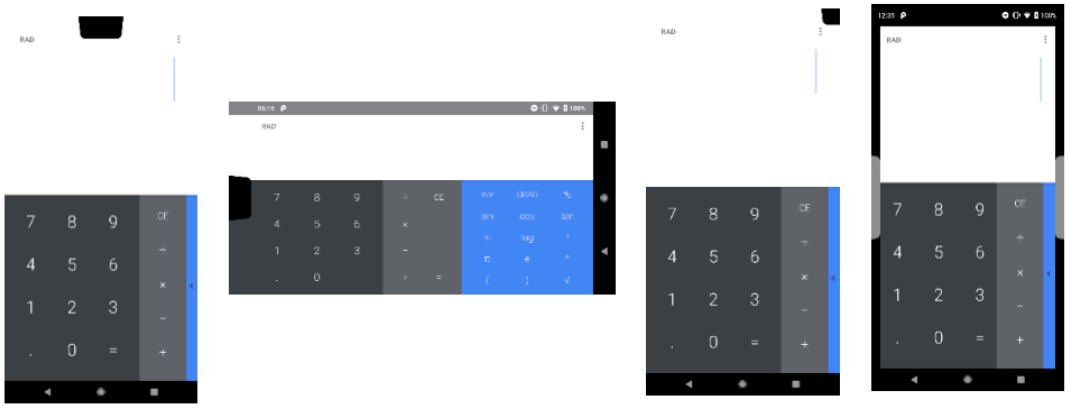
Android P版本刘海屏适配指导

1 背景介绍

谷歌对于刘海屏的设计约束：

1. 刘海区域不能超出系统状态栏的显示区域，刘海高度<=系统状态栏高度
2. 允许屏幕上有多个刘海，但不支持单边有多个刘海
3. **刘海位置可以在屏幕的短边、角上、或者长边。如果刘海在角上，系统认为是在短边**
4. 支持应用界面延伸到短边的刘海区域内。不支持延伸到长边的刘海区域内，长边刘海区留黑

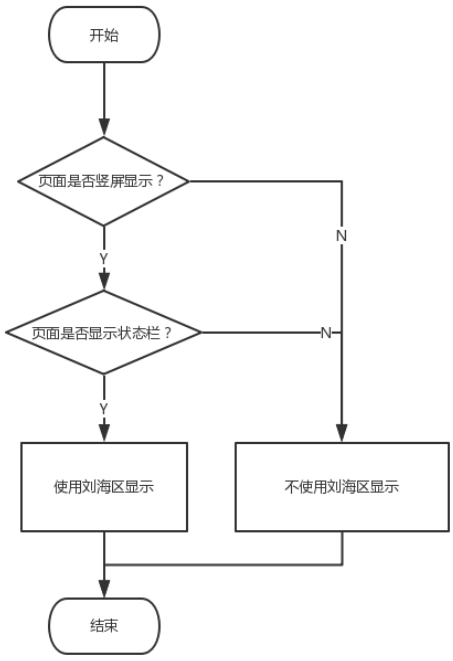


2 刘海屏的 API 介绍

2.1 窗口对刘海区域使用申明 API：layoutInDisplayCutoutMode

1. 默认值：LAYOUT_IN_DISPLAY_CUTOUT_MODE_DEFAULT

✓ 说明：应用不使用 API 或使用默认 API，系统将根据窗口类型自动决策应用是否使用刘海区域，如应用为全屏应用（设置了FLAG_FULLSCRREN 或 SYSTEMUI_FLAG_FULLSCREEN）将不使用，否则使用。



✓ 系统默认处理规则图示：

全屏界面不使用刘海区

有状态栏非全屏界面使用刘海区显示

横屏界面统一不使用刘海区显示

2. 应用申明使用刘海区显示：

LAYOUT_IN_DISPLAY_CUTOUT_MODE_SHORT_EDGES

✓ 说明：窗口申明使用短边刘海区域

注：P版本初期提供 LAYOUT_IN_DISPLAY_CUTOUT_MODE_ALWAYS 标记，但 DP2 升级时，增加了对多个刘海的支持，Google 只允许应用使用短边刘海区，不支持应用使用长边刘海区。

✓ 参考代码：

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
```

```
        WindowManager.LayoutParams lp = getWindow().getAttributes();
        lp.layoutInDisplayCutoutMode =
    WindowManager.LayoutParams.LAYOUT_IN_DISPLAY_CUTOUT_MODE_SHORT_EDGES;
        getWindow().setAttributes(lp);
    }
```

✓ 设置使用刘海区显示的影响：对于所有全屏页面和所有横屏页面是有影响的，设置之后这些页面才能使用刘海区显示

设置默认值

设置使用值

设置默认值

设置使用值

3. 应用申明不使用刘海区显示：

LAYOUT_IN_DISPLAY_CUTOUT_MODE_NEVER

✓ 说明：窗口声明永远不使用刘海区域

✓ 设置不使用刘海区显示的影响：主要影响使用沉浸式状态栏和导航栏的页面有影

响，这种场景应用的布局也是不使用刘海区显示的

默认设置

设置不使用刘海区显示

默认和设置不使用刘海应用布局可显示区域大小是不一样的

2.2 刘海设备及刘海区域 API

1. 判断是否为刘海设备方法

```
DisplayCutout cutout = windowInsets.getDisplayCutout();
if (cutout == null) {
    Log.e(TAG, "cutout==null, is not notch screen");
} else {
    Log.e(TAG, "cutout ! =null, is notch screen");
}
```

2. 获取刘海尺寸接口和安全显示区域接口

所属类	方法	接口说明
android.view.DisplayCutout	List<Rect> getBoundingRects()	返回Rects的列表，每个Rects都是显示屏上非功能区域的边界矩形。设备的**每个短边最多只有一个非功能区域，而长边上则没有**。
android.view.DisplayCutout	int getSafeInsetBottom()	返回安全区域距离屏幕底部的距离，单位是px
android.view.DisplayCutout	int getSafeInsetLeft ()	返回安全区域距离屏幕左边的距离，单位是px
android.view.DisplayCutout	int getSafeInsetRight ()	返回安全区域距离屏幕右边的距离，单位是px
android.view.DisplayCutout	int getSafeInsetTop ()	返回安全区域距离屏幕顶部的距离，单位是px

备注：刘海个数可以是多个：

Line 6291: 05-24 11:27:04.517 11036 11036 E Cutout_test: rect size:2

Line 6292: 05-24 11:27:04.517 11036 11036 E Cutout_test: cutout.getSafeInsetTop():84, cutout.getSafeInsetBottom():84, cutout.getSafeInsetLeft():0, cutout.getSafeInsetRight():0, cutout.rects:Rect**(351, 0 - 729, 84)**，cutout.rects:Rect**(351, 1836 - 729, 1920)**

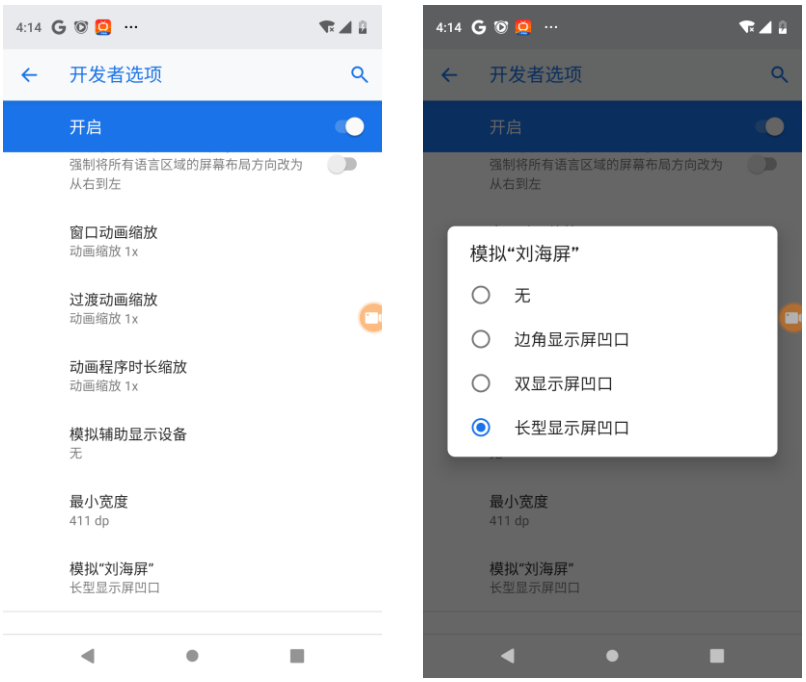
3 适配指导

3.1 使用非刘海屏手机模拟调试

Pixcel非刘海屏手机或者模拟器，通过开发者选项开启模拟刘海屏进行调试：

1. 在开发人员选项屏幕中，向下滚动到绘图部分，然后选择模拟“刘海屏”。

2. 选择刘海尺寸信息



3.2 应用界面需要**使用**刘海区显示适配指导

1. 使用谷歌P版本刘海屏接口需要应用修改（compileSdkVersion和 targetSdkVersion**任意一个**修改到28就可以使用了）：

✓ compileSdkVersion 28

✓ targetSdkVersion 28

2. 页面设置全屏属性SYSTEM_UI_FLAG_FULLSCREEN和 SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN，并设置 LAYOUT_IN_DISPLAY_CUTOUT_MODE_SHORT_EDGES使用刘海区显示

```
getWindow().getDecorView().setSystemUiVisibility(View.SYSTEM_UI_FLAG_FULLSCREEN |
View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
    WindowManager.LayoutParams lp = getWindow().getAttributes();
    lp.layoutInDisplayCutoutMode =

WindowManager.LayoutParams.LAYOUT_IN_DISPLAY_CUTOUT_MODE_SHORT_EDGES;
```

```
    getWindow().setAttributes(lp);
}
```

备注：应用如果想让布局延伸到刘海区显示，除了设置LAYOUT_IN_DISPLAY_CUTOUT_MODE_SHORT_EDGES属性使用刘海区显示，还需要设置SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN

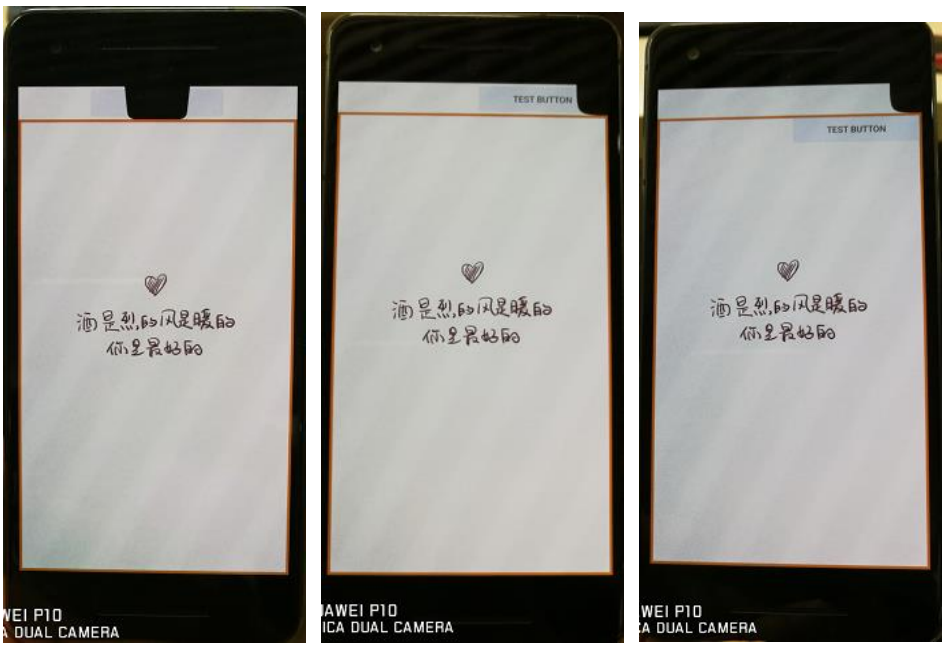
没有设置SYSTEM_UI_FLAG_LAYOUT_FULLSCREE

设置SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN

3. 获取布局安全区域，并调整布局，除背景外的其他布局信息放在安全区域内显示：

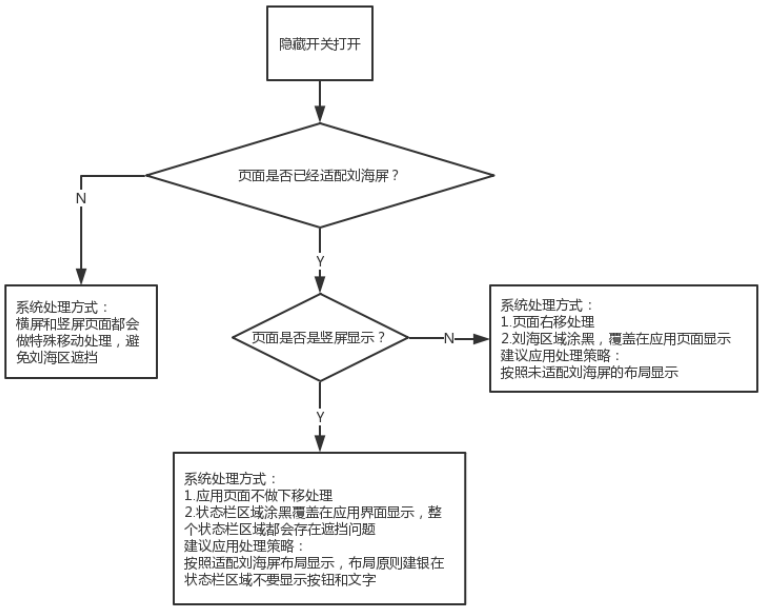
```
contentView.setOnApplyWindowInsetsListener(new View.OnApplyWindowInsetsListener() {
    @Override
    public WindowInsets onApplyWindowInsets(View view, WindowInsets windowInsets) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
            DisplayCutout cutout = windowInsets.getDisplayCutout();
            if (cutout == null) {
                Log.e(TAG, "cutout==null, is not notch screen");
            } else {
                List<Rect> rects = cutout.getBoundingRects();
                if (rects == null || rects.size() == 0) {
                    Log.e(TAG, "rects==null || rects.size()==0, is not notch screen");
                } else {
                    contentView.setPadding(cutout.getSafeInsetLeft(),
cutout.getSafeInsetTop(),
                            cutout.getSafeInsetRight(), cutout.getSafeInsetBottom());
                }
            }
        }
        return windowInsets;
    }
});
```

布局原则：页面背景可以延伸到刘海区危险区域显示，其他布局建议全部放在安全区域布局显示，因为刘海的位置不是固定的，从下图可以看到只有布局放在安全区域（橙色边框标示的区域）显示才能保证不被刘海遮挡。



4. 适配使用刘海区显示，考虑华为独有隐藏刘海的需求

✓ 隐藏开关打开之后，显示规格：



✓ 读取开关状态调用范例：

```
public static final String DISPLAY_NOTCH_STATUS = "display_notch_status";
int mIsNotchSwitchOpen =
Settings.Secure.getInt(getContentResolver(),DISPLAY_NOTCH_STATUS, 0);//0表示“默认”，1表示“隐藏显示区域”
```

✓ 横屏页面适配和谷歌原生差异的说明：

在华为刘海屏手机应用适配使用刘海区显示，竖屏页面和谷歌原生适配没有太大区别，但是应用的**横屏页面**判断是否需要设置布局显示在安全区域判断条件要修改为：手机为刘海屏手机并且隐藏开关未打开，**参考代码**：

```
mIsNotchSwitchOpen =
Settings.Secure.getInt(getContentResolver(),DISPLAY_NOTCH_STATUS, 0);
contentView = findViewById(R.id.safearea);
contentView.setOnApplyWindowInsetsListener(new
View.OnApplyWindowInsetsListener() {
```

```
    @Override
    public WindowInsets onApplyWindowInsets(View view, WindowInsets
windowInsets) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {

            DisplayCutout cutout = windowInsets.getDisplayCutout();
            if(cutout != null && mIsNotchSwitchOpen == 0) {//手机是刘海屏&&隐藏刘
海开关关闭
                contentView.setPadding(cutout.getSafeInsetLeft(),
cutout.getSafeInsetTop(),
                        cutout.getSafeInsetRight(), cutout.getSafeInsetBottom());
            }

        }
        return windowInsets;
    }
});
```

非刘海屏布局

刘海屏布局

非刘海屏布局在隐藏刘海功能开启之后的效果

刘海屏布局在隐藏刘海功能开启之后的效果

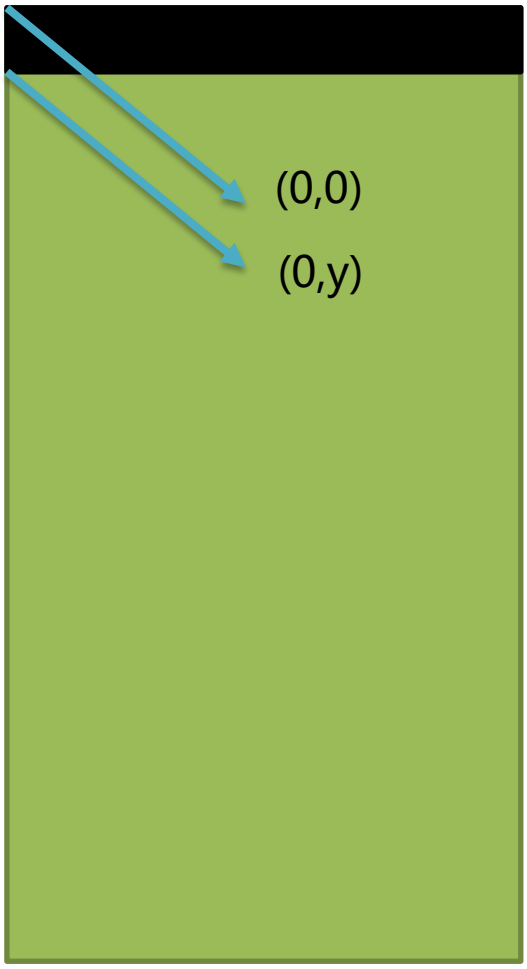
5. 适配使用刘海区显示，应用页面如果支持屏幕旋转，需要考虑横竖屏0度、90度、180度、270度的场景，请旋转手机在这四个场景都测试一下看看布局是否存在遮挡问题；

3.3 应用界面**不想使用**刘海区显示适配指导

应用布局如果不需要延伸到刘海区域显示也是需要适配的，需要考虑非刘海屏和刘海屏手机的差异，否则很容易出现各种UI或者功能问题。

1. 应用全屏页面window坐标原点的差异

- ✓ 非刘海屏，应用全屏页面的window坐标系的坐标原点和整个手机屏幕的坐标系一致，都是(0,0)
- ✓ 刘海屏，应用全屏页面的window坐标系的坐标原点因为系统特殊下移处理发生了变化，已经不再是(0,0)，而是(0,y)---y表示偏移量：状态栏高度



- ✓ 适配指导：在刘海屏手机处理坐标位置的时候，需要考虑**坐标系统一**，要么都用

window的坐标系，要么都用屏幕坐标系，否则计算结果可能存在一个状态栏高度的偏差。

坐标系	接口
窗口坐标系	view.getGlobalVisibleRect() MotionEvent.getX() MotionEvent.getY()
屏幕坐标系	view.getLocationOnScreen() MotionEvent.getRawX() MotionEvent.getRawY()

2. 状态栏高度的差异

不能再假设系统状态栏是固定高度，刘海屏的手机系统状态栏的高度和刘海的高度是有关系的，需要满足：系统状态栏高度>=刘海高度。所以应用在使用沉浸式状态栏的时候，不能把状态栏高度写死，否则就会出现一些UI问题：

计算系统状态栏高度代码：

```
public static int getStatusBarHeight(Context context) {
    int result = 0;
    int resourceId = context.getResources().getIdentifier("status_bar_height",
            "dimen", "android");
    if (resourceId > 0) {
        result = context.getResources().getDimensionPixelSize(resourceId);
    }
    return result;
}
```

3. 计算屏幕分辨率的差异

```
DisplayMetrics dm = getResources().getDisplayMetrics();

int w_screen = dm.widthPixels;

int h_screen = dm.heightPixels;
```

屏幕高度	非刘海屏手机	刘海屏手机
h_screen	屏幕总高度-导航栏高度	屏幕总高度-导航栏高度-刘海高度