所有可点击的UI都具有水波纹效果

Material主题样式

新的设计语言，Material Design对排版、材质、配色、光效、间距、文字大小、交互方式、动画轨迹都做出了建议，以帮助设计者设计出符合Material Design风格的应用。
最关心的还是如何在项目中使用Material Design风格：

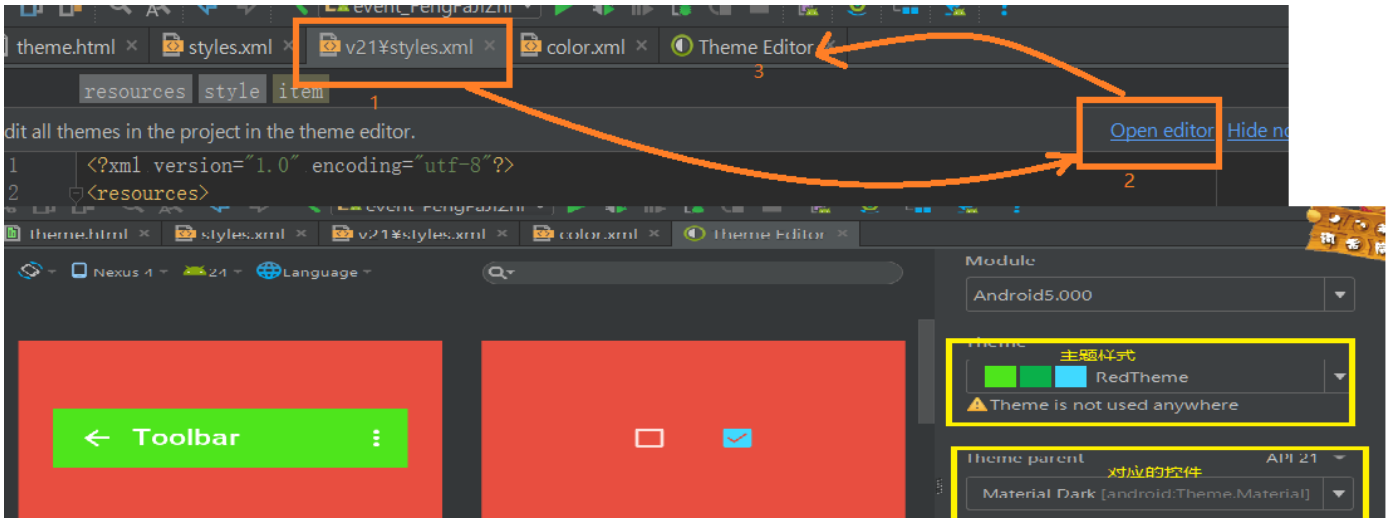
1. 设置应用的 `targetSdkVersion` 和 `targetSdkVersion` 为21
2. 在values目录下的style资源文件中创建一个style，让其继承自 `android:Theme.Material`
3. 在AndroidManifest中指定应用的主题或者Activity的主题为我们设定的样式

谷歌官方我们提供了三种配色风格的Material Design样式：

1. 黑色主题 `Theme.Material`
2. 明亮主题 `Theme.Material.Light`
3. 明亮主题黑色ActionBar `Theme.Material.Light.DarkActionBar`

主题颜色改变方法：在style样式文件中选择对应的组件改变color文件中的颜色值

在style样式文件－－－－>Open Editor进入预览模式改变



Elevation（海拔）属性 凸显布局的层次，建议使用阴影效果，给View添加了一个新的属性：`z` 属性，用于描述视图距离它父视图的高度：

Elevation设置控件阴影的

布局中：android:elevation="10dp"　　代码中：setElevation(10);

View的outlineprovider属性 指定轮廓，不同的轮廓决定了控件不否显示阴影

注意：如果采用图片作为背景，即使在xml布局中指定android:outlineProvider为background也不会显示阴影，只有通过代码中指定轮廓来显示。
在xml布局中，可以通过 android:outlineProvider 来指定轮廓的判定方式：

1. none 即使设置了Z属性，也不会显示阴影
2. background 会按照背景来设置阴影形状
3. bounds 会按照View的大小来描绘阴影
4. paddedBounds 和bounds类似，不过阴影会稍微向右偏移一点

用法：在布局中

android:outlineProvider="none"　设定轮廓

android:elevation="10dp"　　设置阴影

android:background="@drawable/circle_shape"　　设定背景样式，

如果背景为图片则阴影效果要通过代码来设置，指定轮廓

```
ViewOutlineProvider viewOutlineProvider = new ViewOutlineProvider() {
    public void getOutline(View view, Outline outline) {
        // 可以指定形状：圆形，矩形，圆角矩形，path
        outline.setOval(0, 0, view.getWidth(), view.getHeight());
    }
};
View.setOutlineProvider(viewOutlineProvider );
```

View的裁剪功能 想根据轮廓来缩小一个View，则可以通过剪裁

如果一个View指定了轮廓，调用 `setClipToOutline` 方法，就可以根据轮廓来剪裁一个View。想要剪裁轮廓，必须要给View先指定轮廓，并且轮廓是可以被剪裁的，目前只有圆形，矩形，圆角矩形支持剪裁，可以通过outline.canClip()来判断一个轮廓是否支持剪裁。
Path剪裁不会改变View的大小，但是如果Path的范围比View要的bounds要小，则剪裁后会改变View的位置，位置偏移和Z属性有关
代码中的用法：

设置指定轮廓

```
viewOutlineProvider = new ViewOutlineProvider() {
    public void getOutline(View view, Outline outline{
        outline.setOval(0,0,view.getWidth(),view.getHeight());
```

```
        }
    };
cut3.setOutlineProvider(viewOutlineProvider);
```

剪切轮廓

```
cut3.setClipToOutline(true);
```

布局中表达控件

```
<TextView
        android:id="@+id/cut3"
        android:text="矩形剪裁成圆形"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:outlineProvider="paddedBounds"
        android:elevation="10dp"/>
```

Tint属性为UI组件染色做选择器

1、在drawable目录下建立一个xml文件做为点击时的颜色变化的背景文件

2、定义文件内容：

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
        android:src="@drawable/ring"     原始背景图片
        android:tintMode="multiply"      染色的模式
        android:tint="#5677fc" />        将要染色的颜色
```

3、写选择器:

```
<?xml version="1.0" encoding="utf-8"?><selector  xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/tint_bitmap" android:state_pressed="true"/> 暗下去引用染色器
    <item android:drawable="@drawable/ring" />       没有按原始图
</selector>
```

4、UI组件引用

```
<TextView
        android:clickable="true"
        android:layout_marginTop="20dp"
        android:layout_marginBottom="20dp"
        android:layout_marginLeft="20dp"
        android:background="@drawable/tint_selector"/>
```

tint的模式：有十六种 tint的渲染模式有总共有16种，xml文件中可以使用6种

布局中 android:backgroundTint="#5677fc" 染色 android:backgroundTintMode="add" 染色模式

1. `PorterDuff.Mode.CLEAR` 所绘制不会提交到画布上。
2. `PorterDuff.Mode.SRC` 显示上层绘制图片
3. `PorterDuff.Mode.DST` 显示下层绘制图片
4. `PorterDuff.Mode.SRC_OVER` 正常绘制显示，上下层绘制叠盖。
5. `PorterDuff.Mode.DST_OVER` 上下层都显示。下层居上显示。
6. `PorterDuff.Mode.SRC_IN` 取两层绘制交集。显示上层。
7. `PorterDuff.Mode.DST_IN` 取两层绘制交集。显示下层。
8. `PorterDuff.Mode.SRC_OUT` 取上层绘制非交集部分。
9. `PorterDuff.Mode.DST_OUT` 取下层绘制非交集部分。
10. `PorterDuff.Mode.SRC_ATOP` 取下层非交集部分与上层交集部分
11. `PorterDuff.Mode.DST_ATOP` 取上层非交集部分与下层交集部分
12. `PorterDuff.Mode.XOR` 取两层绘制非交集。两层绘制非交集。
13. `PorterDuff.Mode.DARKEN` 上下层都显示。变暗
14. `PorterDuff.Mode.LIGHTEN` 上下层都显示。变亮
15. `PorterDuff.Mode.MULTIPLY` 取两层绘制交集
16. `PorterDuff.Mode.SCREEN` 上下层都显示。

Palette类实现取色

Palette调色板，可以很方便的让我们从图片中提取颜色。并且可以指定提取某种类型的颜色。

1. `Vibrant` 鲜艳的
2. `Vibrant` dark鲜艳的暗色
3. `Vibrant` light鲜艳的亮色
4. `Muted` 柔和的
5. `Muted` dark柔和的暗色

6. `Muted` light柔和的亮色

vector矢量图

矢量图也称为面向对象的图像或绘图图像，是计算机图形学中用点、直线或者多边形等基于数学方程的几何图元表示图像。矢量图形最大的优点是无论放大、缩小或旋转等不会失真；最大的缺点是难以表现色彩层次丰富的逼真图像效果。

矢量图的画法：

在布局UI背景中，引用即可。

```
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
        android:height="300dp"           图片大小
        android:width="300dp"
        android:viewportHeight="40"      虚拟画布大小
        android:viewportWidth="40">
        <path android:fillColor="#ff00ff"      //描述图片形状
       android:pathData="M20.5,9.5c-1.955,0,-3.83,1.268,-4.5,3c-0.67,-1.732,-2.547,-3,-4.5,-3
C8.957,9.5,7,11.432,7,14c0,3.53,3.793,6.257,9,11.5c5.207,-5.242,9,-7.97,9,-11.5C25,11.432,23.043,9.5,20.5,9.5z"/>
</vector>
```

水波纹扩散效果在两种不同的状态间的动画

button默认带有该效果。除了默认的效果外，系统还提供了另外两种效果，我们只把button的背景指定为：

1. ?android:attr/selectableItemBackground
2. ?android:attr/selectableItemBackgroundBorderless

任何view处于可点击状态，都可以使用RippleDrawable来达到水波纹特效。

用法：

```
<Button
        android:layout_width="fill_parent"
        android:layout_height="100dp"
        android:background="?android:attr/selectableItemBackgroundBorderless"/>     水波纹动画
```

圆形展示动画（圆形缩小动画）

控件在被点击之后以圆形缩小动画展示

通过 ViewAnimationUtils.createCircularReveal 来创建一个动画，该api接受5个参数：

1. view 操作的视图   控件
2. centerX 动画开始的中心点X
3. centerY 动画开始的中心点Y
4. startRadius 动画开始半径
5. startRadius 动画结束半径

```
Animator animator1 = ViewAnimationUtils.createCircularReveal(bt1, bt1.getWidth() / 2, bt1.getHeight() / 2, bt1.getWidt
animator1.setInterpolator(new LinearInterpolator());
animator1.setDuration(10000);
animator1.start();
```

从左上角扩展的圆形动画

```
Animator animator2 = ViewAnimationUtils.createCircularReveal(bt2,0,bt2.getHeight(),0,(float) Math.hypot(bt2.getWidth
```

曲线动画 绘制一个动画的路径

```
Path path = new Path();
path.moveTo(100,100);
path.quadTo(1000,300,300,700);
ObjectAnimator mAnimator = ObjectAnimator.ofFloat(curved, View.X, View.Y, path);
Path p = new Path();
p.lineTo(0.6f, 0.9f);
p.lineTo(0.75f, 0.2f);
p.lineTo(1f, 1f);
mAnimator.setInterpolator(new PathInterpolator(p));
mAnimator.setDuration(3000);
mAnimator.start();
```

状态动画（选择器中设置Elevation及动画）

用法，布局UI中

```
<!--android:background="@drawable/state_anim"-->       UI背景可指定也可不指定
            <TextView
                    android:id="@+id/state_anim"
                    android:stateListAnimator="@drawable/state_anim"    指定一个状态动画 选择器
                    android:layout_width="100dp"
                    android:layout_height="100dp"/>
```

写选择器：

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true" android:drawable="@color/control_highlight_blue_grey">
        <set>                                红色可以不要
            <objectAnimator android:propertyName="translationZ"
                            android:duration="200"
                            android:valueTo="20dp"
                            android:valueType="floatType"/>
            <objectAnimator android:propertyName="scaleX"
                            android:duration="200"
                            android:valueTo="0.5"
                            android:valueType="floatType"/>
            <objectAnimator android:propertyName="scaleY"
                            android:duration="200"
                            android:valueTo="0.5"
                            android:valueType="floatType"/>
        </set>
    </item>
    <item android:state_enabled="true" android:state_pressed="false" android:drawable="@color/button_normal_blue_grey"
        <set>                                                                    红色可以不要
            <objectAnimator android:propertyName="translationZ"
                            android:duration="200"
                            android:valueTo="0"
                            android:valueType="floatType"/>
            <objectAnimator android:propertyName="scaleX"
                            android:duration="200"
                            android:valueTo="1"
                            android:valueType="floatType"/>
            <objectAnimator android:propertyName="scaleY"
                            android:duration="200"
                            android:valueTo="1"
                            android:valueType="floatType"/>
        </set>
    </item>
</selector>
```

矢量图动画

布局UI展示：

```
<TextView
        android:id="@+id/vector_anim"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:text="矢量图动画"
        android:background="@drawable/vector_animation"     引用动画
```

```
        />
```

写动画：

```
<?xml version="1.0" encoding="utf-8"?>
<animated-vector xmlns:android="http://schemas.android.com/apk/res/android"
                 android:drawable="@drawable/vector_drawable" >    //默认矢量图
<target
    android:name="vector"
    android:animation="@anim/vector_anim" />    引用动画矢量图
</animated-vector>
```

动画矢量图

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator              第一张动画矢量图
            android:duration="3000"
            android:propertyName="pathData"
            android:valueFrom="M67,750 L500,0 500,0 933,750 67,750         M146,853 A 0.1,500 0 0 1 146,148 M146,148 A
            android:valueTo="M147,853 L147,147 853,147 853,853 147,853     M146,853 A 0.1,500 0 0 1 146,148 M146,148 A
            android:valueType="pathType" />
   <objectAnimator                第二张动画矢量图
            android:startOffset="3000"
            android:duration="3000"
            android:propertyName="pathData"
            android:valueFrom="M147,853 L147,147 853,147 853,853 147,853     M147,853 A 0.1,500 0 0 1 147,147 M147,147
            android:valueTo="M147,853 L147,147 853,147 853,853 147,853       M147,853 A 500,500 0 0 1 147,147 M147,147
            android:valueType="pathType" />
</set>
```

转场动画（界面切换动画）

步骤：

跳转面：点击跳转

1、`intent = new Intent(TransitionsActivity.this, ExplodeActivity.class);`

2、`startActivity(intent, ActivityOptions.makeSceneTransitionAnimation(TransitionsActivity.this).toBundle());`

真正打开面：

在onCreate(Bundle savedInstanceState)方法中

super.onCreate(savedInstanceState);后面

setContentView(R.layout.activity_animation);的前面

```
getWindow().requestFeature(Window.FEATURE_CONTENT_TRANSITIONS);
getWindow().setAllowEnterTransitionOverlap(true);
getWindow().setAllowReturnTransitionOverlap(true);
Explode explode = new Explode();     决定不同动画的关键所在
explode.setDuration(1000);
getWindow().setEnterTransition(explode);
getWindow().setExitTransition(explode);
```

```
Fade fade = new Fade();    淡入淡出动画
```

```
Slide slide = new Slide();  滑动动画
```

在退出界面时：

```
onBackPressed() {
   super.onBackPressed();
   finishAfterTransition();
}
```

RecyclerView控件**可以替代Listview等所有可滑动的View控件，超级加强型**

可以上下，左右，上下九宫格，左右九宫格，上下瀑布流，左右瀑布流等样式

使用方法：

ViewFliper与ViewPaer

都可以实现左右滑动的效果

ViewPager用于实现多页面的切换效果，该类存在于Google的兼容包里面，所以在引用时记得在BuilldPath中加入“android-support-v4.jar”

View切换的控件—ViewFlipper

ViewFilpper类继承于ViewAnimator类。而ViewAnimator类继承于FrameLayout。ViewAnimator类的源码可以看出此类的作用主要是为其中的View切换提供动画效果，如果想要实现滑动翻页的效果，就要了解另外一个类：**Android**.view.GestureDetector类。GestureDetector类中可以用来检测各种手势事件。该类有两个回调接口，分别用来通知具体的事件。

CardView开发出卡片效果

SwipeRefreshLayout刷新控件

ToolBar控件

TextInputLayout控件

FloatingActionButton控件

SnackBar类

TabLayout控件

DrawerLayout控件容器（自带抽屉侧滑动菜单）

用法：

1、布局声明

```
<android.support.v4.widget.DrawerLayout
    android:id="@+id/dl_drawer"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <!--主内容view-->
    <FrameLayout
        android:id="@+id/fl_conest"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </FrameLayout>
    <!--左侧菜单view-->
    <FrameLayout
        android:id="@+id/fl_file"
        android:layout_width="200dp"
        android:layout_height="match_parent"
        android:layout_gravity="left">
    </FrameLayout>
</android.support.v4.widget.DrawerLayout>
```

2、创建View类及对应的子布局

NavigationView控件

ScrollView控件滚屏控件