

# **Android UI Programming 1 Basic**

**Prepared by: Vivian Sun**

Name	Version	Date	Prepared By
Android UI Programming	1.0	2013/4/3	Vivian
Android UI Programming 1 Basic	2.0	2016/10/26	Vivian

## Content

<b>CONTENT .....</b>	<b>3</b>
<b>ABSTRACT .....</b>	<b>4</b>
<b>UI RELATED CONTENTS IN PROJECT .....</b>	<b>7</b>
RES .....	7
ASSETS 目录 .....	8
<b>ANDROID VIEW .....</b>	<b>9</b>
<b>ANDROID LAYOUT.....</b>	<b>12</b>
LinearLayout 线性布局.....	12
RelativeLayout 相对布局.....	13
TableLayout 表格布局.....	15
FrameLayout 框架布局.....	16
AbsoluteLayout 绝对布局 （已废弃） .....	16
Android SDK 工具布局优化工具.....	16
<b>CUSTOMIZED CONTROLS .....</b>	<b>19</b>
为什么需要自定义VIEW .....	19
应用场景.....	19
系统学习ANDROID自定义控件 .....	19
自定义属性.....	20
attrs.xml 中定义key 属性.....	20
XML 中引用自定义控件.....	21
创建UserRadioButton类继承RadioButton .....	22
Code 中修改自定义属性.....	23
优质开源库.....	23
<b>VIEW 事件传递.....</b>	<b>24</b>
基础知识.....	24
传递流程.....	24
View 不处理事件流程图.....	25
View 处理事件流程图.....	26
<b>REFERENCE.....</b>	<b>27</b>

## Abstract

This document shows basic technology of android view. UI Design的概念

UI即User Interface（用户界面）的简称。UI设计则是指对软件的人机交互、操作逻辑、界面美观等各个方面的整体设计。





分享的主题：Android UI Programming相关的内容。

Agenda

Android View

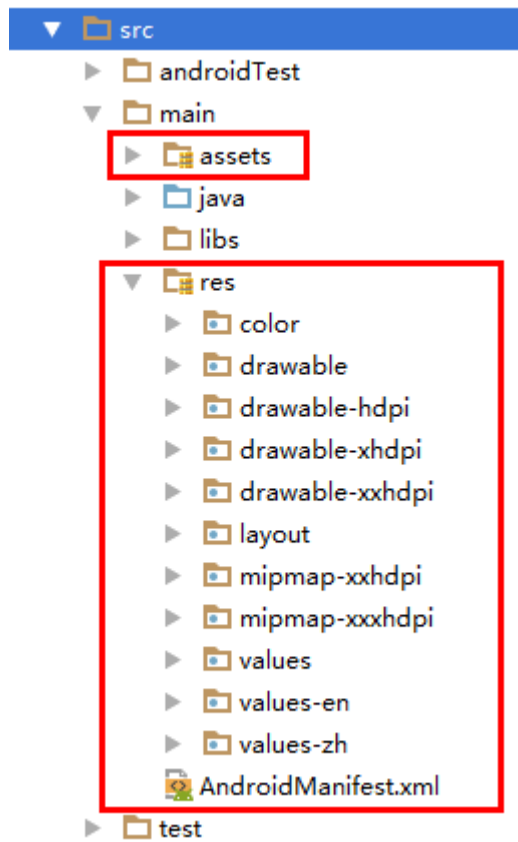
Android Layout

Android Customized View

Android View Event

Android User Interface

## UI related Contents in Project



### res

存放程序所需要的资源文件（可编译的资源文件），常见的目录有：

Path	Description	Type
res/animator/	定义动画属性	XML 文件
res/anim/	被编译进逐帧动画（frame by frame animation）或补间动画 (tweened animation)对象	XML 文件
res/color/	定义颜色状态的列表	XML 文件
res/layout/	存放被编译为屏幕布局(或屏幕的一部分)	XML 文件

<b>res/menu/</b>	用来定义应用的菜单	XML 文件
<b>res/drawable/</b>	.png, .9.png, .jpg, .gif 等	图片文件
<b>res/raw/</b>	直接复制到设备中的任意文件，它们无需编译	Raw 文件
<b>res/values/</b>	存放可以被编译成很多种类型的资源文件 a. array.xml：定义数组 b. colors.xml：定义 color drawable 和颜色的字符串值。 c. dimens.xml 定义尺寸值(dimension value)。 d. strings.xml 定义字符串(string)值。 e. styles.xml 定义样式(style)对象。	XML 格式
<b>res/xml/</b>	任意的 XML 文件，在运行时可通过调用 Resources.getXML()读取	XML 格式

## assets 目录

存放的原生资源文件。例如txt，html 等

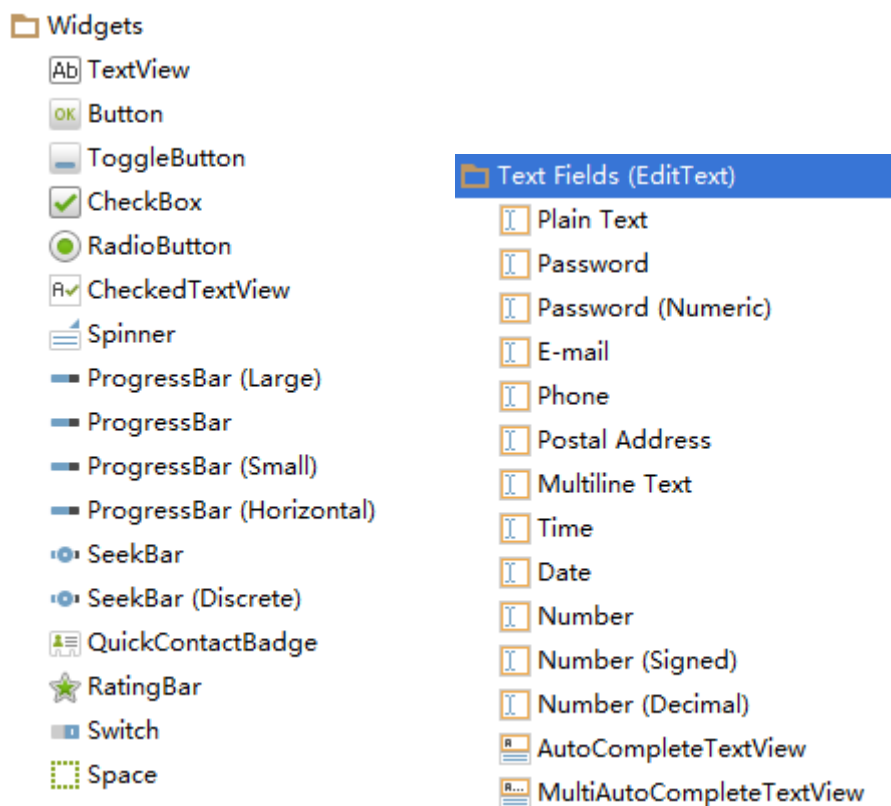


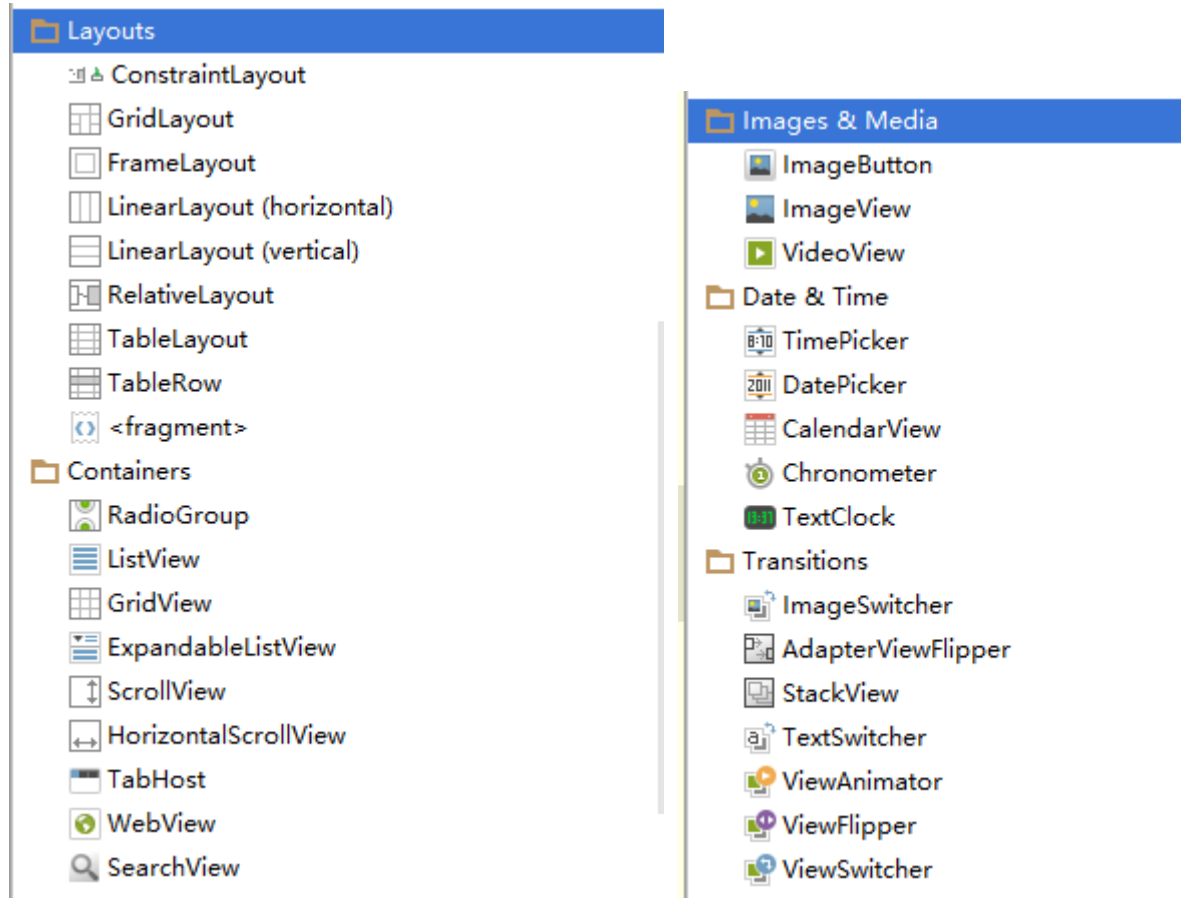
## Android View

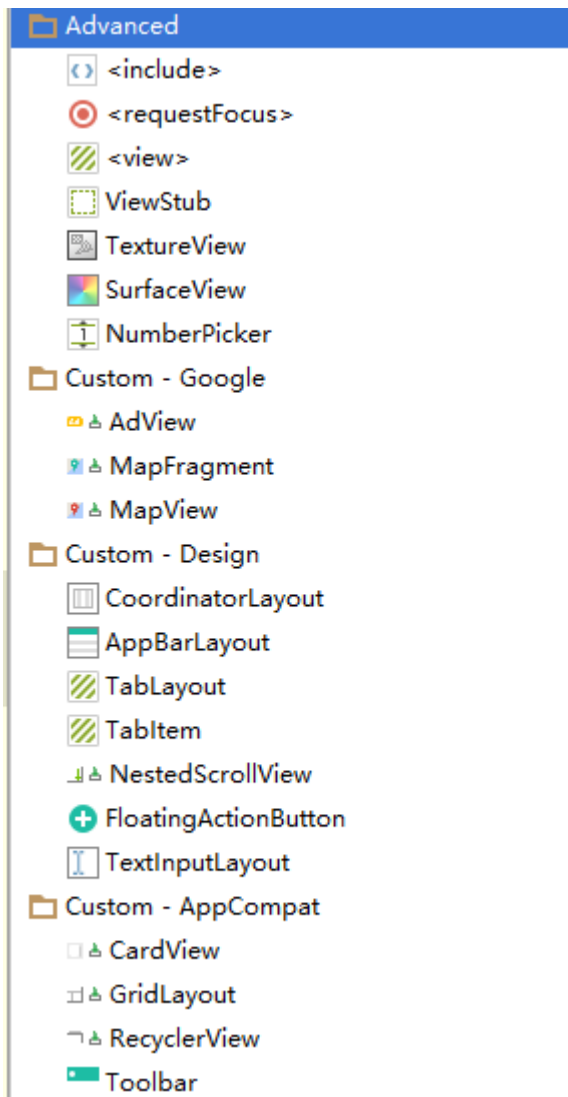
What is Android View?

- ✧ Android UI最基本的单元是View
- ✧ 继承自android.view.View
- ✧ 一个View代表一个控件（widget）
- ✧ 包括button、textview、edittext等。

View类是所有与用户交互的组件的Widgets的基类，Android\_View继承体系图：







## Android Layout

布局就像容器，里面可以装下很多控件。布局里面还可以套用其他的布局。可以实现界面的多样性与设计的灵活性

### LinearLayout 线性布局

所有元素都水平或垂直放置，可以嵌套其它布局



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFFFFF"
    tools:context=".MainActivity" >

    // 这里第一行显示标签为一个水平布局
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <EditText
            android:id="@+id/msg"
            android:inputType="number"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="">
        </EditText>
    </LinearLayout>
```

## RelativeLayout 相对布局

RelativeLayout就是以相对的方式定位布局，允许子元素指定他们相对于其它元素或父元素的位置（通过ID指定）。因此如果第一个元素在屏幕的中央，那么相对于这个元素的其它元素将以屏幕中央的相对位置来排列。

相对布局的属性比较多，但用起来比较灵活. 关键的属性

android:layout\_below 在某元素的下方

android:layout\_above 在某元素的上方

android:layout\_toLeftOf 在某元素的左边

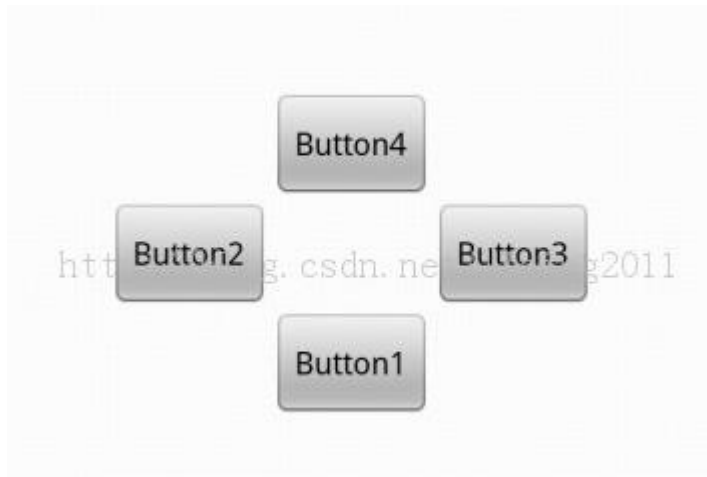
android:layout\_toRightOf 在某元素的右边

android:layout\_alignTop 本元素的上边缘和某元素的的上边缘对齐

android:layout\_alignLeft 本元素的左边缘和某元素的的左边缘对齐

android:layout\_alignBottom 本元素的下边缘和某元素的的下边缘对齐

android:layout\_alignRight 本元素的右边缘和某元素的的右边缘对齐



```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <Button android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_centerHorizontal="true"
        android:text="Button1"
    ></Button>
    <Button android:id="@+id/btn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/btn1"
        android:layout_above="@id/btn1"
        android:text="Button2"
    ></Button>
    <Button android:id="@+id/btn3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/btn1"
        android:layout_above="@id/btn1"
        android:text="Button3"
    ></Button>
    <Button android:id="@+id/btn4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/btn2"
        android:layout_toLeftOf="@id/btn3"
        android:layout_above="@id/btn2"
        android:text="Button4"
    ></Button>
</RelativeLayout>
```

## TableLayout 表格布局

表格布局，类似于HTML的Table和WPF的Grid。

通过TableRow来定义一行，如果一个控件占用多列可以设置android:layout\_span。

默认情况下一个控件是按顺序放置在每一列的(column 0, column 1...), 也可以通过android:layout\_column指定放在哪一列。

如果一列内容过长或者过短，可以通过android:stretchColumns和android:shrinkColumns来增加或者减少此列的宽度。



```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:shrinkColumns="0,1,2" // 设置三列都可以收缩
    android:stretchColumns="0,1,2" // 设置三列都可以拉伸 如果不设置这个，那个显
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TableRow android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <Button android:gravity="center"
            android:padding="10dp"
            android:text="Button1">
        </Button>

        <Button android:gravity="center"
            android:padding="10dp"
            android:text="Button2">
        </Button>
        <Button android:gravity="center"
            android:padding="10dp"
            android:text="Button3">
        </Button>
    </TableRow>
```

## FrameLayout 框架布局

将所有的子元素放在整个界面的左上角，后面的子元素直接覆盖前面的子元素，所以用的比较少。  
所以用的比较少。

## AbsoluteLayout 绝对布局（已废弃）

## Android SDK 工具布局优化工具

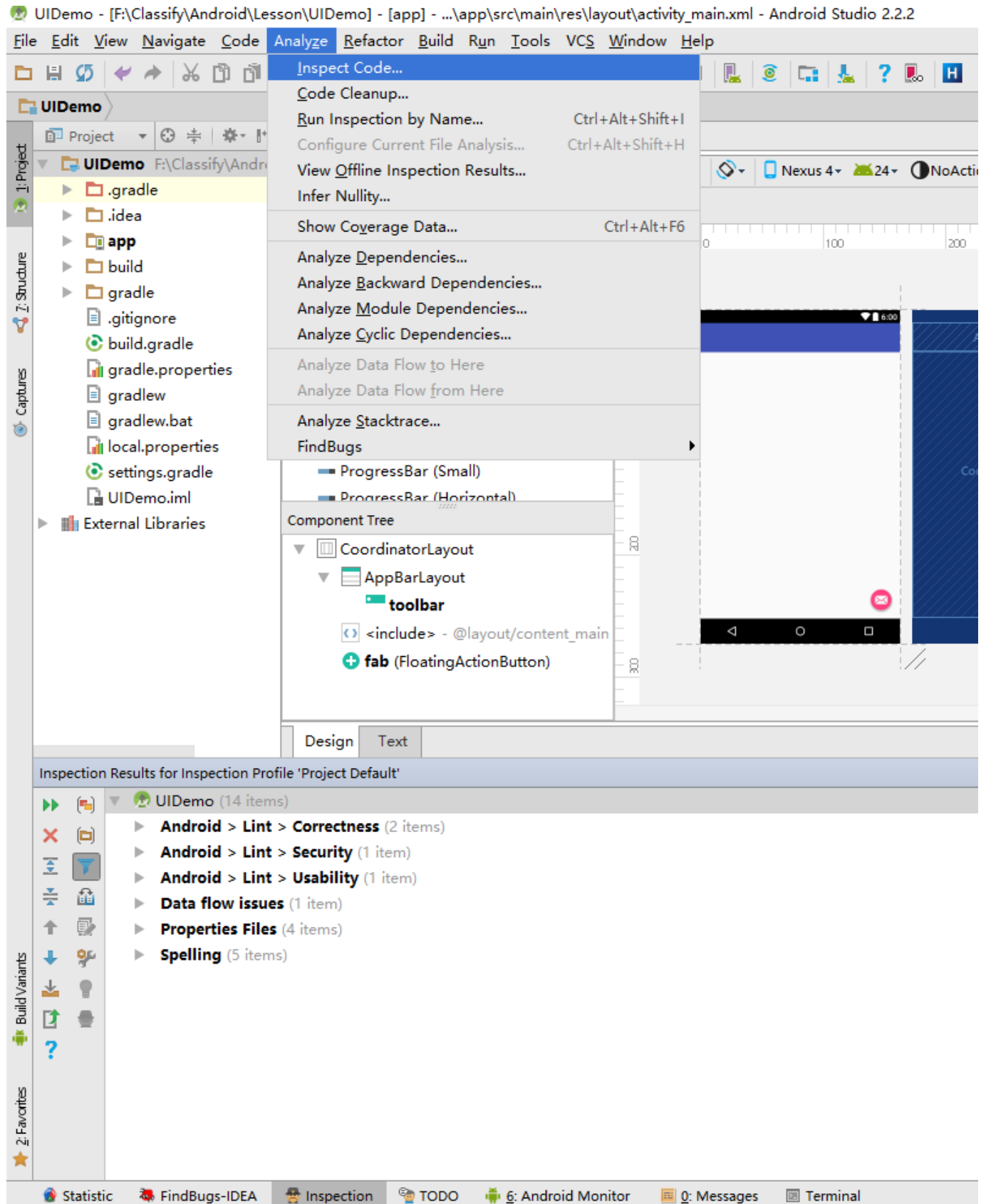
lint（sdk 16之前是layoutopt）

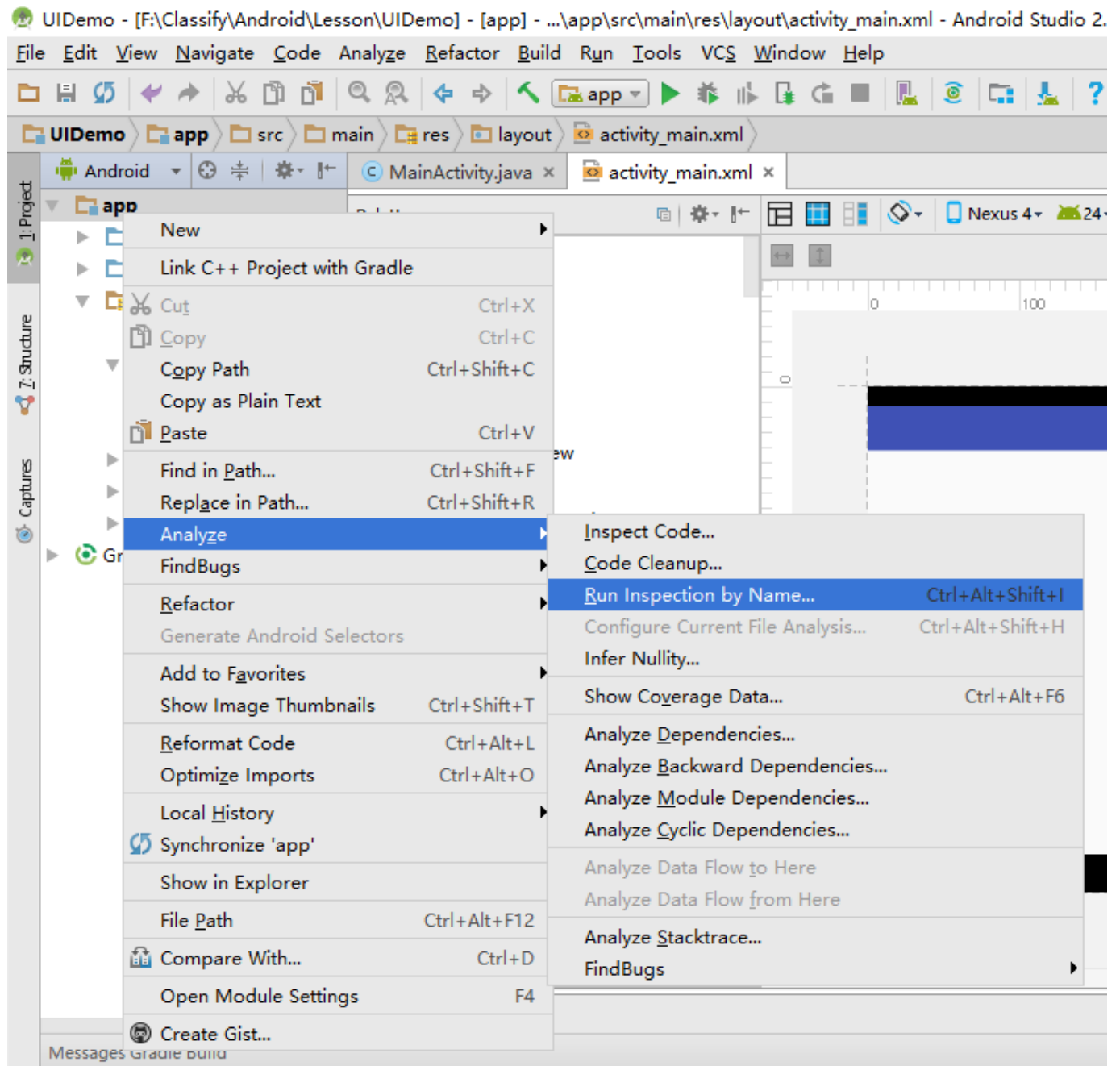
位置：sdk\tools

Android Studio使用Lint进行代码检查：<http://blog.csdn.net/u014651216/article/details/50820748>

android lint-checks <http://tools.android.com/tips/lint-checks>







## Customized Controls

### 为什么需要自定义 View

1. 现有的 View 满足不了你的需求,也没有办法从已有控件派生一个出来;界面元素需要自己绘制。
2. 现有 View 可以满足要求,把它做成自定义 View 只是为了抽象: 为这个自定义 View 提供若干方法,方便调用着操纵 View。通常做法是派生一个已有 View,或者结合 xml 文件直接 inflate。

Attention: 能够用 Android 基础控件解决的问题就尽量用基础控件,其次是用基础控件的组合。如果是确实有必要自定义才考虑自定义。自定义的控件,既需要耗费较长的开发时间,又不一定能保证有基础控件那么高的效率(基础控件都是谷歌优化过了的)。

### 应用场景

- A. 组合控件: 试题控件 (TextView+VideoGroup)、下拉刷新、瀑布流控件、带左/右滑功能的控件、视频控件等。通过 Android 的基础控件 (TextView、CheckBox、Button、ProgressBar 等) 组合而成,
- B. 完全自定义控件: 继承自 View、TextureView 或 SurfaceView,然后重写核心的回调方法。  
比如: webview + loading 动画 (SurfaceView)、  
比如输入法中的手写控件、图文混排控件 (现在很多都是通过 webview 加载网页实现了)、词典取词控件、图表控件、个性化进度条、弹幕显示控件、Markdown 控件、IDE 代码编辑控件等。

### 系统学习 Android 自定义控件

如何系统学习自定义控件 <https://zhuanlan.zhihu.com/p/21995633>

[Android技术专题]自定义View: <https://www.zhihu.com/question/41101031>

## 自定义属性

Android提供了一些基本的控件实现，有时无法满足我们需求。

这时需要自定义控件，在Android基础控件上实现我们想要的功能

下面以RadioButton为例：

我们自定义控件RadioButton，加入自定义属性“key”

### attrs.xml 中定义 key 属性

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3
4 <declare-styleable name="RadioButton">
5 <attr name="key" format="string" />
6 </declare-styleable>
7
8 </resources>
```

**XML** 中引用自定义控件

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res/com.vivian.ui.testcase"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    .....
    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="@string/user_radiobtn"
        android:layout_gravity="center_horizontal">

        <com.user.utils.UserRadioButton
            android:id="@+id/userRadioBtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/user_radiobtn"
            app:key="user key" >
        </com.user.utils.UserRadioButton>

        <com.user.utils.UserRadioButton
            android:id="@+id/userRadioBtn2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/user_radiobtn"
            app:key="user key 2" >
        </com.user.utils.UserRadioButton>
    </RadioGroup>
```

## 创建 UserRadioButton 类继承 RadioButton

把我们自定义的属性跟UserRadioButton类中的变量绑定

```
public class UserRadioButton extends RadioButton {
    .....
    public UserRadioButton(Context context, AttributeSet attrs) {
        super(context, attrs);

        // bind with property values/attrs.xml
        try {
            TypedArray a = context.obtainStyledAttributes(attrs,
                R.styleable.RadioButton);
            this.mUserKey = a.getString(R.styleable.RadioButton_key);
            a.recycle();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public String getKey() {
        return mUserKey;
    }

    public void setKey(String value) {
        mUserKey = value;
    }
}
```

## Code 中修改自定义属性

```
mUserRadioButton = (UserRadioButton) findViewById(R.id.userRadioBtn);  
mUserRadioButton.setKey("modified key");
```

## 优质开源库

awesome-android-ui : <https://github.com/wasabeef/awesome-android-ui>

android-open-project : <https://github.com/Trinea/android-open-project>

## View 事件传递

### 基础知识

A. 所有 Touch 事件都被封装成了 MotionEvent 对象，包括 Touch 的位置、时间、历史记录以及第几个手指(多指触摸)等。

B. 事件类型分为

ACTION\_DOWN, ACTION\_UP, ACTION\_MOVE,

ACTION\_POINTER\_DOWN, ACTION\_POINTER\_UP,

ACTION\_CANCEL,

每个事件都是以 ACTION\_DOWN 开始 ACTION\_UP 结束。

C. 对事件的处理包括三类

传递——dispatchTouchEvent()函数

拦截——onInterceptTouchEvent()函数

消费——onTouchEvent()函数和 onTouchListener

### 传递流程

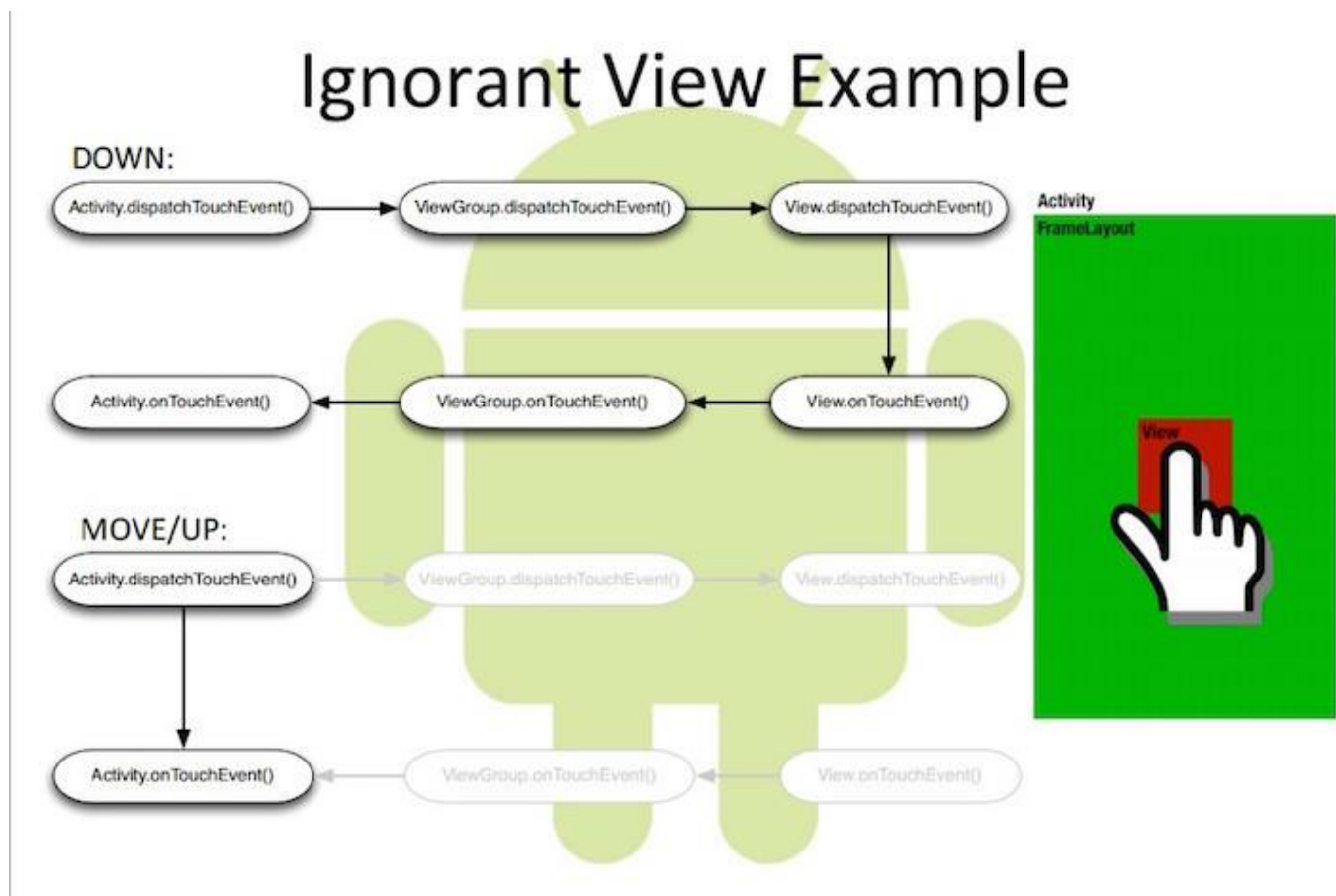
1. 事件从 Activity.dispatchTouchEvent()开始传递。只要没有被停止或拦截，从最上层的 View(ViewGroup)开始一直往下(子 View)传递。子 View 可以通过 onTouchEvent()对事件进行处理。
2. 事件由父 View(ViewGroup)传递给子 View， ViewGroup 可以通过 onInterceptTouchEvent()对事件做拦截，停止其往下传递。



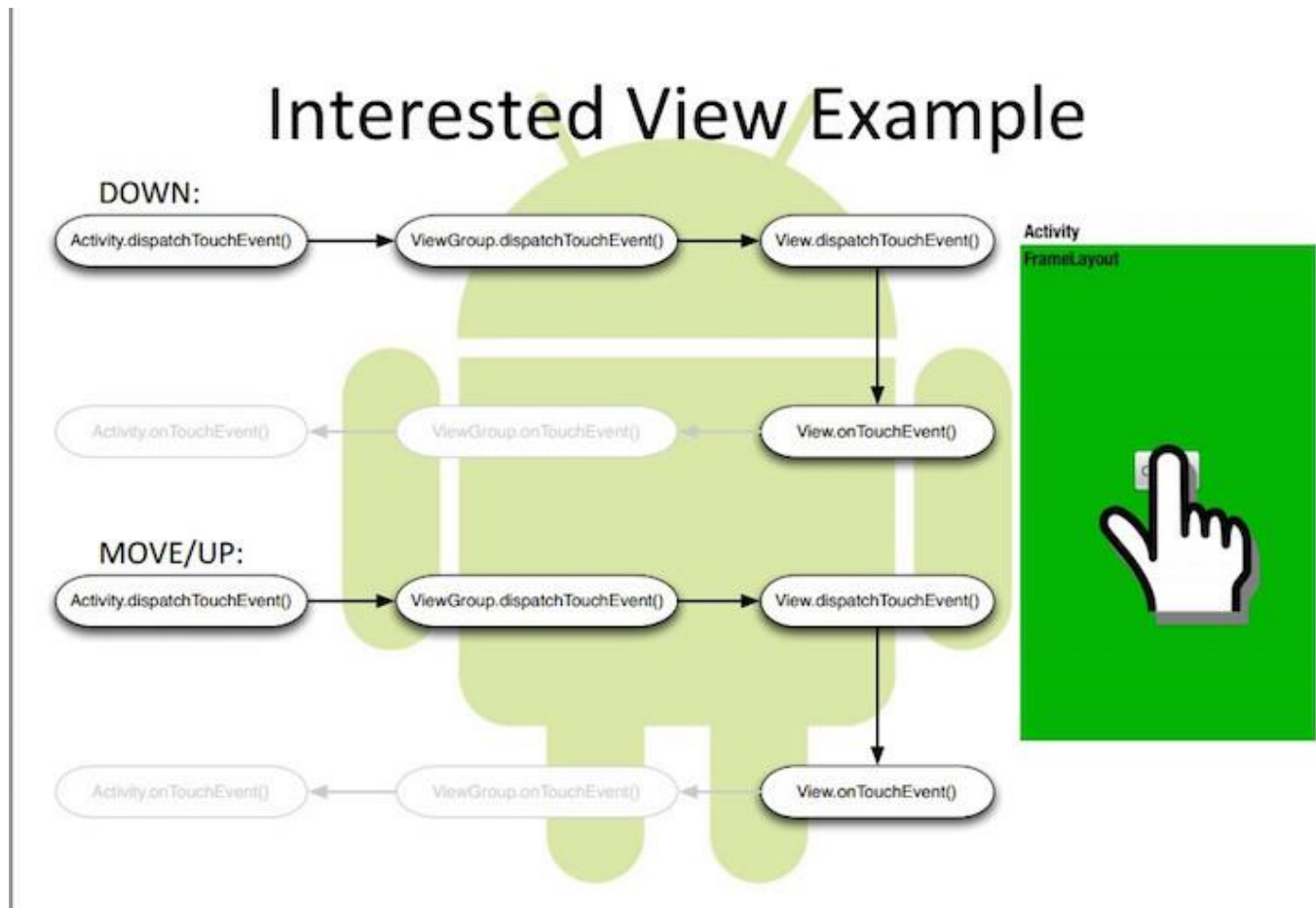
3. 如果事件从上往下传递过程中一直没有被停止，且最底层子 View 没有消费事件，事件会反向往上传递，这时父 View(ViewGroup)可以进行消费，如果还是没有被消费的话，最后会到 Activity 的 onTouchEvent()函数。
4. 如果 View 没有对 ACTION\_DOWN 进行消费，之后的其他事件不会传递过来。
5. onTouchListener 优先于 onTouchEvent()对事件进行消费。

上面的消费即表示相应函数返回值为 true。

### View 不处理事件流程图



## View 处理事件流程图



详见: <https://github.com/android-cn/android-open-project-analysis> 公共技术点之事件传递

## Reference

- [1] 如何系统学习自定义控件  
<https://zhuanlan.zhihu.com/p/21995633>
- [2] [Android技术专题]自定义View  
<https://www.zhihu.com/question/41101031>
- [3] 推翻自己和过往，重学自定义View
- [4] 深入了解View
- [5] Android 深入理解Android中的自定义属性
- [6] 公共技术点之 View 事件传递
- [7] Android 触摸及手势操作GestureDetector
- [8] Android视图SurfaceView的实现原理分析
- [9] 五、优质开源项目
  - awesome-android-ui
  - Android 开源项目分类汇总
  - Android 平台下的原生 Markdown 解析器
  - Android 开源弹幕引擎
  - Android 图文混排控件 MixtureTextView