Android 自定义控件

讲师：杨光福

微博：http://weibo.com/321chinavideo

Day2

1、自定义属性-40

1_创建工程：自定义属性，包名：com.atguigu.autoattrs

建议用 UTF-8 编码

2_创建属性类继承 View

```
public class MyAttrsView extends View {

public MyAttrsView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    /**
     * 布局文件使用的时候通过这个构造方法实例化
     *
     * @param context
     * @param attrs
     */
    public MyAttrsView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    /**
     * 一般在代码里面使用的时候通过这个方法实例化
     *
     * @param context
```

```
     */
    public MyAttrsView(Context context) {
        super(context);
    }

}
```

3_创建工程的属性文件 attrs.xml 和常见属性类型

查看 View 的属性

E:\Android_source\2.3 源码\JB\frameworks\base\core\res\res\values\attra.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- 声明一个属性集合，名字可以随机起，建议有意义 -->
    <declare-styleable name="MyAttrsView">
        <!-- reference 引用  参考 -->
        <attr name="my_bg" format="reference" />
         <!-- 声明integer类型 -->
        <attr name="my_age" format="integer" />
         <!-- 声明string类型 -->
        <attr name="my_name" format="string" />
    </declare-styleable>
</resources>
```

知识拓展

format 常用类型

reference	**引用**
color	**颜色**
boolean	**布尔值**
dimension	**尺寸值**
float	**浮点值**
integer	**整型值**
string	**字符串**
enum	**布尔值**

4_使用自定义类并和自定义属性

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
xmlns:atguigu="http://schemas.android.com/apk/res/com.atguigu.autoattrs
"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
     >
    <com.atguigu.autoattrs.MyAttrsView
        atguigu:my_bg="@drawable/ic_launcher"
        atguigu:my_age="100"
        atguigu:my_name="my_attrs_view"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</RelativeLayout>
```

5_得到属性值

```
public MyAttrsView(Context context, AttributeSet attrs) {
        super(context, attrs);
        for(int i=0;i<attrs.getAttributeCount();i++){
            String name = attrs.getAttributeName(i);
            String value = attrs.getAttributeValue(i);
            System.out.println("name="+name+",value="+value);
        }
    }
```

布局文件解析和类的属性的关系

得到属性的原理，显示解析用 pull 解析 xml 文件，解析成键值对，对象的封装成 AttributeSet 对象;根据元素名,有类的全名就可以反射实例化该类,得到 AttributeSet 该类，就得到属性值。

```
public class MyAttrsView extends View {

    public MyAttrsView(Context context, AttributeSet attrs, int defStyle)
{
        super(context, attrs, defStyle);
    }
```

```
    /**
     * 布局文件使用的时候通过这个构造方法实例化
     *
     * @param context
     * @param attrs
     */
    public MyAttrsView(Context context, AttributeSet attrs) {
        super(context, attrs);
        for(int i=0;i<attrs.getAttributeCount();i++){
            String name = attrs.getAttributeName(i);
            String value = attrs.getAttributeValue(i);
//          System.out.println("name="+name+",value="+value);
        }

        //BitmapFactory.decodeResource(getResources(), id);
        //建议用系统工具
        /**
         * 第一个实例化的参数属性集合
         * 第二参数是定义在attrs.xml文件的属性
         */
        TypedArray ta = context.obtainStyledAttributes(attrs,
R.styleable.MyAttrsView);
        int indexCount = ta.getIndexCount();
        for(int i=0;i<indexCount;i++){
            int index = ta.getIndex(i);
            switch (index) {
            case R.styleable.MyAttrsView_my_age:
                int age = ta.getInt(index, 10);
                System.out.println("age=="+age);

                break;
            case R.styleable.MyAttrsView_my_bg:
                Drawable drawable = ta.getDrawable(index);
                System.out.println("drawable=="+drawable);

                break;
            case R.styleable.MyAttrsView_my_name:
                String name = ta.getString(index);
                System.out.println("name=="+name);

                break;
```

```
            default:
                break;
            }
        }

    }

    /**
     * 一般在代码里面使用的时候通过这个方法实例化
     *
     * @param context
     */
    public MyAttrsView(Context context) {
        super(context);
    }

}
```

6_把取到的属性值给给画出来

```
@Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawText(name+"-"+age, 0, 40, paint);
        canvas.drawBitmap(bitmap, 0, 40, paint);
 }



public MyAttrsView(Context context, AttributeSet attrs) {
        super(context, attrs);
        paint = new Paint();
        paint.setColor(Color.GREEN);
        paint.setAntiAlias(true);
        for(int i=0;i<attrs.getAttributeCount();i++){
            String name = attrs.getAttributeName(i);
            String value = attrs.getAttributeValue(i);
//          System.out.println("name="+name+",value="+value);
        }

        //BitmapFactory.decodeResource(getResources(), id);
```

```
        //建议用系统工具
        /**
         * 第一个实例化的参数属性集合
         * 第二参数是定义在attrs.xml文件的属性
         */
        TypedArray ta = context.obtainStyledAttributes(attrs,
R.styleable.MyAttrsView);
        int indexCount = ta.getIndexCount();
        for(int i=0;i<indexCount;i++){
            int index = ta.getIndex(i);
            switch (index) {
            case R.styleable.MyAttrsView_my_age:
               age = ta.getInt(index, 10);
                System.out.println("age=="+age);

                break;
            case R.styleable.MyAttrsView_my_bg:
                Drawable drawable = ta.getDrawable(index);
                //drawable --转换成bitmap
                BitmapDrawable bd = (BitmapDrawable) drawable;
                bitmap = bd.getBitmap();
                System.out.println("drawable=="+drawable);

                break;
            case R.styleable.MyAttrsView_my_name:
                name = ta.getString(index);
                System.out.println("name=="+name);

                break;

            default:
                break;
            }
        }
    }
```

2、仿 viewPager-实现滑动-50

演示做好的时候是怎么个样子

1_创建项目：仿 ViewPager,包名：com.atguigu.myscrollview

2_创建 MyScrollView 继承 ViewGroup,并在布局中使用

```
public class MyScrollView extends ViewGroup {

@Override
protected void onLayout(boolean changed, int l, int t, int r, int b) {
}

public MyScrollView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    public MyScrollView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }
    public MyScrollView(Context context) {
        super(context);
    }

}
```

布局文件中使用

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <com.atguigu.myscrollview.MyScrollView
        android:id="@+id/msv"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />

</RelativeLayout>
```

3_代码中实例化 MyScrollView,并添加页面

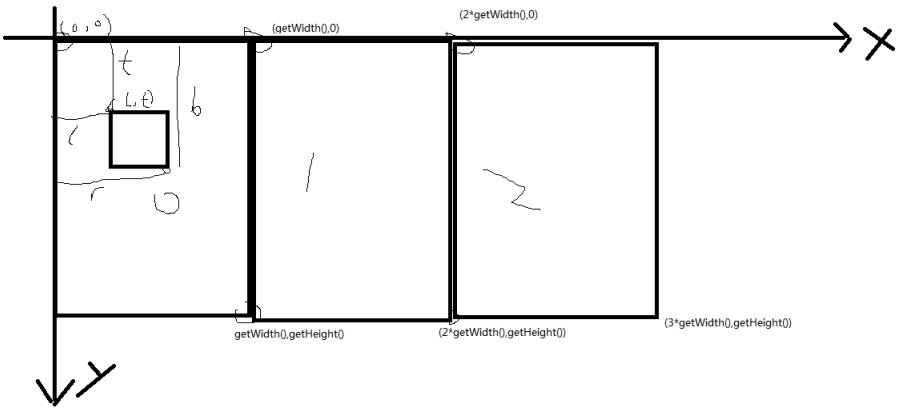
```
public class MainActivity extends Activity {
    private MyScrollView msv;

    private int[] ids = { R.drawable.a1, R.drawable.a2, R.drawable.a3,
```

```
            R.drawable.a4, R.drawable.a5, R.drawable.a6 };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        msv = (MyScrollView) findViewById(R.id.msv);
        //添加view 到自定义MyScrollView 类里面
        for(int i=0;i<ids.length;i++){
            ImageView image = new ImageView(this);
            image.setBackgroundResource(ids[i]);
            msv.addView(image);
        }
    }
}
```

4_onLayout 方法实现

```
/**
     * 指定当前View的位置
     * 如果当前View是viewGroup的话，应该在此方法指定子View的位置
     */
    @Override
    protected void onLayout(boolean changed, int l, int t, int r, int b) {
        for(int i=0;i<getChildCount();i++){
            //获取指定下标的子View
            View child = getChildAt(i);
            //填充满只有一个页面的时候
//          child.layout(0, 0, getWidth(), getHeight());
            //绘制出最后一张图片
            //支持多个图片滑动
            child.layout(0+i*getWidth(),    0,    getWidth()+i*getWidth(),
getHeight());
        }
    }
```

画图分析



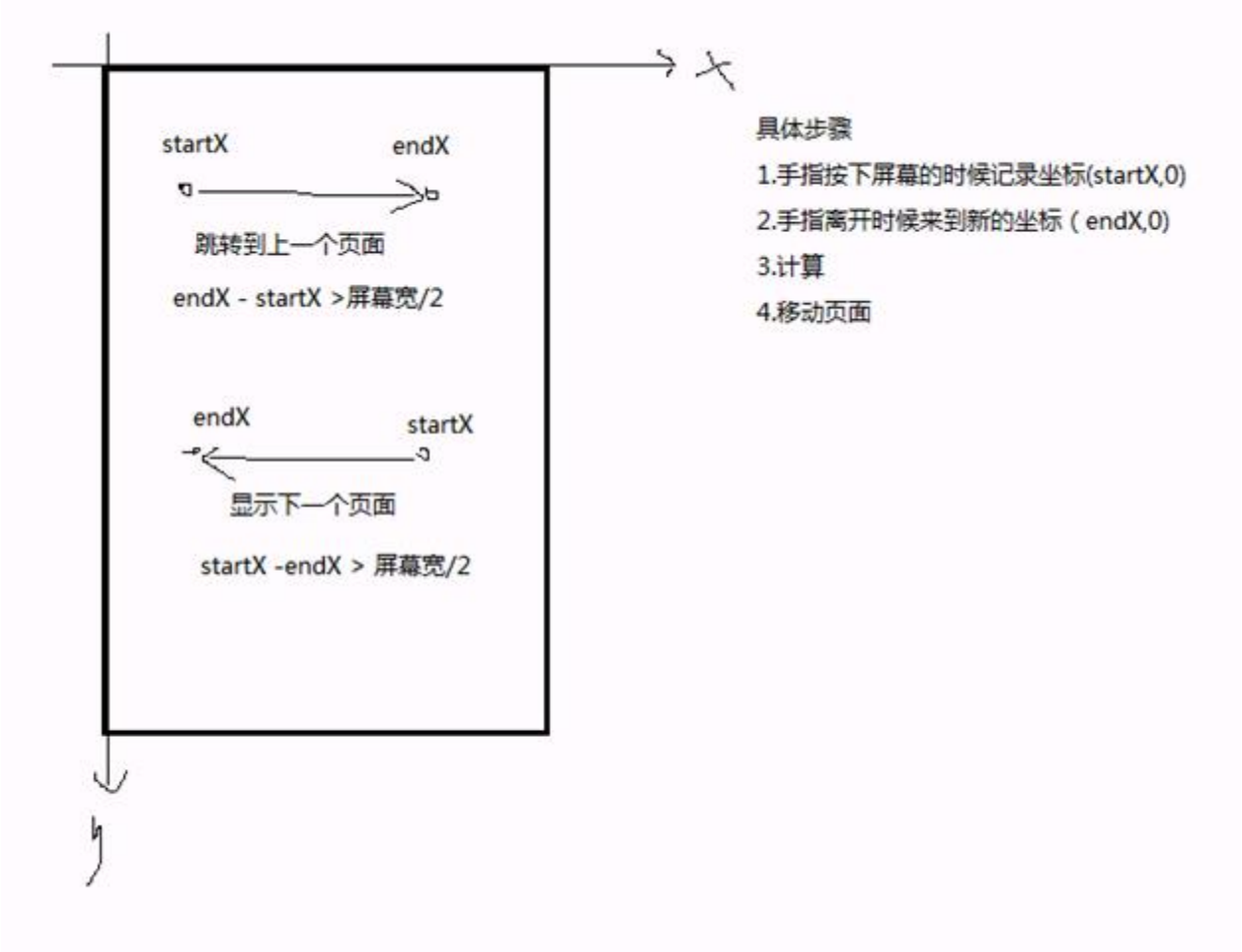
5_用手势识别监听手指在屏幕上触摸滑动

```
/**
     * 系统提供的工具
     * 手势识别器-解析手势
     */
    private GestureDetector detector;
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        super.onTouchEvent(event);
        detector.onTouchEvent(event);
        return true;
    }
    private void initView(Context context) {
        detector = new GestureDetector(context, new
GestureDetector.SimpleOnGestureListener(){

            /**
             * 当手指在屏幕上触摸滑动的时候回调这个方法
             */
            @Override
            public boolean onScroll(MotionEvent e1, MotionEvent e2,
                    float distanceX, float distanceY) {
                /**
                 * 移动View中内容
                 * 竖直方法设置为0
                 * x：距离X的方向的距离
                 * y：距离Y的方向的距离
```

```
                 */
                scrollBy((int)distanceX, 0);
                /**
                 * 移动到某个点
                 * x,y是坐标点
                 * scrollTo(int x, int y)
                 */
                return true;
            }

        });
    }
```

画图理解 scrollTo(int x,int y)

6_滑动到某个位置后自动到合适位置下标停留



代码如下：

```
/**
     * 手指第一次按下的X的坐标
     */
    private int startX = 0;
    /**
     * 当前页面的下标
     */
    private int curIndex;

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        super.onTouchEvent(event);
        detector.onTouchEvent(event);
        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN://手指按下
            //1.手指按下第一次的X的坐标
            startX = (int) event.getX();
            break;

        case MotionEvent.ACTION_MOVE://手指在屏幕移动

            break;
        case MotionEvent.ACTION_UP://手指离开屏幕
            //2.移动到新的X坐标
            int endX = (int) event.getX();
            //3.计算
            int tempIndex = curIndex;
            if(endX-startX>getWidth()/2){
                //移动到上一个页面
                tempIndex--;
            }else if(startX-endX>getWidth()/2){
                //移动到下一个页面
                tempIndex++;
            }

            //4.移动指定的某个页面
            moveTo(tempIndex);

            break;
        }
```

```
        return true;
    }

    /**
     * 移动到指定页面并且屏蔽异常
     * @param tempIndex
     */
    private void moveTo(int tempIndex) {
        if(tempIndex <0){
            tempIndex = 0;
        }

        if(tempIndex >= getChildCount()-1){
            tempIndex = getChildCount()-1;
        }
        curIndex = tempIndex;
        //移动到指定的的子View上
        scrollTo(curIndex*getWidth(), 0);
    }
```

滑动到某一个页面的算法

后几张的图片的清晰度不一样，原因就是对 drawable-hdpi 里面的图片进行压缩了。本身模拟器刚好适合 drawable-mdip 目录的图片的。

3、仿 viewPager 自动回弹效果优化-50

1_简单回顾

2_解决移动生硬

讲解原理

```
//得到要移动的距离
  int distance = curIndex*getWidth() - getScrollX();
//下面是瞬间移动，者效果不好，我们需要在有些时间的移动。
//scrollTo(curIndex*getWidth(), 0);
```

3_写工具类 MyScroller 并记录开始时间

```
public class MyScroller {
    private Context context;
    /**
     * 基准点的x坐标
     */
    private float startX;
    /**
     * 基准点的y坐标
     */
    private float startY;
    /**
     * x方向要移动的距离
     */
    private int distanceX;
    /**
     * y方向要移动的距离
     */
    private int distanceY;
    /**
     * 开始执行动画的时间
     */
    private long startTime;
    /**
     * 用于判断动画是否结束
     */
    private boolean isFinish;

    public MyScroller(Context context){
        this.context = context;
```

```
    }
    /**
     * 开始执行动画
     * @param startX 基准点的x坐标
     * @param startY 基准点的y坐标
     * @param distanceX x方向要移动的距离
     * @param distanceY y方向要移动的距离
     */
    public void startScroll(float startX,float startY,int distanceX,int
distanceY){
        this.startX = startX;
        this.startY = startY;
        this.distanceX = distanceX;
        this.distanceY = distanceY;
        //开始执行动画的时间
        this.startTime = SystemClock.uptimeMillis();
        this.isFinish = false;

    }

}
```

4_工具类中计算时间和偏移量

```
/**
     * 总共的运动的时间
     */
    private long totalTime = 500;
    /**
     * 当前要移动的X方向距离
     */
    private float curX;
    public float getCurX() {
        return curX;
    }
    public void setCurX(float curX) {
        this.curX = curX;
    }
    /**
     * 计算滑动的偏移量
     * true,还在滑动
     * false说明已经结束
     */
```

```
    public boolean computeScrollOffset(){
        if(isFinish){
            return false;
        }
        long endTime = SystemClock.uptimeMillis();
        //花了多少时间
        long passTime = endTime - startTime;
        if(passTime < totalTime){
            //运动还在进行

            //x方向滑动的速度= 距离/时间
            long velocityX = distanceX / totalTime;
            //当前移动的距离
            this.curX = startX  + passTime * distanceX / totalTime;

        }else{
            //动画结束 ，时间到了 滑动该停止
            this.curX = startX  + distanceX ;
            isFinish = true;
        }

        return true;
    }
```

5_使用工具类-完整代码

```
public class MyScrollView extends ViewGroup {

    /**
     * 得到View的位置 如果这个View是ViewGroup的话，还可以指定子view的位置
     */
    @Override
    protected void onLayout(boolean changed, int l, int t, int r, int b) {
        for (int i = 0; i < getChildCount(); i++) {
            View childview = getChildAt(i);
            // 指定孩子的位置
            childview.layout(i * getWidth(), 0, getWidth() + i * getWidth(),
                    getHeight());

        }
```

```
    }

    // 手势识别器
    // 1.定义手势识别器
    private GestureDetector detector;

    private void initView(Context context) {
        scroller = new MyScroller(context);
        // 2.实例化手势识别器
        detector = new GestureDetector(context,
                new GestureDetector.SimpleOnGestureListener() {

                    @Override
                    public boolean onScroll(MotionEvent e1, MotionEvent e2,
                            float distanceX, float distanceY) {
                        /**
                         * 更加滑动的距离去移动View x:在屏幕上距离X坐标的距离
y:在屏幕上距离Y坐标的距离
                         */
                        scrollBy((int) distanceX, 0);
                        /**
                         * 移动到指定的坐标 x: 坐标系中的X的坐标 y:坐标系中的Y
的坐标
                         */
                        // scrollTo(x, y);

                        return true;
                    }

                });
    }
    /**
     * 手指第一次按下屏幕的X轴的坐标
     */
    private int startX = 0;
    /**
     * 当前显示的子View的下标
     */
    private int curIndex  = 0;
    // 3.使用手势识别器-把屏幕的滑动，给手势识别器解析给我们直接使用
    @Override
```

```
public boolean onTouchEvent(MotionEvent event) {
    detector.onTouchEvent(event);
    switch (event.getAction()) {
    case MotionEvent.ACTION_DOWN://按下
        //1.记录坐标
        startX = (int) event.getX();
        break;

    case MotionEvent.ACTION_MOVE://滑动

        break;
    case MotionEvent.ACTION_UP://手指离开屏幕
        //2.来电新的坐标
        int endX = (int) event.getX();
        //3.计算位置，得出该定位到那个子View的坐标
        int tempIndex = curIndex;
        if(endX - startX > getWidth()/2){
            //显示上一个子View
            tempIndex --;
        }else if(startX - endX >= getWidth()/2){
            //显示下一个子View
            tempIndex ++;
        }
        //4.移动到指定下标的子View
        moveTo(tempIndex);

        break;
    }
    return true;
}

private MyScroller scroller;
/**
 * 移动到指定的View上面
 * @param tempIndex
 */
private void moveTo(int tempIndex) {
    //屏幕非法操作
    if(tempIndex < 0){
        tempIndex = 0;
    }
```

```
        if(tempIndex > getChildCount()-1){
            tempIndex =  getChildCount()-1;
        }

        curIndex = tempIndex;

        //定位到指定的子View里
        //得到要移动的距离
        int distance = curIndex*getWidth() - getScrollX();
        scroller.startScroll(getScrollX(), 0, distance, 0);
        //下面是瞬间移动，者效果不好，我们需要在有些时间的移动。
//      scrollTo(curIndex*getWidth(), 0);
        /**
         * invalidate 会导致computeScroll 的执行，该方法是空的
         */
        invalidate();


    }

    @Override
    public void computeScroll() {
        if(scroller.computeScrollOffset()){

            float curX = scroller.getCurX();
            scrollTo((int)curX, 0);
            /**
             * 会导致computeScroll 的执行
             */
            invalidate();

        }
    }

    public MyScrollView(Context context, AttributeSet attrs, int defStyle)
{
        super(context, attrs, defStyle);
        initView(context);
    }

    public MyScrollView(Context context, AttributeSet attrs) {
        super(context, attrs);
```

```
        initView(context);
    }

    public MyScrollView(Context context) {
        super(context);
        initView(context);
    }

}
```

6_使用系统的 Scroller

```
public class MyScrollView extends ViewGroup {

    /**
     * 得到View的位置 如果这个View是ViewGroup的话，还可以指定子view的位置
     */
    @Override
    protected void onLayout(boolean changed, int l, int t, int r, int b) {
        for (int i = 0; i < getChildCount(); i++) {
            View childview = getChildAt(i);
            // 指定孩子的位置
            childview.layout(i * getWidth(), 0, getWidth() + i * getWidth(),
                    getHeight());

        }

    }

    // 手势识别器
    // 1.定义手势识别器
    private GestureDetector detector;

    private void initView(Context context) {
        scroller = new Scroller(context);
        // 2.实例化手势识别器
        detector = new GestureDetector(context,
                new GestureDetector.SimpleOnGestureListener() {

                    @Override
                    public boolean onScroll(MotionEvent e1, MotionEvent e2,
                            float distanceX, float distanceY) {
```

```
                    /**
                     * 更加滑动的距离去移动View x:在屏幕上距离X坐标的距离 y:在屏幕上距离Y坐标的距离
                     */
                    scrollBy((int) distanceX, 0);
                    /**
                     * 移动到指定的坐标 x：坐标系中的X的坐标 y:坐标系中的Y的坐标
                     */
                    // scrollTo(x, y);

                    return true;
                }

            });
    }
    /**
     * 手指第一次按下屏幕的X轴的坐标
     */
    private int startX = 0;
    /**
     * 当前显示的子View的下标
     */
    private int curIndex  = 0;
    // 3.使用手势识别器-把屏幕的滑动，给手势识别器解析给我们直接使用
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        detector.onTouchEvent(event);
        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN://按下
            //1.记录坐标
            startX = (int) event.getX();
            break;

        case MotionEvent.ACTION_MOVE://滑动

            break;
        case MotionEvent.ACTION_UP://手指离开屏幕
            //2.来电新的坐标
            int endX = (int) event.getX();
            //3.计算位置，得出该定位到那个子View的坐标
            int tempIndex = curIndex;
```

```
            if(endX - startX > getWidth()/2){
                //显示上一个子View
                tempIndex --;
            }else if(startX - endX >= getWidth()/2){
                //显示下一个子View
                tempIndex ++;
            }
            //4.移动到指定下标的子View
            moveTo(tempIndex);

            break;
        }
        return true;
    }

    private Scroller scroller;
    /**
     * 移动到指定的View上面
     * @param tempIndex
     */
    private void moveTo(int tempIndex) {
        //屏幕非法操作
        if(tempIndex < 0){
            tempIndex = 0;
        }

        if(tempIndex > getChildCount()-1){
            tempIndex =  getChildCount()-1;
        }

        curIndex = tempIndex;

        //定位到指定的子View里
        //得到要移动的距离
        int distance = curIndex*getWidth() - getScrollX();
        scroller.startScroll(getScrollX(), 0, distance, 0);
        //下面是瞬间移动，者效果不好，我们需要在有些时间的移动。
//      scrollTo(curIndex*getWidth(), 0);
        /**
         * invalidate 会导致computeScroll 的执行，该方法是空的
         */
        invalidate();
```

```
    }

    @Override
    public void computeScroll() {
        if(scroller.computeScrollOffset()){

            float curX = scroller.getCurrX();
            scrollTo((int)curX, 0);
            /**
             * 会导致computeScroll 的执行
             */
            invalidate();

        }
    }

    public MyScrollView(Context context, AttributeSet attrs, int defStyle)
{
        super(context, attrs, defStyle);
        initView(context);
    }

    public MyScrollView(Context context, AttributeSet attrs) {
        super(context, attrs);
        initView(context);
    }

    public MyScrollView(Context context) {
        super(context);
        initView(context);
    }

}
```

4、添加 RadioGroup 点击实现双向跳转-29

1_布局文件加上 RadioGroup 并在代码实例化

```
<RelativeLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:orientation="horizontal" />


    <com.atguigu.myscrollview.MyScrollView
        android:id="@+id/msv"
        android:layout_below="@id/radioGroup"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />

</RelativeLayout>
```

在 onCreate 实例化

```
private RadioGroup radioGroup;
radioGroup = (RadioGroup) findViewById(R.id.radioGroup);

//遍历MyScrollView有多少个子View，就给RadioGroup添加多少个RadioButton
        for(int i=0;i<msv.getChildCount();i++){
            RadioButton button = new RadioButton(this);
            radioGroup.addView(button);
            if(i == 0){
                button.setChecked(true);
            }
        }
```

2_定义接口 PageChangeListener 监听页面变化

```
public class MyScrollView extends ViewGroup {

    ......................

    /**
```

```
     * 移动到指定的View上面
     * @param tempIndex
     */
    private void moveTo(int tempIndex) {
        //屏幕非法操作
        if(tempIndex < 0){
            tempIndex = 0;
        }

        if(tempIndex > getChildCount()-1){
            tempIndex =  getChildCount()-1;
        }

        curIndex = tempIndex;

        if(changeListener != null){
            changeListener.moveTo(tempIndex);
        }
        //得到播放动画的距离
        int distanceX = curIndex*getWidth() - getScrollX();

        scroller.startScroll(getScrollX(), 0, distanceX, 0);

        //定位到指定的子View里,下面这个方法是瞬间播放，我们需要慢慢的播放
//      scrollTo(curIndex*getWidth(), 0);
        /**
         * invalidate()这个方法会导致computeScroll();
         */
        invalidate();


    }

    ......................


    private PageChangeListener changeListener;

    public PageChangeListener getChangeListener() {
        return changeListener;
    }
```

```
    /**
     * 由外界实例化传进来
     * @param changeListener
     */
    public void setChangeListener(PageChangeListener changeListener) {
        this.changeListener = changeListener;
    }

    /**
     * 监听页面改变时得到下标
     * @author afu
     *
     */
    public interface PageChangeListener{
        /**
         * 当页面改变的时候移动到指定的下标
         * @param curIndex 指定下标
         */
        public void moveTo(int curIndex);
    }


        ......................
}
```

3_使用接口

```
//遍历MyScrollView有多少个子View，就给RadioGroup添加多少个RadioButton
        for(int i=0;i<msv.getChildCount();i++){
            RadioButton button = new RadioButton(this);
            button.setId(i);
            radioGroup.addView(button);
            if(i == 0){
                button.setChecked(true);
            }
        }

        msv.setChangeListener(new PageChangeListener() {

            @Override
            public void moveTo(int curIndex) {
//              for(int i=0;i<radioGroup.getChildCount();i++){
```

```
//
    ((RadioButton)radioGroup.getChildAt(curIndex)).setChecked(false);
//            }
//
    ((RadioButton)radioGroup.getChildAt(curIndex)).setChecked(true);
                radioGroup.check(curIndex);
            }
        });
```

4_监听点击 RadioButton 就跳转对应页面

```
private void moveTo(int tempIndex);
```

变成公共的

```
public void moveTo(int tempIndex);
```

```
radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener() {

            //刚好RadioButton的ID(checkedId)和MyScrollView子View的下标()一
样
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                msv.moveTo(checkedId);
            }
        });
```

5、自定义 viewGroup 添加测试页面--28

1_点击最左边和最右边的 RadioButton 会有问题并解决

解决思路：使用系统自带的 Scroller；

```
public class MyScrollView extends ViewGroup {

    ......................

    // 手势识别器
    // 1.定义手势识别器
    private GestureDetector detector;
```

```
    private void initView(Context context) {
        //实例化偏移量计算工具
        scroller = new Scroller(context);
        ......................
    }
    ......................

    private Scroller scroller;

    /**
     * 移动到指定的View上面
     * @param tempIndex
     */
    public void moveTo(int tempIndex) {
        //屏幕非法操作
        if(tempIndex < 0){
            tempIndex = 0;
        }

        if(tempIndex > getChildCount()-1){
            tempIndex =  getChildCount()-1;
        }

        curIndex = tempIndex;

        //调用接口实现类的moveTo
        if(changeListener != null){
            changeListener.moveTo(curIndex);
        }

        //得到播放动画的距离
        int distanceX = curIndex*getWidth() - getScrollX();

        scroller.startScroll(getScrollX(),          0,          distanceX,
0,Math.abs(distanceX));

        //定位到指定的子View里,下面这个方法是瞬间播放，我们需要慢慢的播放
//      scrollTo(curIndex*getWidth(), 0);
        /**
         * invalidate()这个方法会导致computeScroll();
         */
        invalidate();
```

```
    }

    ......................
}
```

2_写任意测试页面里面包含 ScrollVivew

布局文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/darker_gray"
    android:orientation="vertical" >

    <Spinner
        android:id="@+id/spinner1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <ProgressBar
        android:id="@+id/progressBar1"
        style="?android:attr/progressBarStyleLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <ScrollView
        android:id="@+id/scrollView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical" >

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Large Text"
```

```
android:textAppearance="?android:attr/textAppearanceLarge" />

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge" />

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge" />

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge" />

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge" />

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge" />
```

```
        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge" />

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge" />

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge" />

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge" />

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge" />

        <TextView
```

```
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge" />

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge" />

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Large Text"

android:textAppearance="?android:attr/textAppearanceLarge" />
        </LinearLayout>
    </ScrollView>

</LinearLayout>
```

3_代码添加布局文件到 MyScrollView 中

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        msv = (MyScrollView) findViewById(R.id.msv);
        //添加页面(可以图片也可以是布局）
        for(int i=0;i<ids.length;i++){
            ImageView child = new ImageView(this);
            child.setBackgroundResource(ids[i]);
            msv.addView(child);
        }

        //添加页面
```

```
        View view = View.inflate(this,R.layout.test, null);
        msv.addView(view,2);

        ..............................

    }
```

运行演示无法看到效果。

4_解决无法看到效果--在 MyScrollView 中重写 onMeasure

```
/**
 * 对View进行测量
 * 如果当前View是ViewGroup的话，那么ViewGroup有义务对每个子View测量大小
*/
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        super.onMeasure(widthMeasureSpec, heightMeasureSpec);
        for(int i=0;i<getChildCount();i++){
            View child = getChildAt(i);
            child.measure(widthMeasureSpec, heightMeasureSpec);
        }
    }
```

5_解释原因

最外层的 View 没有被调用 measure，里面中的子 View：ProgressBar、和 Spinner、ScrollView 等等没有被调用 measure 方法，没有测量就不知道有大小，没有大小就没有高和宽，就不知道怎么画出来。

1、测量过程：

父亲测量子 View，如果父亲是 ViewGroup，父亲有义务测量每一个孩子。形成树状结构测量。如果是 View 测量自身，也就是设置自己的大小。

2.问题：刚才其他显示图片的页面，我们没有测量大小为什么有能显示？

当我们没有写 measure 方法的时候，第二个页面其实也是显示出来的；也就是说我们并没有去测量，我们只是在 onLayout 方法中强制设置每个 MyScrollView 的**一级子 view** 的固定大小，并没有考虑它自身要多大，它想要多大。

但是对于**一级子 View 的子 View** 我们是没有测量的，所以不能显示。

系统的 ViewGroup（线性布局、相对布局），给子 View 指定大小的时候，会考虑子 View

想要多大，测量出来了，有多大才给多大。

```
public final int getWidth() {
      return mRight - mLeft;
 }
public final int getHeight() {
      return mBottom - mTop;
   }
```

宽度：右边的距离减掉左边的距离
高度：底部的距离减掉顶部的距离

3.问题：先有子 View 的大小还是现有父 View 的大小？
不确定；
例如：一个按钮，边是个线性布局，高和宽是包裹类型；现在是先知道子 View 才知道父 View 有多大
如果，父 View 是填充窗体，子 View 也是填充窗体，现在是先知道父 View 大小，才知道子 View 大小。

4.所有的 View 的大小，仅计算一次是不能完成的，一般要计算多次。

理解参数，答应出来看看

```
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
      super.onMeasure(widthMeasureSpec,
      System.out.println(widthMeasureSpec+"   :   "+heightMeasureSpec);
   }
```

答应结果：
1073742144   :   1073742193
是很大的数；这个数字包含了宽度的信息，还包含了给孩子是包裹计算，还是填充计算等信息。这两个信息是可以分开来的。Int 类型最大值是 20 多亿，表示屏幕宽已经足够。1 万个像素足够。

int 类型在内存中占 32 位
4.前两位用来表示宽度，后面的 30 位表示大小；

分开宽度和大小

```
int size = MeasureSpec.getSize(widthMeasureSpec);
int mode = MeasureSpec.getMode(widthMeasureSpec);
System.out.println(size+"   :   "+mode);
```

```
 320   :   1073741824
```

第一个是宽，第一个是大小是一种模式

宽：取反~， &求与得到的。

知识拓展：

它常用的三个函数：

1.static int getMode(int measureSpec):根据提供的测量值(格式)提取模式(上述三个模式之一)

2.static int getSize(int measureSpec):根据提供的测量值(格式)提取大小值(这个大小也就是我们通常所说的大小)

3.static int makeMeasureSpec(int size,int mode):根据提供的大小值和模式创建一个测量值(格式)这个类的使用呢，通常在 view 组件的 onMeasure 方法里面调用但也有少数

6_onMesaure 总结

系统的 onMesaure 中所干的事：

1、根据 widthMeasureSpec 求得宽度 width，和父 view 给的模式

2、根据自身的宽度 width 和自身的 padding 值，相减，求得子 view 可以拥有的宽度 newWidth

3、根据 newWidth 和模式求得一个新的 MeasureSpec 值：

```
MeasureSpec.makeMeasureSpec(newSize, newmode);
```

4、用新的 MeasureSpec 来计算子 view

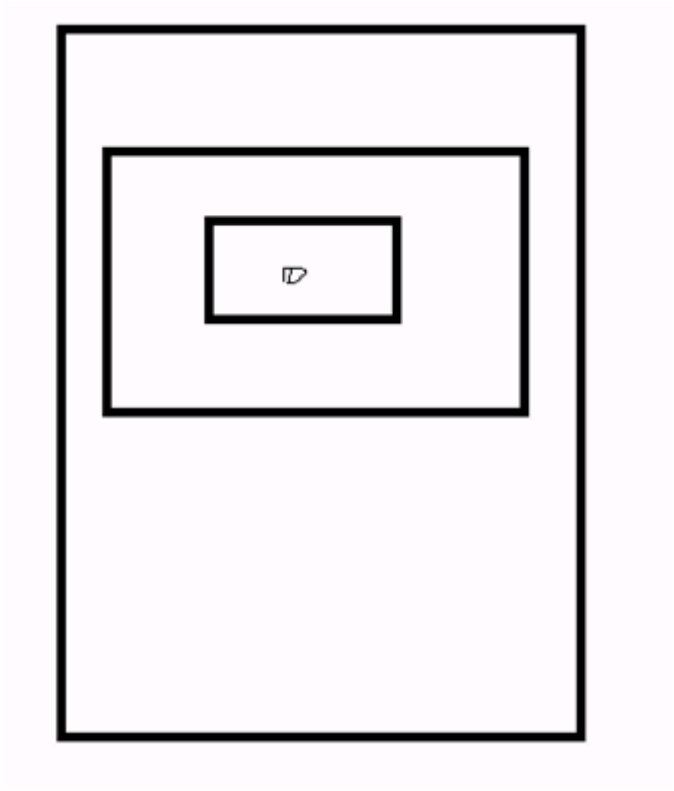
6、自定义 viewGroup 事件中断和消费-37

1_来到测试页面演示左右不能滑动的 bug

演示 bug,并口头说一下原因。

看第一张图片并事件机制

点击小图片，谁先收到事件？



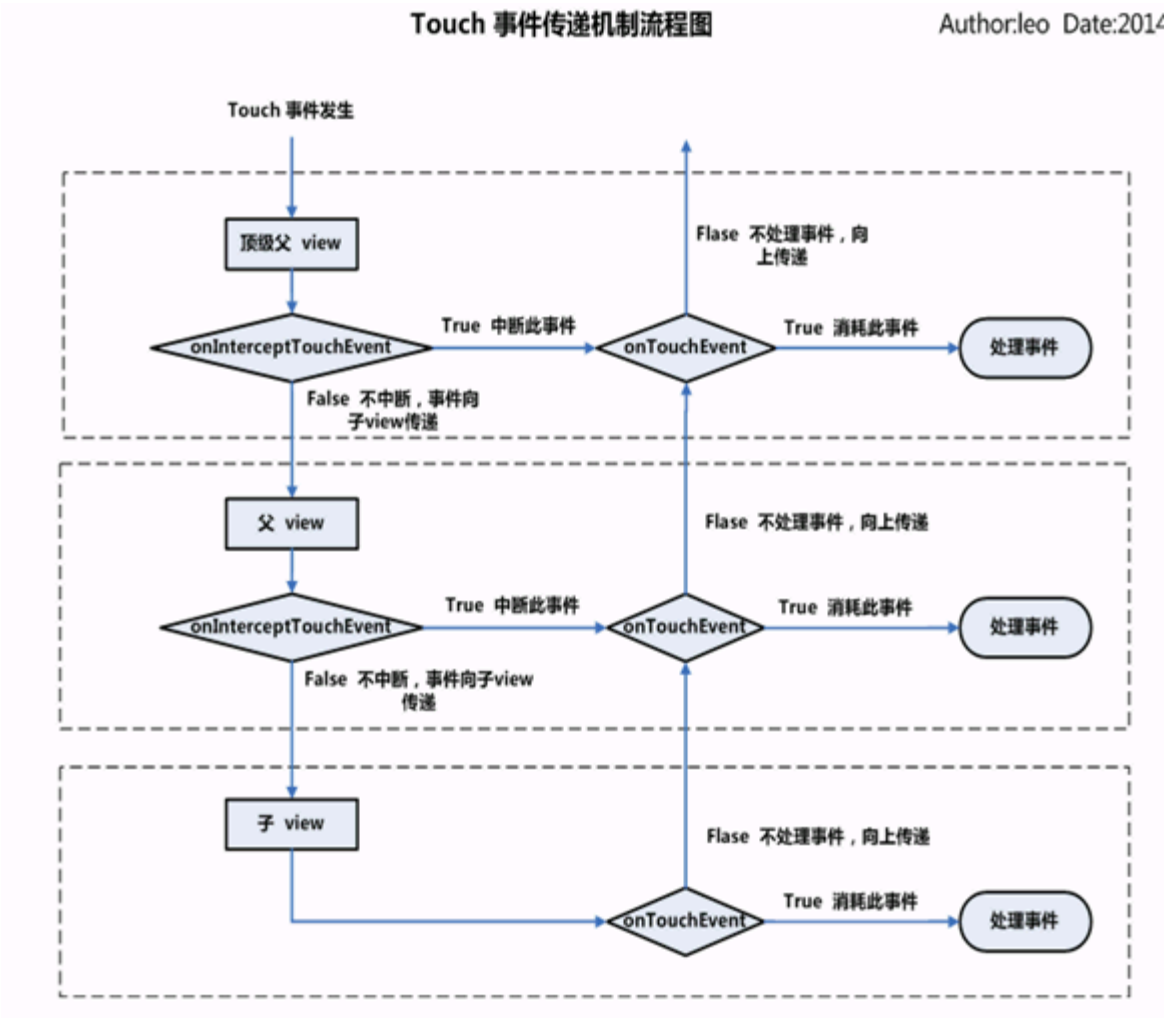
2_重写 onInterceptTouchEvent

Intercept 拦截，截断

```
/**
     * 是否中断事件的传递，默认返回false,意思为，不中断，按正常情况，传递事件
     * 如果为true，就将事件中断，直接执行自己的onTounchEvent方法
     */
    @Override
    public boolean onInterceptTouchEvent(MotionEvent ev) {
        return true;
    }
```

左右滑动可以了，但是上下滑动不行了。
导致的原因是：事件被中断掉了，也就是被消耗或者说消费了，也就不会传给它的孩子了。
就没法实现上下滑动了。

再看第二个图，讲解原因。



讲解梨故事。

3_解决孩子无法上下滑动问题

导致的原因，父亲直接中断了事件的传递。

我们需要根据情况区分是左右滑动，还是上下滑动，如果是左右滑动事件就终止，如果是上下滑动就传递给孩子。

```
/**
     *  第一次按下的X的坐标
     */
    private float downX;
    /**
     *  第一次按下的Y的坐标
     */
    private float downY;
```

```
    /**
     * 是否中断事件的传递，默认返回false,意思为，不中断，按正常情况，传递事件
     * 如果为true，就将事件中断，直接执行自己的onTounchEvent方法
     */
    @Override
    public boolean onInterceptTouchEvent(MotionEvent ev) {
        boolean result = false;
        // 如果水平方向滑动的距离大于竖直方向滑动，就是左右滑动，中断事件；否则事件继续传递；
        switch (ev.getAction()) {
        case MotionEvent.ACTION_DOWN:
            //1.第一次按下坐标
            downX =  ev.getX();
            downY =  ev.getY();
            break;

        case MotionEvent.ACTION_MOVE:
            //2.来到新的坐标
            float newdownX =  ev.getX();
            float newdownY =  ev.getY();
            //3.计算距离
            int distanceX = (int) Math.abs(newdownX - downX);
            int distanceY = (int) Math.abs(newdownY - downY);
            //distanceX > 10 防止抖动为1左右的情况
            if(distanceX > distanceY && distanceX > 10){
                result = true;
            }

            break;
        case MotionEvent.ACTION_UP:

            break;
        }
        return result;
    }
```

解释原因

4_解决左右滑动的 bug

分析原因打日志看看 onTouchEvent **里的按下事件、移动、离开**是否执行，onInterceptTouchEvent 的按下事件、移动、离开是否执行；

问题？回顾左右滑动，是如果实现的？---找到 bug 的原因，是应用 onTouchEvent 没有按下事件，就没有起始值，直接计算得到的值有问题。

对应 onTouchEvent 的注释

onInterceptTouchEvent 返回true,把当前事件中断，我们当前的View才执行onTouchEvent事件； 否则返回false，当前view是没有机会执行onTouchEvent 事件的；

解决方案代码

```
/**
     * 是否中断事件的传递，默认返回false,意思为，不中断，按正常情况，传递事件
     * 如果为true，就将事件中断，直接执行自己的onTounchEvent方法
     */
    @Override
    public boolean onInterceptTouchEvent(MotionEvent ev) {
        boolean result = false;
        // 如果水平方向滑动的距离大于竖直方向滑动，就是左右滑动，中断事件；否则事件继续传递；
        switch (ev.getAction()) {
        case MotionEvent.ACTION_DOWN:
            detector.onTouchEvent(ev);
            //1.第一次按下坐标
            downX =  ev.getX();
            downY =  ev.getY();
            Log.e(TAG, "ACTION_DOWN");
            break;

        ..........................
    }
```

5_引出事件分发 dispatchTouchEvent

事件的分发一般不用重新写，除非你想改变 Android 事件默认机制。建议看代码的话看 2.2 的，比较清晰，后面的版本修改了 bug.导致看起来复杂。

7、用已经学过的知足学习事件分发小案例-10

带领同学本看看代码。

8、水波纹-基本功能-30

先演示做好的效果

1_创建工程：水波纹 com.atguigu.ware

创建自定义水波纹类 MyWare

```
public class MyWare extends View {

    public MyWare(Context context, AttributeSet attrs) {
        super(context, attrs);
    }
}
```

布局文件使用

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <com.atguigu.ware.MyWare
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />

</RelativeLayout>
```

2_画圆环图形

```
public class MyWare extends View {

    private Paint paint;


    public MyWare(Context context, AttributeSet attrs) {
        super(context, attrs);
        initView(context);
    }

    private void initView() {
        paint = new Paint();
        //抗锯齿
        paint.setAntiAlias(true);
```

```
        paint.setColor(Color.RED);
        /**
         * 画线条
         */
        paint.setStyle(Style.STROKE);
        //设置圆环的宽度
        paint.setStrokeWidth(20);

    }

    /**
     * 测量，需要写任何东西，也可以删除，应用布局文件已经写了宽度。
     *
     */
    @Override
   protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        super.onMeasure(widthMeasureSpec, heightMeasureSpec);
    }

    /**
     * 绘制内容
     */
    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawCircle(200, 200, 100, paint);
    }
}
```

3_设置圆环图形动画

```
public class MyWare extends View {

    private Paint paint;

    /**
     * 圆环的半径
     */
    private int radius;
```

```
    public MyWare(Context context, AttributeSet attrs) {
        super(context, attrs);
        initView(context);
    }

    private void initView(Context context) {
        paint = new Paint();
        //抗锯齿
        paint.setAntiAlias(true);
        paint.setColor(Color.RED);
        /**
         * 画线条
         */
        paint.setStyle(Style.STROKE);
        //设置圆环的宽度
//      paint.setStrokeWidth(20);


        //开始设置圆环初始值
        radius = 30;
        paint.setStrokeWidth(radius/3);

        //设置透明度,0完全透明，255完全不透明
        paint.setAlpha(255);

    }

    private Handler handler = new Handler(){
        public void handleMessage(android.os.Message msg) {
            //1.改变圆环的参数：半径、宽、透明
            //半径
            radius +=5;
            //透明度
            int alpha = paint.getAlpha();
            alpha -= 10;

            //alpha 是对alpha%255 的结果值作为透明
            if(alpha <= 20){
                alpha = 0;
            }
            paint.setAlpha(alpha);
```

```
            paint.setStrokeWidth(radius/3);
            //2.刷新 -导致onDraw方法重新执行
            invalidate();

        };
    };
    /**
     * 绘制内容
     */
    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawCircle(200, 200, radius, paint);
        if(paint.getAlpha()>0){
            handler.sendEmptyMessageDelayed(0, 50);
        }
    }

    ..................................

}
```

4_按哪里圆心就在哪里

```
public class MyWare extends View {
     ..................
    /**
     * 绘制内容
     */
    @Override
    protected void onDraw(Canvas canvas) {
        if(pointX!=0&&pointY!=0){
            canvas.drawCircle(pointX, pointY, 100, paint);
            if(paint.getAlpha()>0){
                handler.sendEmptyMessageDelayed(0, 50);
            }
        }

    }
    /**
     * 圆心的X坐标
     */
    private float pointX;
```

```
    /**
     * 圆心的Y坐标
     */
    private float pointY;

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN://按下屏幕
            pointX = event.getX();
            pointY = event.getY();

            //对圆进行初始化
            initView(null);

            //刷新
            invalidate();
            break;

        default:
            break;
        }
        return true;
    }

    ..........................
}
```

水波纹的完成-如何参照别人代码
参照代码：
读代码
在拷贝过来的过程中，最好遵照一定的原则：

1.拷贝只拷贝我们需要的，这样不至于软件很大。
2.拷贝的过程中参照软件运行的顺序先后拷贝。
3.拷贝过程中决定命名不合适的地方最好修改。