**SlidingActivity：**

1.初始化界面，包括整体布局，左布局，中间布局，右布局，需要提交；

2.设置监听事件函数，如果是第一个viewpager滑动，则显示左侧边界面，如果是最后viewpager，则显示右侧边界面

**Fragment：**

1.设置各个界面的fragment以及每个fragment中的操作

2.设置viewpagerfragment，需要初始化viewpager，设置显示左侧边以及右侧边的方法

**SlidingMenu：**

继承相对布局而自定义的侧边栏菜单布局

package com.example.view;

import com.example.discoverelectricity.R;

import android.app.Activity;

import android.content.Context;

import android.util.AttributeSet;

import android.view.Display;

import android.view.MotionEvent;

import android.view.VelocityTracker;

import android.view.View;

import android.view.ViewConfiguration;

import android.view.WindowManager;

import android.widget.RelativeLayout;

import android.widget.Scroller;

public class SlidingMenu extends RelativeLayout {

    private View mSlidingView;

    private View mMenuView;

    private View mDetailView;

    private RelativeLayout bgShade;

    private int screenWidth;

    private int screenHeight;

    private Context mContext;

    private Scroller mScroller;

    private VelocityTracker mVelocityTracker;

    private int mTouchSlop;

    private float mLastMotionX;

    private float mLastMotionY;

    private static final int VELOCITY = 50;

    private boolean mIsBeingDragged = true;

    private boolean tCanSlideLeft = true;

    private boolean tCanSlideRight = false;

    private boolean hasClickLeft = false;

    private boolean hasClickRight = false;

    public SlidingMenu(Context context) {

        super(context);

        init(context);

    }

    private void init(Context context) {

        mContext = context;

        bgShade = new RelativeLayout(context);

        mScroller = new Scroller(getContext());

        mTouchSlop = ViewConfiguration.get(getContext()).getScaledTouchSlop();

        WindowManager windowManager = ((Activity) context).getWindow()

                .getWindowManager();

        Display display = windowManager.getDefaultDisplay();

        screenWidth = display.getWidth();

        screenHeight = display.getHeight();

        LayoutParams bgParams = new LayoutParams(screenWidth, screenHeight);

        bgParams.addRule(RelativeLayout.CENTER\_IN\_PARENT);

        bgShade.setLayoutParams(bgParams);

    }

    public SlidingMenu(Context context, AttributeSet attrs) {

        super(context, attrs);

        init(context);

    }

    public SlidingMenu(Context context, AttributeSet attrs, int defStyle) {

        super(context, attrs, defStyle);

        init(context);

    }

    public void addViews(View left, View center, View right) {

        setLeftView(left);

        setRightView(right);

        setCenterView(center);

    }

    public void setLeftView(View view) {

        LayoutParams behindParams = new LayoutParams(LayoutParams.WRAP\_CONTENT,

                LayoutParams.FILL\_PARENT);

        addView(view, behindParams);

        mMenuView = view;

    }

    public void setRightView(View view) {

        LayoutParams behindParams = new LayoutParams(LayoutParams.WRAP\_CONTENT,

                LayoutParams.FILL\_PARENT);

        behindParams.addRule(RelativeLayout.ALIGN\_PARENT\_RIGHT);

        addView(view, behindParams);

        mDetailView = view;

    }

    public void setCenterView(View view) {

        LayoutParams aboveParams = new LayoutParams(LayoutParams.FILL\_PARENT,

                LayoutParams.FILL\_PARENT);

        LayoutParams bgParams = new LayoutParams(screenWidth, screenHeight);

        bgParams.addRule(RelativeLayout.CENTER\_IN\_PARENT);

        View bgShadeContent = new View(mContext);

        bgShadeContent.setBackgroundDrawable(getResources().getDrawable(

                R.drawable.shade\_bg));

        bgShade.addView(bgShadeContent, bgParams);

        addView(bgShade, bgParams);

        addView(view, aboveParams);

        mSlidingView = view;

        mSlidingView.bringToFront();

    }

    @Override

    public void scrollTo(int x, int y) {

        super.scrollTo(x, y);

        postInvalidate();

    }

    @Override

    public void computeScroll() {

        if (!mScroller.isFinished()) {

            if (mScroller.computeScrollOffset()) {

                int oldX = mSlidingView.getScrollX();

                int oldY = mSlidingView.getScrollY();

                int x = mScroller.getCurrX();

                int y = mScroller.getCurrY();

                if (oldX != x || oldY != y) {

                    if (mSlidingView != null) {

                        mSlidingView.scrollTo(x, y);

                        if (x < 0)

                            bgShade.scrollTo(x + 20, y);// 背景阴影右偏

                        else

                            bgShade.scrollTo(x - 20, y);// 背景阴影左偏

                    }

                }

                invalidate();

            }

        }

    }

    private boolean canSlideLeft = true;

    private boolean canSlideRight = false;

    public void setCanSliding(boolean left, boolean right) {

        canSlideLeft = left;

        canSlideRight = right;

    }

    /\* 拦截touch事件 \*/

    @Override

    public boolean onInterceptTouchEvent(MotionEvent ev) {

        final int action = ev.getAction();

        final float x = ev.getX();

        final float y = ev.getY();

        switch (action) {

        case MotionEvent.ACTION\_DOWN:

            mLastMotionX = x;

            mLastMotionY = y;

            mIsBeingDragged = false;

            if (canSlideLeft) {

                mMenuView.setVisibility(View.VISIBLE);

                mDetailView.setVisibility(View.INVISIBLE);

            }

            if (canSlideRight) {

                mMenuView.setVisibility(View.INVISIBLE);

                mDetailView.setVisibility(View.VISIBLE);

            }

            break;

        case MotionEvent.ACTION\_MOVE:

            final float dx = x - mLastMotionX;

            final float xDiff = Math.abs(dx);

            final float yDiff = Math.abs(y - mLastMotionY);

            if (xDiff > mTouchSlop && xDiff > yDiff) {

                if (canSlideLeft) {

                    float oldScrollX = mSlidingView.getScrollX();

                    if (oldScrollX < 0) {

                        mIsBeingDragged = true;

                        mLastMotionX = x;

                    } else {

                        if (dx > 0) {

                            mIsBeingDragged = true;

                            mLastMotionX = x;

                        }

                    }

                } else if (canSlideRight) {

                    float oldScrollX = mSlidingView.getScrollX();

                    if (oldScrollX > 0) {

                        mIsBeingDragged = true;

                        mLastMotionX = x;

                    } else {

                        if (dx < 0) {

                            mIsBeingDragged = true;

                            mLastMotionX = x;

                        }

                    }

                }

            }

            break;

        }

        return mIsBeingDragged;

    }

    /\* 处理拦截后的touch事件 \*/

    @Override

    public boolean onTouchEvent(MotionEvent ev) {

        if (mVelocityTracker == null) {

            mVelocityTracker = VelocityTracker.obtain();

        }

        mVelocityTracker.addMovement(ev);

        final int action = ev.getAction();

        final float x = ev.getX();

        final float y = ev.getY();

        switch (action) {

        case MotionEvent.ACTION\_DOWN:

            if (!mScroller.isFinished()) {

                mScroller.abortAnimation();

            }

            mLastMotionX = x;

            mLastMotionY = y;

            if (mSlidingView.getScrollX() == -getMenuViewWidth()

                    && mLastMotionX < getMenuViewWidth()) {

                return false;

            }

            if (mSlidingView.getScrollX() == getDetailViewWidth()

                    && mLastMotionX > getMenuViewWidth()) {

                return false;

            }

            break;

        case MotionEvent.ACTION\_MOVE:

            if (mIsBeingDragged) {

                final float deltaX = mLastMotionX - x;

                mLastMotionX = x;

                float oldScrollX = mSlidingView.getScrollX();

                float scrollX = oldScrollX + deltaX;

                if (canSlideLeft) {

                    if (scrollX > 0)

                        scrollX = 0;

                }

                if (canSlideRight) {

                    if (scrollX < 0)

                        scrollX = 0;

                }

                if (deltaX < 0 && oldScrollX < 0) { // left view

                    final float leftBound = 0;

                    final float rightBound = -getMenuViewWidth();

                    if (scrollX > leftBound) {

                        scrollX = leftBound;

                    } else if (scrollX < rightBound) {

                        scrollX = rightBound;

                    }

                } else if (deltaX > 0 && oldScrollX > 0) { // right view

                    final float rightBound = getDetailViewWidth();

                    final float leftBound = 0;

                    if (scrollX < leftBound) {

                        scrollX = leftBound;

                    } else if (scrollX > rightBound) {

                        scrollX = rightBound;

                    }

                }

                if (mSlidingView != null) {

                    mSlidingView.scrollTo((int) scrollX,

                            mSlidingView.getScrollY());

                    if (scrollX < 0)

                        bgShade.scrollTo((int) scrollX + 20,

                                mSlidingView.getScrollY());

                    else

                        bgShade.scrollTo((int) scrollX - 20,

                                mSlidingView.getScrollY());

                }

            }

            break;

        case MotionEvent.ACTION\_CANCEL:

        case MotionEvent.ACTION\_UP:

            if (mIsBeingDragged) {

                final VelocityTracker velocityTracker = mVelocityTracker;

                velocityTracker.computeCurrentVelocity(100);

                float xVelocity = velocityTracker.getXVelocity();// 滑动的速度

                int oldScrollX = mSlidingView.getScrollX();

                int dx = 0;

                if (oldScrollX <= 0 && canSlideLeft) {// left view

                    if (xVelocity > VELOCITY) {

                        dx = -getMenuViewWidth() - oldScrollX;

                    } else if (xVelocity < -VELOCITY) {

                        dx = -oldScrollX;

                        if (hasClickLeft) {

                            hasClickLeft = false;

                            setCanSliding(tCanSlideLeft, tCanSlideRight);

                        }

                    } else if (oldScrollX < -getMenuViewWidth() / 2) {

                        dx = -getMenuViewWidth() - oldScrollX;

                    } else if (oldScrollX >= -getMenuViewWidth() / 2) {

                        dx = -oldScrollX;

                        if (hasClickLeft) {

                            hasClickLeft = false;

                            setCanSliding(tCanSlideLeft, tCanSlideRight);

                        }

                    }

                }

                if (oldScrollX >= 0 && canSlideRight) {

                    if (xVelocity < -VELOCITY) {

                        dx = getDetailViewWidth() - oldScrollX;

                    } else if (xVelocity > VELOCITY) {

                        dx = -oldScrollX;

                        if (hasClickRight) {

                            hasClickRight = false;

                            setCanSliding(tCanSlideLeft, tCanSlideRight);

                        }

                    } else if (oldScrollX > getDetailViewWidth() / 2) {

                        dx = getDetailViewWidth() - oldScrollX;

                    } else if (oldScrollX <= getDetailViewWidth() / 2) {

                        dx = -oldScrollX;

                        if (hasClickRight) {

                            hasClickRight = false;

                            setCanSliding(tCanSlideLeft, tCanSlideRight);

                        }

                    }

                }

                smoothScrollTo(dx);

            }

            break;

        }

        return true;

    }

    private int getMenuViewWidth() {

        if (mMenuView == null) {

            return 0;

        }

        return mMenuView.getWidth();

    }

    private int getDetailViewWidth() {

        if (mDetailView == null) {

            return 0;

        }

        return mDetailView.getWidth();

    }

    void smoothScrollTo(int dx) {

        int duration = 500;

        int oldScrollX = mSlidingView.getScrollX();

        mScroller.startScroll(oldScrollX, mSlidingView.getScrollY(), dx,

                mSlidingView.getScrollY(), duration);

        invalidate();

    }

    /\*

     \* 显示左侧边的view

     \*/

    public void showLeftView() {

        int menuWidth = mMenuView.getWidth();

        int oldScrollX = mSlidingView.getScrollX();

        if (oldScrollX == 0) {

            mMenuView.setVisibility(View.VISIBLE);

            mDetailView.setVisibility(View.INVISIBLE);

            smoothScrollTo(-menuWidth);

            tCanSlideLeft = canSlideLeft;

            tCanSlideRight = canSlideRight;

            hasClickLeft = true;

            setCanSliding(true, false);

        } else if (oldScrollX == -menuWidth) {

            smoothScrollTo(menuWidth);

            if (hasClickLeft) {

                hasClickLeft = false;

                setCanSliding(tCanSlideLeft, tCanSlideRight);

            }

        }

    }

    /\* 显示右侧边的view \*/

    public void showRightView() {

        int menuWidth = mDetailView.getWidth();

        int oldScrollX = mSlidingView.getScrollX();

        if (oldScrollX == 0) {

            mMenuView.setVisibility(View.INVISIBLE);

            mDetailView.setVisibility(View.VISIBLE);

            smoothScrollTo(menuWidth);

            tCanSlideLeft = canSlideLeft;

            tCanSlideRight = canSlideRight;

            hasClickRight = true;

            setCanSliding(false, true);

        } else if (oldScrollX == menuWidth) {

            smoothScrollTo(-menuWidth);

            if (hasClickRight) {

                hasClickRight = false;

                setCanSliding(tCanSlideLeft, tCanSlideRight);

            }

        }

    }

}