Android自定义控件之基本原理

前言：

在日常的Android开发中会经常和控件打交道，有时Android提供的控件未必能满足业务的需求，这个时候就需要我们实现自定义一些控件，今天先大致了解一下自定义控件的要求和实现的基本原理。

自定义控件相关文章地址：

Android自定义控件之基本原理

Android自定义控件之自定义属性

Android自定义控件之自定义组合控件

Android自定义控件之自定义ViewGroup实现标签云

自定义控件要求：

1. 应当遵守Android标准的规范(命名，可配置，事件处理等)。

2. 在XML布局中可配置控件的属性。

3. 对交互应当有合适的反馈，比如按下，点击等。

4. 具有兼容性， Android版本很多，应该具有广泛的适用性。

自定义控件学习步骤：

　　1 .View的工作原理

　　2 .编写View类

　　3.为View类增加属性

　　4 .绘制屏幕

　　5. 响应用户消息

　　6 .自定义回调函数

自定义控件两种方式：

　　1. 继承ViewGroup

例如：ViewGroup、LinearLayout、FrameLayout、RelativeLayout等。

　　2. 继承View

例如：View、TextView、ImageView、Button等。

自定义控件基本绘制原理：

View的绘制基本上由measure()、layout()、draw()这个三个函数完成

1.）测量-Measure过程是计算视图大小，View measure过程相关方法主要有三个：

public final void measure(int widthMeasureSpec, int heightMeasureSpec)

protected final void setMeasuredDimension(int measuredWidth, int measuredHeight)

protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec)

measure调用onMeasure，onMeasure测量宽度、高度然后调用setMeasureDimension保存测量结果，measure，setMeasureDimension是final类型，view的子类不需要重写，onMeasure在view的子类中重写。

关于MeasureSpec：

(1) UPSPECIFIED :父容器对于子容器没有任何限制,子容器想要多大就多大.

(2) EXACTLY父容器已经为子容器设置了尺寸,子容器应当服从这些边界,不论子容器想要多大的空间.

(3) AT\_MOST子容器可以是声明大小内的任意大小.

2.）布局-Layout过程用于设置视图在屏幕中显示的位置，View layout过程相关方法主要要三个：

public void layout(int l, int t, int r, int b)

protected boolean setFrame(int left, int top, int right, int bottom)

protected void onLayout(boolean changed, int left, int top, int right, int bottom)

layout通过调用setFrame（l,t,r,b），l,t,r,b即子视图在父视图中的具体位置，onLayout一般只会在自定义ViewGroup中才会使用

3.)绘制-draw过程主要用于利用前两步得到的参数，将视图显示在屏幕上，到这里也就完成了整个的视图绘制工作。

public void draw(Canvas canvas)

protected void onDraw(Canvas canvas)

通过调用draw函数进行视图绘制，在View类中onDraw函数是个空函数，最终的绘制需求需要在自定义的onDraw函数中进行实现，比如ImageView完成图片的绘制，如果自定义ViewGroup这个函数则不需要重载。

自定义控件示例：

这里先介绍继承View的方式为例，其实ViewGroup最终的继承的也是View。这里 模拟一个需求场景，需要一个圆形显示百分比。

复制代码

public class PercentView extends View {

private final static String TAG = PercentView.class.getSimpleName();

private Paint mPaint;

private RectF oval;

public PercentView(Context context) {

super(context);

init();

}

public PercentView(Context context, AttributeSet attrs) {

super(context, attrs);

init();

}

public PercentView(Context context, AttributeSet attrs, int defStyleAttr) {

super(context, attrs, defStyleAttr);

init();

}

private void init(){

mPaint = new Paint();

mPaint.setAntiAlias(true);

oval=new RectF();

}

@Override

protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {

super.onMeasure(widthMeasureSpec, heightMeasureSpec);

int widthMode = MeasureSpec.getMode(widthMeasureSpec);

int widthSize = MeasureSpec.getSize(widthMeasureSpec);

int heightMode = MeasureSpec.getMode(heightMeasureSpec);

int heightSize = MeasureSpec.getSize(heightMeasureSpec);

Log.e(TAG, "onMeasure--widthMode-->" + widthMode);

switch (widthMode) {

case MeasureSpec.EXACTLY:

break;

case MeasureSpec.AT\_MOST:

break;

case MeasureSpec.UNSPECIFIED:

break;

}

Log.e(TAG, "onMeasure--widthSize-->" + widthSize);

Log.e(TAG, "onMeasure--heightMode-->" + heightMode);

Log.e(TAG, "onMeasure--heightSize-->" + heightSize);

}

@Override

protected void onLayout(boolean changed, int left, int top, int right, int bottom) {

super.onLayout(changed, left, top, right, bottom);

Log.e(TAG, "onLayout");

}

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);

mPaint.setColor(Color.GRAY);

// FILL填充, STROKE描边,FILL\_AND\_STROKE填充和描边

mPaint.setStyle(Paint.Style.FILL\_AND\_STROKE);

int with = getWidth();

int height = getHeight();

Log.e(TAG, "onDraw---->" + with + "\*" + height);

float radius = with / 4;

canvas.drawCircle(with / 2, with / 2, radius, mPaint);

mPaint.setColor(Color.BLUE);

oval.set(with / 2 - radius, with / 2 - radius, with / 2

+ radius, with / 2 + radius);//用于定义的圆弧的形状和大小的界限

canvas.drawArc(oval, 270, 120, true, mPaint); //根据进度画圆弧

}

}

复制代码

在布局中如何使用

复制代码

<LinearLayout 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:orientation="vertical">

<com.whoislcj.views.PercentView

android:layout\_width="wrap\_content"

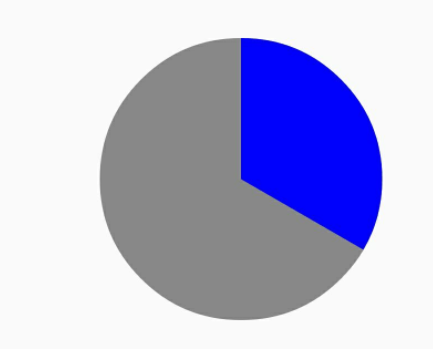
android:layout\_height="wrap\_content"

android:layout\_margin="10dp" />

</LinearLayout>

复制代码

显示效果：



如果布局文件改成

复制代码

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

<LinearLayout 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:orientation="vertical">

<com.whoislcj.views.PercentView

android:layout\_width="150dp"

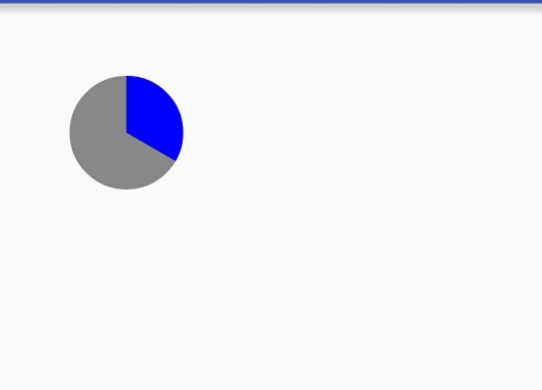
android:layout\_height="150dp"

android:layout\_margin="10dp" />

</LinearLayout>

复制代码

显示效果变成



总结：

本篇主要介绍Android自定义控件的基本绘制原理，会在下一篇中介绍如何自定义属性。

https://www.cnblogs.com/whoislcj/p/5708778.html