浅谈Android中的Drawable

在Android开发应用中，少不了和图片打交道，精美的UI页面，富有个性的App icon等都是一个好的Android应用所具备的。为了展示一个较好的视觉效果，app中的图片资源应该选择多大的，放在哪个位置做好,本文就针对Android中的drawable资源来简单说道说道。

1. Drawable 和 Mipmap

现在Android工程下面有很多mipmap-xxx级别的文件夹，很多开发人员对drawbale和mipmap文件夹分不清楚，把图片放在mipmap下面好像也行，App能正常显示。那这两者有什么区别呢？其实Google开发者文档已经明确了两者的使用场景



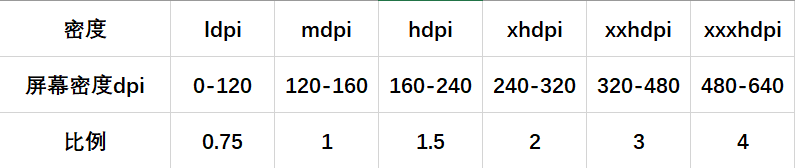
Drawable存放位图文件，平常开发使用的App内部的一些图片都放置在此目录下。而mipmap存放App启动器图片，也就是我们的launcher\_icon，显示在App上的桌面图标。

因为android系统对放在mipmap下的图片内部做了优化，提高了系统渲染图片的速度和图片质量，减少了GPU的压力。

1. PX、DPI、DP

介绍完drawable和mipmap的区别后，接下来重点讲讲drawabale。App的大部分图片都需要放在drawable-xxx目录下，那怎么放呢？依据的规则是什么呢？首先，我们介绍几个基本概念。

* Px：屏幕实际的像素,1280×1920的屏幕在横向有1280个像素，在纵向有1920个像素
* Dpi：屏幕密度,指每英寸上的像素点数。Android根据不同的dpi将Android设备分成多个显示级别,例如:ldpi、mdpi、hdpi、xhdpi等
* Dp：密度无关的像素。使用了这个单位，我们在不同屏幕密度的设备上显示的长度就会是相同的。

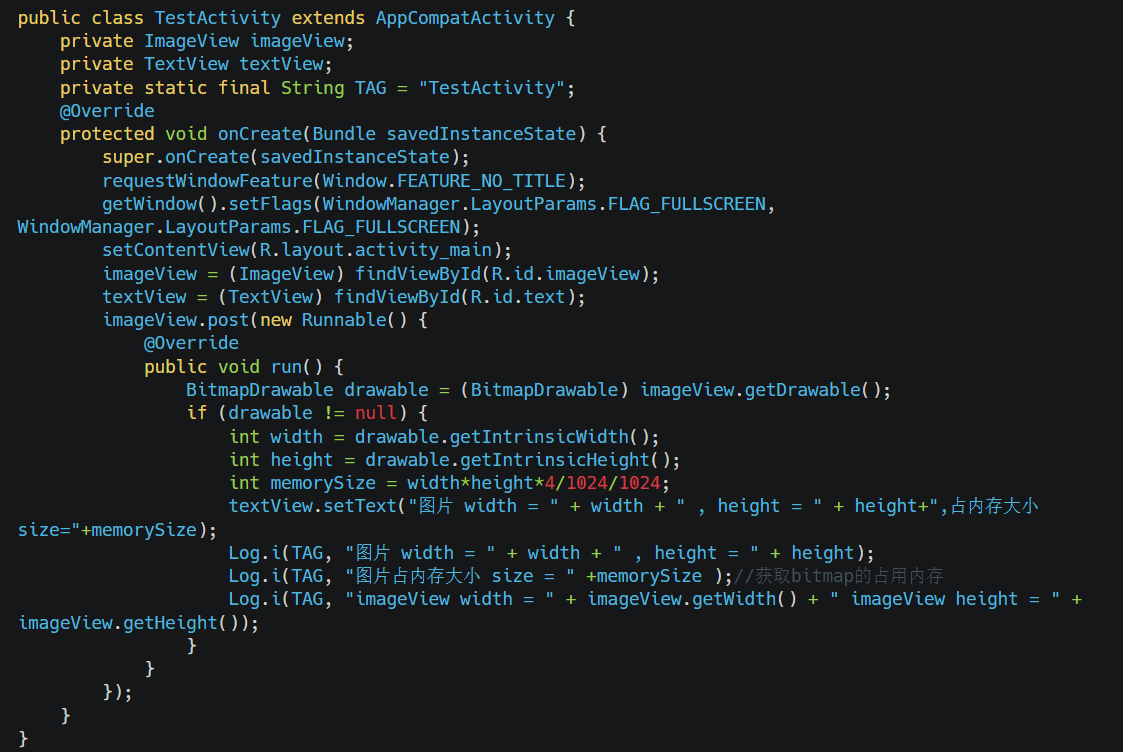
那dp是如何转换成px在手机上显示的呢？谷歌官方将屏幕密度为160dpi的中密度设备屏幕作为基准屏幕，在这个屏幕中1dp = 1px。其他屏幕密度的设备按照比例换算如下图: 

例如在hdpi的设备上1dp=1.5px，在xxhdpi的设备上1dp=3px。

1. Drawable的适配

在Android工程中有许多以drawable开头的文件夹，如drawable-xhdpi/drawable-xxhdpi等，在UI设计阶段，UI设计师一般需要针对drawable-hdpi、drawable-xhdpi、drawable-xxhdpi三种(此三种较为主流)不同的屏幕密度设备设计三套图片，然后将图片分别放置于对应的drawable-xxx目录。这是一般的做法，此方式能比较好的适配市场上各个机型，UI显示效果较为接近一致。但是同时也带来一个问题，如果APP中的每个图片都设计三套，那生成的APK就会很大，免不了遭受客户的吐槽。那能不能只设计一套图片，达到不错的适配效果，如果可以，设计适配多大屏幕密度的设备呢？

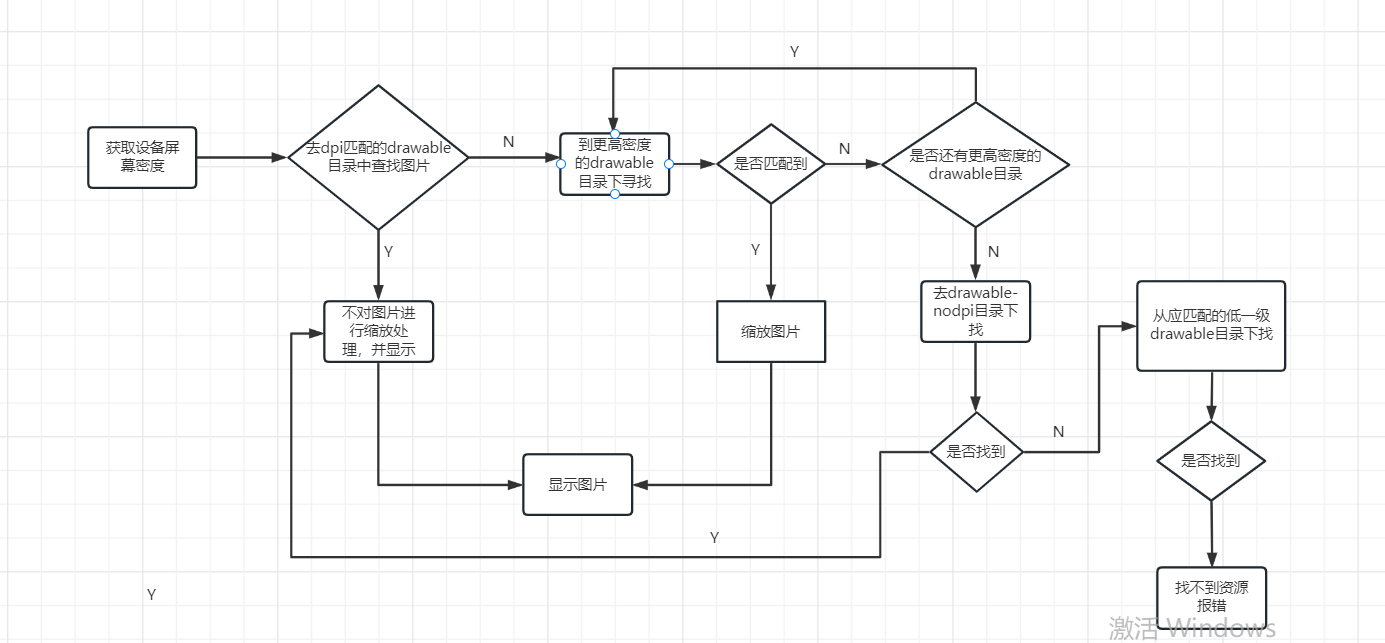
我们先来看下同一张图片在不同的drawable文件夹下在同一个设备上的表现，分别打印出不同场景下的图片的大小以及占用内存情况，简单代码如下:



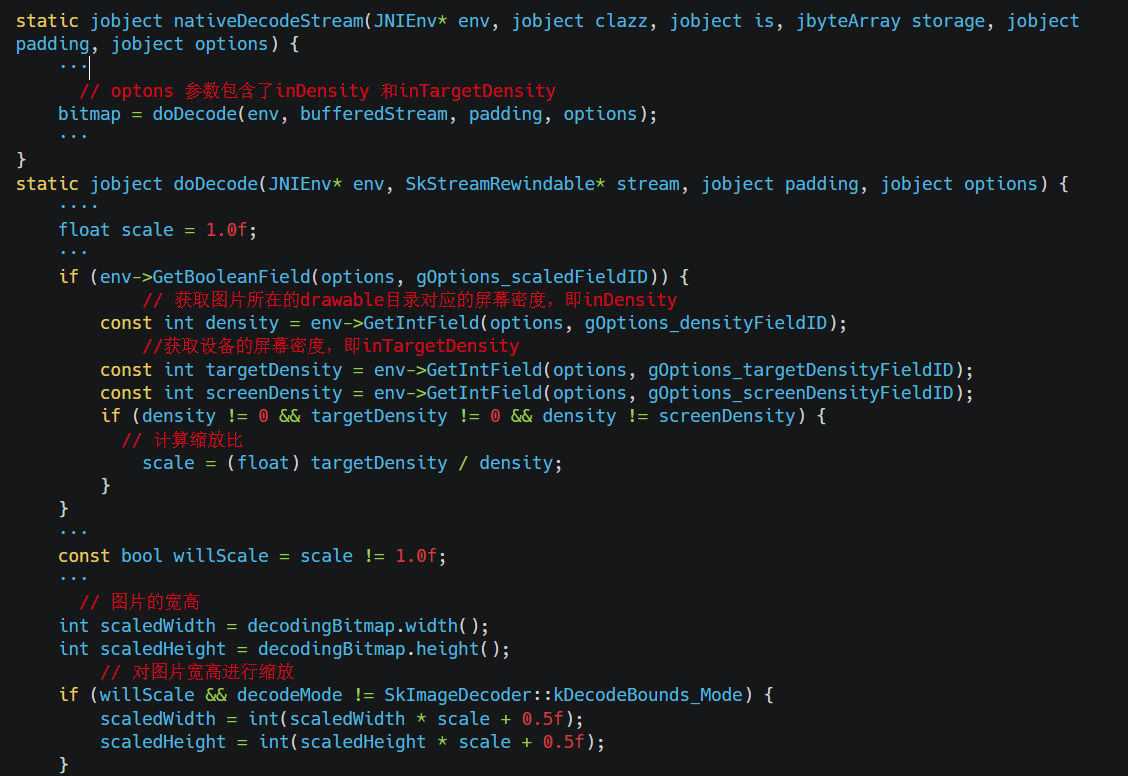
我们准备的是一张600\*960的图片，将此图片分别放在不同的drawable目录下，运行在设备密度480dpi的设备上，得到的结果如下:



可以看到，同一张图片放在不同的目录下，在同一个手机上，会生成不同大小的bitmap，并且占据的内存大小也不相同(分辨率越高的目录下，占据的内存越小)。图片在不同的设备上出现了缩放的现象，这是因为当我们使用资源id来去引用一张图片时，Android会使用一些规则来去帮我们找最匹配的图片。比如我的手机屏幕密度是xxhdpi，那么drawable-xxhdpi文件夹下的图片就是最匹配的图片。因此，当我引用android\_test这张图时，如果drawable-xxhdpi文件夹下有这张图就会优先被使用，在这种情况下，图片是不会被缩放的，保持原图大小。但是，如果drawable-xxhdpi文件夹下没有这张图时， 系统就会自动去其它文件夹下找这张图了，优先会去更高密度的文件夹下找这张图片，我们当前的场景就是drawable-xxxhdpi文件夹，然后发现这里也没有android\_test这张图，接下来会尝试再找更高密度的文件夹，发现没有更高密度的了，这个时候会去drawable-nodpi文件夹找这张图，发现也没有，那么就会去更低密度的文件夹下面找，依次是drawable-xhdpi -> drawable-hdpi -> drawable-mdpi -> drawable-ldpi。那么比如说现在终于在drawable-hdpi文件夹下面找到android\_test这张图了，但是系统会认为你这张图是专门为较低密度的设备所设计的，如果直接将这张图在当前的较高密度设备上使用就有可能会出现像素过低的情况，于是系统自动帮我们做了这样一个放大操作。那么同样的道理，如果系统是在drawable-xxxhdpi文件夹下面找到这张图的话，它会认为这张图是为更高密度的设备所设计的，如果直接将这张图在当前设备上使用就有可能会出现像素过高的情况，于是会自动帮我们做一个缩小的操作。大致流程如下:



从代码层面看android系统加载解析drawable目录下图片使用的是decodeResource,最终调用到natvie层的doDecode方法，大致逻辑如下图:



我们可以看到Android系统调用doDecode()时传入了两个重要的参数，分别是inDensity 和inTargetDensity 。然后拿targetDensity和denisty进行比值，拿到缩放比，再将图片的宽高按照int(scaledWith\*scale+0.5f)的公式进行换算，得到最终的图片宽高！我们上面做的测试600\*960图片大小，设备屏幕密度480dpi，如果放置在hdpi(240dpi)目录下，图片宽高分别为600\*（480/240）=1200 960\*（480/240）=1920，图片进行了缩放。

基于上面的分析，只设计一套UI效果图是可以满足App的日常需求的。因为图片可以在不同的设备上自动进行缩放，自动适配。那应该设计一套适配哪种dpi的呢? 其实不难分析出来，如果制作适配hdpi的图片，并且放置位置也是drawable-hdpi目录中，其他除了mdpi的设备在显示的时候都是放大原图，dpi越大的设备显示的图片就会越模糊，那如果是制作适配xxhdpi的图片，在除了xxxhdpi的设备上显示的都是缩小图，不会出现模糊的情况，效果更佳，并且目前市场上主流的设备都素都是xxhdpi的机器，因此总结下来，制作一套适配xxhdpi的UI效果，能满足大部分的情况。

四．总结

以上就是关于android中drawable资源的一些介绍，希望能给大家在平常的开发中带来一点帮助。