|  |  |
| --- | --- |
| dp和px，那些不得不吐槽的故事Android平台图片文字元素单位浅析 [android](http://search.sina.com.cn/?c=blog&q=android&by=tag)  [开发](http://search.sina.com.cn/?c=blog&q=%BF%AA%B7%A2&by=tag)  [单位](http://search.sina.com.cn/?c=blog&q=%B5%A5%CE%BB&by=tag)  [设计](http://search.sina.com.cn/?c=blog&q=%C9%E8%BC%C6&by=tag)  [dp](http://search.sina.com.cn/?c=blog&q=dp&by=tag)  [px](http://search.sina.com.cn/?c=blog&q=px&by=tag)  [sp](http://search.sina.com.cn/?c=blog&q=sp&by=tag) |  |

一个优秀的手机软件，不仅要有精巧的功能，流畅的速度，让人赏心悦目的UI也往往是用户选择的重要理由。作为移动产品的PM，也需要了解一些在UI设计中的基本知识。

**1. px和pt，一对好伙伴**

在视觉设计中，我们经常会见到以下两个单位： px和磅数pt。

Px ，是英文单词pixel的缩写，意为像素。在设计领域中，像素是用来计算数码影像的一种单位。计算机中显示的图像并非连续的线条组成，而是由许多肉眼看不见的小点组成。如果把把影像放大数倍，会发现这些连续色调其实是由许多色彩相近的小点所组成，这些小点就是构成影像的最小单位“像素”。由于是最小的独立显示单位，px均为整数，不会出现0.5px的情况。

**图一** 看这个色彩鲜艳的LED灯（原图大小）

[dp和px，那些不得不吐槽的故事——Android平台图片文字元素单位浅析](http://photo.blog.sina.com.cn/showpic.html#blogid=6499f8f101014ipq&url=http://s4.sinaimg.cn/orignal/6499f8f1hbd7b87ba5df3)

**图二** 你能想象这才是他的本来面目吗？（放大之后）



Pt，则是point的缩写，一般音译为磅数，也有人直译为点数，请大家自行转换。这是使用在印刷领域的单位，一磅等于1/72英寸。在国际上一般会用pt作为字体的单位。

一般情况下，设计师们采用px来进行标注设计图中的图形，用pt来标注文字。在过去很多年的视觉设计中，px和pt的配合堪称天衣无缝，把PC打扮得花枝招展，堪称史上最佳配合的好伙伴。

**2. 那些年，我们一起追过的Android**

在突如其来的移动互联网时代，px和pt的配合变得不那么和谐。原因就在于Android携众高矮不一，胖瘦各异的爱妃来到世人面前，给他们化妆就变成横看成岭侧成峰，远近高低各不同了。

由于Google对Android的开放政策，使得所有厂商都可以加入到智能手机的研发中来。于是乎，Android手机一时间百花齐放，出现了各种不同的手机样式。比如：

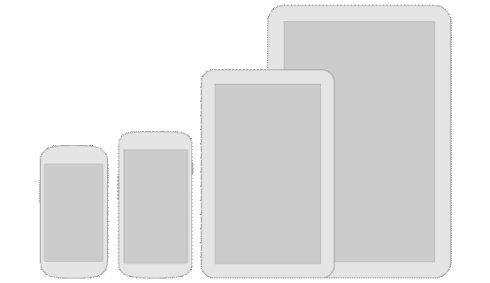
**表一** Android分辨率规格（宽\*高）：

240\*320 320\*240 320\*480 480\*800 480\*854 640\*960 720\*1280，……还有平板

|  |  |  |
| --- | --- | --- |
| 分辨率 | 比率 | 市场对应手机 |
| 320×480 | 2:3 | 三星S5830 Galaxy Ace、三星 I5700、三星I7500、华为U8500 |
| 索尼爱立信X8 (E15i) 、华为 C8600、Motorola XT502 |
| Motorola ME600、Motorola CLIQ、Motorola ME501 |
| LG P503、LG GT540、LG GW620、LG P500 |
| HTC G6 Legend、HTC Aria (G9)、酷派 D530、酷派 W711 |
| 夏普 SH8118U、天语 W606 、中兴 R750、首派 A60 |
| 240x320 | 3:4 | HTC A3366 Wildfire、HTC A3380 、HTC A3360 |
| Motorola ME511 、Motorola XT301 、Motoroal XT300 |
| 华为 C8500、三星 I5508、中兴 X850、索尼爱立信 U201、SAMSUNG S5570、 |
| 480x800 | 3:5 | HTC Desire (G7)、HTC A9191 、HTC 双擎T9199 |
| HTC A9191、HTC Desire HD (G10) 、HTC Desire Z |
| HTC Incredible S (S710e)、 HTC EVO 4G、三星 I9003 |
| 三星 I9010、三星 I9088、三星 I909、三星 I9088、三星I9000 |
| 三星I909 (SPhone)、三星I9020、三星T959、三星W899 |
| 酷派N930、Google Nexus S、、酷派 N930、戴尔Venue |
| 联想W101 乐phone (1.6系统非原生)、中兴 V9E、中兴 V880 |
| 联想W100乐phone(1.6)、联想C101、夏普 SH8128 、 |
| 夏普 SH7218U(新机)、Motorola A1680、华为 U8800 |
| 戴尔Mini5 (Streak) 、Google Nexus One (G5) |
| 600x1024 | 75:128 | 三星 Galaxy P1000 |
| 480x854 |  | Motorola XT702、Motorola XT711 |
| Motorola ME722 (里程碑2)、Motorola XT800 |
| Motorola ME811 (Droid X)、Motorola ME525 |
| Motorola XT800+ 、Motorola XT806Ix |
| 索尼爱立信 X10i |

**图三** Android屏幕物理尺寸规格

屏幕物理尺寸3.2、3.5、3.75、4.0、4.3、5.0、7、9.3、10.1



为了更好的描述这种多样化引发的问题，我们需要引入一个新的概念density，也就是密度。它指的是在一定尺寸的物理屏幕上显示像素的数量，一般使用dpi(dots per inch，每英寸像素数)作为单位。比如一台分辨率为240x320，屏幕物理尺寸为1.5英寸x2英寸的手机，他的密度可以用分辨率/尺寸，240/1.5或者320/2，密度为160dpi。对于用户来说，密度越大的屏幕显示效果越精细，因为每英寸显示的像素数量更多。

如果是要使用px和pt给程序贴图，假设生成了一条240px的直线，在240\*320的屏幕上刚好可以铺满，而放到480x800的分辨率下才刚刚到一半的位置。

如果要解决这样的问题，那么程序在实现的时候就必须为每个分辨率都写一个配置，否则就会出现图形在低密度的屏幕上显示的更大，而在高密度的屏幕上显示的更小。最麻烦的事情在于，一旦出现新的分辨率，所有程序都没办法直接运行了。这明显不符合手机快速发展的客观规律。

3. 新的搭档，dp和sp

为了解决这个问题，Google为Android引入了一套新的单位dp和sp。

首先登场的是dp，Density Independent Pixel，可以翻译为密度无关像素。和px相比，dp在不同密度的屏幕中实际显示比例将保持一致。根据规定，一个dp相当于160dpi屏幕中的一个px。在320dpi的屏幕中，一个dp相当于2个px。通过这样的成比例放缩，Android解决了需要多个不同屏幕中的大小显示问题。

为了便于对多屏幕进行管理，**Android对屏幕密度做了重新的规定，将密度与分辨率绑定起来，如下表**

**表二 Android屏幕密度**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 密度  屏幕大小 | Low density  (120)ldpi | Medium density  (160)mdpi | High density  (240)hdpi | Extra high density  （320）xdpi |
| Small screen  2.5寸到3.2寸 | 240x320 |  |  |  |
| Normal screen  3寸到4.3寸 | 240x400  240x432 | 320x480 | 480x800  480x854  540x960 | 640x960 |
| Large screen 4寸到7寸 | 480x800  480x854 | 480x800  480x854 |  |  |

**具体而言，当屏幕的的分辨率为320x480时，无论实际的物理尺寸大小，密度都会被设置为160dpi，也就意味着这个设备上，一行最多可以放置320dp，而对于480x800的设备，只需屏幕尺寸在3寸到4.3寸之间，密度都会被设置为240dpi，一行同样可以放置320dp。**

因此在开发中，用dp作为单位，只需要设置一次，就可以适配到多个屏幕上。

同样，sp，scaled pixels，这个单位也采用了和dp同样的设计理念，将需要独立设置的pt转化为可以自动适配的sp，从而解决了文字的优化显示。

4. 设计的疑惑

既然android提供了这么方便的适配机制，设计师的效果图也如此惊艳，为什么最后实现出来的效果经常会让人大跌眼镜，列表的高度不对，文字的大小不对。原因何在呢？

原因也正好出现在设计和实现的单位转换上。

设计师在设计的过程中，一般仍然采用了px和pt作为标注单位。根据目前的主流分辨率和发展趋势来看，我们将480x800定义为设计图的基准版本。通过查询上边的密度对应表，我们可以看到在这个分辨率下1dp=1.5px。

举例而言，19px的线条，转化成dp，应该使用19px/1.5= 12.67，由于dp无法支持小数点，所以一般会增加0.5dp，再四舍五入，变为13dp。当他显示到屏幕中的时候，仍然会对应成为真实的像素13dp\*1.5=19.5，最终显示将会变为20px。这样的差异可能会体现在软件的各个地方，在每一个px都是体验问题的时代，就变得让人难以接受了。

看看下表的对应关系

**表三 dp、sp在屏幕中的实际显示效果（480x800分辨率下测得）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **sp/dp值** | **实测pt值** | **理论px值** | **修正偏移** | **实测px** |
| 1 | 7 | 1.5 | 1.5+0.5 | 2 |
| 2 | 8 | 3 | 1.5x2 | 3 |
| 3 | 9 | 4.5 | 1.5x3+0.5 | 5 |
| 4 | 10 | 6 | 1.5x4 | 6 |
| 5 | 10 | 7.5 | 1.5x5-0.5 | 7 |
| 6 | 12 | 9 | 1.5x6 | 9 |
| 7 | 13.5 | 10.5 | 1.5x7+0.5 | 11 |
| 8 | 14 | 12 | 1.5x8 | 12 |
| 9 | 16.5 | 13.5 | 1.5x9+0.5 | 14 |
| 10 | 17 | 15 | 1.5x10 | 15 |
| 11 | 17 | 16.5 | 1.5x11+0.5 | 17 |
| 12 | 19 | 18 | 1.5x12 | 18 |
| 13 | 20 | 19.5 | 1.5x13-1.5 | 18 |
| 14 | 22 | 21 | 1.5x14+1 | 20 |
| 15 | 23 | 22.5 | 1.5x15+-1.5 | 21 |
| 16 | 25 | 24 | 1.5x16+1 | 23 |
| 17 | 25 | 25.5 | 1.5x17-1.5 | 24 |
| 18 | 26 | 27 | 1.5x18-1 | 26 |
| 19 | 27 | 28.5 | 1.5x19-1.5 | 27 |
| 20 | 28 | 30 | 1.5x20-1 | 29 |
| 21 | 略 | 31.5 | 1.5x21-1.5 | 30 |
| 22 | 略 | 33 | 1.5x22-1 | 32 |
| 23 | 略 | 34.5 | 1.5x23-1.5 | 33 |
| 24 | 略 | 36 | 1.5x24-1 | 35 |
| 25 | 略 | 37.5 | 1.5x25-1.5 | 36 |
| 26 | 略 | 39 | 1.5x26-1 | 38 |
| 27 | 略 | 40.5 | 1.5x27-1.5 | 39 |
| 28 | 略 | 42 | 1.5x28+1 | 41 |

第一列是在android程序中使用的dp/sp单位，第二列是sp实际生成的文字的pt大小，第三列是dp根据密度定义推算出的理论大小，第四列是dp理论大小和实际显示大小的差距，第五列是dp在手机屏幕上实际显示的大小。（以上数值都是在480x800的分辨率下测得）

从图中不难看出，许多在ps中可以轻易实现的线条长度和字体大小在Android的实际显示中无法正确展示。

5. 启示和疑问

对于PM和UI设计来说，我们关注的是在Android平台的开发中，如何保证设计效果被良好的实现。这里提供两个解决方案：

1. 使用320x480作为基础分辨率，因为密度和Android定义的标准设备密度一致，均为160dpi。视觉设计图上的元素可以和屏幕显示一一对应，最大程度保障了设计效果和实现的一致性。缺陷是在适配高分辨率时，需要再出一套高分辨率切图，工作量较大。同时在高分辨率机型上，由于显示不准确仍然需要调整。

2. 使用480x800作为基础分辨率。在设计元素尺寸和间距时，采用系统可以显示的长度和文字大小，从而保证实现时的RD反复调试，但是找不到合适大小的尴尬局面。同时，高分辨率资源可以在低分辨率上自动压缩复用，需要调整的元素相对较少。

Google目前对Android采取的开放式政策，让它的发展势头如火如荼，机型种类众多。但是高速发展中，为开发者埋下无数的隐忧。除了文中分析的显示错位的问题，还有包括像双卡双待，系统数据库字段修改等问题，也很容易引发程序出现各类奇怪的bug。这也提醒Android产品的设计师和PM注意，需要经常切换各种手机试用，发现其中的问题。

最后，存留有一个疑问，笔者尚未能得到答案。Android在修正0.5像素无法显示的问题时采用的修正值非常奇怪，并且没有发现明显的规律。如有同学清楚，望不吝赐教。

## [android中像素单位dp、px、pt、sp的比较](http://www.cnblogs.com/chiao/archive/2011/07/07/2100216.html)

dp(dip): device independent pixels(设备独立像素). 不同设备有不同的显示效果,这个和设备硬件有关，一般我们为了支持WVGA、HVGA和QVGA 推荐使用这个，不依赖像素。  
  
px: pixels(像素). 不同设备显示效果相同，一般我们HVGA代表320x480像素，这个用的比较多。  
  
pt: point，是一个标准的长度单位，1pt＝1/72英寸，用于印刷业，非常简单易用；  
  
sp: scaled pixels(放大像素). 主要用于字体显示best for textsize。  
  
由此，根据 google 的建议，TextView 的字号最好使用 sp 做单位，而且查看TextView的源码可知 Android 默认使用 sp 作为字号单位。

还有, google代码里边所用的单位都是以px为默认单位的.

apk的资源包中，当屏幕density=240时使用hdpi标签的资源

当屏幕density=160时，使用mdpi标签的资源

当屏幕density=120时，使用ldpi标签的资源。

在每英寸160点的显示器上，1dp = 1px。

下面是几种不同单位的相互转换.

public static int dip2px(Context context, float dipValue){   
final float scale = context.getResources().getDisplayMetrics().density;   
return (int)(dipValue \* scale + 0.5f);   
}   
public static int px2dip(Context context, float pxValue){   
final float scale = context.getResource().getDisplayMetrics().density;   
return (int)(pxValue / scale + 0.5f);   
}   
public static int dip2px(Context context, float dipValue){   
final float scale = context.getResources().getDisplayMetrics().density;   
return (int)(dipValue \* scale + 0.5f);   
}   
public static int px2dip(Context context, float pxValue){   
final float scale = context.getResource().getDisplayMetrics().density;   
return (int)(pxValue / scale + 0.5f);   
}

**下面说下如何获取分辨率:**

    在一个Activity的onCreate方法中，写入如下代码：  
        DisplayMetrics metric = new DisplayMetrics();  
        getWindowManager().getDefaultDisplay().getMetrics(metric);  
        int width = metric.widthPixels;  // 屏幕宽度（像素）  
        int height = metric.heightPixels;  // 屏幕高度（像素）  
        float density = metric.density;  // 屏幕密度（0.75 / 1.0 / 1.5）  
        int densityDpi = metric.densityDpi;  // 屏幕密度DPI（120 / 160 / 240）

在 Android 中，  1pt 大概等于 2.22sp

以上供参考，如果 UI 能够以 sp 为单位提供设计是最好的，如果设计中没有 sp的概念，则开发人员也可以通过适当的换算取近似值。  
  
过去，程序员通常以像素为单位设计计算机用户界面。例如，定义一个宽度为300像素的表单字段，列之间的间距为5个像素，图标大小为16×16像素 等。这样处理的问题在于，如果在一个每英寸点数（dpi）更高的新显示器上运行该程序，则用户界面会显得很小。在有些情况下，用户界面可能会小到难以看清 内容。与分辨率无关的度量单位可以解决这一问题。

Android支持下列所有单位。  
px（像素）：屏幕上的点。  
in（英寸）：长度单位。  
mm（毫米）：长度单位。  
pt（磅）：1/72英寸。  
dp（与密度无关的像素）：一种基于屏幕密度的抽象单位。在每英寸160点的显示器上，1dp = 1px。  
dip：与dp相同，多用于android/ophone示例中。  
sp（与刻度无关的像素）：与dp类似，但是可以根据用户的字体大小首选项进行缩放。