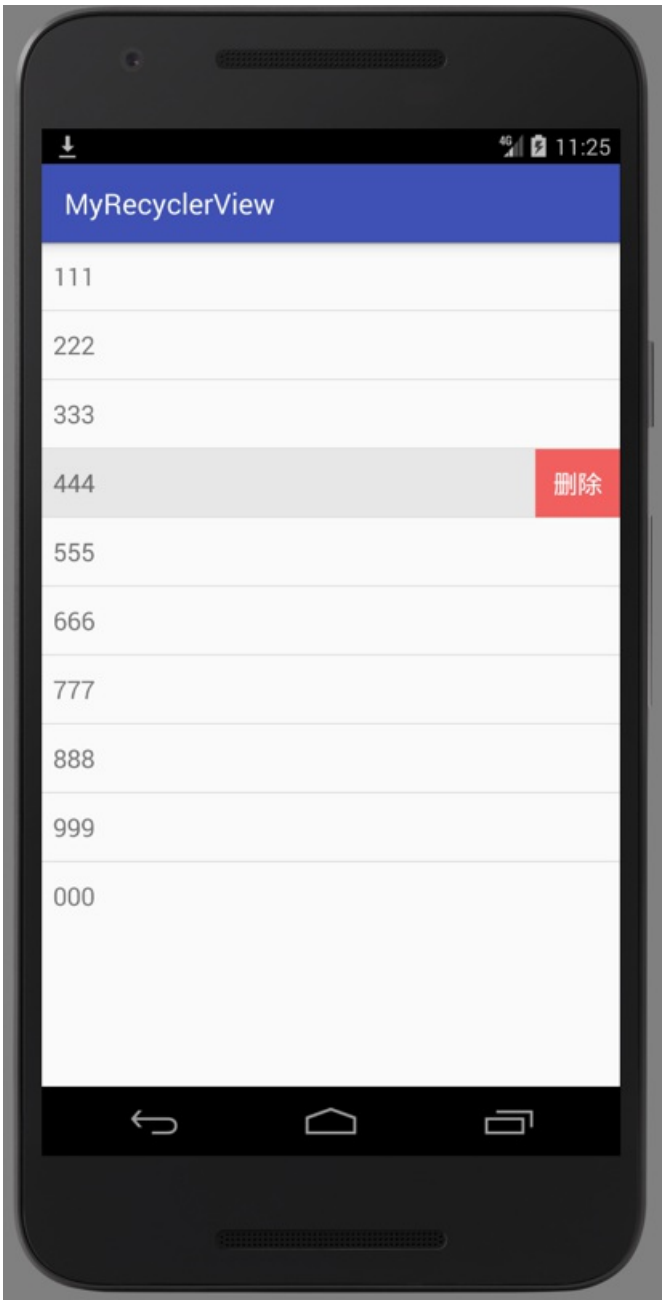
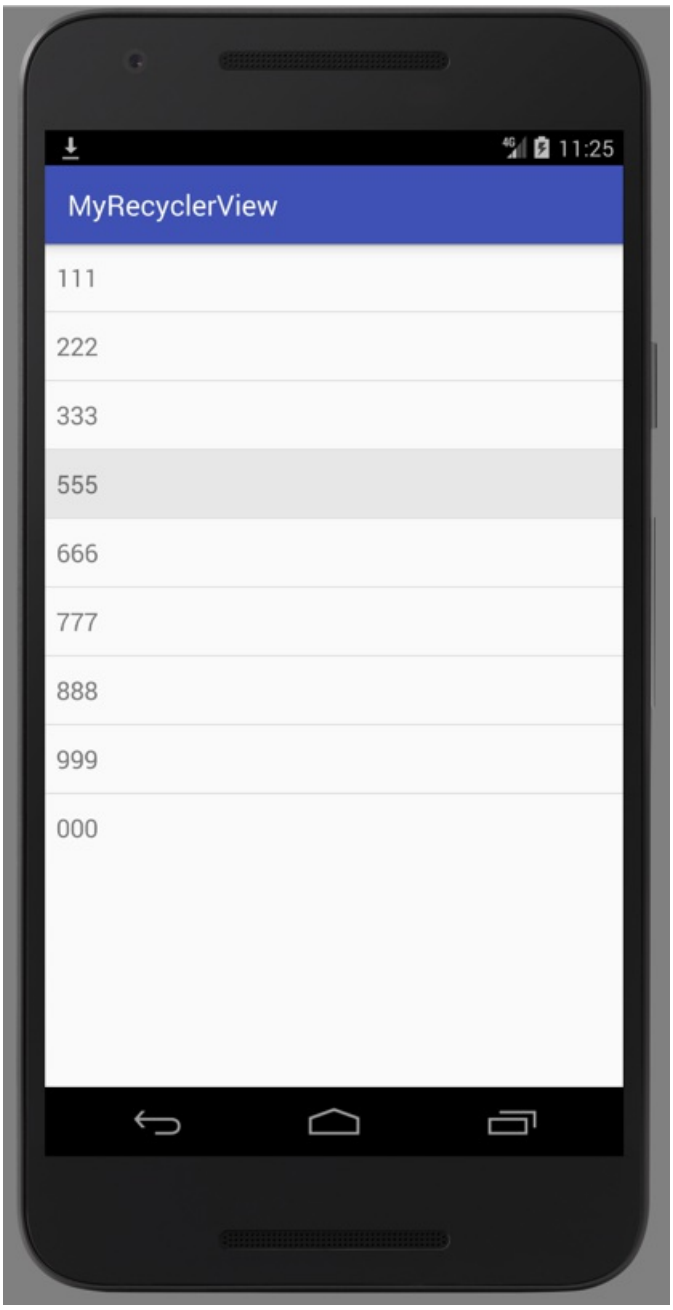
> 实现了一个可侧滑删除的listView，这个view是一个继承自listView的自定义view， 实现侧滑删除，可以通过很多种方式，今天我介绍的方式是通过PopupWindow的方式来实现的。

效果图





思路

当一个listView在屏幕上显示的时候，它上面(屏幕上面)发生的各种事件，我们是可以捕捉到的，我们只需要判断一下是不是我们需要的事件，如果是的话，就产生反馈，对事件进行处理，否则就不处理即可。 当我们发现用户是在一个item上面产生了滑动事件，并且是从右向左滑，并且满足我们对有效滑动长度的定义的话，那么这次事件我们就判断是有效的，我们就计算到相应的位置，并且产生相应的删除的按钮就可以了。 当我们发现用户的单击事件的时候，我们就让删除的按钮消失就可以了。

实现思路来自于：鸿洋的博客

实现代码

```
/**
 *  Created by linSir
 *  date at 2017/5/1.
 *  describe: listView，主要是实现可以侧滑删除
 */

public class MyListView extends ListView {

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

    private int touchSlop;  //用户滑动的最小距离
    private boolean isSliding;  //是否相应滑动
    private int xDown;  //按下的x坐标
    private int yDown;  //按下的y坐标
```

```
    private int yDown;  //按下的y坐标
    private int xMove;  //手指移动时x的坐标
    private int yMove;  //手指移动是y的坐标
    private LayoutInflater mInflater;   //一个layoutInflater
    private PopupWindow mPopupWindow;   //弹出一个用于展示的popupWindow
    private int mPopupWindowHeight;     //该展示的popupWindow的高度
    private int mPopupWindowWidth;      //该展示的popupWindow的宽度

    private TextView delete;    //侧滑后删除的按钮
    private DeleteClickListener mListener;  //点击删除后回调的接口
    private View mCurrentView;  //当前展示删除按钮的view
    private int mCurrentViewPos;    //当前展示删除按钮的view的位置(下标)

    /**
     * 该自定义view的构造方法
     */
    public MyListView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        mInflater = LayoutInflater.from(context);   //一个Inflater
        touchSlop = ViewConfiguration.get(context).getScaledTouchSlop();    //最小的滑动距离

        View view = mInflater.inflate(R.layout.delete_btn, null);   //找到删除按钮的view
        delete = (TextView) view.findViewById(R.id.delete);     //找到删除按钮的控件


        mPopupWindow = new PopupWindow(view, LinearLayout.LayoutParams.WRAP_CONTENT,
                LinearLayout.LayoutParams.WRAP_CONTENT);    //弹出的popupWindow

        mPopupWindow.getContentView().measure(0, 0);    //初始化
        mPopupWindowHeight = mPopupWindow.getContentView().getMeasuredHeight(); //获取到该view的高度
        mPopupWindowWidth = mPopupWindow.getContentView().getMeasuredWidth();   //获取到该view的宽度
    }

    /**
     * 触摸事件的派发
     */
    @Override public boolean dispatchTouchEvent(MotionEvent ev) {

        int action = ev.getAction();
        int x = (int) ev.getX();
        int y = (int) ev.getY();

        switch (action) {
            case MotionEvent.ACTION_DOWN:   //action_down 即点击事件，这个时候需要关闭popupWindow
                xDown = x;
                yDown = y;

                if (mPopupWindow.isShowing()) {
                    dismissPopWindow();
                    return false;
                }

                mCurrentViewPos = pointToPosition(xDown, yDown);    //根据x,y坐标获取到自己的下标
                View view = getChildAt(mCurrentViewPos - getFirstVisiblePosition());//当前可见view的小标
                mCurrentView = view;

                break;

            case MotionEvent.ACTION_MOVE:   //当发生移动时间的时候
                xMove = x;
                yMove = y;
                int dx = xMove - xDown;
                int dy = yMove - yDown;

                if (xMove < xDown && Math.abs(dx) > touchSlop && Math.abs(dy) < touchSlop) { //判断向左
                    isSliding = true;   //满足这个条件就符合了打开的popupWindow的条件
                }
                break;
        }

        return super.dispatchTouchEvent(ev);

    }


    @Override public boolean onTouchEvent(MotionEvent ev) {

        if (mCurrentView == null) {     //判断当前的view不存在之后，则直接return不进行处理这次时间
            return false;
        }

        int action = ev.getAction();
```

```
        if (isSliding) {
            switch (action) {
                case MotionEvent.ACTION_MOVE:
                    int[] location = new int[2];
                    mCurrentView.getLocationOnScreen(location);
                    mPopupWindow.update();

                    delete.setHeight(getMeasuredHeight()/getChildCount());    //计算出来每一个条目的高度

                    mPopupWindow.showAtLocation(mCurrentView, Gravity.LEFT | Gravity.TOP,
                            location[0] + mCurrentView.getWidth(), location[1] + mCurrentView.getHeig
                                    - mPopupWindowHeight );      //设置显示的位置

                    delete.setOnClickListener(new OnClickListener() {
                        @Override public void onClick(View view) {
                            if (mListener != null) {
                                mListener.onClickDelete(mCurrentViewPos);
                                mPopupWindow.dismiss();
                            }
                        }
                    });

                    break;

                case MotionEvent.ACTION_UP:
                    isSliding = false;

                    break;
            }


            return true;
        }
        return super.onTouchEvent(ev);

    }


    private void dismissPopWindow() {
        if (mPopupWindow != null && mPopupWindow.isShowing()) {
            mPopupWindow.dismiss();
        }

    }


    public void setDelButtonClickListener(DeleteClickListener listener) {
        mListener = listener;
    }

}
```

```
/**
 *  Created by linSir
 *  date at 2017/5/1.
 *  describe: 用于点击删除按钮的回调
 */

public interface DeleteClickListener {

    void onClickDelete(int position);

}
```

```
//测试用例

public class MainActivity extends AppCompatActivity {

    private MyListView mListView;
    private ArrayAdapter<String> mAdapter;
    private List<String> mDatas;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mListView = (MyListView) findViewById(R.id.id_listview);
        mDatas = new ArrayList<String>(Arrays.asList("111", "222", "333", "444", "555", "666",
                "777", "888", "999", "000"));
        mAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, mDatas);
        mListView.setAdapter(mAdapter);

        mListView.setDelButtonClickListener(new DeleteClickListener() {
            @Override public void onClickDelete(int position) {
                Toast.makeText(MainActivity.this, position + " : " + mAdapter.getItem(position), Toas
                mAdapter.remove(mAdapter.getItem(position));

            }
        });

        mListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                Toast.makeText(MainActivity.this, position + " : " + mAdapter.getItem(position), Toas
            }
        });
    }
}
```

```
//主界面布局文件

<?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:orientation="vertical"
    >

  <com.dotengine.linsir.myrecyclerview.MyListView
      android:id="@+id/id_listview"
      android:layout_width="match_parent"
      android:layout_height="wrap_content">


  </com.dotengine.linsir.myrecyclerview.MyListView>

</LinearLayout>
```

```
//删除按钮的布局

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content">

    <TextView
        android:id="@+id/delete"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="删除"
        android:textSize="18sp"
        android:gravity="center"
        android:textColor="#FFF"
        android:background="#b4f72626"
        android:paddingLeft="12dp"
        android:paddingRight="12dp"
        />

</LinearLayout>
```

以上便是这次分享的自定义view，最近一直在看自定义view，还有事件传递机制这里，也写了很多测试程序，有空的时候会分享出来的~然后再强调一下，本文的全部思路来自于张鸿洋的博客。