**Android 仿知乎创意广告 广告还能这么玩？**

2017年12月05日 09:45:02 [鸿洋\_](https://me.csdn.net/lmj623565791) 阅读数：20076 标签： [android](http://so.csdn.net/so/search/s.do?q=android&t=blog)[view](http://so.csdn.net/so/search/s.do?q=view&t=blog) 更多

个人分类： [【Android 精彩案例】](https://blog.csdn.net/lmj623565791/article/category/2680597)

 版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/lmj623565791/article/details/78714705

本文已在我的公众号[hongyangAndroid](https://github.com/hongyangAndroid/hongyangWeixinArticles)原创首发。   
转载请标明出处：   
<http://blog.csdn.net/lmj623565791/article/details/78714705>   
本文出自张鸿洋的博客

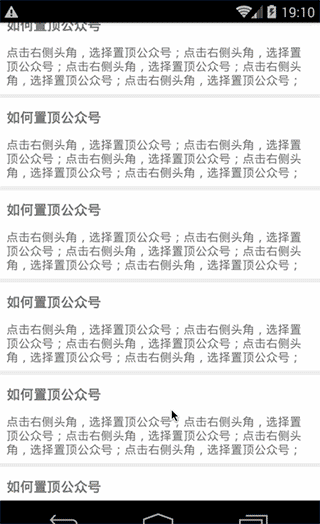
**一、概述**

貌似前段时间刷知乎看到的一种非常有特色的广告展现方式，即在列表页，某一个Item显示背后部分广告图，随着列表滚动，会逐渐展示全部图片。

刚看到的时候就想实现一哈，一直比较懒，公众号后台也有人问如何实现，今天来给大家讲解下，当然了，目前一些自定义View已经不算难题，所以本文的讲解会做一些实现思路引导，相信不会是那么枯燥的文章，希望对大家有一定的帮助。

恩，现在知乎上已经找不到该效果了，试了多个历史版本也没找到，那只能贴实现的效果图了~

效果图如下：





2选1，你喜欢哪个效果图呢~~

**二、思路**

好了，抛开别的，确定下本文的目标：

实现在列表中展示某张图片：

往上滚动：在图片刚出现时展示顶部部分，随着滚动部分展示全部   
往下滚动：在图片刚出现时展示底部部分，随着滚动部分展示全部

换句话说，我们需要在列表滚动时，改变图片显示的部分。

两个点：

* 捕获列表滚动的dy，不管是ListView还是RecyclerView相信这一点都能做到
* 图片显示部分变化，我们可以利用canvas.translate

结合一下，就是，监听列表的滚动dy，传给我们的图片控件，设置translate，然后绘制。

到这里，思路非常清晰，这个东西肯定能做了。

初步方案：自定义一个View，自己去绘制bitmap，对外暴露setDy(dy)，然后根据dy做canvas偏移重绘即可。

有了初步方案，基本不慌了，那么再想想？

能否利用已有的控件，比如ImageView呢？

肯定可以，这样省去了我们去声明一个接受图片的属性，我们编写一个子类，依然是通过设置src去使用。

那继承ImageView实现一波再说。

开始码代码前，力推下我的公众号：



专注于Android的相关技术~

**三、实现**

首先我们先写个假的列表，鉴于RV用的越来越多，就用RecyclerView吧。

**布局**

主布局文件，一个RecyclerView即可：

<?xml version="1.0" encoding="utf-8"?>

<android.support.v7.widget.RecyclerView

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

android:id="@+id/id\_recyclerview"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

/>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9

item布局文件：

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="@drawable/item\_bg"

android:gravity="center">

<com.imooc.rvimageads.AdImageViewVersion1

android:id="@+id/id\_iv\_ad"

android:layout\_width="match\_parent"

android:layout\_height="180dp"

android:scaleType="matrix"

android:src="@mipmap/grsm"

android:visibility="gone" />

<TextView

android:layout\_margin="12dp"

android:id="@+id/id\_tv\_title"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="这是title"

android:textSize="16dp"

android:textStyle="bold" />

<TextView

android:id="@+id/id\_tv\_desc"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/id\_tv\_title"

android:layout\_marginLeft="12dp"

android:layout\_marginRight="12dp"

android:layout\_marginBottom="12dp"

android:text="这是描述" />

</RelativeLayout>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36

很简单，先不用管AdImageViewVersion1类，这将是我们具体的实现类。   
通过布局文件，可以看到，我们只使用了一个item布局文件，然后通过visible,gone控制展示不同形态。

**Activity**

public class MainActivity extends AppCompatActivity {

private RecyclerView mRecyclerView;

private LinearLayoutManager mLinearLayoutManager;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mRecyclerView = findViewById(R.id.id\_recyclerview);

List<String> mockDatas = new ArrayList<>();

for (int i = 0; i < 100; i++) {

mockDatas.add(i + "");

}

mRecyclerView.setLayoutManager(mLinearLayoutManager = new LinearLayoutManager(this));

mRecyclerView.setAdapter(new CommonAdapter<String>(MainActivity.this,

R.layout.item,

mockDatas) {

@Override

protected void convert(ViewHolder holder, String o, int position) {

if (position > 0 && position % 6 == 0) {

holder.setVisible(R.id.id\_tv\_title, false);

holder.setVisible(R.id.id\_tv\_desc, false);

holder.setVisible(R.id.id\_iv\_ad, true);

} else {

holder.setVisible(R.id.id\_tv\_title, true);

holder.setVisible(R.id.id\_tv\_desc, true);

holder.setVisible(R.id.id\_iv\_ad, false);

}

}

});

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38

仅仅是设置数据了，Adapter这里用了

compile 'com.zhy:base-rvadapter:3.0.3'

* 1

你可以随便用一个你自己喜欢的Adapter封装类。

到这里，一个列表页就显示出来了，并且每隔6个会显示成图片。

不截图了，脑补下…

现在才正式开始实现。

**自定义AdImageView**

public class AdImageViewVersion1 extends AppCompatImageView {

public AdImageViewVersion1(Context context, @Nullable AttributeSet attrs) {

super(context, attrs);

}

private RectF mBitmapRectF;

private Bitmap mBitmap;

private int mMinDy;

@Override

protected void onSizeChanged(int w, int h, int oldw, int oldh) {

super.onSizeChanged(w, h, oldw, oldh);

mMinDy = h;

Drawable drawable = getDrawable();

if (drawable == null) {

return;

}

mBitmap = drawableToBitamp(drawable);

mBitmapRectF = new RectF(0, 0,

w,

mBitmap.getHeight() \* w / mBitmap.getWidth());

}

private Bitmap drawableToBitamp(Drawable drawable) {

if (drawable instanceof BitmapDrawable) {

BitmapDrawable bd = (BitmapDrawable) drawable;

return bd.getBitmap();

}

int w = drawable.getIntrinsicWidth();

int h = drawable.getIntrinsicHeight();

Bitmap bitmap = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB\_8888);

Canvas canvas = new Canvas(bitmap);

drawable.setBounds(0, 0, w, h);

drawable.draw(canvas);

return bitmap;

}

*// ... 省略一些代码*

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45

因为我们要绘制，所以这里我们把drawable转成bitmap，然后我们默认要显示最底部，所以需要一个最小的偏移，即控件高度。

这些事情，我们都在onSizeChanged做了。

并且我们根据当前控件宽度，对bitmap进行了缩放，并将缩放后的尺寸存在了mBitmapRectF中，以便于绘制。

那么接下来就是绘制了，还记得绘制过程中，我们主要利用translate来控制绘制的区域，所以我们还要对外暴露一个setDy方法，so，我们的代码大致是这样的：

private int mDy;

public void setDy(int dy) {

if (getDrawable() == null) {

return;

}

mDy = dy - mMinDy;

if (mDy <= 0) {

mDy = 0;

}

if (mDy > mBitmapRectF.height() - mMinDy) {

mDy = (int) (mBitmapRectF.height() - mMinDy);

}

invalidate();

}

@Override

protected void onDraw(Canvas canvas) {

if (mBitmap == null) {

return;

}

canvas.save();

canvas.translate(0, -mDy);

canvas.drawBitmap(mBitmap, null, mBitmapRectF, null);

canvas.restore();

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27

setDy的时候，我们做了一个边界判断，最小的情况，我们偏移-mMinDy，显示图片的底部。   
最大的时候，我们便宜图片高度-mMinDy，显示顶部部分。

所以我们对传入的值做了最小与最大值判断。

那么在绘制的时候，就简单了，先translate dy距离，然后绘制即可。

到这里我们的自定义View部分就结束了，代码很少~

**结合RecyclerView**

接下来就是在RecyclerView滚动时，给我们传入dy即可。

mRecyclerView.addOnScrollListener(new RecyclerView.OnScrollListener() {

@Override

public void onScrolled(RecyclerView recyclerView, int dx, int dy) {

super.onScrolled(recyclerView, dx, dy);

int fPos = mLinearLayoutManager.findFirstVisibleItemPosition();

int lPos = mLinearLayoutManager.findLastCompletelyVisibleItemPosition();

for (int i = fPos; i <= lPos; i++) {

View view = mLinearLayoutManager.findViewByPosition(i);

AdImageViewVersion1 adImageView = view.findViewById(R.id.id\_iv\_ad);

if (adImageView.getVisibility() == View.VISIBLE) {

adImageView.setDy(mLinearLayoutManager.getHeight() - view.getTop());

}

}

}

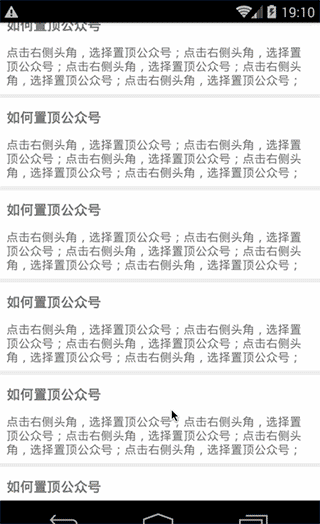
});

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16

通过addOnScrollListener监听，当滚动时，拿到所有可见的Item，找出正在显示图片的Item。然后调用setDy，dy的值为mLinearLayoutManager.getHeight() - view.getTop()，当View从最底部出现的时候为0，当View到达最顶部的时候为当前rv的高度。

你可以合理的利用setDy传入的值，做移动差，显示区域从上到下等，都可以。

这样就完成了~~



一句话实现：即滚动时不断改变dy，然后translate绘制即可。

**四、再想想**

看着这个代码，好像drawableToBitamp看起来非常不爽，也是比较耗内存的部分。我们再想想：

**本身Drawable就是能绘制的，为什么我们要转成bitmap呢？**

好像有道理，ImageView本身绘制的就是Drawable，我们需要控制的就是这个Drawable的绘制范围要足够大，不能被控件本身的宽高所影响，导致图片被压扁。

好像有那么一个方法：

drawable.setBounds()*;*

* 1

那就简单了，去除drawable2bitmap的代码，直接利用原本的绘制即可，我们唯一要做的就是设置bounds，做一个translate dy即可。

完整代码：

public class AdImageView extends AppCompatImageView {

*// 删除构造方法*

private int mDx;

private int mMinDx;

public void setDx(int dx) {

if (getDrawable() == null) {

return;

}

mDx = dx - mMinDx;

if (mDx <= 0) {

mDx = 0;

}

if (mDx > getDrawable().getBounds().height() - mMinDx) {

mDx = getDrawable().getBounds().height() - mMinDx;

}

invalidate();

}

@Override

protected void onSizeChanged(int w, int h, int oldw, int oldh) {

super.onSizeChanged(w, h, oldw, oldh);

mMinDx = h;

}

public int getDx() {

return mDx;

}

@Override

protected void onDraw(Canvas canvas) {

Drawable drawable = getDrawable();

int w = getWidth();

int h = (int) (getWidth() \* 1.0f / drawable.getIntrinsicWidth() \* drawable.getIntrinsicHeight());

drawable.setBounds(0, 0, w, h);

canvas.save();

canvas.translate(0, -getDx());

super.onDraw(canvas);

canvas.restore();

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43

短短的代码就实现了，这样看起来顺眼多了~~

再贴下效果图：



效果图主要看字，你懂的！

好了，本篇总结：

看到当看一个效果，可以先对它进行拆分，找出关键点，针对每个关键点，考虑可行性。

如果确定每个点都可行，那么基本的方案就出来了。

有了基本的方案，不要着急写，再想想还有无改善空间。

例子比较简单，have a nice day ~~

<https://github.com/hongyangAndroid/demo_rvadimage>