Hongyang

生命不息，奋斗不止，万事起于忽微，量变引起质变

目录视图 摘

个人资料

鸿洋_

关注 发私信

访问： 4963541次

积分： 34195

等级： BLOG 8

排名： 第65名

原创： 175篇 转载： 0篇

译文： 6篇 评论： 7872条

我的微信公众号

长期为您推荐优秀博文、开源项目、视频等，进入还有好玩的等着你，欢迎扫一扫。

联系方式

新动态

给我写信

QQ群：

429757068

264950424

463081660

请勿重复加群，Thx

文章分类

【Android 5.x】 (8)

【Android 精彩案例】 (35)

【Android 源码解析】 (29)

Android 属性动画（Property Animation）完全解析（上）

标签： Android Property Animation

2014-07-25 09:34 95840人阅读 评论(49)

分类： 【Android 源码解析】（28） 【android 进阶之路】（61）

目录(?) [+]

转载请标明出处：http://blog.csdn.net/lmj623565791/article/details/38067475

1、概述

Android提供了几种动画类型：View Animation 、Drawable Animation 、Property Animation 。View Animation相当简单的缩放、平移、旋转、透明度基本的动画，且有一定的局限性。比如：你希望View有一个颜色的切换动画；你希望可以你希望当动画停止时，View的位置就是当前的位置；这些View Animation都无法做到。这就是Property Animation产生的原细介绍Property Animation的用法。至于Drawable Animation，嗯，略~

2、相关API

Property Animation故名思议就是通过动画的方式改变对象的属性了，我们首先需要了解几个属性：

Duration动画的持续时间，默认300ms。

Time interpolatio

干嘛的了，定义动画的变化率。

Repeat count and behavior：重复次数、以及重复模式；可以定义重复多少次；重复时从头开始，还是反向。

Animator sets: 动画集合，你可以定义一组动画，一起执行或者顺序执行。

Frame refresh delay：帧刷新延迟，对于你的动画，多久刷新一次帧；默认为10ms，但最终依赖系统的当前状态；基本不用

相关的类

ObjectAnimator 动画的执行类，后面详细介绍

ValueAnimator 动画的执行类，后面详细介绍

AnimatorSet 用于控制一组动画的执行：线性，一起，每个动画的先后执行等。

AnimatorInflater 用户加载属性动画的xml文件

TypeEvaluator 类型估值，主要用于设置动画操作属性的值。

TimeInterpolator 时间插值，上面已经介绍。

总的来说，属性动画就是，动画的执行类来设置动画操作的对象的属性、持续时间，开始和结束的属性值，时间差值等，然后参数动态的变化对象的属性。

3、ObjectAnimator实现动画

之所以选择ObjectAnimator为第一个~~是因为，这个实现最简单~~一行代码，秒秒钟实现动画，下面看个例子：

布局文件：

```
[html]
<RelativeLayout 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"
```

【Android 自定义控件实战】 (29)
【Android 自定义控件之起步】 (7)
【Android 快速开发】 (11)
【Android 原生开发游戏】 (3)
【Java 并发专题】 (15)
【android 进阶之路】 (62)
【Java 设计模式】 (10)
【Android 百度地图】 (4)
【html5 css3精彩案例】 (14)
【Android github 控件】 (10)
【Android 基础】 (16)
【Javascript 】 (9)
【rabbitMQ 用法】 (5)

友情链接

郭霖的博客
夏安明的博客
任玉刚的博客
敬佩的孔老师
foruok的订阅号程序视界
泡在网上的日子

我的微博

微博
鸿洋_ 陕西 西安
加关注
[喵喵][good]
Android_CJJ：本来是想漂亮的不像实力派 做出来是“漂亮的不像实力派”咳 我深深的叹了口气 apk下载地址：http://t.cn/R4zTg7u GitHub地址：http://t.cn/R4zTg7m 请轻点吐槽.....
TA 的粉丝 (4396) 全部»
风林松山 johnny-y Leeovo

博客专栏

HTML5 & CSS3 实战
文章：11篇
阅读：81214

设计模式融入生活
文章：10篇
阅读：46883

Android 精彩案例
文章：67篇
阅读：2245401

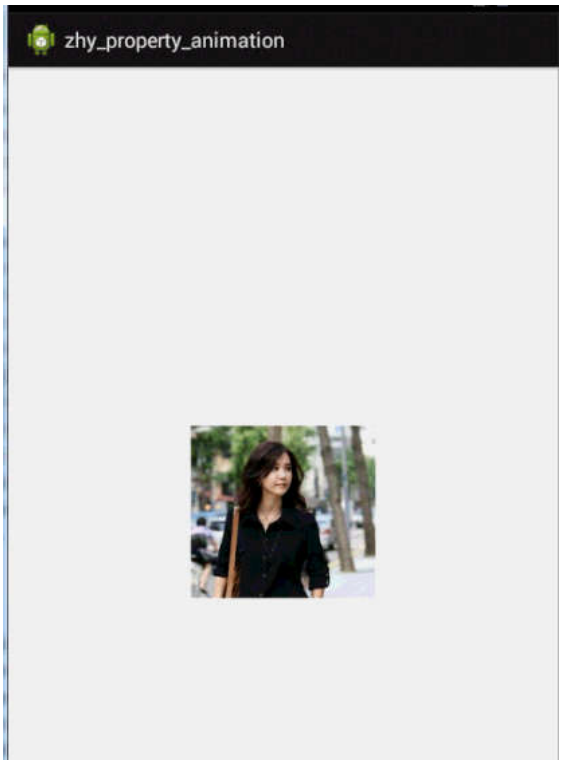
```
    android:id="@+id/id_container" >

    <ImageView
        android:id="@+id/id_ball"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:src="@drawable/mv"
        android:scaleType="centerCrop"
        android:onClick="rotateyAnimRun"
        />

</RelativeLayout>
```

很简单，就一张妹子图片~

Activity代码：

```java
package com.example.zhy_property_animation;

import android.animation.ObjectAnimator;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;

public class ObjectAnimActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.xml_for_anim);
    }

    public void rotateyAnimRun(View view)
    {
         ObjectAnimator//
         .ofFloat(view, "rotationX", 0.0F, 360.0F)//
         .setDuration(500)//
         .start();
    }

}
```

效果：



是不是一行代码就能实现简单的动画~~

对于ObjectAnimator

1、提供了ofInt、ofFloat、ofObject，这几个方法都是设置动画作用的元素、作用的属性、动画开始、结束、以及中间的任

当对于属性值，只设置一个的时候，会认为当然对象该属性的值为开始（getPropName反射获取），然后设置的值为终点。一个为开始、一个为结束~~~

动画更新的过程中，会不断调用setPropName更新元素的属性，所有使用ObjectAnimator更新某个属性，必须得有getter的时候）和setter方法~

2、如果你操作对象的该属性方法里面，比如上例的setRotationX如果内部没有调用view的重绘，则你需要自己按照下面方式

```java
anim.addUpdateListener(new AnimatorUpdateListener()
        {
            @Override
            public void onAnimationUpdate(ValueAnimator animation)
            {
//              view.postInvalidate();
//              view.invalidate();
            }
        });
```

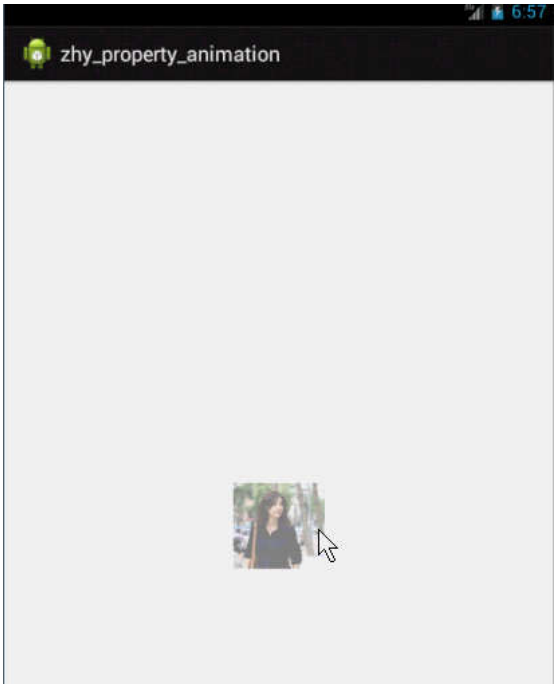
3、看了上面的例子，因为设置的操作的属性只有一个，那么如果我希望一个动画能够让View既可以缩小、又能够淡出（3个属scaleX,scaleY,alpha），只使用ObjectAnimator咋弄？

想法是不是很不错，可能会说使用AnimatorSet啊，这一看就是一堆动画塞一起执行，但是我偏偏要用一个ObjectAnimator看代码：

```java
public void rotateyAnimRun(final View view)
{
    ObjectAnimator anim = ObjectAnimator//
            .ofFloat(view, "zhy", 1.0F,  0.0F)//
            .setDuration(500);//
    anim.start();
    anim.addUpdateListener(new AnimatorUpdateListener()
    {
        @Override
        public void onAnimationUpdate(ValueAnimator animation)
        {
            float cVal = (Float) animation.getAnimatedValue();
            view.setAlpha(cVal);
            view.setScaleX(cVal);
            view.setScaleY(cVal);
        }
    });
}
```

把设置属性的那个字符串，随便写一个该对象没有的属性，就是不管~~咱们只需要它按照时间插值和持续时间计算的那个值用~

效果：



这个例子就是想说明一下，有时候换个思路不要被API所约束，利用部分API提供的功能也能实现好玩的效果~~~

比如：你想实现抛物线的效果，水平方向100px/s，垂直方向加速度200px/s*s ，咋实现呢~~可以自己用ObjectAnimator计

4、其实还有更简单的方式，实现一个动画更改多个效果：使用propertyValuesHolder

```java
public void propertyValuesHolder(View view)
    {
        PropertyValuesHolder pvhX = PropertyValuesHolder.ofFloat("alpha", 1f,
                0f, 1f);
        PropertyValuesHolder pvhY = PropertyValuesHolder.ofFloat("scaleX", 1f,
                0, 1f);
        PropertyValuesHolder pvhZ = PropertyValuesHolder.ofFloat("scaleY", 1f,
                0, 1f);
        ObjectAnimator.ofPropertyValuesHolder(view, pvhX, pvhY,pvhZ).setDuration(1000).start();
    }
```

4、ValueAnimator实现动画

和ObjectAnimator用法很类似，简单看一下用view垂直移动的动画代码：

```java
public void verticalRun(View view)
    {
        ValueAnimator animator = ValueAnimator.ofFloat(0, mScreenHeight
                - mBlueBall.getHeight());
        animator.setTarget(mBlueBall);
        animator.setDuration(1000).start();
    }
```

给你的感觉是不是，坑爹啊，这和ValueAnimator有毛线区别~但是仔细看，你看会发现，没有设置操作的属性~~也就是说

任何效果的，没有指定属性~

这就是和ValueAnimator的区别之处：ValueAnimator并没有在属性上做操作，你可能会问这样有啥好处？我岂不是还得手动

好处：不需要操作的对象的属性一定要有getter和setter方法，你可以自己根据当前动画的计算值，来操作任何属性，记得上

一个动画能够让View既可以缩小、又能够淡出（3个属性scaleX,scaleY,alpha）】吗？其实就是这么个用法~

实例：

布局文件：

```html
<RelativeLayout 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"
```

```
    android:id="@+id/id_container"

    >

    <ImageView
        android:id="@+id/id_ball"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/bol_blue" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:orientation="horizontal" >

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="verticalRun"
            android:text="垂直" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="paowuxian"
            android:text="抛物线" />

    </LinearLayout>

</RelativeLayout>
```

左上角一个小球，底部两个按钮~我们先看一个自由落体的代码：

```
[java]
/**
     * 自由落体
     * @param view
     */
    public void verticalRun( View view)
    {
        ValueAnimator animator = ValueAnimator.ofFloat(0, mScreenHeight
                - mBlueBall.getHeight());
        animator.setTarget(mBlueBall);
        animator.setDuration(1000).start();
//      animator.setInterpolator(value)
        animator.addUpdateListener(new AnimatorUpdateListener()
        {
            @Override
            public void onAnimationUpdate(ValueAnimator animation)
            {
                mBlueBall.setTranslationY((Float) animation.getAnimatedValue());
            }
        });
    }
```

与ObjectAnimator不同的就是我们自己设置元素属性的更新~虽然多了几行代码，但是貌似提高灵活性~

下面再来一个例子，如果我希望小球抛物线运动【实现抛物线的效果，水平方向100px/s，垂直方向加速度200px/s*s 】，分时间有关系，但是根据时间的变化，横向和纵向的移动速率是不同的，我们该咋实现呢？此时就要重写TypeValue的时候了，化的同时，需要返回给对象两个值，x当前位置，y当前位置：

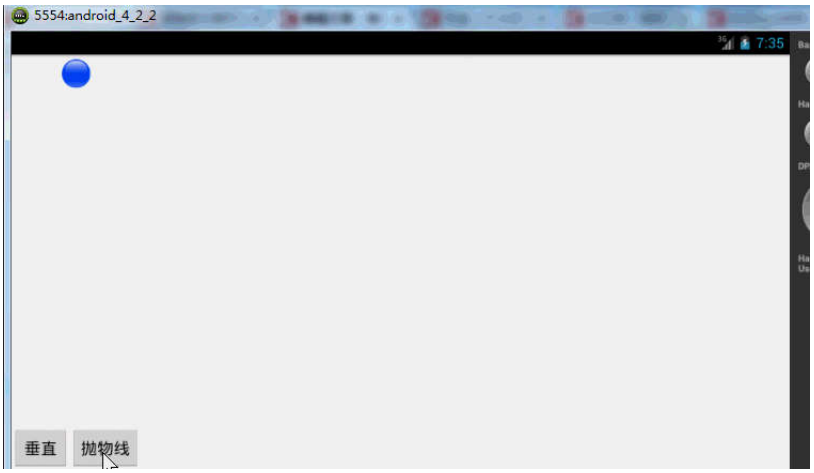
代码：

```
[java]
/**
     * 抛物线
     * @param view
     */
    public void paowuxian(View view)
    {

        ValueAnimator valueAnimator = new ValueAnimator();
        valueAnimator.setDuration(3000);
        valueAnimator.setObjectValues(new PointF(0, 0));
        valueAnimator.setInterpolator(new LinearInterpolator());
        valueAnimator.setEvaluator(new TypeEvaluator<PointF>()
        {
            // fraction = t / duration
            @Override
            public PointF evaluate(float fraction, PointF startValue,
```

```
                    PointF endValue)
            {
                Log.e(TAG, fraction * 3 + "");
                // x方向200px/s ，则y方向0.5 * 10 * t
                PointF point = new PointF();
                point.x = 200 * fraction * 3;
                point.y = 0.5f * 200 * (fraction * 3) * (fraction * 3);
                return point;
            }
        });

        valueAnimator.start();
        valueAnimator.addUpdateListener(new AnimatorUpdateListener()
        {
            @Override
            public void onAnimationUpdate(ValueAnimator animation)
            {
                PointF point = (PointF) animation.getAnimatedValue();
                mBlueBall.setX(point.x);
                mBlueBall.setY(point.y);

            }
        });
    }
```

可以看到，因为ofInt,ofFloat等无法使用，我们自定义了一个TypeValue，每次根据当前时间返回一个PointF对象，（Point是x,y的单位一个是float,一个是int;RectF,Rect也是）PointF中包含了x,y的当前位置～然后我们在监听器中获取，动态设置属

效果图：



有木有两个铁球同时落地的感觉～～对，我应该搞两个球～～ps:物理公式要是错了，就当没看见哈

自定义TypeEvaluator传入的泛型可以根据自己的需求，自己设计个Bean。

好了，我们已经分别讲解了ValueAnimator和ObjectAnimator实现动画；二者区别；如何利用部分API，自己更新属性实现

TypeEvaluator实现我们的需求；但是我们并没有讲如何设计插值，其实我觉得把，这个插值默认的那一串实现类够用了～～

个超级变态的~嗯~所以：略。

5、监听动画的事件

对于动画，一般都是一些辅助效果，比如我要删除个元素，我可能希望是个淡出的效果，但是最终还是要删掉，并不是你透明

位置，所以我们需要知道动画如何结束。

所以我们可以添加一个动画的监听:

```
[java]
public void fadeOut(View view)
    {
        ObjectAnimator anim = ObjectAnimator.ofFloat(mBlueBall, "alpha", 0.5f);

        anim.addListener(new AnimatorListener()
        {

            @Override
            public void onAnimationStart(Animator animation)
            {
                Log.e(TAG, "onAnimationStart");
            }

            @Override
            public void onAnimationRepeat(Animator animation)
            {
                // TODO Auto-generated method stub
                Log.e(TAG, "onAnimationRepeat");
```

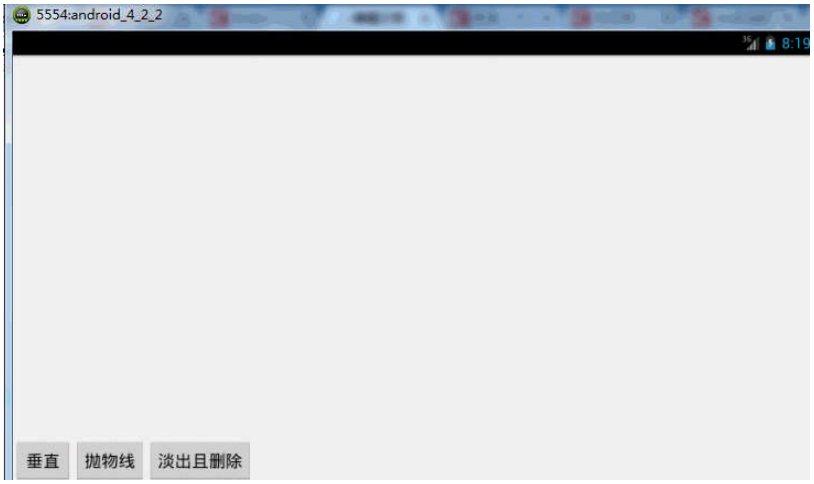
```
            }

            @Override
            public void onAnimationEnd(Animator animation)
            {
                Log.e(TAG, "onAnimationEnd");
                ViewGroup parent = (ViewGroup) mBlueBall.getParent();
                if (parent != null)
                    parent.removeView(mBlueBall);
            }

            @Override
            public void onAnimationCancel(Animator animation)
            {
                // TODO Auto-generated method stub
                Log.e(TAG, "onAnimationCancel");
            }
        });
        anim.start();
    }
```

这样就可以监听动画的开始、结束、被取消、重复等事件~但是有时候会觉得，我只要知道结束就行了，这么长的代码我不能用AnimatorListenerAdapter

```
[java]
anim.addListener(new AnimatorListenerAdapter()
{
    @Override
    public void onAnimationEnd(Animator animation)
    {
        Log.e(TAG, "onAnimationEnd");
        ViewGroup parent = (ViewGroup) mBlueBall.getParent();
        if (parent != null)
            parent.removeView(mBlueBall);
    }
});
```

AnimatorListenerAdapter继承了AnimatorListener接口，然后空实现了所有的方法~

效果图：



animator还有cancel()和end()方法：cancel动画立即停止，停在当前的位置；end动画直接到最终状态。

6、AnimatorSet的使用

实例：

布局文件:

```
[html]
<RelativeLayout 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"
    android:id="@+id/id_container"

    >

    <ImageView
        android:id="@+id/id_ball"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:src="@drawable/bol_blue" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:orientation="horizontal" >

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="togetherRun"
            android:text="简单的多动画Together" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="playWithAfter"
            android:text="多动画按次序执行" />


    </LinearLayout>

</RelativeLayout>
```

继续玩球~

代码：

```
[java]
package com.example.zhy_property_animation;

import android.animation.AnimatorSet;
import android.animation.ObjectAnimator;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.animation.LinearInterpolator;
import android.widget.ImageView;

public class AnimatorSetActivity extends Activity
{
    private ImageView mBlueBall;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.anim_set);

        mBlueBall = (ImageView) findViewById(R.id.id_ball);

    }

    public void togetherRun(View view)
    {
        ObjectAnimator anim1 = ObjectAnimator.ofFloat(mBlueBall, "scaleX",
                1.0f, 2f);
        ObjectAnimator anim2 = ObjectAnimator.ofFloat(mBlueBall, "scaleY",
                1.0f, 2f);
        AnimatorSet animSet = new AnimatorSet();
        animSet.setDuration(2000);
        animSet.setInterpolator(new LinearInterpolator());
        //两个动画同时执行
        animSet.playTogether(anim1, anim2);
        animSet.start();
    }

    public void playWithAfter(View view)
    {
        float cx = mBlueBall.getX();

        ObjectAnimator anim1 = ObjectAnimator.ofFloat(mBlueBall, "scaleX",
                1.0f, 2f);
        ObjectAnimator anim2 = ObjectAnimator.ofFloat(mBlueBall, "scaleY",
                1.0f, 2f);
        ObjectAnimator anim3 = ObjectAnimator.ofFloat(mBlueBall,
                "x",  cx ,  0f);
        ObjectAnimator anim4 = ObjectAnimator.ofFloat(mBlueBall,
                "x", cx);

        /**
         * anim1，anim2,anim3同时执行
```

```
54.          * anim4接着执行
55.          */
56.         AnimatorSet animSet = new AnimatorSet();
57.         animSet.play(anim1).with(anim2);
58.         animSet.play(anim2).with(anim3);
59.         animSet.play(anim4).after(anim3);
60.         animSet.setDuration(1000);
61.         animSet.start();
62.     }
63. }
```

写了两个效果：

第一：使用playTogether两个动画同时执行，当然还有playSequentially依次执行~~

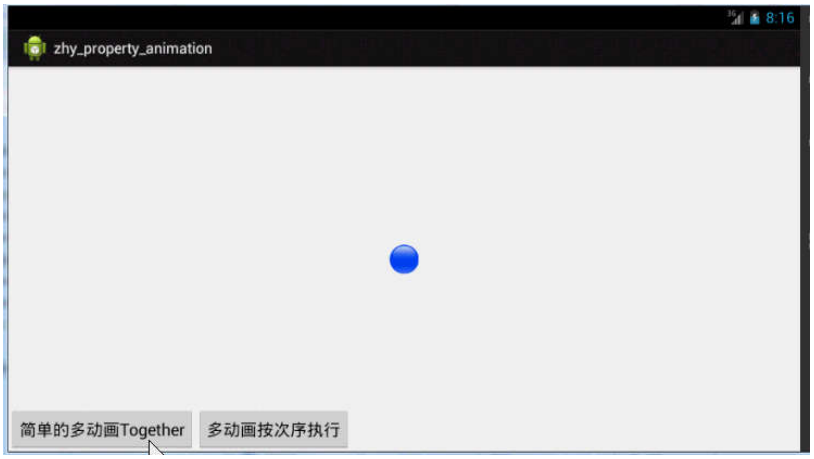
第二：如果我们有一堆动画，如何使用代码控制顺序，比如1，2同时；3在2后面；4在1之前等~就是效果2了

有一点注意：animSet.play().with();也是支持链式编程的，但是不要想着狂点，比如

animSet.play(anim1).with(anim2).before(anim3).before(anim5); 这样是不行的，系统不会根据你写的这一长串来决

以麻烦你按照上面例子的写法，多写几行：

效果图：



好了，由于篇幅~~关于属性动画还有点知识：

1、xml文件创建属性动画

2、布局动画

3、View的animate方法等。

那就考虑写到下一篇了，不过核心的功能就这些了~~

对了，如果使用11以下的SDK ，请导入nineoldandroids动画库，用法基本完全一致~

源码点击下载

- 上一篇 Android 自定义 ViewPager 打造千变万化的图片切换效果
- 下一篇 Android 属性动画（Property Animation） 完全解析 （下）

我的同类文章