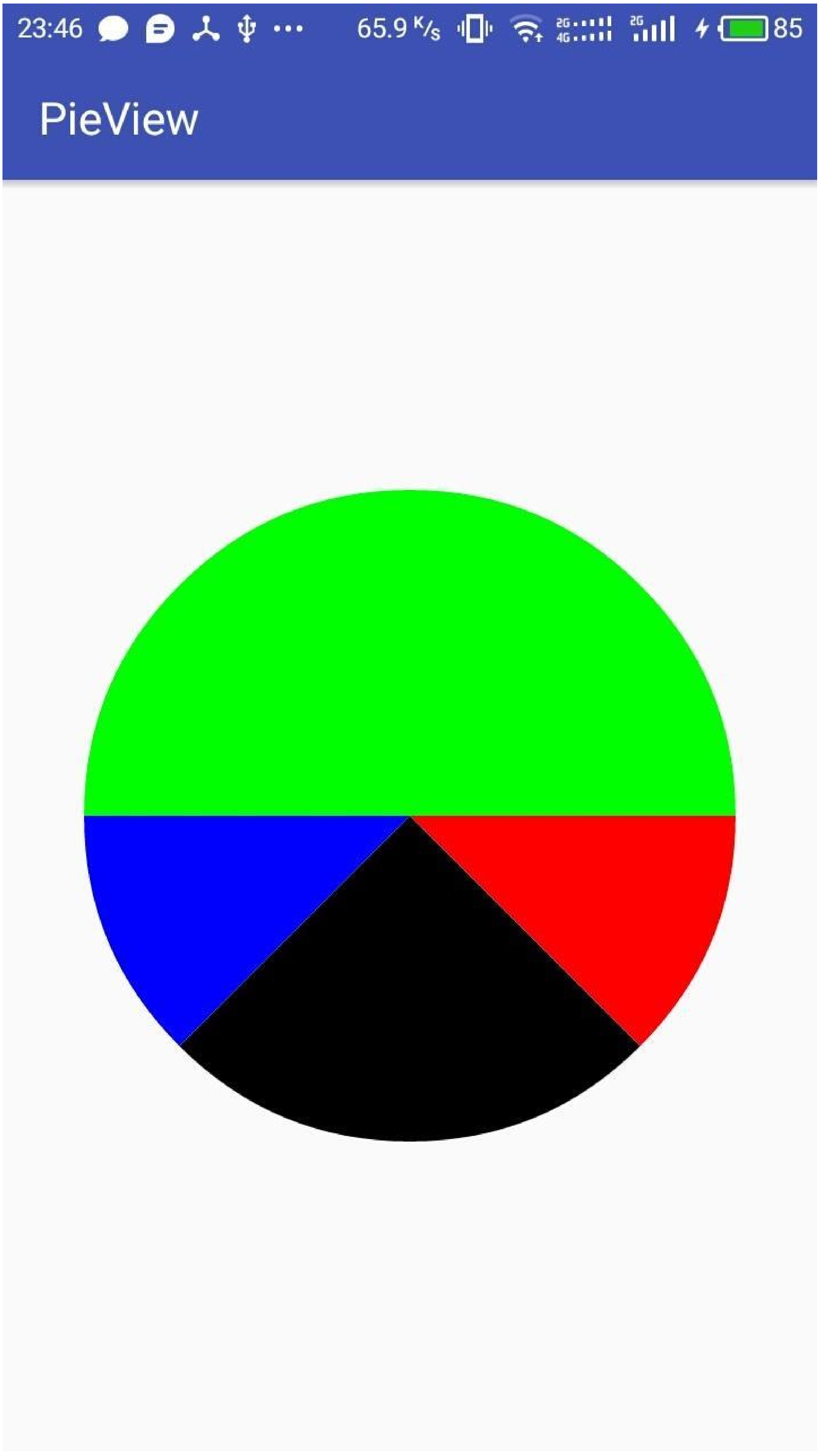
自定义view基础入门——实现饼状图

> 自定义view应该是Android开发的基本功吧，最近无聊打算再重头过一遍自定义view，今天写的是一个比较简单的demo了，是一个自定义的饼状图，我是根据自定义view教程学习的。


23:46
65.9 K/s
85
PieView

其实这个自定义view还是挺简单的，只需要让用户传入一个list，然后根据list里面的数据，找出不同数据占的权重，然后在绘制扇形的过程中，上不同的色就可以了，当然这只是一个入门级的自定义view。

核心代码：

```java
public class PieData {

    private String name;    //颜色
    private float value;    //数值
    private float percentage;   //百分比

    private int color = 0;      //颜色
    private float angle = 0;    //角度

    public PieData(@NonNull String name, @NonNull float value) {
        this.name = name;
        this.value = value;
    }
}
```

```java
/**
 *  Created by linSir
 *  date at 2017/5/22.
 *  describe: 自定义的饼状图
 */

public class PieView extends View {

    private int[] mColors = {Color.RED, Color.BLACK, Color.BLUE, Color.GREEN, Color.YELLOW};

    private float mStartAngle = 0;  //初始化绘制的角度

    private ArrayList<PieData> mData;   //数据

    private int mWidth, mHeight;    //宽，高

    private Paint mPaint = new Paint();


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

    public PieView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        mPaint.setStyle(Paint.Style.FILL);  //设置画笔的模式为填充
        mPaint.setAntiAlias(true);  //设置抗锯齿
    }


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

    @Override protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        if (null == mData)
            return;
        float currentStartAngle = mStartAngle;
        canvas.translate(mWidth / 2, mHeight / 2);
        float r = (float) (Math.min(mWidth, mHeight) / 2 * 0.8);
        RectF rect = new RectF(-r, -r, r, r);

        for (int i = 0; i < mData.size(); i++) {
            PieData pie = mData.get(i);
            mPaint.setColor(pie.getColor());
            canvas.drawArc(rect, currentStartAngle, pie.getAngle(), true, mPaint);
            currentStartAngle += pie.getAngle();
        }
    }

    /**
     * 设置起始角度
     */
    public void setStartAngle(int mStartAngle) {
        this.mStartAngle = mStartAngle;
        invalidate();    //刷新
        invalidate();    //刷新
    }


    public void setData(ArrayList<PieData> mData) {
        this.mData = mData;
        initData(mData);
        invalidate();
    }

    private void initData(ArrayList<PieData> mData) {
        if (null == mData || mData.size() == 0)
            return;

        float sumValue = 0;
        for (int i = 0; i < mData.size(); i++) {
            PieData pie = mData.get(i);
            sumValue += pie.getValue();
            int j = i % mColors.length;
            pie.setColor(mColors[j]);
        }

        float sumAngle = 0;
        for (int i = 0; i < mData.size(); i++) {
            PieData pie = mData.get(i);

            float percentage = pie.getValue() / sumValue;   //占的百分比
            float angle = percentage * 360;

            pie.setPercentage(percentage);
            pie.setAngle(angle);
            sumAngle += angle;

        }

    }

}
```

```
public class MainActivity extends AppCompatActivity {

    private PieView mPieView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mPieView = (PieView) findViewById(R.id.mPieView);

        ArrayList<PieData> list = new ArrayList<PieData>();
        PieData data = new PieData("test",1);
        PieData data2 = new PieData("test",2);
        PieData data3 = new PieData("test",1);
        PieData data4 = new PieData("test",4);

        list.add(data);
        list.add(data2);
        list.add(data3);
        list.add(data4);

        mPieView.setData(list);

    }
}
```

写到这里，我们的自定义view就写完了，效果如上图所示，当然这只能算是自定义view家族中最为简单的自定义view了，在之后的学习生活中吧，我打算总结一下自定义view的基本知识点，还有一些常见的问题，并且多做一些例子，大家一起讨论~

本文代码的源码链接