巧用ViewPager 打造不一样的广告轮播切换效果

标签： viewpager android 广告轮播 banner
2016-05-09 08:47 1290人阅读 评论(18) 收藏 举报

分类： 【Android 精彩案例】（36）

目录(?) [+]

> 转载请标明出处：
> http://blog.csdn.net/lmj623565791/article/details/51339751；
> 本文出自:【张鸿洋的博客】

一、概述

如果大家关注了我的**微信**公众号的话，一定知道我在5月6号的时候推送了一篇文章，文章名为Android超高仿QQ附近的人搜索展示（一），通过该文可以利用ViewPager实现单页显示多个Item且能够添加一些炫酷的动画效果。我当时阅读这篇文章的时候，简单做了下记录，然后想了想，可以按照该思路做一个比较特殊轮播效果，如图：

其实看到这个大家肯定不陌生，对于ViewPager切换有个很出名的库叫JazzViewPager，没错，我又跑了下JazzyViewPager的例子，看看有什么动画效果可以借鉴的，ok，最终呢，产生以下几个效果图。

贴效果图前，简单说下我的公众号，恩，我是在上周决定正式开始好好打理的，目前很多东西都在尝试阶段，当然支持大家的投稿，目前存在一些文章过长，或者代码过长的排版问题，不过都在尝试改善与解决，以及对推送文章的选材都在考虑，所以多谢大家的支持，也欢迎大家的关注（二维码在侧栏），相信我一定会做的更好。

此外，针对不好阅读的问题，大家可以通过该仓库，看到所有推送文章的一个列表，https://github.com/hongyangAndroid/hongyangWeixinArticles该仓库会和公众号推送的文章同步更新。

下面进入正题，本文主要是利用ViewPager做类似上图风格的Banner，这种Banner在app上不是很常见，不过在web端还有tv的app上还是很常见的。

不过原理很简单，说到核心，就两个地方:

- `android:clipChildren="false"`
- `viewPager.setPageTransformer`

很久之前也写过类似的文章，可以参考

- Android 自定义 ViewPager 打造千变万化的图片切换效果
- Android 实现个性的ViewPager切换动画 实战PageTransformer（兼容Android3.0以下）

二、效果图

- Rotate Down

- Rotate Up

- ScaleIn

贴三个意思下，恩，更多效果见https://github.com/hongyangAndroid/MagicViewPager.

三、ViewPager一屏显示多个页面

ok，首先说明下控件，上述效果采用的控件是 ViewPager ，大家都清楚哇，使用 ViewPager 一般我们都是一屏幕显示一个页面，那么如何做到一屏显示多个页面呢？

> ViewPager如何做到一屏显示多个页面呢？

原理就一个属性 Android:clipChildren="false",该属性的意思就是在子View进行绘制时不要去裁切它们的显示范围。ok，知道要使用这个属性之后，剩下的事情就不麻烦了：

我们的布局文件这么写：

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="160dp"
    android:clipChildren="false"
    android:layout_centerInParent="true"
    android:background="#aadc71ff"
    >
    <android.support.v4.view.ViewPager
        android:id="@+id/id_viewpager"
        android:layout_width="match_parent"
        android:layout_marginLeft="60dp"
        android:layout_marginRight="60dp"
        android:clipChildren="false"
        android:layout_height="120dp"
        android:layout_gravity="center"
        >
    </android.support.v4.view.ViewPager>

```

```
19 </FrameLayout>
```

我们设置了 ViewPager 外层控件以及 ViewPager 都设置了 android:clipChildren="false" 。

我们的ViewPager的宽度是 match_parent ，左后个设置了 60dp 的边距，就是为了显示出左右部分的Page.

接下来可以对 ViewPager 设置Adapter等相关属性。

```
1  public class MainActivity extends AppCompatActivity
2  {
3
4      private ViewPager mViewPager;
5      private PagerAdapter mAdapter;
6
7      int[] imgRes = {R.drawable.a, R.drawable.b, R.drawable.c...};
8
9      @Override
10     protected void onCreate(Bundle savedInstanceState)
11     {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14
15         mViewPager = (ViewPager) findViewById(R.id.id_viewpager);
16          //设置Page间间距
17         mViewPager.setPageMargin(20);
18         //设置缓存的页面数量
19         mViewPager.setOffscreenPageLimit(3);
20         mViewPager.setAdapter(mAdapter = new PagerAdapter()
21         {
22             @Override
23             public Object instantiateItem(ViewGroup container, int position)
24             {
25                 ImageView view = new ImageView(MainActivity.this);
26                 view.setImageResource(imgRes[position]);
27                 container.addView(view);
28                 return view;
29             }
30
31             @Override
32             public void destroyItem(ViewGroup container, int position, Object object)
33             {
34                 container.removeView((View) object);
35             }
36
37             @Override
38             public int getCount()
39             {
```

```
                return imgRes.length;
            }

            @Override
            public boolean isViewFromObject(View view, Object o)
            {
                return view == o;
            }
        });

    }
```

ok，没有任何复杂的地方，注意

```
 //设置Page间间距
mViewPager.setPageMargin(20);
```

以及

```
//设置缓存的页面数量
 mViewPager.setOffscreenPageLimit(3);
```

我们这里最多可见就是3页。

此时运行：

可以看到，我们已经实现了单屏幕显示出多个page，而且是ViewPager所以肯定可以左右滑动。

这么看，是不是非常简单，接下来就是加特效了，大家都清楚对于ViewPager可以通过设置 PageTransformer 来利用属性动画来设置特效，注意目前该方法添加的动画在3.0即以上的手机中有效，因为3.0以下并不存在属性动画，所以 setPageTransformer 内部加了个判断，不过现在已经几乎没有3.0以下的手机了，但是如果你非要较真，参考文章开始时给出的两篇文章，里面有解决方案。

四、给ViewPager加特效

这里我们简单抽取两个动画效果来讲，其实以前的文章里面也有详细的描述，所以不准备花费太多的时间描述。

（1）AlphaPageTransformer

首先讲个最简单的动画，叫 AlphaPageTransformer ，顾名思义就是一个渐变的变化，那么我们的步骤是这样的：

- 实现 AlphaPageTransformer implements ViewPager.PageTransformer
- 调用 viewPager.setPageTransformer(new AlphaPageTransformer())

对于 ViewPager.PageTransformer 就一个方法需要实现

```
#AlphaPageTransformer

private static final float DEFAULT_MIN_ALPHA = 0.5f;
private float mMinAlpha = DEFAULT_MIN_ALPHA;

public void pageTransform(View view, float position)
{
    if (position < -1)
    {
        view.setAlpha(mMinAlpha);
    } else if (position <= 1)
    { // [-1,1]

        if (position < 0) //[0, -1]
        {
            float factor = mMinAlpha + (1 - mMinAlpha) * (1 + position);
            view.setAlpha(factor);
        } else//[1, 0]
        {
            float factor = mMinAlpha + (1 - mMinAlpha) * (1 - position);
            view.setAlpha(factor);
        }
    } else
    { // (1,+Infinity]
        view.setAlpha(mMinAlpha);
    }
}
```

代码非常简短，简单的介绍下，可以看到postion主要分为

- [-Infinity,-1)
- (1,+Infinity]
- [-1,1]

这三个区间，对于前两个，拿我们的页面上目前显示的3个Page来说，前两个分别对应左右两个露出一点的Page，那么对于alpha值，只需要设置为最小值即可。

对于 [-1,1] ，这个就需要详细分析了，我们这里拿：第一页->第二页这个过程来说，主要看position的变化

第1页->第2页

- 页1的postion变化为：从0到-1
- 页2的postion变化为：从-1到0

第一页到第二页，实际上就是左滑，第一页到左边，第二页成为currentItem到达中间，那么对应alpha的变化应该是：

- 页1到左边，对应alpha应该是：1到minAlpha
- 页2到中间，成为currentItem，对应alpha应该是：minAlpha到1

分析到这就是写代码了：

对于页1

```
//注意该代码判断在(position <= 1)的条件内
if (position < 0) //[0, -1]
{
    float factor = mMinAlpha + (1 - mMinAlpha) * (1 + position);
    view.setAlpha(factor);
}
```

position是0到-1的变化

那么 1+position 就是从1到0的变化

(1 - mMinAlpha) * (1 + position) 就是 1 - mMinAlpha 到0的变化

再加上一个mMinAlpha，就变为1到mMinAlpha的变化。

其实绕来绕去就是为了实现factor是1到minAlpha的变化，具体这样的算式，每个人的思路可能不同，但是达到相同的效果即可。

同理，页2是minAlpha到1的变化。

对应算式(postion为1到0变化)

```
1  float factor = mMinAlpha + (1 - mMinAlpha) * (1 - position);
```

这个留给大家自己算，或者自己去总结出一个相同结果的算式。

ok，当我们完成 AlphaPageTransformer 的编码，然后ViewPager设置后，效果就是这样的：

（2）RotateDownPageTransformer

再介绍个 RotateDownPageTransformer ，因为这个涉及到旋转中心的变化，即：

```
1  view.setPivotX();
2  view.setPivotY();
```

直接看代码:

```
1  private static final float DEFAULT_MAX_ROTATE = 15.0f;
2  private float mMaxRotate = DEFAULT_MAX_ROTATE;
3
4  public void pageTransform(View view, float position)
5  {
6      if (position < -1)
7      { // [-Infinity,-1)
8          // This page is way off-screen to the left.
9          view.setRotation(mMaxRotate * -1);
```

```
        view.setPivotX(view.getWidth());
        view.setPivotY(view.getHeight());

    } else if (position <= 1)
    { // [-1,1]

        if (position < 0)//[0, -1]
        {
            view.setPivotX(view.getWidth() * (0.5f + 0.5f * (-position)));
            view.setPivotY(view.getHeight());
            view.setRotation(mMaxRotate * position);
        } else//[1,0]
        {
            view.setPivotX(view.getWidth() * 0.5f * (1 - position));
            view.setPivotY(view.getHeight());
            view.setRotation(mMaxRotate * position);
        }
    } else
    { // (1,+Infinity]
        // This page is way off-screen to the right.
        view.setRotation(mMaxRotate);
        view.setPivotX(view.getWidth() * 0);
        view.setPivotY(view.getHeight());
    }
}
```

经过上面的分析，我们直接锁定到第一页到第二页时，第一页的相关变化的代码：

```
if (position < 0)//[0, -1]
{
    float factor = view.getWidth() * (0.5f + 0.5f * (-position));
    view.setPivotX(factor);
    view.setPivotY(view.getHeight());
    view.setRotation(mMaxRotate * position);
}
```

第一页开始时滑动时，旋转中心上图原点，即（width/2，height）.

第一页滑动结束时，旋转中心在左边页面的右下角，即（width,height）.

恩，这个旋转中心的位置是我自己定义的，不一定是最好的效果，如果有必要大家可以自己选择，保证良好的显示效果。

可以看到旋转中心的纵坐标没有发生变化，主要看横坐标

```
1  float factor = view.getWidth() * (0.5f + 0.5f * (-position));
```

position为0到-1，乘以-0.5之后，变为0到0.5

在加上0.5，变为0.5到1的变化

再乘以width，即变为width/2到width的变化。

对应我们的旋转中心x是需要从width/2到width，是不是刚好匹配。

旋转中心的变化说明白了；再简单说下，角度的变化，第一页到达左边页面的状态，角度是-15度，开始状态是0度，那么变化就是0到-15度之间，因为position是0到-1之间变化，所以直接乘以15即可

```
1  float rotation = position * 15f
```

好了，经过上面的分析，本文就基本结束了，有兴趣可以下载源码多分析几个，或者创造几个动画效果，千万不要忘了告诉我，我可以加入到这个动画库中。

五、总结

本文的内容其实涉及到的API实际上比较多，再多的动画其实质性的原理都是一样的，关键在于找规律，所以带大家梳理一下步骤：

1. 确定View需要变化的属性
2. 确定该属性的初始值，终值
3. 确定该View对应的position变化的梯度
4. 根据position的变化梯度，计算出需要变化的属性的变化梯度
5. 剩下的就是调用属性动画的API了

ok，相信通过该步骤大家一定能够自己去定义出形态各异的动画，此外，切记如果学习，一定要尝试编写，看一看就认为了解的，可能有些坑是发现不了的。

源码地址：https://github.com/hongyangAndroid/MagicViewPager