那些你应该知道却不一定知道的 — View坐标分析汇总

2016-08-24 安卓应用频道

(点击上方公众号，可快速关注)

> 来源：伯乐在线专栏作者 - ImmortalZ
> 链接：http://android.jobbole.com/84285/
> 点击 → 了解如何加入专栏作者

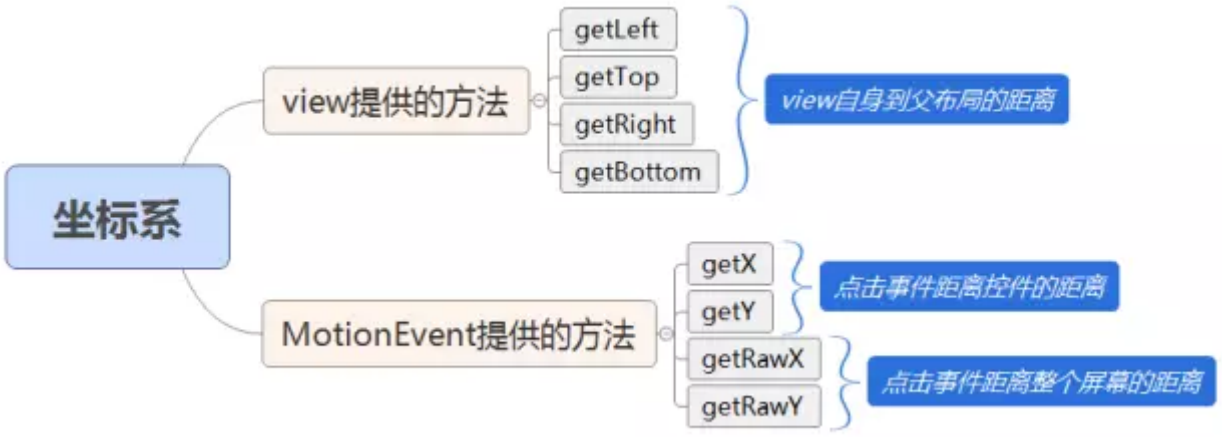
一.概述

> 网上关于Android 的view坐标挺多的，写这篇的目的是因为网上搜到的文章大多较简单，几乎都是简单的介绍下获取的几个方法坐标的几个方法罢了，但在实战中，你会发现可能你学会的那几个获取坐标的方法并没有正确的使用，导致当你要计算坐标的时候可能会试过几遍才找到正确的办法（其实这也正是我容易混淆的地方，所以特地写篇博客记录下）

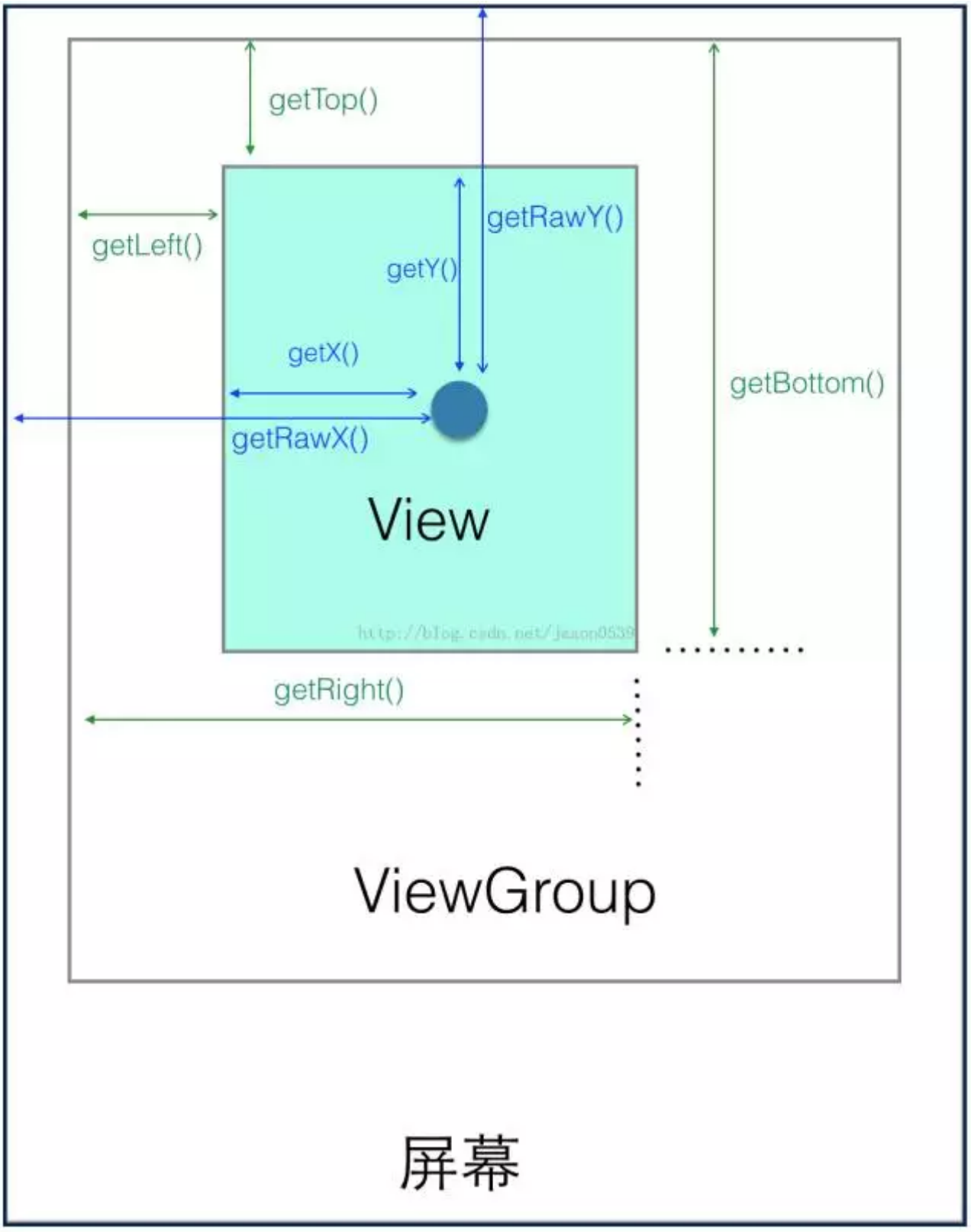
关于那几个获取坐标的方法我就懒得说了

（这篇博客有记载，大家可以去看看
http://blog.csdn.net/jason0539/article/details/42743531）

大体的方法就是这些了



下面借用那篇博客的一张图：



view提供的方法

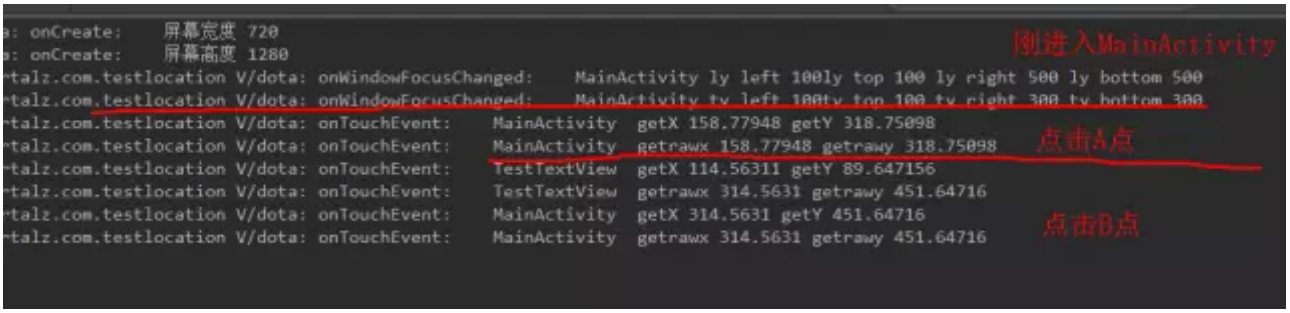
> getTop：获取到的，是view自身的顶边到其父布局顶边的距离
> getLeft：获取到的，是view自身的左边到其父布局左边的距离
> getRight：获取到的，是view自身的右边到其父布局左边的距离
> getBottom：获取到的，是view自身的底边到其父布局顶边的距离

MotionEvent提供的方法

> getX()：获取点击事件相对控件左边的x轴坐标，即点击事件距离控件左边的距离
> getY()：获取点击事件相对控件顶边的y轴坐标，即点击事件距离控件顶边的距离
> getRawX()：获取点击事件相对整个屏幕左边的x轴坐标，即点击事件距离整个屏幕左边的距离
> getRawY()：获取点击事件相对整个屏幕顶边的y轴坐标，即点击事件距离整个屏幕顶边的距离

下面做个测试

分别点击A点，B点后效果



这里需要注意的是：

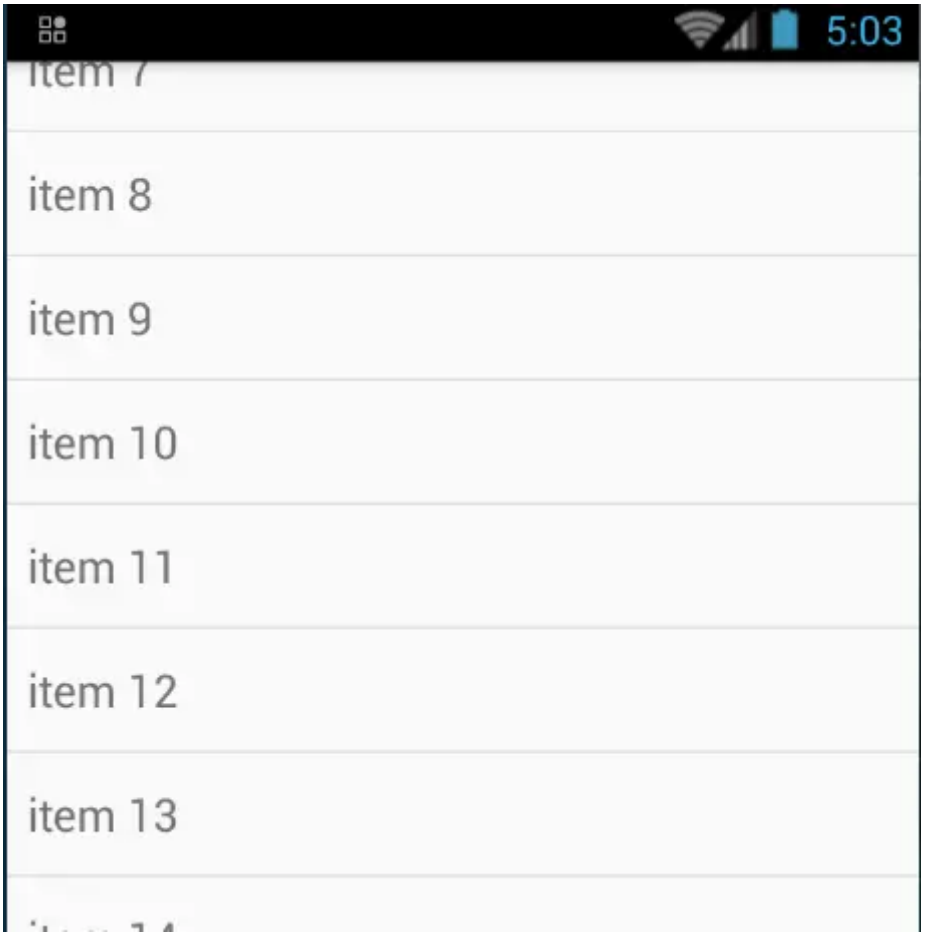
> 点击B点后（可以看到先是回调TestTextView中的onTouchEvent方法，然后才是MainActivity中的onTouchEvent，因为我在二者的onTouchEvent方法中都没有进行点击事件的消费处理，所以会往上传递，突然扯到了事件分发机制，2333~这里就是突然想补充一点，还是扯回坐标吧）
>
> 1.TestTextView中getY和getRawY取得的值不一样，这点我们可以理解

> 2.MainActivity中getY和getRawY取得的值一样！（我们注意到点击A,B点都是如此）

这里我们得到一条启发：

> getY和getRawY这一系动作都是由MotionEvent来定义产生的。是得看最后MotionEvent是被哪个View所消耗。如果MotionEvent没有被任何View所消耗，最终返回Activity则getY和getRawY则一致。如果被View所消耗，则具体情况具体分析，getY，getRawY可能一致，也可能不一致。

测试2：类似在ListView这种有滚动轴的控件中会是什么样的呢？



这个打印我忘记截图完整了，这里我说明下打印的log我是在ListView所在的activity中的dispatchTouchEvent，之所以不在activity的onTouchEvent中打印是因为ListView中的item消费了事件，那么这个activity的onTouchEvent就不会打印出Log了。

贴上代码

```
@Override
    public boolean dispatchTouchEvent(MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            LogUtil.m("SecondActivity  getX " + event.getX() + " getY " + event.getY());
```

```
            LogUtil.m("SecondActivity  getrawx " + event.getRawX() + " getrawy " + event.getRawY());
        }
        return super.dispatchTouchEvent(event);
    }
```

```
SecondActivity  getX 294.59085 getY 60.95238
SecondActivity  getrawx 294.59085 getrawy 60.95238
```

这次我们点击的是item7的顶部。
（PS：这时可能会有点好奇，我们明明点击的接近是item7的顶部，为啥得到的Y指却不是接近0呢，原因后面讲）

这里我们得到一条启发：

> 对于这种滑动的ViewGroup，我们在获取ViewGroup的坐标值时并不需要考虑它到底滑动了多少（实际滑动的我们应该看作是ViewGroup中的View在滑动）

二.获取

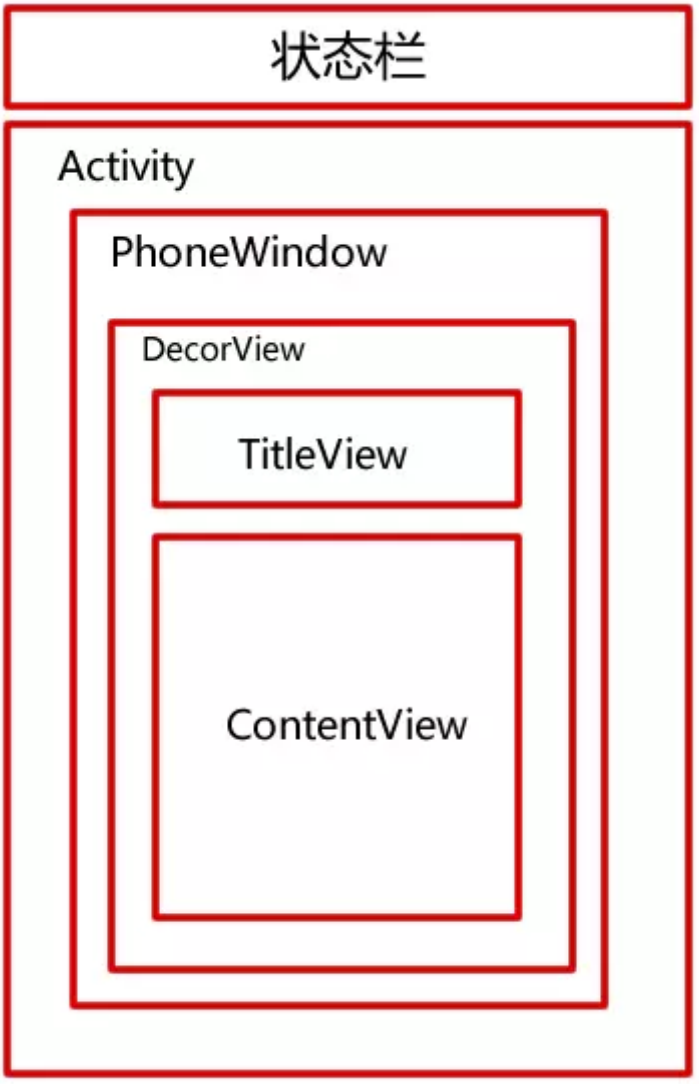
在上面我们留下了一个疑问：我们明明点击的接近是item7的顶部，得到的Y指却不是接近0。

原因在于getRawY返回的是点击事件距离整个屏幕顶边的距离，所以点击item7的顶部，得到的Y值其实就是状态栏的值。

> 当然我们有时候碰到的不仅就只有状态栏，有时候是状态栏与标题栏并存的。所以我们在获取ViewGroup的Y值是一定要注意是否需要减去状态栏，标题栏（如果有）的高度，否则计算得到的Y值并不是正确的。

这里为了让大家更清晰的了解，大家可以看看这篇文章http://bbs.51cto.com/thread-1072344-1.html（Android4.0窗口机制和创建过程分析 ）

下面我们用图来初略说明（这是我的理解，有误欢迎指正）



From:http://blog.csdn.net/mr_immortalZ

现在我们再来说说怎么取得坐标值的时机

> 因为MotionEvent提供的获取坐标的方法是在页面完完全全显示在用户眼前且用户点击后才会使用到的方法，所以并不存在获取不到的问题，下面就论述下**view提供的获取坐标方法**

看到这，你可能会说，那还不简单当布局被加载出来的时候，我们去获取不就OK了吗？

于是我们就会看到这样的错误：在一个Activity的onCreate方法中，设置完setContentView后，就开始View的getLeft，getTop等方法，结果发现为0，为啥？

原因就是：

> **对于View，ViewGroup来说，width、height、top、left等属性值是在Measure与Layout过程完成之后才开始正确赋值的，而Measure与Layout却都晚于onCreate方法执行，所以onCreate中getLeft根本就取不到值！**

那要是我们想要在onCreate中取到我们想要的值，我们应该怎么做呢？
大家可以参考这两篇博文
（http://www.cnblogs.com/kissazi2/p/4133927.html）
（http://blog.csdn.net/codezjx/article/details/45341309）

我觉得写得很好了

归纳如下：

1. 监听Draw/Layout事件：ViewTreeObserver
2. 将一个runnable添加到Layout队列中：View.post()
3. 重写View的onLayout方法
4. 重写Activity的onWindowFocusChanged方法,在该方法中获取

这里我推荐2,4这两种，即

```
view.post(new Runnable() {
    @Override
    public void run() {
        view.getHeight();
    }
});
```

或者

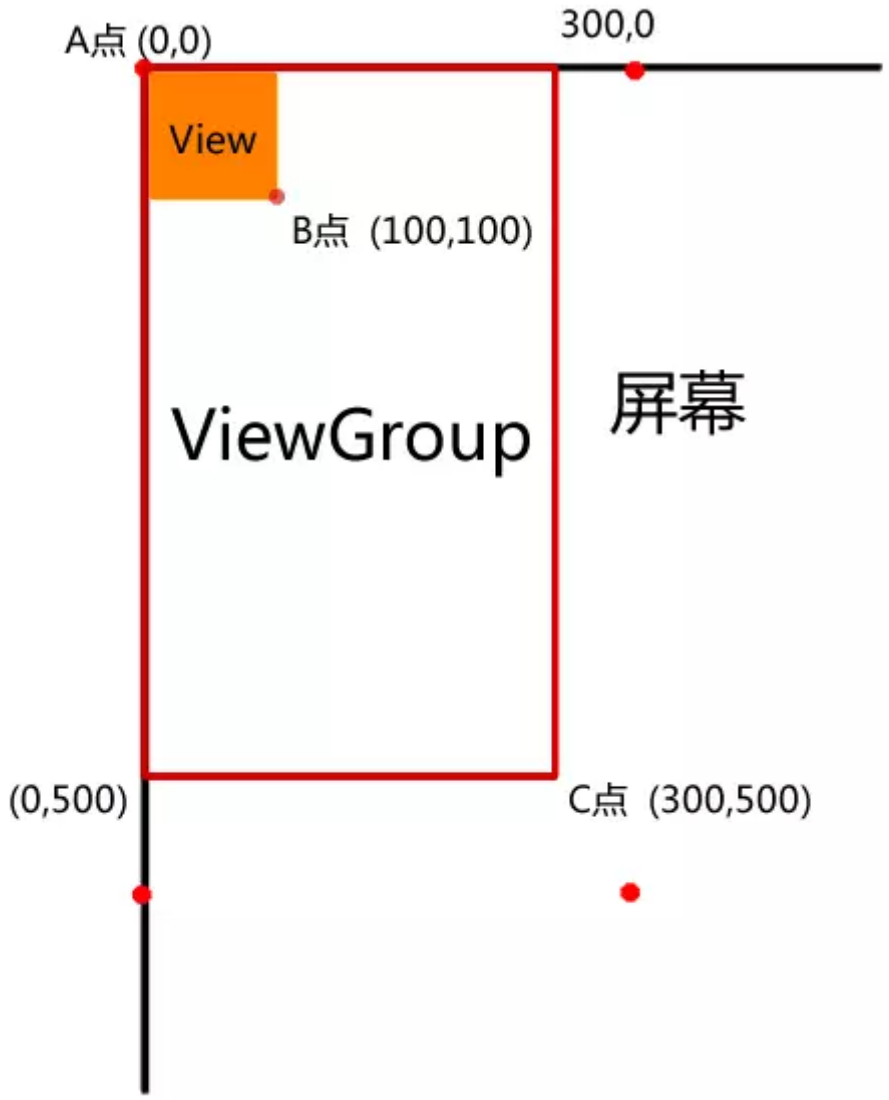
```
@Override
public void onWindowFocusChanged(boolean hasFocus) {
    super.onWindowFocusChanged(hasFocus);
    //此处可以正常获取width、height等
}
```

三.计算

> 现在对坐标系是咋样的，我们已经了解了。
>
> 对啥时候获取，以及获取后是否需要纠正得到正确的Y，我们也已经分析了。
>
> 下面就来说说本文的重头戏 ——— 坐标的计算。
>
> （之所以说是重头戏，是因为我们之前得到的都是一个点的正确坐标，现在我们需要做的是在得到多个正确坐标后，进行正确的计算，这样我们才能实现滑动这样炫酷的效果`(∩_∩)′）

首先我们需要建立一个概念

> 在Android的坐标系中，原点在屏幕左上角，向右x为正，向下y为正。



（为了好计算，图片中的坐标单位是px）

（下面就以这个为例，我们要将View从原点移动到（200,400）的位置，即B点与C点重合）

想实现View移动大致有这几种方式（代码见下面）

XML文件

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <LinearLayout
        android:id="@+id/ly"
        android:background="#EFAA88"
        android:layout_centerInParent="true"
        android:layout_width="300px"
        android:layout_height="500px">
        <mr_immortalz.com.testlocation.TestTextView
            android:background="#aabbcc"
            android:id="@+id/tv"
            android:text="你好"
            android:layout_width="100px"
            android:layout_height="100px" />
    </LinearLayout>

</LinearLayout>
```

TestTextView也很简单，就是

```
/**
 * Created by Mr_immortalZ on 2016/4/16.
 * email : mr_immortalz@qq.com
 */
public class TestTextView extends TextView {
    public TestTextView(Context context) {
        super(context);
    }

    public TestTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public TestTextView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                /*LogUtil.m("TestTextView  getX "+event.getX()+" getY "+event.getY());
                    LogUtil.m("TestTextView  getrawx "+event.getRawX()+" getrawy
"+event.getRawY());*/
                //layout(getLeft() + 200, getTop() + 400, getRight() + 200, getBottom() + 400);

                /*offsetLeftAndRight(200);
                    offsetTopAndBottom(400);*/
                /* ViewGroup.MarginLayoutParams lp = (ViewGroup.MarginLayoutParams) getLayoutParams();
                    lp.leftMargin = getLeft() + 200;
                    lp.topMargin = getTop() + 400;
                    setLayoutParams(lp);*/

                //((View)getParent()).scrollTo(-200,-400);

                //scrollTo(-50,-10);

                //scrollTo(300, 500);

                //((View)getParent()).scrollBy(-200,-400);
                /*AnimatorSet set = new AnimatorSet();
```

```
            set.playTogether(
                    ObjectAnimator.ofFloat(this, "translationX", 200),
                    ObjectAnimator.ofFloat(this,"translationY", 400)

            );
            set.start();*/
        TranslateAnimation anim = new TranslateAnimation(0, 200, 0, 400);
        anim.setFillAfter(true);
        startAnimation(anim);

        LogUtil.m("移动后 getX " + getX() + "  getY " + getY());
        LogUtil.m("移动后 getLeft " + getLeft() + "tv getTop " + getTop()
                + " tv getRight " + getRight() + " tv getBottom " + getBottom());
        break;
    }
    return true;
  }
}
```

我们移动到指定位置的有7种方式

1.layout

```
layout(getLeft() + 200, getTop() + 400, getRight() + 200, getBottom() + 400);
```

移动后getLeft等值改变

2.offsetLeftAndRight、offsetTopAndBottom

```
offsetLeftAndRight(200);
offsetTopAndBottom(400);
```

移动后getLeft等值改变

3.修改LayoutParams

```
ViewGroup.MarginLayoutParams lp = (ViewGroup.MarginLayoutParams) getLayoutParams();
        lp.leftMargin = getLeft() + 200;
        lp.topMargin = getTop() + 400;
        setLayoutParams(lp);
```

移动后getLeft等值不改变

```
onWindowFocusChanged:    移动前 getLeft 0tv getTop 0 tv getRight 100 tv getBottom 100
onTouchEvent:    移动后 getLeft 0tv getTop 0 tv getRight 100 tv getBottom 100
```

4.scrollTo

```
((View)getParent()).scrollTo(-200,-400);
```

移动后getLeft等值不改变

```
onWindowFocusChanged:    移动前 getLeft 0tv getTop 0 tv getRight 100 tv getBottom 100
onTouchEvent:    移动后 getLeft 0tv getTop 0 tv getRight 100 tv getBottom 100
```

5.scrollBy

```
((View)getParent()).scrollBy(-200,-400);
```

移动后getLeft等值不改变

```
onWindowFocusChanged:    移动前 getLeft 0tv getTop 0 tv getRight 100 tv getBottom 100
onTouchEvent:    移动后 getLeft 0tv getTop 0 tv getRight 100 tv getBottom 100
```

对于scrollTo、scrollBy需要注意的有两个问题

问题1：

> 移动计算值 = 最开始点坐标 – 最后移动到的坐标
> 原因是因为最终会调用这个方法
> —— invalidateInternal(l – scrollX, t – scrollY, r – scrollX, b – scrollY, true, false);
> 其中l,t,r,b为原来坐标点，scrollX,scrollY为目标坐标点，只有当目标坐标点值是负数时，移动到的位置才为正数！
> 例如scrollTo ，我们要从（0,0）移动到（200,400）这个点，根据上面的公式可知为负值

问题2

> 为什么需要加上 ((View)getParent())
>
> TestTextView本身是View，scrollTo、scrollBy移动的都是View的Content，如果不加的话，使用的效果则是TestTextView的文字位置变化，而TestTextView本身不会变化。
> 如果在ViewGroup中使用scrollTo、scrollBy，则移动的是ViewGroup中的View.我们这里需要让TestTextView移动，则需要先 ((View)getParent())，然后再((View)getParent()).scrollTo...

6.属性动画

我就以ObjectAnimator为例子

```
AnimatorSet set = new AnimatorSet();
        set.playTogether(
            ObjectAnimator.ofFloat(this, "translationX", 200),
            ObjectAnimator.ofFloat(this,"translationY", 400)

        );
        set.start();
```

移动后getLeft等值不改变

```
onWindowFocusChanged:    移动前 getLeft 0tv getTop 0 tv getRight 100 tv getBottom 100
onTouchEvent:    移动后 getLeft 0tv getTop 0 tv getRight 100 tv getBottom 100
```

7.位移动画

```
TranslateAnimation anim = new TranslateAnimation(0,200,0,400);
        anim.setFillAfter(true);
        startAnimation(anim);
```

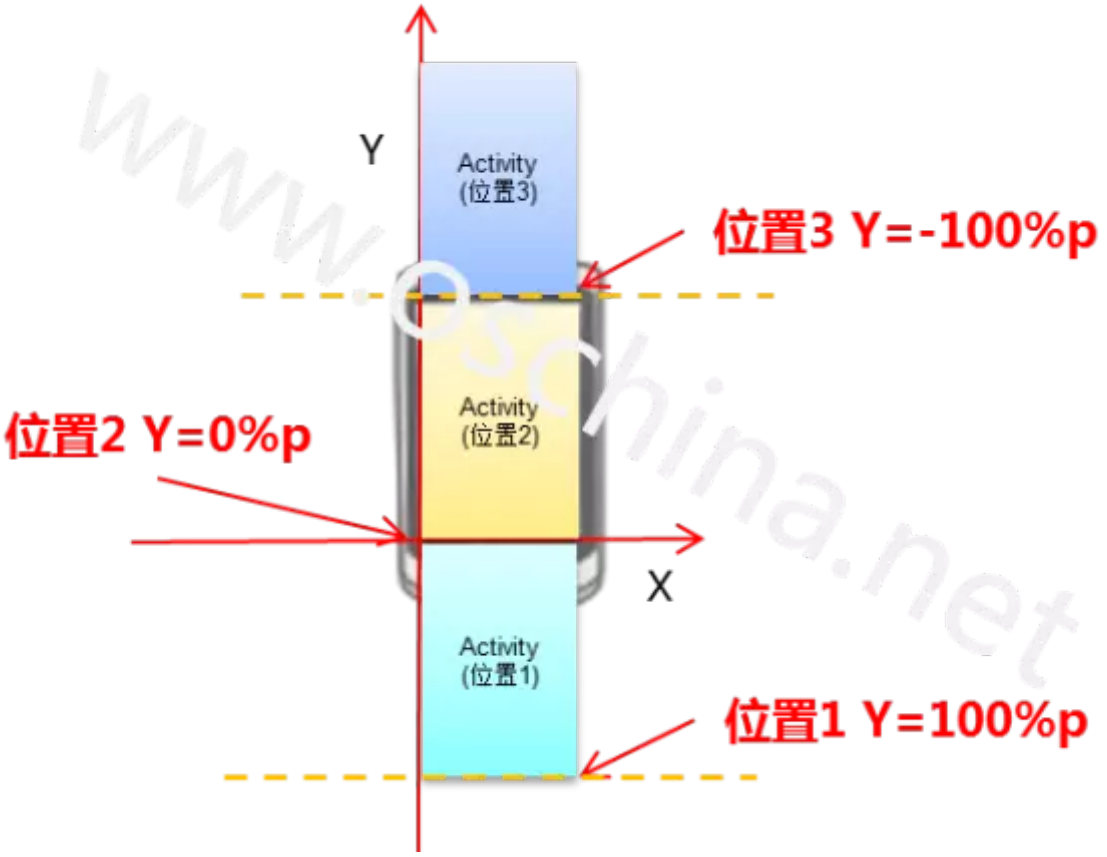
移动后getLeft等值不改变

```
onWindowFocusChanged:    移动前 getLeft 0tv getTop 0 tv getRight 100 tv getBottom 100
onTouchEvent:    移动后 getLeft 0tv getTop 0 tv getRight 100 tv getBottom 100
```

关于位移动画的补充点：

我们经常用这样的需求，要求一个popupwindow从屏幕底部弹出或者从屏幕顶部弹出。
这里的位移设置同样还是如此（原点在屏幕左上角，向右x为正，向下y为正）

这篇博文可以参考学习下，下面这张神图也是来自这篇博文（Android动画之translate(位移动画)）
http://www.cnblogs.com/bavariama/archive/2013/01/29/2881225.html



注意点

属性动画是真实改变View的位置的，虽然属性动画、位移动画的getLeft等没有改变，但是属性动画的getX、getY是改变了的，位移动画的getX、getY仍未改变！

最后再来回顾下这张图

四.小结

> 坐标分析上面都分析完了，基本上涵盖了自定义View坐标计算、滑动、事件分发等常见场景的坐标问题，希望大家能从中得到收获。
> 水平很菜，有错误的地方欢迎指正，大家一起学习进步！
>
> ☻ 反正撸完这篇，我算是对于坐标系有了更深刻的认识，给自己点个zan~❤

（上面为部分 T 恤款式）

网店地址：https://fanpinshe.taobao.com

淘口令：复制以下红色内容，然后打开手淘即可购买

范品社，使用￥极客T恤￥抢先预览（长按复制整段文案，打开手机淘宝即可进入活动内容）

阅读原文