还没有完成，勿看勿看~！！！！

转载请注明：https://blog.csdn.net/feather_wch/article/details/80097833

Material Design动画

触摸反馈

1、触摸反馈(点击后水波特效)

> 按钮中添加属性(两种)：

```
<Button
    xxxxxx
    android:text="有界水波纹"
    android:background="?android:attr/selectableItemBackground"/>

<Button
    xxxxxx
    android:text="无界水波纹"
    android:background="?android:attr/selectableItemBackgroundBorderless"/>
```

揭露效果

2、Reveal Effect实例

> 1. 通过 View的ViewAnimationUtils的方法
> 2. 揭露效果 的动画的圆心是以 目标View的左上角确定坐标系的

```
button.post(new Runnable() {
    @Override
    public void run() {
        //圆心
        int cx = button.getWidth() / 2;
        int cy = button.getHeight() / 2;
        //最终半径
        int finalRadius = Math.max(button.getWidth(), button.getHeight());

        Animator animator = ViewAnimationUtils.createCircularReveal(button, cx, cy, 0, finalRac
        animator.setDuration(5000);
        animator.start();
    }
});
```

3、崩溃错误-解决办法：“揭露动画”需要在View被attache后调用

```
java.lang.RuntimeException: Unable to start activity ComponentInfo{xxx}: java.lang.IllegalStateException: C
```

```
//1. The Runnable will run after the view's creation. No need to post delay.

view.post(new Runnable()
{
       @Override
       public void run(){
           //create your anim here
       }
});

//2. Adding GlobalLayoutListener to wait for the view to be inflated worked for me.
 tv.getViewTreeObserver().addOnGlobalLayoutListener(new ViewTreeObserver.OnGlobalLayoutListener
        @Override
        public void onGlobalLayout() {
            if (android.os.Build.VERSION.SDK_INT < Build.VERSION_CODES.JELLY_BEAN) {
                tv.getViewTreeObserver().removeGlobalOnLayoutListener(this);
            } else {
                tv.getViewTreeObserver().removeOnGlobalLayoutListener(this);
            }
            Animator anim = ViewAnimationUtils.createCircularReveal(tv, a, b, 0, radius);
            anim.start();
        }
    });


//3. Or you can do this.
tv.postDelayed(new Runnable() {
    @Override
    public void run() {
          //start your animation here
    }
}, 1000);

//4. Call your Animator logic from onResume(). That way you'll be sure all views have been atta
onResume(){
  //TODO reveal nimation
}
```

转场动画

Activity传统转场动画

4、mActivity.overridePendingTransition()

```
//启动Activity
public void onStartActivity(){

    Intent intent = new Intent(getActivity(),UpgradeActivity.class);
    mActivity.startActivity(intent);
    mActivity.overridePendingTransition(R.anim.activity_zoom_enter_in,
}
```

5、注意点

- 1、必须在 StartActivity() 或 finish() 之后立即调用;
- 2、而且在 2.1 以上版本有效;
- 3、手机设置-显示-动画，要开启状态.

ActivityOptions

6、ActivityOptions

1. MD风格的Activity过场动画辅助类
2. 是一个静态类, 兼容包是 `ActivityOptionsCompat` (android.support.v4.app.ActivityOptionsCompat)
3. 提供了6种方法，用于实现过场动画。

7、ActivityOptions的过场动画

过场动画(ActivityOptions的方法)	介绍
makeSceneTransitionAnimation(activity, sharedElement, sharedElementName)	单一View的过场动画
makeSceneTransitionAnimation(activity, sharedElements)	多个View的过场动画
makeClipRevealAnimation()	揭露效果
makeCustomAnimation()	自定义动画
makeScaleUpAnimation()	渐变缩放

8、转场动画简单实例

```
//第一个Activity
Intent intent=new Intent(FirstActivity.this,SecondActivity.class);
startActivity(intent, ActivityOptions.makeSceneTransitionAnimation(FirstActivity.this).toBundle
```

```
//第二个Activity-需要在setContentView()前面设置
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    //TODO 淡入淡出
    getWindow().setEnterTransition(new Fade().setDuration(2000));  //进入动画
    getWindow().setExitTransition(new Fade().setDuration(2000));  //退出动画

    //加快动画且更生动
    getWindow().setAllowEnterTransitionOverlap(true);

    setContentView(R.layout.activity_second);
}
```

9、系统提供的转场动画一共有三种

> 1. 渐变： `Fade()`
> 2. 边缘进入/退出: `Slide()`
> 3. 从场景中心移进/移出： `Explode()`

10、自定义动画实现方法(makeCustomAnimation):

```
//1. 自定义动画(指定动画的ID)
ActivityOptionsCompat.makeCustomAnimation(this, R.anim.translate_in, R.anim.translate_none);

//调用该方法进行退出动画。
ActivityCompat.finishAfterTransition(this);
```

11、揭露效果的实现方法(makeClipRevealAnimation):

> 1. 需要在 `Style`中 设置透明度:

```
<style name="AppTheme.Translucent" parent="Theme.AppCompat.Light.NoActionBar">
    <item name="android:windowBackground">@android:color/transparent</item>
</style>
```

12、单一共享元素(makeSceneTransitionAnimation):

> 1-两个需要共享的 `View` 使用 `android:transitionName="stars_head_img"` 进行标注。
> 2-第一个Activity中调用：(第二个Activity可以进行绑定或者不绑定)

```
startActivity(intent,
    ActivityOptions.makeSceneTransitionAnimation(
                StarActivity.this, //第一个Activity
                imageView //第一个View
                , "stars_head_img").toBundle()); //标注的名字
```

> 3-第二个Activity绑定相关View

```
ViewCompat.setTransitionName(imageView, "stars_head_img");
```

13、多元素共享(makeSceneTransitionAnimation(...Pair))

```
Intent intent=new Intent(StarActivity.this,MDActivity.class);
//设置多个共享元素的Pair
Pair<View, String> pairOne = new Pair<View, String>(holder.mHeadImg, "stars_head_img");
Pair<View, String> pairTwo = new Pair<View, String>(holder.mNameTxt, "stars_name_txt");
Pair<View, String> pairThreee = new Pair<View, String>(holder.mAgeTxt, "stars_age_txt");
//启动新的Activity
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    startActivity(intent,
            ActivityOptions.makeSceneTransitionAnimation(
                    StarActivity.this,
                    pairOne, pairTwo, pairThreee).toBundle());
}
```

> 建议两个元素的大小内容完全一致，才能保证流畅过渡。

转场动画相关效果

1、弹性效果

```
getWindow().setEnterTransition(new Explode().setDuration(1000).setInterpolator(new BounceInterp
getWindow().setExitTransition(new Explode().setDuration(1000));
```

> 适合在 `startActivity`前进行指定

```
Intent intent=new Intent(StarActivity.this,MDActivity.class);
Pair<View, String> pairOne = new Pair<View, String>(holder.mHeadImg, "stars_head_img");
Pair<View, String> pairTwo = new Pair<View, String>(holder.mNameTxt, "stars_name_txt");
Pair<View, String> pairThreee = new Pair<View, String>(holder.mAgeTxt, "stars_age_txt");
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {

    getWindow().setEnterTransition(new Explode().setDuration(500).setInterpolator(new BounceInt
    getWindow().setExitTransition(new Explode().setDuration(500));

    startActivity(intent,
            ActivityOptions.makeSceneTransitionAnimation(
                    StarActivity.this,
                    pairOne, pairTwo, pairThreee).toBundle());
}
```

Android几种预设动画

```
change_bounds
change_clip_bounds
change_transition
change_image_transition
change_scroll
```

changeBounds:改变目标视图的布局边界。
changeClipBounds:裁剪目标视图边界。
changeTransform:改变目标的缩放比例和旋转角度。
changeImageTransform:改变目标图片的大小和缩放比例。
overlay

1、

参考资料

1. ActivityOptions
2. Material Design动画