*/\*\*  
 \* Created by 聂军 on 2017/5/18.   
 \* 自定义控件两种写法  
 \* 1、继承现有的布局容器 不用担心我的控件如何摆放，只需要把原生控件组合在一起变成一个自定义控件  
 \* 2、继承view或者viewgrooup 需要从头到尾的自己写  
 \*/***public class** ToggleButton **extends** View {  
 **private final** Bitmap **down**;  
 **private final** Bitmap **up**;  
 **private boolean isOpen**;  
 **private final int max**;  
 **private int left**;  
 **private int downX**;  
 **private int downY**;  
 **private long downTimes**;  
 **private boolean isChange**;  
 */\*\*  
 \* 一、构造方法：  
 \* 创建对象，数据初始化操作  
 \* \*/* **public** ToggleButton(Context context, @Nullable AttributeSet attrs) {  
 **super**(context, attrs);  
 *//1、获取图片* **down** = BitmapFactory.*decodeResource*(getResources(), R.mipmap.***down***);  
 **up** = BitmapFactory.*decodeResource*(getResources(), R.mipmap.***up***);  
 **max** = **down**.getWidth() - **up**.getWidth();  
 *//4、写好attrs后，要让开关的状态真正发生改变* TypedArray typedArray = context.obtainStyledAttributes(attrs, R.styleable.***ToggleButton***);  
 *//获取状态，第二个参数是如果xml里面没有设置显示的默认状态* **isOpen** = typedArray.getBoolean(R.styleable.***ToggleButton\_isOpen***,**false**);  
 *//调用recycle主要是为了缓存。当recycle被调用后，这就说明这个对象从现在可以被重用了* typedArray.recycle();  
 *//5、获取开关位置* **left** = **isOpen** ? **max** : 0;  
 }  
 */\*\*  
 \* 二、测量：  
 \* \*/* @Override  
 **protected void** onMeasure(**int** widthMeasureSpec, **int** heightMeasureSpec) {  
 *//调用父类（view）的setMeasuredDimension方法，这里测量的是整个view的宽高、需要注释掉，重写我们自己的setMeasuredDimension方法进行测量  
// super.onMeasure(widthMeasureSpec, heightMeasureSpec);  
 //3、调用setMeasuredDimension测量底部照片的宽高* setMeasuredDimension(**down**.getWidth(),**down**.getHeight());  
 }  
 */\*\*  
 \* 三、排版布局：   
 \* 继承View：view的子类都是单控件、例如button textview edittext等 ----->实现功能，不需要关注排版  
 \* 继承ViewGroup： viewgroup是容器、例如五大布局、listview、gridview等 ----->需要给儿子排版  
 \* 本例继承view、不需要管它  
 \* \*/* @Override  
 **protected void** onLayout(**boolean** changed, **int** left, **int** top, **int** right, **int** bottom) {  
 **super**.onLayout(changed, left, top, right, bottom);  
 }  
 */\*\*  
 \* 四、绘制：  
 \* \*/* @Override  
 **protected void** onDraw(Canvas canvas) {  
 **super**.onDraw(canvas);  
 *//2、如果不进行测量就绘制，也可以绘制到画布上，但是控件是整个view的大小，可以在控件上面加个背景颜色进行认证  
 //想要绘制成我们自己需要的大小，必须先进行测量* canvas.drawBitmap(**down**,0,0,**null**);  
 *//5、把0改成left，即开关的位置* canvas.drawBitmap(**up**,**left**,0,**null**);  
 }  
  
 */\*\*  
 \*五、 触摸事件的处理  
 \* \*/* @Override  
 **public boolean** onTouchEvent(MotionEvent event) {  
  
 **switch** (event.getAction()) {  
 **case** MotionEvent.***ACTION\_DOWN***:*//手指按下  
 //七、按下的时候记录是否改变状态* **isChange** = **isOpen**;  
 *//2、记录按下点的坐标* **downX** = (**int**) event.getX();  
 **downY** = (**int**) event.getY();  
 **downTimes** = SystemClock.*uptimeMillis*();  
 Log.*e*(**"ToggleButton"**, **"按下的点："** + **downX**);  
 **break**;  
 **case** MotionEvent.***ACTION\_MOVE***:*//手指移动  
 //3、记录移动点的坐标，移动点坐标减去按下点的坐标就等于移动的距离* **int** moveX = (**int**) event.getX();  
 Log.*e*(**"ToggleButton"**, **"移动的点"** + moveX);  
 **int** dx = moveX - **downX**;  
 Log.*e*(**"ToggleButton"**, **"移动的距离"** + dx);  
 *//4、left加上移动的距离就是当前移动到的位置* **left** += dx;  
 *//5、处理越界问题* **if** (**left** < 0) {  
 **left** = 0;  
 }  
 **if** (**left** > **max**){  
 **left** = **max**;  
 }  
 *//6、一定记得重绘* invalidate();  
 *//7、把当前移动的点赋值给按下的点，可使平滑移动* **downX** = moveX;  
 **break**;  
 **case** MotionEvent.***ACTION\_UP***:*//手指抬起  
 //9、处理点击动作 按下一个点有误差（容错范围与容错时间），记录按下点与时间，抬起点与时间* **int** upX = (**int**) event.getX();  
 **int** upY = (**int**) event.getY();  
 **if** (SystemClock.*uptimeMillis*() - **downTimes** < 500 && Math.*abs*(upX - **downX**) < 5 && Math.*abs*(upY - **downY**) < 5) {  
 *//此处是点击操作* **if** (!**isOpen**) {  
 *//执行打开操作  
 //判断点击的有效范围* **if** (**downX** > **up**.getWidth() && **downX** < **down**.getWidth()) {  
 *//执行打开* **isOpen** = **true**;  
 **left** = **max**;  
 }**else** {  
 **left** = 0;  
 }  
 invalidate();  
 }**else** {  
 *//执行关闭操作  
 //判断点击的有效范围* **if** (**downX** > 0 && **downX** < **max**) {  
 *//执行关闭* **isOpen** = **false**;  
 **left** = 0;  
 }**else** {  
 **left** = **max**;  
 }  
 invalidate();  
 }  
 }**else** {  
 *//此处是滑动后的回弹  
 //8、处理松手后的回弹* **if** (**left** < **max**/2){  
 **left** = 0;  
 **isOpen** = **false**;  
 }**else** {  
 **left** = **max**;  
 **isOpen** = **true**;  
 }  
 invalidate();  
 }  
 *//加个判断条件isOpen != isChange，开关的状态不等于改变的状态才进入里面执行* **if** (**listener** != **null** && **isOpen** != **isChange**) {  
 **listener**.OnToggleChange(**isOpen**);  
 }  
 **break**;  
 **default**:  
 **break**;  
 }  
 *//1、默认父类自己处理，这里我们需要自己（togglebutton）处理触摸事件，改为true  
// return super.onTouchEvent(event);* **return true**;  
 }  
 **private** OnToggleButtonChangeListener **listener**;  
 **public void** setOnToggleButtonChangeListener(OnToggleButtonChangeListener listener){  
 **this**.**listener** = listener;  
 }  
 */\*\*  
 \* 六、接口回掉，让外部能知道打开还是关闭  
 \* \*/* **public interface** OnToggleButtonChangeListener{  
 **void** OnToggleChange(**boolean** isOpen);  
 }  
}