# Android笔记（中篇）

[Android笔记（中篇） 1](#_Toc460599776)

[1 适配 2](#_Toc460599777)

[2 控件 3](#_Toc460599778)

[2.1 属性说明 3](#_Toc460599779)

[2.2 系统默认控件 4](#_Toc460599780)

[2.2.1 ImageView 4](#_Toc460599781)

[2.2.2 AdapterView 4](#_Toc460599782)

[2.2.3 视频播放 5](#_Toc460599783)

[2.2.4 Menu 5](#_Toc460599784)

[2.2.5 WebView 5](#_Toc460599785)

[2.2.6 Dialog 5](#_Toc460599786)

[2.3 自定义控件 6](#_Toc460599787)

[2.3.1 基本概念 6](#_Toc460599788)

[2.3.2 控件自定义的方式 6](#_Toc460599789)

[2.3.3 自定义控件的实现过程 7](#_Toc460599790)

[2.3.4 自定义控件的属性 7](#_Toc460599791)

[3 其他 9](#_Toc460599792)

[3.1 PopupWindow 9](#_Toc460599793)

[4 效果实现 10](#_Toc460599794)

[4.1 侧拉菜单效果 10](#_Toc460599795)

[4.2 10](#_Toc460599796)

[5 事件 11](#_Toc460599797)

[6 资源 12](#_Toc460599798)

[6.1 资源的使用规则 12](#_Toc460599799)

[6.2 系统资源 12](#_Toc460599800)

[6.3 attrs.xml文件 12](#_Toc460599801)

[6.4 Style.xml文件 13](#_Toc460599802)

[6.5 AttributeSet 14](#_Toc460599803)

[7 动画 15](#_Toc460599804)

[7.1 图片操作 15](#_Toc460599805)

[7.2 动画 15](#_Toc460599806)

[7.2.1 基础 15](#_Toc460599807)

[7.2.2 View Animation 补间动画 16](#_Toc460599808)

[7.2.3 Drawable Animation 帧动画 16](#_Toc460599809)

[7.2.4 ObjectAnimator属性动画 16](#_Toc460599810)

[7.2.5 疑问 16](#_Toc460599811)

1. 适配
2. Android selects the version of each resource that best matches the current device, by comparing the device configuration information with the qualifiers in the resource directory name.

比较device configuration和res的qualifier，实现适配。

1. the device's smallestWidth does not change when the screen's orientation changes.
2. API8之后，提供了night和notnight的qualifier，可以通过把资源文件放入相应的文件夹，为软件提供自动根据昼夜变化更新软件视图的功能。
3. Providing default resources is important not only because your application might run on a configuration you had not anticipated, but also because new versions of Android sometimes add configuration qualifiers that older versions do not support.

比如night模式，可以将资源文件分成night和notnight两种，但是如果用户的api是8之前，程序就会报错，所以default资源必须提供。

1. When you have a resource that you'd like to use for more than one device configuration (but do not want to provide as a default resource), you do not need to put the same resource in more than one alternative resource directory. Instead, you can (in some cases) create an alternative resource that acts as an alias for a resource saved in your default resource directory.

安卓使用别名的方式可以将default资源文件夹中的资源提供给不同的qualifier资源。相应qualifier文件夹中需要定义对应名称的xml文件。具

体示例见API DEVELOP ->GUIDE ->APP RESOURCE ->PROVIDING RESOURCES->CREATING ALIAS RESOURCES

1. 通过在values, values-v4等文件夹中的style.xml文件中定义同一个自定义主题，但是parent选择符合相应API等级的theme，可以实现当用户系统API等级不同时，使用不同主题的功能。
2. Gravity属性的start，end是根据书写习惯确定位置，左侧开始还是右侧开始，这样当国家改变时，自动改变。
3. listView条栏目的图片尽量使用dp，不用包裹内容（因为包裹内容的话，就是使用图片的大小，而图片的大小就是px；但是如果父容器对view的大小有限制的话，应该也不是，图片会自己缩放），因为图片的px是确定的，所以屏幕分辨率不同时，图片显示的大小基本是不变的。而dp的话，图片会根据屏幕分辨率调整大小。
4. 控件
   1. 方法
5. offsetLeftAndRight()：在刷新不可见的控件的时候，未必会实时invalidate()
   1. 属性说明
6. Weight属性实在onLayout的时候处理还是onMeasure的时候处理？
7. 设置View.GONE属性的控件，当运行时由GONE—>VISIBLE，控件在显示到屏幕的同时，会将原有位置的控件挤走。前提是这些控件在布局时是互相不覆盖的。
8. Visible属性可以设置为invisible和gone，invisible不可见但占位，gone不可见不占位
9. TextSize参数

这个参数优先级可能比较高，padding参数的改变并不会迫使TextSize改变。

1. Padding参数

下面所有的分析，都是在布局文件里进行设置。如果在代码中直接setPadding()，还是可以在各个方向使用负数。

View的内容的尺寸是不会因为padding参数的变化而变化的，即内容的尺寸不会被压缩，也不会被放大。

padding参数如果把View的内容挤没了，View在显示的时候还是会显示原来的大小，因为padding是填补空白，空白也是内容。但是如果padding是负数，那可能控件（wrap\_content）就会不占空间（ListView的item还是会占用空间，header和item是不同的）。

padding参数可以为负数，但是有没有效果与View本身layout\_width和layout\_height的类型有关。

View的padding参数的实际效果，与View本身layout\_width和layout\_height的类型有关。分析如下（均以layout\_width进行说明，layout\_height相同）：

* 1. layout\_width为wrap\_content
     1. padding参数为负数

这种情况下，padding参数top, bottom为负数是无效的；但是left和right是有效的，并且会使内容显示不完全，即隐藏内容。

* + 1. padding参数为正数
       1. padding和内容加一起之后，大于屏幕宽度

会将内容挤出屏幕。

* + - 1. padding和内容加一起之后，小于屏幕宽度

会扩大View的宽高，以使View可以将内容和padding参数占用的空间同时显示出来。

* 1. layout\_width为精确值（××dp, match\_parent）
     1. padding参数为正数
        1. left, top

会将内容整体往对面推动。

* + - 1. Right, bottom

保持View内容的左侧位置不动，然后向左或向上推动内容，但有严格的限制，即内容变形时，内容的边界不能超出View的边界，如果内容在top, bottom, right三个方向上均已经被挤压到View的边界，那就表示已经挤不动了，那么系统就会去覆盖内容，被覆盖的内容区域为padding所占空间与内容所占空间重叠的部分。

* + 1. padding参数为负数

只有left起作用，会将内容往左侧拉拽。其他三个方向的参数设置成负数是无效的。

1. dip, dpi, px, density, sp
   1. dip和px都是像素的意思，只不过量度不同而已

dip（即dp）：density independent pixel，密度无关像素。1 dip=每英寸160像素的屏幕上的一个点。

px：像素

* 1. dp和sp

sp分为几个等级：小、正常、大、超大，允许用户自定义文字尺寸。

在屏幕密度不变的情况下，单个dp的值就确定并且改变不了了，因为dp=density\*dp(每英寸160像素的屏幕上的一个点的大小)。所以屏幕密度确定了，控件的尺寸也就定死了。当用户在当前屏幕密度不变的情况下需要将字体变大或缩小时，如果字体用dp设置尺寸，那么字体就没有办法调整了，因为1dp的大小已经定死了。而如果用sp作为单位，那么即使屏幕密度确定，单位sp的尺寸仍然可以根据用户自定义的文字尺寸进行调整，从而达到控件大小保持不变，字体大小进行调整的效果。

* 1. dpi：dots per inch，每英寸的点数，即每英寸像素数

density：数值为DPI / (160像素/英寸)，即相对于标准屏幕的比例。

* 1. 系统默认控件
     1. ViewPager

1. ViewPager对事件的处理方式：
2. onInterceptTouchEvent():

只根据水平滑动速度判断是否拦截事件，也就是说正常慢速滑动，不会被拦截。

1. onTouchEvent()
   * 1. ImageView
2. adjustViewBounds：ImageView是否根据src属性的图片自动缩放自身的尺寸。
   * 1. AdapterView
3. AdapterView继承的是ViewGroup，所以onItemClick其实就是对ViewGroup中的各个节点设置点击事件，所以如果加了header和footer，onItemClick(position)中的position与adapter中数据position并不是对应的。
4. 典型的MVC模式

View→AdapterView

Model→cursor游标，list集合

Controller→Adapter

1. 源码分析
   1. View hierarchy中的View是存在list集合中的。
   2. 视图回收器类是在absListView中定义的。
   3. ListView中展示多种类型的View时，adapter中的getViewTypeCount()方法返回值后，这个值会被传给RecycleBin（视图回收器）中，试图回收器会调用自己的setViewTypeCount()方法，创建一个数组（数组的元素是集合），来保存废弃的view。所以在adapter中的getItemVIewType()方法中返回的表示view类型的整型数值必须是0~（viewTypeCount-1），因为存储废弃view的数组的下标就是这个范围。
   4. getFirstVisiblePosition() 方法：在第一个item顶端显示出来的时候才会返回索引0.
2. Listview
   1. ListView的header和footer是跟着条目移动的，只不过是可以隐藏而已。
   2. header和footer本来的目的可能只不过是在listView的上下两端添加一个条目而已，通过header和footer添加的条目只会被添加到列表的顶端和底端，这样就不需要我们人为的对数据集合进行操作。因为每次数据集合添加新数据的时候，我们都需要将新数据插入到最后一个元素的前方，从而保证footer的数据不会被改变，google应该是为了简化这种代码的编写，直接给我们提供了一个header和一个footer。
   3. header和footer可以添加多个，并依次排列显示。
   4. header和footer需要的布局参数是AbsListView.LayoutParams，addHeaderView时增加的View采用ViewGroup比较合适，因为直接加入到header和footer的View调整布局参数没有效果（原因待查），所以需要先加一个ViewGroup，然后在ViewGroup中对View进行相应的调整。
      1. 视频播放
3. SurfaceView
   1. 该控件是重量级控件，在创建时底层启动了两个子线程。
4. VideoView是谷歌自己继承SurfaceView实现视频播放的类。只可以播放MP4和3gp
5. 开源项目vitamio，可以播放大部分视频格式。
   * 1. Menu
6. Menu内选项的点击事件，注意，不是在activity中注册onClickListener，而是重写Activity里的onCreateOptionsMenu（）和onMenuOpened（）两个方法。这里说明设置点击事件不一定非要自己注册事件监听器。也可以采用回调方法的形式。
   * 1. WebView
7. WebSetting的有些设置不支持wap网页，html可以。
8. javaScript支持需要设置。（通过WebSetting）
9. 页面缩放和双击缩放功能也需要设置。
   * 1. Dialog
10. Dialog的按钮设置监听事件：
    1. 直接在Dialog上增加按钮，代码如下：

|  |
| --- |
| import android.content.DialogInterface.OnClickListener;  builder.setPositiveButton("确定", **new** OnClickListener() {  @Override  **public** **void** onClick(DialogInterface dialog, **int** which) {  System.out.println(**this**);  //下载对话框  showDownloadDialog(updateinfo);  }  }); |

这种方式是在setPositiveButton()方法中直接传入事件监听器，这时的事件监听器就要是DialogInterface.OnClickListener。

* 1. 通过setView()增加按钮，代码如下：

|  |
| --- |
| import **import** android.view.View;  Button cancelButton =  (Button)findViewById(R.id.*home\_activity\_sign\_up\_dialog\_et\_cancel\_btn*);  cancelButton.setOnClickListener(**new** OnClickListener() {  @Override  **public** **void** onClick(View v) {    }  }; |

这种方式中setOnClickListener()方法是View类下的，所以onClickListener也要是View类下的。

* 1. 第一种方式其实也可以找到button，然后用View类下的onClickListener，就是代码多余了，因为明显setPositiveButton()的时候直接可以设置监听事件。代码如下：

|  |
| --- |
| import android.view.View;  builder.setPositiveButton("确定", **null**);  AlertDialog dialog2 = builder.show();  Button button = dialog2.getButton(DialogInterface.BUTTON\_POSITIVE);  button.setOnClickListener(**new** OnClickListener() {  @Override  **public** **void** onClick(View v) {  // **TODO** Auto-generated method stub  }  }); |

dialog与Activity是会互相抢界面的可交互性的，那监听事件到底是怎么分配的？包括fragment在内

* 1. 自定义控件
     1. 基本概念

1. 构造函数：defStyle赋值为0，系统会默认调用内部的控件样式
2. onDraw方法提供的canvas是控件的left、right、top、bottom包围的范围，不是整个屏幕。
   * 1. 控件自定义的方式
3. 继承View类，重写onDraw()方法
4. 继承具体的控件类，重写其中的某些方法，并将一些固有属性在构造中进行设置。示例见跑马灯效果的TextView。
5. 半xml、半Java代码：



如上图所示的条目，在setting视图界面上实现时可以采用三种方式：

* + 1. Xml文件直接生成

activity\_setting.xml布局文件中重复下面的代码，需要多少个条目就重复多少遍。

优点：在第一次实现界面的时候，可能是三种方式中最速度的方式。

缺点：TextView或ImageView任何一个需要修改某些属性，工作量可想而知。

|  |
| --- |
| <TextView  android:id=*"@+id/setting\_item\_tv\_name"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_centerVertical=*"true"*  android:text=*"自动更新xxx"*  android:textSize=*"22sp"*  android:textStyle=*"bold"* />  <ImageView  android:id=*"@+id/setting\_item\_iv\_toggle"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_alignParentRight=*"true"*  android:layout\_centerVertical=*"true"*  android:src=*"@drawable/settting\_on"* /> |

* + 1. 使用include标签

将上述xml代码封装成一个layout（setting\_item\_layout），然后使用include引入该layout，进而生成setting界面。

优点：相比前一种，当需要修改属性时，工作量大幅减少。

缺点：采用include的方式，只能在include标签下设置属性，这些属性只可以控制setting\_item\_layout整体的效果。相对于style这种局部抽取的方式，include不但无法覆盖setting\_item\_layout中各组件的属性，更不能设置setting\_item\_layout里面各个组件的属性。

* + 1. 自定义控件

由于上面两种方式存在诸多问题，所以才有了这种半xml、半Java代码的实现形式。这种方式其实是深化了include，解决了include不能灵活修改内部组件属性的问题。

这种方式最初也是加载整个setting\_item\_layout布局。与include的区别在于获得布局后，通过java代码拿到了布局里具体的组件，从而实现对各组件属性的修改。到此，其实已经可以满足正常使用的要求。

但既然将setting\_item\_layout整体当做一个控件，当然有一些方法和变量可以操作和配置布局里的组件。但这些方法和变量不可能设置为各个组件的属性，各组件的属性系统已经定义好了，肯定不可能修改系统定义。所以为了将这些方法和变量封装为整体的属性，就需要实现自定义属性。

* + 1. 自定义控件的实现过程

1. 半xml、半Java代码：
   * 1. 继承布局类，由java代码构造控件的界面
     2. 在attrs.xml文件中定义控件的自定义属性
     3. 自定义控件的属性
2. 自定义控件的属性可以通过两种方式进行设置
   1. Attrs.xml中自定义属性，在layout.xml文件中相应View的属性中进行定义

这种方式需要在View的构造方法中手动提取attributeSet中的属性值，并赋值给View中相应的成员变量或调用属性设置方法（描述属性的方法或成员变量）

实现步骤：

1. Attrs中定义自定义属性
2. Layout.xml中配置属性
3. View的构造方法中编写提取设置属性的代码
   1. Attrs.xml中自定义主题，主题中包含一个自定义属性。然后style.xml中实现attrs中自定义的主题，并将相应属性进行实现。

这种方式因为不需要在layout.xml中配置属性，也不需要在attrs定义数个自定义属性，并且View的构造方法中基本不用写什么代码，所以相对来说涉及的文件比较少，感觉适合开源框架使用。

实现步骤：

1. Attrs中定义自定义主题并在该主题下定义一个属性
2. Style中实现attrs中定义的主题的自定义属性
3. 其他
   1. PopupWindow
4. 不能在Activity的onCreate方法中show，会报token null is not valid。
5. 效果实现
   1. 侧拉菜单效果
6. 使用开源项目SlidingMenu：

继承SlidingMenu提供的Activity，代码实现。示例，zhbj。

1. 使用系统自带的DrawerLayout：

DrawerLayout继承的ViewGroup，直接在需要实现侧拉菜单效果的Activity中加载以DrawerLayout作为根元素的布局文件即可。

示例，googlePlay。



1. 事件
2. 监听器listener是对用户操作的监听，这个操作作用于控件本身。对监听器接口的定义，命名的时候应该以动作本身命名，不应该描述功能，因为这个动作对应于什么功能应该是由控件的使用者决定的，控件本身只对动作提供功能接口。
3. MotionEvent参数，getX()返回的坐标是相对于控件左上角的。
4. 注册事件也可以采取重写相关回调方法的形式。
5. 事件被细分到了具体的每个动作，一般从event对象中可以获取。
6. 事件注册的位置分析：

Activity中的所有View控件有完整的View hierarchy层级结构。

* 1. Activity中

Activity中注册的事件，只可以作用到Activity的根布局的下一层级的控件。比如对listView注册onClickListener，Activity只可以响应点击整个listView的事件，而不区分点击的是哪个item（这里有个特例，如果注册onItenClickListener则可以响应item层级的事件，但是item里的各控件是无法区分的）。所以fragment的内各控件的事件监听器应该是在fragment中注册的。

* 1. fragment中

在自己内部注册事件监听器。

* 1. listView中

分两种，可以注册针对自身listView控件的事件监听器，也可以注册针对item的事件监听器。但是前者可能没有什么意义。

注意，对于item中的各控件，只在Activity中注册onItemClickListener监听器的话，事件是不会区分item内部的控件的，只把item当成一个整体。京东商城的商品列表就为这种情况，每个商品是一个item，item包含各种信息（价格、销量、优惠），点击事件只能够进入商品的详细介绍页面，而没办法区分显示各种信息的item内部控件。

1. 资源
   1. 资源的使用规则
2. Asset和res的选择

While uncommon, you might need access your original files and directories. If you do, then saving your files in res/ won't work for you, because the only way to read a resource from res/ is with the resource ID. Instead, you can save your resources in the assets/ directory.

Files saved in the assets/ directory are not given a resource ID, so you can't reference them through the R class or from XML resources. Instead, you can query files in the assets/ directory like a normal file system and read raw data using AssetManager.

However, if all you require is the ability to read raw data (such as a video or audio file), then save the file in the res/raw/ directory and read a stream of bytes using openRawResource().

1. 主题的设置

在Activity和Application设置的主题，主题的参数才会被应用到所有的View。如果自定义控件需要从obtainStyledAttributes()方法中获取属性参数，那么包含属性参数的主题样式必须设置给Activity或Application。

但是这里要注意下面两个问题：

1. 布局文件编辑窗口设置的主题样式的作用？

布局文件的主题样式在预览布局文件的时候会被调用。因为系统在预览布局文件的时候并不知道该布局文件会被放到哪个Activity中，所以此时Activity设置成什么主题，并不会影响布局文件的预览，系统只能通过布局文件编辑窗口自身配置的主题样式进行预览。但是这个样式仅仅是在预览时发挥作用，一旦程序运行，布局文件窗口设置的这个样式就会失效。

1. 自定义控件的属性需要从当前主题中获取

这种情况，Activity和布局文件编辑窗口都需要设置成所需的主题。因为实际运行的时候属性是从Activity的主题中获取，而预览时是从布局文件窗口获取。当然，布局文件窗口的主题设置不正确，只会造成不能正确预览布局效果，并不会真正影响程序的运行。

* 1. 系统资源

1. 系统提供的style和theme
   1. 官方文档Develop ->Styles and Themes中有详细的定义
      1. Android Styles (styles.xml)
      2. Android Themes (themes.xml)
   2. attrs.xml文件

仅用来负责声明属性及其定义方式。属性并不区分是布局文件的属性还是主题的属性，布局文件和主题其实在作用到一个View时，作用是相同的，只不过主题里包含了不止一个View的属性，而布局文件里的属性一般来说都是针对具体View的。

|  |
| --- |
| <declare-styleable name="ViewPagerIndicator">  <attr name="vpiTabPageIndicatorStyle" format="reference"/> |

只有在上面的attrs.xml中声明了属性，才能在下面的style.xml文件中对属性进行定义。attrs.xml中声明属性的目的，其实就是声明name，如果attrs.xml中未声明该属性，name中是不能写这个属性的。

|  |
| --- |
| <item name=”attr”>\*\*\*</item> |

所以说attrs.xml只是用来声明属性，至于属性在哪里被使用，都是由style.xml文件决定的。

attrs.xml里定义的属性，一个属性就是一个单独的个体，就仅仅代表一个单一的属性。而style.xml中的一个style，可以包含多个属性。两者一个是单独个体，一个是属性的集合。

TabPagerIndicator开源框架的逻辑是：

1. 首先在attrs.xml中声明了vpiTabPageIndicatorStyle属性。
2. 然后在style.xml中定义了vpiTabPageIndicatorStyle属性的具体参数（一个style引用：style下包括了几个属性）。

前两步有一点需要注意：vpiTabPageIndicatorStyle虽然是单个属性，但是在2）中赋给这个属性的值是一个style引用，这个引用指向了几个属性的组合，也就是说一个属性可以通过指向一个style，一次性获得多个属性。

1. 最后在代码中调用obtainStyledAttributes()方法，从主题中获取vpiTabPageIndicatorStyle属性的参数。
2. 之所以这个属性被定义到了主题中，主要是由于代码获取属性参数用的是obtainStyledAttributes()方法，这个方法表示从当前主题中获取属性，所以才需要我们在使用的时候将该属性定义到主题中。如果代码中获取该属性参数的方式是通过attributeSet获得，那属性就可以直接定义到布局文件中，而不需要特意定义到主题中。
3. 属性可以在style.xml和layout.xml两个文件中的任意一个中进行定义，这其实都没有什么关系，关键是要看代码需要从什么位置获取属性参数，代码决定了我们定义属性的位置，而不是反过来。
   1. Style.xml文件
4. 基本概念
   1. Theme就是一个大而全的style。
   2. style是attr的上一级，一个style包含多个attribute。
   3. 同一个style可以用于单个的View，也可以用于Activity or Application。Style之间并没有差别。
   4. 在定义Activity和Application的主题，以及View的样式的时候，可以将任何系统能够是别的属性赋予三者，即使有些属性是三者并不能够处理的。这样之所以不会报错，是因为三者会在加载属性的时候忽略掉自己不能够识别的。因为加载属性的时候，代码是用switch语句编写的，所以case里没有相应的项，自然就不会接收不存在的属性。

Keep in mind that all View objects don't accept all the same style attributes, so you should normally refer to the specific View class for supported style properties. However, if you apply a style to a View that does not support all of the style properties, the View will apply only those properties that are supported and simply ignore the others.

* 1. 布局文件layout.xml的Theme在哪里起作用，如果他并不是一个Activity的根布局？会不会被所在的Activity的主题覆盖，还是它去覆盖Activity的？

When you apply a style to a single View in the layout, the properties defined by the style are applied only to that View. If a style is applied to a ViewGroup, the child View elements will not inherit the style properties—only the element to which you directly apply the style will apply its properties. However, you can apply a style so that it applies to all View elements—by applying the style as a theme.

Style的作用域只局限于当前的View或ViewGroup自己，如果需要style作用到所有View，需要将style定义为theme。

To apply a style definition as a theme, you must apply the style to an Activity or application in the Android manifest.

一般来说，将style定义为theme，都需要继承系统的theme，不然的话需要重新定义所有theme需要的属性。但是，即使style通过继承系统的theme实现自定义的theme，根据上面这段的描述，也需要被定义到manifest中Activity或Application的theme属性中，才能够被系统识别为theme。

Theme类的内部类Theme中的obtainStyledAttributes()方法，各个控件在构造中进行调用，目的就是获取当前主题中关于自己的属性，当前主题是需要在Activity或Application中定义的。布局文件layout.xml中定义的theme，这个方法有可能不会把它识别为当前的主题。

1. Style的继承
   1. 继承系统自带的style

继承系统的style，必须使用parent属性

|  |
| --- |
| <style name="GreenText" parent="@android:style/TextAppearance">  <item name="android:textColor">#00FF00</item>  </style> |

* 1. 继承自己的style

继承自己的style，一般不使用parent属性。比如自己定义了一个CodeFont属性，直接创建一个style，然后name属性定义为CodeFont.Red，就表示继承了CodeFont。

|  |
| --- |
| <style name="CodeFont.Red"> <item name="android:textColor">#FF0000</item>  </style> |

* 1. AttributeSet

Attributeset提取数据的问题分析

1. 动画
   1. 图片操作
2. 基础知识
   1. BMP格式图片的大小 = 图片像素 \* 每个像素的大小

所以图片的像素数确定以后，不论图片的内容是空白还是复杂的图像，大小是一样的。因为描述每一像素的bit数是完全相同的。

显示器的每个像素的显像元件由三原色控制，所以可以随意描述每个像素的颜色。

* 1. jpg,bmp,png的图片格式，在android下加载的时候，会忽略原格式，自动按照android下的图片格式ARGB重新计算图片的大小。android系统识别不同的图片格式，但是计算大小均采用ARGB形式计算。
  2. 图片的像素如果和当前手机屏幕的像素相同时，图片的大小会远小于VM Heap的大小。比如320\*480的屏幕，改大小的图片才不到0.6M。

1. 代码实现
   1. BitmapFactory从硬盘读取图片的时候，就已经将图片赋值给相应的变量并加载到内存。
   2. 图片动态缩放：
      1. 假设图片宽为500，屏幕宽为150
      2. 在计算缩放比的时候scale = 500/150 = 3，实际是5.3。
      3. 所以按照3缩放的时候，实际图片还是略大于ImageView。但是因为ImageView本身具有动态缩放图片的能力，所以小于1的部分由ImageView消化了。
   3. 原图不可以被修改
   4. Bitmap旋转时，画布
   5. onTouchListener(MotionEvent event)中，event返回的坐标是触摸点相对于ImageView的坐标，而拿这个坐标在bitmap上绘制图形的时候，坐标是相对于bitmap图像边界的。所以如果想让触摸的坐标和bitmap绘图的坐标完全重叠，就要换算坐标。最简单的方法就是bitmap正好完全填充ImageView。
   6. 利用画布canvas的matrix旋转、镜像等操作的时候，最初由bitmap绘制的画布边界不会被改变，所以当图像旋转的时候，会旋转出画布，从而图像会有一部分显示不出来。
   7. Matrix在操作的时候，set（prefix）的函数表示清空前一次操作，完全重绘；post表示在上次的基础上进行修改。
   8. 动画
      1. 基础
2. 与Matrix实现动画效果的区别：

Matrix是对控件中的图像进行操作，而非控件。所以自定义动画的时候也应该是针对控件进行操作，这样效果会更好。

1. 对布局使用动画，布局内的控件也会跟着展示动画效果。这些控件展示动画的坐标，系统自己计算。
2. anim和drawable文件夹中都可以放动画xml，但是这两个xml可以定义的属性并不相同。两个文件夹中的动画不是一回事儿，anim是动画，drawable是动画Animation drawable。例，ProgressBar背景动画重新定义的时候，需要在drawable文件夹中定义Animation drawable，而不是anim里定义动画，这说明ProgressBar的进度动画是帧动画。

这是因为补间动画放在anim，属性动画放在animator，帧动画放在drawable。

* + 1. View Animation 补间动画

1. 控件真实坐标并没有被改变，动画展示的仅仅是一个效果。
   * 1. Drawable Animation 帧动画
2. 可以通过XML和代码两种方式实现，XML定义animation\_list，然后通过getDrawable获取AnimationDrawable，然后作为背景给到一个控件，这时，AnimationDrawable的start和stop方法就可以开启和关闭帧动画。底层通过Handler定时更新控件背景实现。
   * 1. ObjectAnimator属性动画
3. 属性动画有ObjectAnimator和ValueAnimator两种（AnimatorSet暂时不算），如果一个控件实现动画效果的属性本来就有set和get方法，那么直接使用ObjectAnimator即可。只有当需要实现动画效果的属性系统默认没有提供set和get方法时，才需要用ValueAnimator加监听器的方式来间接实现。
4. ValueAnimator本身并没有对任何控件进行操作，它只不过是根据要求，产生一系列的值。所以需要监听动画，然后获取实时变动的值，然后拿这个值对控件执行自定义操作，从而实现动画效果。
5. 控件真实坐标被改变。
6. 使对象动画化，所以说不一定是ImageView，其他控件应该也可以。但是有个问题，就是控件外观的实际性改变会影响临近的控件，即整个窗体的布局。
7. 属性动画实际上是通过设置控件的各个属性来改变控件的状态来实现动画效果。所以其实可以通过设置属性实现动画。
   * 1. 疑问
8. 执行setAnimation()的是控件，也就是说对控件实施动画。而不是仅仅是控件中的图像。动画是不是在子线程中运行？动画会不会阻塞主线程？动画会不会暂时剥夺Activity的可交互性？
9. 如果在控件执行setAnimation()后，再重新将另外一张图片加载给控件，这个动画显示的是之前的图像，还是另外加载的这张？
10. 动画演示的过程，比如缩放，会不会影响（占用，能不能再说）其他控件的控件？

补间动画和帧动画不会，属性动画由于改变了真实控件的外观参数，可能会影响其他控件。