# 鸿蒙开源第三方组件——SlidingMenu\_ohos侧滑菜单组件

## 前言

基于安卓平台的SlidingMenu侧滑菜单组件（https://github.com/jfeinstein10/SlidingMenu），实现了鸿蒙化迁移和重构，代码已经开源到（https://gitee.com/isrc\_ohos/sliding-menu\_ohos），欢迎各位下载使用并提出宝贵意见！

## 背景

          SlidingMenu\_ohos提供了一个侧滑菜单的导航框架，使菜单可以隐藏在手机屏幕的左侧、右侧或左右两侧。当用户使用时，通过左滑或者右滑的方式调出，既节省了主屏幕的空间，也方便用户操作，在很多主流APP中都有广泛的应用。

## 效果展示

由于菜单从左右两侧调出的显示效果相似，此处仅以菜单从左侧调出为例进行效果展示。

组件未启用时，应用显示主页面。单指触摸屏幕左侧并逐渐向右滑动，菜单页面逐渐显示，主页面逐渐隐藏。向右滑动的距离超过某个阈值时，菜单页面全部显示，效果如图1所示。



图1 菜单展示和隐藏效果图

## Sample解析

Sample部分的内容较为简单，主要包含两个部分。一是创建SlidingMenu\_ohos组件的对象，可根据用户的实际需求，调用Library的接口，对组件的具体属性进行设置。二是将设置好的组件添加到Ability中。下面将详细介绍组件的使用方法。

1、导入SlidingMenu类

import com.jeremyfeinstein.slidingmenu.lib.SlidingMenu;

2、设置Ability的布局

此布局用作为主页面的布局，在组件隐藏的时候显示。

DirectionalLayout directionalLayout =

(DirectionalLayout)LayoutScatter.getInstance(this).parse(ResourceTable.Layout\_activity\_main,null,false); setUIContent(directionalLayout);

3、实例化组件的对象

SlidingMenu slidingMenu = null;

try {

*//初始化SlidingMenu实例*

slidingMenu = new SlidingMenu(this);

} catch (IOException e) {

e.printStackTrace();

} catch (NotExistException e) {

e.printStackTrace();

}

4、设置组件属性

此步骤可以根据具体需求，设置组件的位置、触发范围、布局、最大宽度等属性。

*//设置菜单放置位置*

slidingMenu.setMode(SlidingMenu.LEFT);

*//设置组件的触发范围*

slidingMenu.setTouchScale(100);

*//设置组件的布局*

slidingMenu.setMenu(ResourceTable.Layout\_layout\_left\_menu);

*//设置菜单最大宽度*

slidingMenu.setMenuWidth(800);

5、关联Ability

attachToAbility()方法是Library提供的重要方法，用于将菜单组件关联到Ability。其参数SLIDING\_WINDOW和SLIDING\_CONTENT是菜单的不同模式，SLIDING\_WINDOW模式下的菜单包含Title / ActionBar部分，菜单需在整个手机页面上显示，如图2所示；SLIDING\_CONTENT模式下的菜单不包括包含Title / ActionBar部分，菜单可以在手机页面的局部范围内显示，如图3所示。

try {

*//关联Ability，获取页面展示根节点*

slidingMenu.attachToAbility(directionalLayout,this, SlidingMenu.SLIDING\_WINDOW);

} catch (NotExistException e) {

e.printStackTrace();

} catch (WrongTypeException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}



 图2 SLIDING\_WINDOW展示效果图



图3 SLIDING\_CONTENT展示效果图

## Library解析

Library的工程结构如下图所示，CustomViewAbove表示主页面，CustomViewBehind表示菜单页面，SlidingMenu主要用于控制主页面位于菜单页面的上方，还可以设置菜单的宽度、触发范围、显示模式等属性。为了方便解释，以下均以手指从左侧触摸屏幕并向右滑动为例进行讲解，菜单均采用SLIDING\_WINDOW的显示模式。

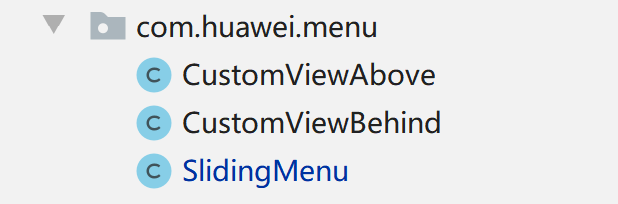


图4 Library的工程结构

1、CustomViewAbove主页面

CustomViewAbove需要监听触摸、移动、抬起和取消等Touch事件，并记录手指滑动的距离和速度。

（1）对Touch事件的处理

Touch事件决定了菜单的显示、移动和隐藏。例如：在菜单的触发范围内，手指向右滑动（POINT\_MOVE）时，菜单会跟随滑动到手指所在位置。手指抬起（PRIMARY\_POINT\_UP）或者取消滑动（CANCEL）时，会依据手指滑动的距离和速度决定菜单页面的下一状态是全部隐藏还是全部显示。

switch (action) {

*//按下*

case TouchEvent.PRIMARY\_POINT\_DOWN:

.....

mInitialMotionX=mLastMotionX=ev.getPointerPosition(mActivePointerId).getX();

break;

*//滑动*

case TouchEvent.POINT\_MOVE:

......

*//菜单滑动到此时手指所在位置（x）*

left\_scrollto(x);

break;

*//抬起*

case TouchEvent.PRIMARY\_POINT\_UP:

......

*//获得菜单的下一状态（全屏显示或者全部隐藏）*

int nextPage = determineTargetPage(pageOffset, initialVelocity,totalDelta);

*//设置菜单的下一状态*

setCurrentItemInternal(nextPage,initialVelocity);

......

endDrag();

break;

*//取消*

case TouchEvent.CANCEL:

......

*//根据菜单当前状态mCurItem设置菜单下一状态*

setCurrentItemInternal(mCurItem);

*//结束拖动*

endDrag();

break;

}

（2）对滑动的距离和速度的处理

手指抬起时，滑动的速度和距离分别大于最小滑动速度和最小移动距离，判定此时的操作为快速拖动，菜单立即弹出并全部显示，如图5所示。

private int determineTargetPage(float pageOffset, int velocity, int deltaX) {

*//获得当前菜单状态，0：左侧菜单正在展示，1：菜单隐藏，2：右侧菜单正在展示*

int targetPage = getCurrentItem();

*//针对快速拖动的判断*

if (Math.abs(deltaX) > mFlingDistance && Math.abs(velocity) > mMinimumVelocity) {

if (velocity > 0 && deltaX > 0) {

targetPage -= 1;

} else if (velocity < 0 && deltaX < 0){

targetPage += 1;

}

}

}



图5快速拖动效果图

 当手指抬起并且不满足快速拖动标准时，需要根据滑动距离判断菜单的隐藏或显示。若菜单已展开的部分超过自身宽度的1/2，菜单立即弹出全部显示，，效果图如图1所示；若不足自身宽度的1/2，则立即弹回全部隐藏，效果图如图6所示。

*//获得当前菜单状态，0：左侧菜单正在展示，1：菜单隐藏，2：右侧菜单正在展示*

switch (mCurItem){

case 0:

targetPage=1-Math.round(pageOffset);

break;

case 1:

*//菜单隐藏时，首先要判断此时菜单的放置状态是左侧还是右侧*

if(current\_state == SlidingMenu.LEFT){

targetPage = Math.round(1-pageOffset);

}

if(current\_state == SlidingMenu.RIGHT){

targetPage = Math.round(1+pageOffset);

}

break;

case 2:

targetPage = Math.round(1+pageOffset);

break;

}



 图6 缓慢拖动效果图

（3）菜单显示和隐藏的实现

主页面的左侧边线与手指的位置绑定，当手指向右滑动时，主页面也会随手指向右滑动，在这个过程中菜单页面渐渐展示出来，实现菜单页面随手指滑动慢慢展开的视觉效果。

void setCurrentItemInternal(int item,int velocity) {

*//获得菜单的目标状态*

item = mViewBehind.getMenuPage(item);

mCurItem = item;

final int destX = getDestScrollX(mCurItem);

*/\*菜单放置状态为左侧，通过设置主页面的位置实现菜单的弹出展示或弹回隐藏*

*1.destX=0,主页面左侧边线与屏幕左侧边线对齐，菜单被全部遮挡，实现菜单弹回隐藏*

*2.destX=MenuWidth，主页面左侧边线向右移动与菜单总宽度相等的距离，实现菜单弹出展示\*/*

if (mViewBehind.getMode() == SlidingMenu.LEFT) {

mContent.setLeft(destX);

mViewBehind.scrollBehindTo(destX);

}

......

}

*// 菜单放置在左侧时的菜单滑动操作*

public void left\_scrollto(float x) {

*//当menu的展示宽度大于最大宽度时仅展示最大宽度*

if(x>getMenuWidth()){

x=getMenuWidth();

}

*//主页面（主页面左侧边线）和菜单（菜单右侧边线）分别移动到指定位置X*

mContent.setLeft((int)x);

mViewBehind.scrollBehindTo((int)x);

}

2、CustomViewBehind 菜单页面

CustomViewBehind为菜单页面，逻辑相比于主页面简单许多。主要负责根据主页面中的Touch事件改变自身状态值，同时向外暴露接口，用于设置或者获取菜单页面的最大宽度、自身状态等属性。

*// 设置菜单最大宽度*

public void setMenuWidth(int menuWidth) {

this.menuWidth = menuWidth;

}

*// 获得菜单最大宽度*

public int getMenuWidth() {

return menuWidth;

}

3.  SlidingMenu

分别实例化CustomViewAbove和CustomViewBehind的对象，并按照主页面在上菜单页面在下的顺序分别添加到SlidingMenu的容器中。

*//添加菜单子控件*

addComponent(mViewBehind, behindParams);

*//添加主页面子控件*

addComponent(mViewAbove, aboveParams);

## 项目贡献人

徐泽鑫 郑森文 朱伟 陈美汝 王佳思 张馨心