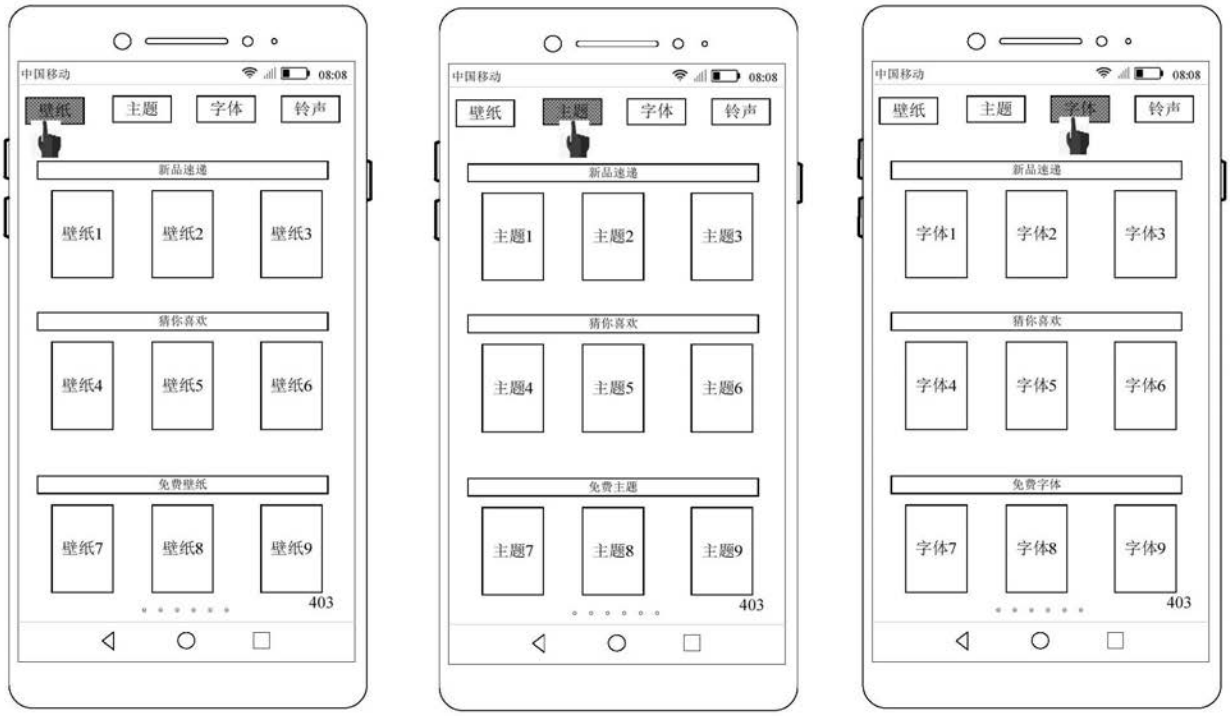

中国移动
08:08
壁纸
主题
字体
铃声
新品速递
壁纸1
壁纸2
壁纸3
猜你喜欢
壁纸4
壁纸5
壁纸6
免费壁纸
壁纸7
壁纸8
壁纸9
403
中国移动
08:08
壁纸
主题
字体
铃声
新品速递
主题1
主题2
主题3
猜你喜欢
主题4
主题5
主题6
免费主题
主题7
主题8
主题9
403
中国移动
08:08
壁纸
主题
字体
铃声
新品速递
字体1
字体2
字体3
猜你喜欢
字体4
字体5
字体6
免费字体
字体7
字体8
字体9
403

图6

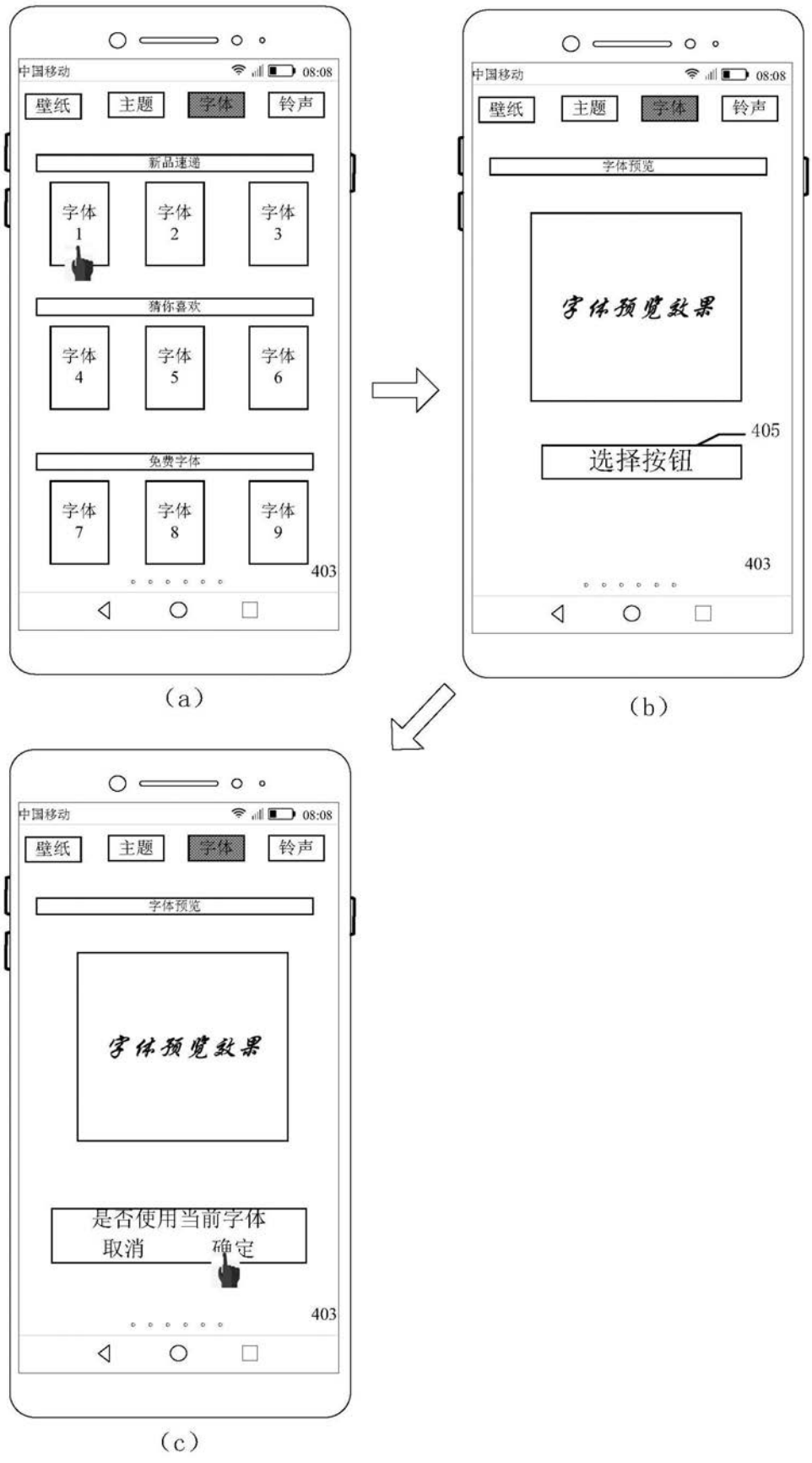

中国移动
08:08
壁纸
主题
字体
铃声
新品速递
字体1
字体2
字体3
猜你喜欢
字体4
字体5
字体6
免费字体
字体7
字体8
字体9
403
(a)
字体预览
字体预览效果
405
选择按钮
403
(b)
是否使用当前字体
取消
确定
403
(c)

图7

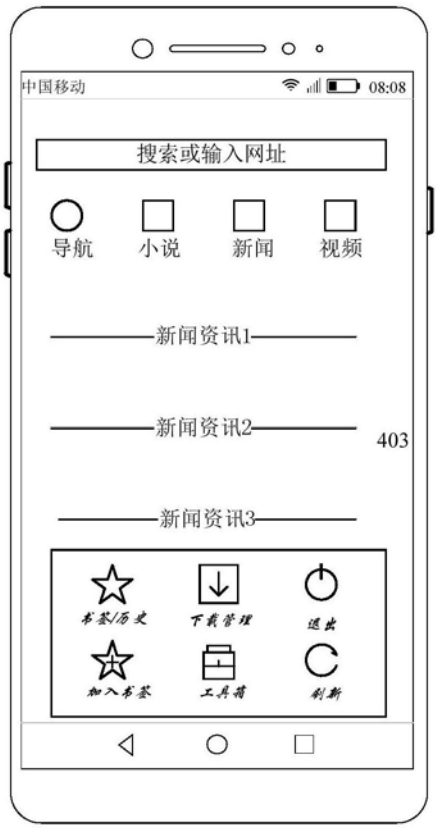

中国移动
08:08
搜索或输入网址
导航
小说
新闻
视频
新闻资讯1
新闻资讯2
403
新闻资讯3
书签/历史
下载管理
退出
加入书签
工具箱
刷新

图8

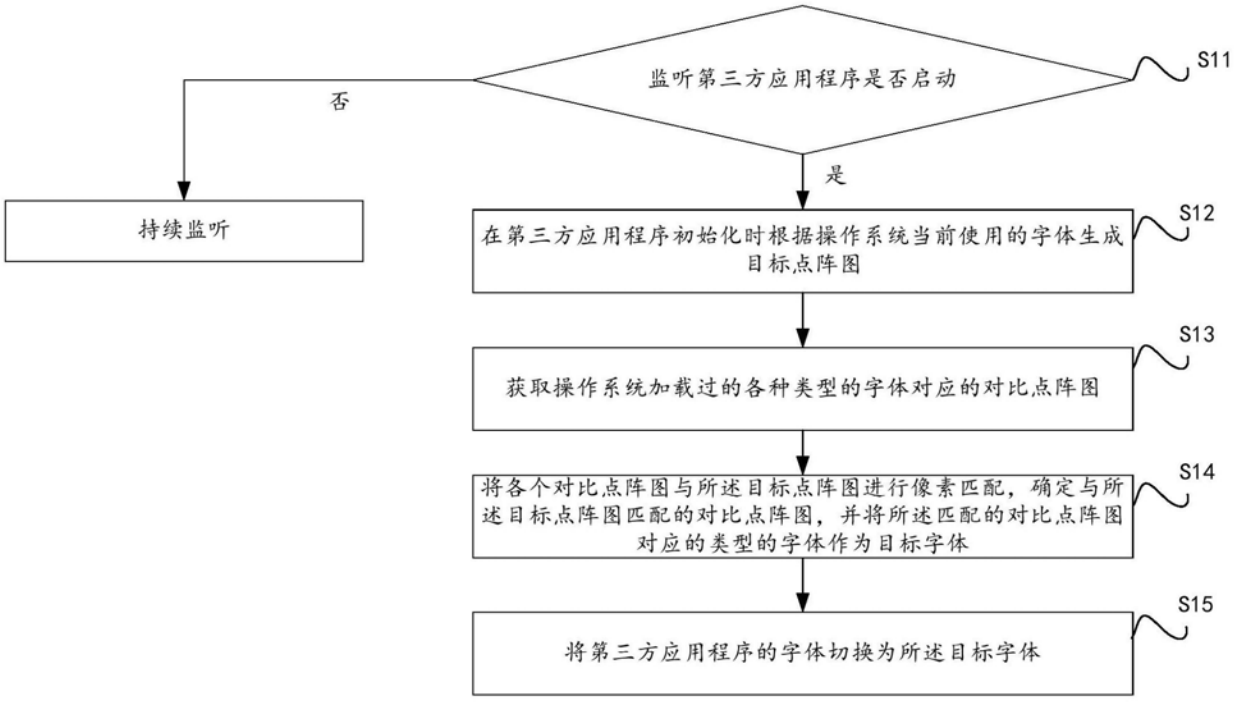

监听第三方应用程序是否启动
S11
否
是
持续监听
在第三方应用程序初始化时根据操作系统当前使用的字体生成目标点阵图
S12
获取操作系统加载过的各种类型的字体对应的对比点阵图
S13
将各个对比点阵图与所述目标点阵图进行像素匹配，确定与所述目标点阵图匹配的对比点阵图，并将所述匹配的对比点阵图对应的类型的字体作为目标字体
S14
将第三方应用程序的字体切换为所述目标字体
S15

图9

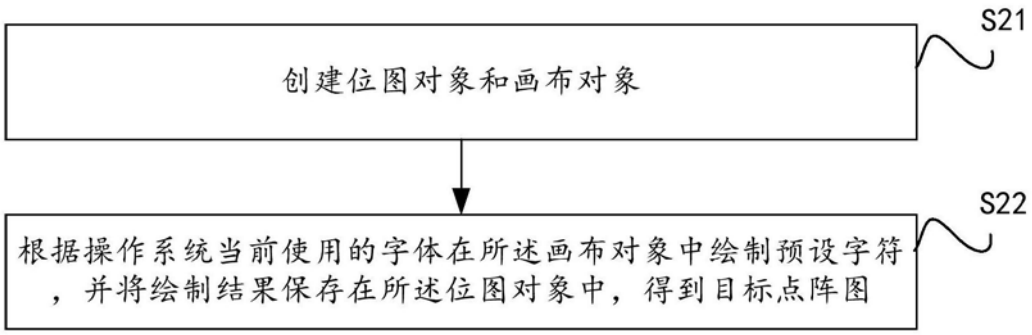

S21
创建位图对象和画布对象
S22
根据操作系统当前使用的字体在所述画布对象中绘制预设字符，并将绘制结果保存在所述位图对象中，得到目标点阵图

图10

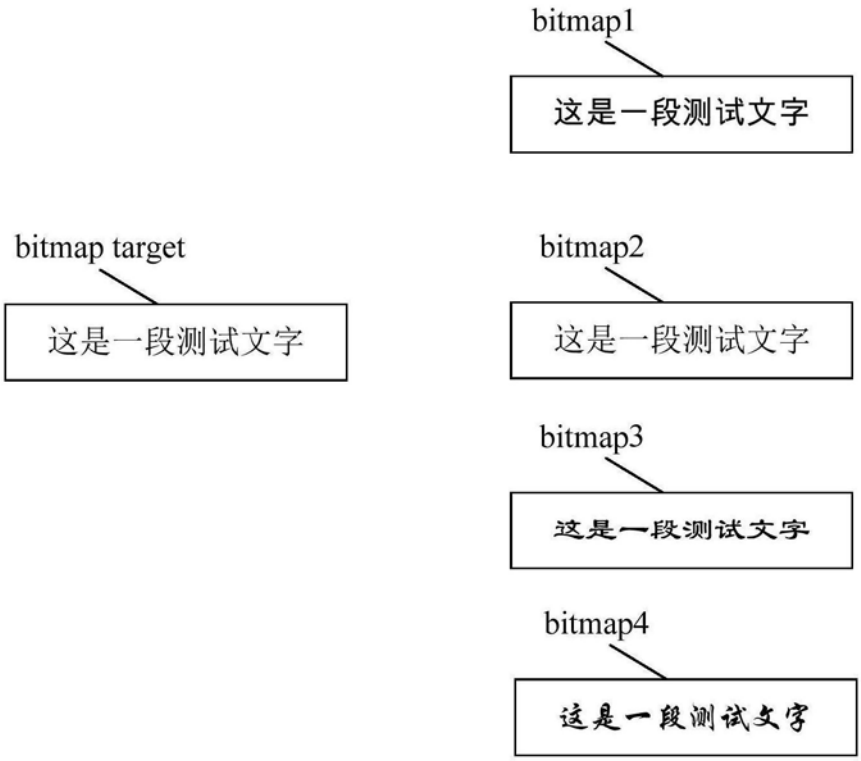

bitmap1
这是一段测试文字
bitmap target
这是一段测试文字
bitmap2
这是一段测试文字
bitmap3
这是一段测试文字
bitmap4
这是一段测试文字

图11

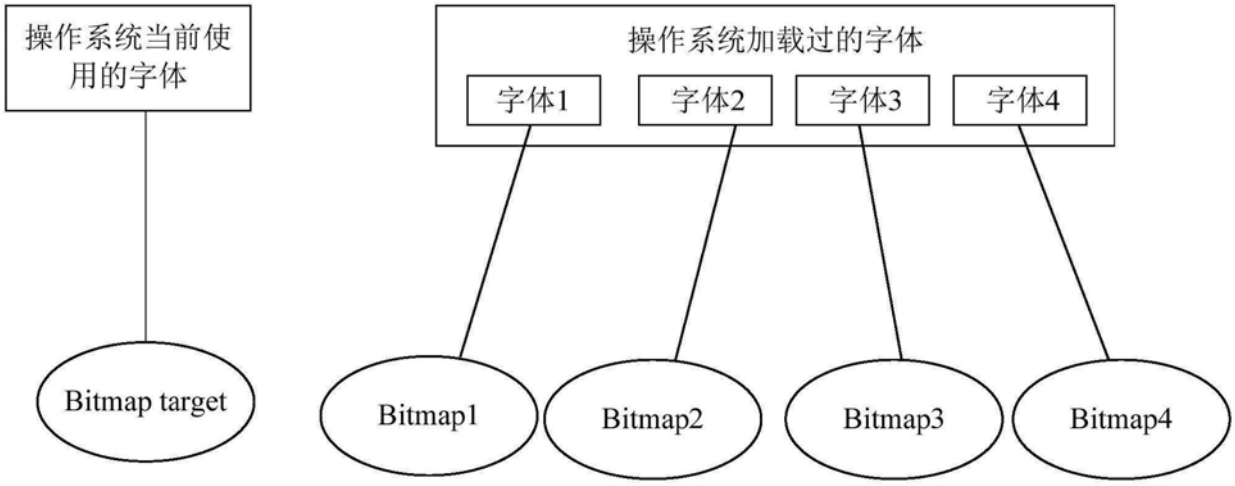

操作系统当前使用的字体
Bitmap target
操作系统加载过的字体
字体1
字体2
字体3
字体4
Bitmap1
Bitmap2
Bitmap3
Bitmap4

图12

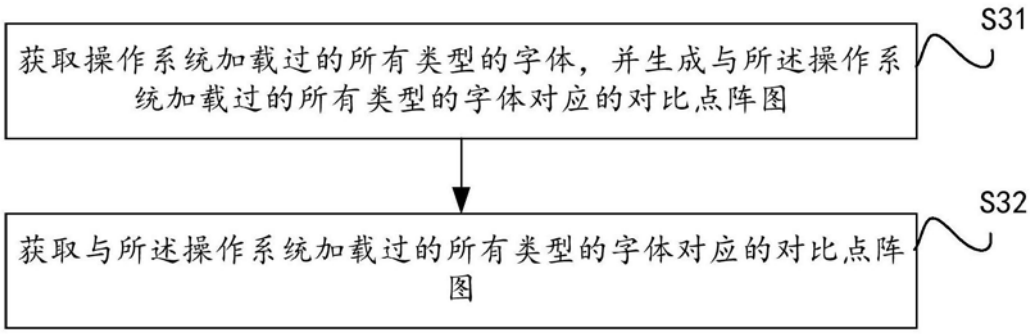

S31
获取操作系统加载过的所有类型的字体，并生成与所述操作系统加载过的所有类型的字体对应的对比点阵图
S32
获取与所述操作系统加载过的所有类型的字体对应的对比点阵图

图13

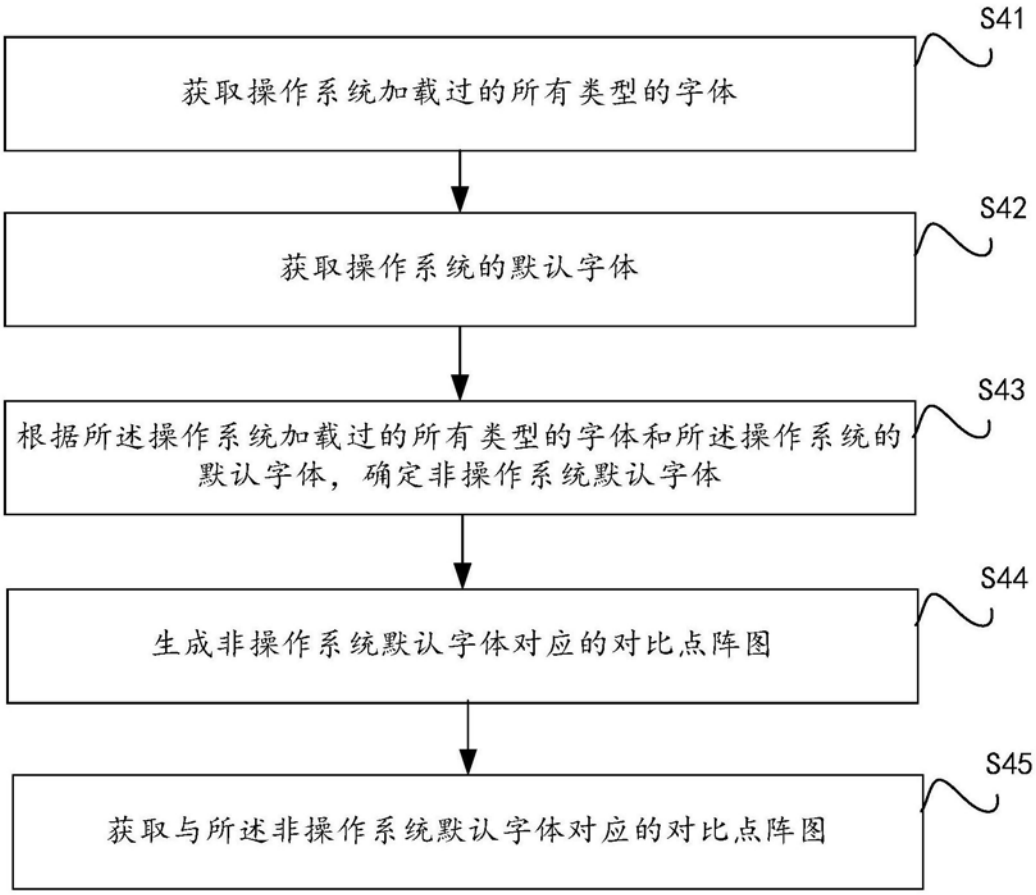

S41
获取操作系统加载过的所有类型的字体
S42
获取操作系统的默认字体
S43
根据所述操作系统加载过的所有类型的字体和所述操作系统的默认字体，确定非操作系统默认字体
S44
生成非操作系统默认字体对应的对比点阵图
S45
获取与所述非操作系统默认字体对应的对比点阵图

图14

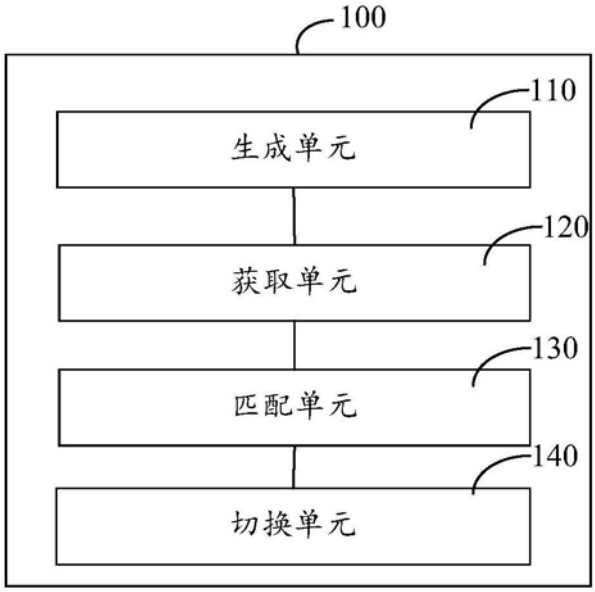

100
110
生成单元
120
获取单元
130
匹配单元
140
切换单元

图15