Android Scaletype源码分析及自定义Matrix缩放规则

Feb 28, 2017 | 技术 | Hits 0 Comments

在上一篇中，我详细的总结了ImageView各种ScaleType对应的缩放效果。对于Matrix这种强大的缩放方式并没有详细介绍，在这一篇中会主要说一下这种缩放方式。
让我们先从源码中看一下上一篇中介绍的几种缩放方式都是怎么实现的，然后再通过Matrix的方式来实现一下上一篇最后没有实现的3种自定义缩放规则(宽度撑满并居下、高度撑满居左、高度撑满居右)

源码分析

打开ImageView的源码，找到如下方法：

```
@Override
protected boolean setFrame(int l, int t, int r, int b) {
    final boolean changed = super.setFrame(l, t, r, b);
    mHaveFrame = true;
    configureBounds();
    return changed;
}
```

其中有一个configureBounds()方法。我们需要的代码就在这个方法里面。首先看一下**FIT_XY**这种方式是如何实现的

1. FIT_XY源码实现

```
if (dwidth <= 0 || dheight <= 0 || ScaleType.FIT_XY == mScaleType) {
    // If the drawable has no intrinsic size, or we're told to
    //  scaletofit, then we just fill our entire view.
    mDrawable.setBounds(0, 0, vwidth, vheight);
```

```
5       mDrawMatrix = null;
6     }
```

上述代码中的 `vwidth与vheight` 是ImageView控件大小减去padding后剩下的值即可容纳内容的最大空间。可以看到这种缩放方式非常简单粗暴，直接通过改变Drawable的绘制区域来达到，并未进行任何的Matrix操作。

2. FIT_START/FIT_CENTER/FIT_END源码实现

之所以把它们3个的放在一起说，是因为它们的实现逻辑是一样,在configureBounds()方法中有如下代码如下：

```
1     // Generate the required transform.
2     mTempSrc.set(0, 0, dwidth, dheight);
3     mTempDst.set(0, 0, vwidth, vheight);
4     mDrawMatrix = mMatrix;
5     mDrawMatrix.setRectToRect(mTempSrc, mTempDst, scaleTypeToScaleToFit(mScaleType));
```

可以看到这里用到了 `setRectToRect` 方法，这个方法可以根据变换前后的两个矩形得到对应的变换矩阵。上面代码中mTempSrc即是内容矩形大小也是变换的源矩形大小，mTempDst是ImageView控件矩形大小也是目标矩形大小。
这个方法还有一个参数 Matrix.ScaleToFit 它用来指定缩放选项,一共有4个选项：

> 1. FILL: 独立的缩放X,Y，所以这个选项可能会改变图片的长宽比（其实效果与FIT_XY一样）
> 2. START: 这种缩放方式的效果对应ImageView的FIT_START
> 3. CENTER: 这种缩放方式的效果对应ImageView的FIT_CENTER
> 4. END: 这种缩放方式的效果对应ImageView的FIT_END

也即是ImageView的这3种缩放规则直接使用的Matrix内置的这几种变换规则实现。你也可以自己用如下代码验证一下：

```
1     @Override
2         protected boolean setFrame(int l, int t, int r, int b) {
3             boolean changed = super.setFrame(l, t, r, b);
4             test();
5             return changed;
6         }
7
8         private void test() {
9             final int dw= getDrawable().getIntrinsicWidth();
```

```
            final int dh= getDrawable().getIntrinsicHeight();

            final float vw= getWidth() - getPaddingLeft() - getPaddingRight();
            final float vh= getHeight() - getPaddingTop() - getPaddingBottom();

            mTempSrc.set(0, 0, dw, dh);
            mTempDst.set(0, 0, vw, vh);
            Log.e("ghui", "mTempSrc: " + mTempSrc.toShortString());
            Log.e("ghui", "mTempDst: " + mTempDst.toShortString());
            Matrix matrix = getImageMatrix();
            matrix.reset();
            Log.e("ghui", "before: " + matrix.toShortString());
            matrix.setRectToRect(mTempSrc, mTempDst, Matrix.ScaleToFit.START);
            Log.e("ghui", "after: " + matrix.toShortString());
            setImageMatrix(matrix);
        }
```

这里不再赘述。

3.CENTER

CENTER的源码实现对应如下代码:

```
else if (ScaleType.CENTER == mScaleType) {
// Center bitmap in view, no scaling.
  mDrawMatrix = mMatrix;
  mDrawMatrix.setTranslate(Math.round((vwidth - dwidth) * 0.5f),
  Math.round((vheight - dheight) * 0.5f));
}
```

可以看到这里是直接通过改变Matrix的Translate将内容从左上角移动到了控件的中心，使内容的中心与控件的中心对齐。并未做任何的缩放操作，
和我们上一篇中总结的效果一致。

4.CENTER_CROP

这种方式的对应的源码如下:

```
else if (ScaleType.CENTER_CROP == mScaleType) {
  mDrawMatrix = mMatrix;

  float scale;
  float dx = 0, dy = 0;

```

```
    if (dwidth * vheight > vwidth * dheight) {
        scale = (float) vheight / (float) dheight;
        dx = (vwidth - dwidth * scale) * 0.5f;
    } else {
        scale = (float) vwidth / (float) dwidth;
        dy = (vheight - dheight * scale) * 0.5f;
    }

    mDrawMatrix.setScale(scale, scale);
    mDrawMatrix.postTranslate(Math.round(dx), Math.round(dy));
  }
```

这里会首先比较内容的宽高比与控件的宽高比的大小，若内容的宽高比更大（内容更宽）则以控件高度与内容高度的比作为缩放比例。反之则会将控件宽度与内容宽度的比作为缩放的比例。这样实现的效果即是将内容中更小的一者（宽或高）缩放到与控件对应的一方一样。也与上一篇中总结的效果一样。

接着会通过postTranslate方法将内容移动到控件中心，计算方法与CENTER中的一样。

5. CENTER_INSID

```
  else if (ScaleType.CENTER_INSIDE == mScaleType) {
    mDrawMatrix = mMatrix;
    float scale;
    float dx;
    float dy;

    if (dwidth <= vwidth && dheight <= vheight) {
        scale = 1.0f;
    } else {
        scale = Math.min((float) vwidth / (float) dwidth,
                (float) vheight / (float) dheight);
    }

    dx = Math.round((vwidth - dwidth * scale) * 0.5f);
    dy = Math.round((vheight - dheight * scale) * 0.5f);

    mDrawMatrix.setScale(scale, scale);
    mDrawMatrix.postTranslate(dx, dy);
  }
```

可以看到若内容的宽高都小于控件的宽高，这种方式的Scale为1，即不会去缩放。否则的话则会去选择一个更小的缩放比例（控件的宽高与内容的宽高比，取其中较小的一个）

效果也与上一篇中总结的一致。

Matrix-实现自定义的缩放效果。

其实通过学习上面的源码，自定义实现 宽度撑满并居上、宽度撑满并居下、高度撑满居左、高度撑满居右这几种效果也是很简单的。先把代码贴出来：

```
@Override
    protected boolean setFrame(int l, int t, int r, int b) {
        boolean changed = super.setFrame(l, t, r, b);
        if (getScaleType() == ScaleType.MATRIX) transformMatrix();
        return changed;
    }

    private void transformMatrix() {
        Matrix matrix = getImageMatrix();
        matrix.reset();
        float h = getHeight();
        float w = getWidth();
        float ch = getDrawable().getIntrinsicHeight();
        float cw = getDrawable().getIntrinsicWidth();
        float widthScaleFactor = w / cw;
        float heightScaleFactor = h / ch;
        if (alignType == AlignType.LEFT) {
            matrix.postScale(heightScaleFactor, heightScaleFactor, 0, 0);
        } else if (alignType == AlignType.RIGHT) {
            matrix.postTranslate(w - cw, 0);
            matrix.postScale(heightScaleFactor, heightScaleFactor, w, 0);
        } else if (alignType == AlignType.BOTTOM) {
            matrix.postTranslate(0, h - ch);
            matrix.postScale(widthScaleFactor, widthScaleFactor, 0, h);
        } else { //default is top
            matrix.postScale(widthScaleFactor, widthScaleFactor, 0, 0);
        }
        setImageMatrix(matrix);
    }

    public enum AlignType {
        LEFT(0),
        TOP(1),
        RIGHT(2),
        BOTTOM(3);
        AlignType(int ni) {
            nativeInt = ni;
        }
        final int nativeInt;
    }
```

首先看第一种缩放方式：

1.高度撑满居左

因为ImageView的内容默认会居左上，所以这里可以直接对matrix执行缩放操作，将高度缩放到撑满控件。如下图所示:

2.高度撑满居右

因为要居右，所以首先将内容移动到右上角，然后再以右上角为中心执行缩放操作。如下图所示：

3.宽度撑满并居下

因为要居下，所以首先将内容移动到左下角，然后再以左下角为中心执行缩放操作。

4.宽度撑满并居上

这种方式更简单，因为内容默认在左上角，所以这里可以直接执行缩放操作即可。

源码

ImageScaleTutorial

版权声明

> 文章版权归本人所有，如需转载需在明显位置处保留作者信息及原文链接！

Android 分享到

◂ V2er免费版发布啦 对于Android中ImageView的ScaleType你的理解可能是错的 ▸

0条评论 ghui.me 登录

推荐 分享 评分最高

开始讨论...

通过以下方式登录

或注册一个 DISQUS 帐号 ?

姓名

来做第一个留言的人吧！

在 GHUI.ME 上还有

15款你可能不知道的精致Mac应用

2条评论 • 2年前

ghui Zhang — 赞, 很实用.

iShanbay(如果你也是一个在用Alfred的扇贝网用户)

7条评论 • 2年前

ghui Zhang — 你点击授权后，回跳转到这个页面：https://ghui.me/post/2017/0... ...

留言

1条评论 • 2年前

ghui Zhang — □□□测北京联通不fq没事

About

1条评论 • 2年前

屠夫9441 — 原来你是懒投资的员工啊~我可是懒投资的忠实用户.....

订阅 在您的网站上使用 Disqus添加 Disqus添加 Disqus 隐私政策隐私政策隐私

Copyright © 2017 ghui | Visitor: