dip device independent pixels(设备独立像素). 不同设备不同的显示效果,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。

在android上开发的程序将会在不同分辨率的手机上运行。为了让程序外观不至于相差太大，所以引入了dip的概念。比如定义一个矩形10 x 10dip.在分辨率为160dpi的屏上，比如G1，正好是10 x 10像素。而在240 dpi的屏，则是15 x 15像素.换算公式为pixs = dips \* (density/160). density就是屏的分辨率

这里要特别注意dip与屏幕密度有关，而屏幕密度又与具体的硬件有关，硬件设置不正确，有可能导致dip不能正常显示。在屏幕密度为160的显示屏上，1dip=1px，有时候可能你的屏幕分辨率很大如480\*800，但是屏幕密度没有正确设置比如说还是160，那么这个时候凡是使用dip的都会显示异常，基本都是显示过小。

dip的换算： dip（value）=(int) (px（value）/1.5 + 0.5)

dp （与密度无关的像素）：同dip是一样的。 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。

px pixels（像素） 屏幕上的点，不同设备不同的显示屏显示效果相同，这是绝对像素，是多少就永远是多少不会改变。一般我们HVGA代表320x480像素，这个用的比较多。

sp scaled pixels — best for text size (放大像素)（与刻度无关的像素）：主要处理字体的大小。与dp类似，但是可以根据用户的字体大小首选项进行缩放。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。

使用建议：

根据google的推荐，像素统一使用dip，字体统一使用sp

为了使用户界面能够在现在和将来的显示器类型上正常显示，建议大家始终使用sp作为文字大小的单位，将dip作为其他元素的单位。当然，也可以考虑使用矢量图形，而不是用位图

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。

其他单位：(不经常用)

pt（磅） 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；

在 Android 中，1pt 大概等于 2.22sp

in inches（英寸）：长度单位。

mm millimeters（毫米）：长度单位。

---------------------------

举个例子区别px和dip：

px就是像素，如果用px,就会用实际像素画，比个如吧，用画一条长度为240px的横线，在480宽的模拟器上看就是一半的屏宽，而在320宽的模拟器上看就是2／3的屏宽了。

而dip，就是把屏幕的高分成480分，宽分成320分。比如你做一条160dip的横线，无论你在320还480的模拟器上，都是一半屏的长度。

dip与px转换的方法

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.getResources().getDisplayMetrics().density;

return (int)(pxValue / scale + 0.5f);

}