# 安卓中实现圆形头像~

大方向上讲,实现圆形图片的展示，在安卓中分 为两种大情况， 一种是自定义一个view+bitmapShader,另外一种方式我们可以重写一个drawable。

> 第一种情况

```
/**
 * Created by linSir on 16/7/30.自定义的头像的ImageVIew
 */
public class CircleImageView extends ImageView {

    private static final ScaleType SCALE_TYPE = ScaleType.CENTER_CROP;

    private static final Bitmap.Config BITMAP_CONFIG = Bitmap.Config.ARGB_8888;
    private static final int COLORDRAWABLE_DIMENSION = 1;

    private static final int DEFAULT_BORDER_WIDTH = 0;
    private static final int DEFAULT_BORDER_COLOR = Color.BLACK;

    private final RectF mDrawableRect = new RectF();
    private final RectF mBorderRect = new RectF();

    private final Matrix mShaderMatrix = new Matrix();
    private final Paint mBitmapPaint = new Paint();
    private final Paint mBorderPaint = new Paint();

    private int mBorderColor = DEFAULT_BORDER_COLOR;
    private int mBorderWidth = DEFAULT_BORDER_WIDTH;

    private Bitmap mBitmap;
    private BitmapShader mBitmapShader;
    private int mBitmapWidth;
    private int mBitmapHeight;

    private float mDrawableRadius;
    private float mBorderRadius;

    private boolean mReady;
    private boolean mSetupPending;

    public CircleImageView(Context context) {
        super(context);
    }

    public CircleImageView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public CircleImageView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        super.setScaleType(SCALE_TYPE);

        TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.CircleImageView, defStyle, 0

        mBorderWidth = a.getDimensionPixelSize(R.styleable.CircleImageView_border_width, DEFAULT_BORD
        mBorderColor = a.getColor(R.styleable.CircleImageView_border_color, DEFAULT_BORDER_COLOR);

        a.recycle();

        mReady = true;

        if (mSetupPending) {
            setup();
            mSetupPending = false;
        }
    }

    @Override
    public ScaleType getScaleType() {
        return SCALE_TYPE;
    }

    @Override
    public void setScaleType(ScaleType scaleType) {
        if (scaleType != SCALE_TYPE) {
            throw new IllegalArgumentException(String.format("ScaleType %s not supported.", scaleType
```

```
        }
    }

    @Override
    protected void onDraw(Canvas canvas) {
        if (getDrawable() == null) {
            return;
        }

        canvas.drawCircle(getWidth() / 2, getHeight() / 2, mDrawableRadius, mBitmapPaint);
        canvas.drawCircle(getWidth() / 2, getHeight() / 2, mBorderRadius, mBorderPaint);
    }

    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        super.onSizeChanged(w, h, oldw, oldh);
        setup();
    }

    public int getBorderColor() {
        return mBorderColor;
    }

    public void setBorderColor(int borderColor) {
        if (borderColor == mBorderColor) {
            return;
        }

        mBorderColor = borderColor;
        mBorderPaint.setColor(mBorderColor);
        invalidate();
    }

    public int getBorderWidth() {
        return mBorderWidth;
    }

    public void setBorderWidth(int borderWidth) {
        if (borderWidth == mBorderWidth) {
            return;
        }

        mBorderWidth = borderWidth;
        setup();
    }

    @Override
    public void setImageBitmap(Bitmap bm) {
        super.setImageBitmap(bm);
        mBitmap = bm;
        setup();
    }

    @Override
    public void setImageDrawable(Drawable drawable) {
        super.setImageDrawable(drawable);
        mBitmap = getBitmapFromDrawable(drawable);
        setup();
    }

    @Override
    public void setImageResource(int resId) {
        super.setImageResource(resId);
        mBitmap = getBitmapFromDrawable(getDrawable());
        setup();
    }

    private Bitmap getBitmapFromDrawable(Drawable drawable) {
        if (drawable == null) {
            return null;
        }

        if (drawable instanceof BitmapDrawable) {
            return ((BitmapDrawable) drawable).getBitmap();
        }

        try {
            Bitmap bitmap;

            if (drawable instanceof ColorDrawable) {
                bitmap = Bitmap.createBitmap(COLORDRAWABLE_DIMENSION, COLORDRAWABLE_DIMENSION, BITMAP
            } else {
```

```
            } else {
                bitmap = Bitmap.createBitmap(drawable.getIntrinsicWidth(), drawable.getIntrinsicHeigh
            }

            Canvas canvas = new Canvas(bitmap);
            drawable.setBounds(0, 0, canvas.getWidth(), canvas.getHeight());
            drawable.draw(canvas);
            return bitmap;
        } catch (OutOfMemoryError e) {
            return null;
        }
    }

    private void setup() {
        if (!mReady) {
            mSetupPending = true;
            return;
        }

        if (mBitmap == null) {
            return;
        }

        mBitmapShader = new BitmapShader(mBitmap, Shader.TileMode.CLAMP, Shader.TileMode.CLAMP);

        mBitmapPaint.setAntiAlias(true);
        mBitmapPaint.setShader(mBitmapShader);

        mBorderPaint.setStyle(Paint.Style.STROKE);
        mBorderPaint.setAntiAlias(true);
        mBorderPaint.setColor(mBorderColor);
        mBorderPaint.setStrokeWidth(mBorderWidth);

        mBitmapHeight = mBitmap.getHeight();
        mBitmapWidth = mBitmap.getWidth();

        mBorderRect.set(0, 0, getWidth(), getHeight());
        mBorderRadius = Math.min((mBorderRect.height() - mBorderWidth) / 2, (mBorderRect.width() - mB

        mDrawableRect.set(mBorderWidth, mBorderWidth, mBorderRect.width() - mBorderWidth, mBorderRect
        mDrawableRadius = Math.min(mDrawableRect.height() / 2, mDrawableRect.width() / 2);

        updateShaderMatrix();
        invalidate();
    }

    private void updateShaderMatrix() {
        float scale;
        float dx = 0;
        float dy = 0;

        mShaderMatrix.set(null);

        if (mBitmapWidth * mDrawableRect.height() > mDrawableRect.width() * mBitmapHeight) {
            scale = mDrawableRect.height() / (float) mBitmapHeight;
            dx = (mDrawableRect.width() - mBitmapWidth * scale) * 0.5f;
        } else {
            scale = mDrawableRect.width() / (float) mBitmapWidth;
            dy = (mDrawableRect.height() - mBitmapHeight * scale) * 0.5f;
        }

        mShaderMatrix.setScale(scale, scale);
        mShaderMatrix.postTranslate((int) (dx + 0.5f) + mBorderWidth, (int) (dy + 0.5f) + mBorderWidt

        mBitmapShader.setLocalMatrix(mShaderMatrix);
    }

}
```

```
<declare-styleable name="CircleImageView">
        <attr name="border_width" format="dimension"/>
        <attr name="border_color" format="color"/>
    </declare-styleable>
```

```
<com.example.lin_sir_one.tripbuyer.customview.CircleImageView
                android:id="@+id/view"
                android:layout_width="70dp"
                android:layout_height="70dp"
                android:layout_below="@+id/hint2_address_details"
                android:layout_marginLeft="28dp"
                android:layout_marginTop="18dp"
                android:src="@mipmap/linsir"
                app:border_color="#FFF"
                app:border_width="1dp"/>
```

只要这样就可以实现圆形头像的功能了，可以看一下下图↓。


> 下面我们简单的介绍一下我们代码

- 这里我们首先需要将外界传进来的一个drawable的图片转换成一个bitmap
- 然后我们画了一个圆形的圆
- 把他们放在一起，就能实现一个具有外圈圆的头像
- 当然我们这里面还有很多自定义的属性，就不一一介绍了，大家可以copy下去当做工具类

> 本文参考我一直崇拜的偶像，鸿神的文章，他的博客地址 www.zhanghongyang.com 他对实现圆形头像的简介 http://blog.csdn.net/lmj623565791/article/details/41967509