


## 如何处理手势冲突 | 手势导航连载 (三)



谷歌开发者   
已认证的官方帐号

已关注

8 人赞同了该文章



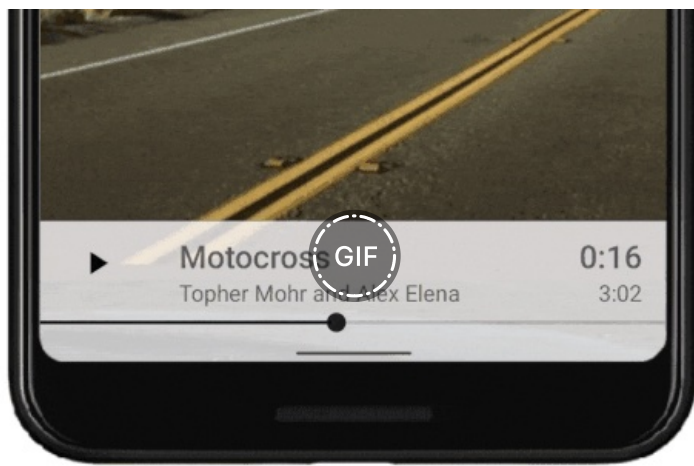
作者 / Chris Banes, Android 开发者关系团队工程师

我们将在近期为大家带来一个关于 "手势导航" 的系列连载，本文是手势导航连载的第三篇，如果您希望查看前两篇文章，请点击下方链接：

- [开启全面屏体验 | 手势导航 \(一\)](#)
- [处理视觉冲突 | 手势导航 \(二\)](#)

在上一篇文章中，我们讨论完了从边到边绘制应用内容。从这一篇文章开始我们将介绍如何处理您的应用和 Android 10 中新引入的系统交互手势之间的冲突。

首先让我们来理解一下什么是 "手势冲突 (gesture conflict)"。我们来看一个例子，比如下面这个音乐播放应用，该应用允许用户通过拖动进度条 (SeekBar) 来快进或快退当前歌曲。



不幸的是，进度条太靠近主屏手势区域 (Home Screen Gesture Area)，所以当用户在该区域滑动时，系统把它错误地判断为用户是要执行快速切换应用的操作，这也会让用户感到困惑。支持手势导航的任何屏幕边缘区域都可能发生类似情况。有很多可能导致冲突的例子，例如：导航抽屉 (DrawerLayout)、多图展示 (ViewPager)、进度条 (SeekBar)，甚至在列表上进行滑动操作也有可能出现冲突。

那么，如何解决这个问题呢？我们准备了一张流程图帮助大家快速做出决策：



△ 请点击图片放大查看

\* 注解: 非粘性沉浸模式: 用户可以通过在系统栏上滑动来退出沉浸模式。

粘性沉浸模式: 用户可以通过在系统栏上滑动来暂时退出沉浸模式

这里我们向您进一步解释一下流程图里的内容。

#### 问题 1: 应用需要隐藏导航栏或状态栏吗?

流程图里的第一个问题, 询问您应用的主要使用场景是否需要隐藏导航和/或状态栏。所谓 "隐藏", 是指让它们根本不可见。这并不意味着让您的应用实现从边到边的全屏状态。

需要隐藏的原因可能包括:

- 使用`FLAG_FULLSCREEN` WindowManager 开关。注意, 这个效果也可以通过 `android:windowFullscreen` 主题设置来实现, 或者扩展一个 `Theme.XXX.Fullscreen` 的衍生控件。
- 使用`SYSTEM_UI_FLAG_FULLSCREEN` 这个系统 UI 可见性开关。
- 使用沉浸模式中的系统 UI 可见性开关: `SYSTEM_UI_FLAG_IMMERSIVE`或 `SYSTEM_UI_FLAG_IMMERSIVE_STICKY`。

一般来说, 游戏、视频播放器、照片应用、绘图应用等会在这个问题中回答 "是"。

#### 问题 2: 主要的 UI 需要在交互区域内/附近使用滑动操作吗?

这个问题是在询问, 应用的界面是否在手势导航交互区域内或附近包含任何需要用户滑动操作的组件。(包括在后退和返回主屏按钮区域滑动)



- 游戏屏幕上的控件往往非常靠近屏幕左/右边缘，或靠近屏幕底部。
- 某些游戏需要在屏幕上滑动操作一个元素，而这个元素可能出现在屏幕的任何位置，例如平台动作类的游戏。

除了游戏之外，有一些常见的 UI 也可能在这里回答 "是"：

- 图片裁切 UI，其中用于裁切图片的控制点可能位于屏幕左/右边缘附近。
- 绘图应用，用户可以在屏幕画布上绘图 (自然也是滑动操作)。

### 问题 3: 常用的视图/控件位于手势交互区域内/附近吗？

这个问题应该简单一些。注意，这个问题也包括那些占据屏幕较大区域，且包括了手势交互区域的视图/控件。比如 `DrawerLayout` 或尺寸较大的 `ViewPager`。

### 问题 4: 该视图/控件需要滑动拖动交互吗？

这个紧接着问题 3。在问题 3 中回答 "是" 的视图，是否需要用户在其上滑动或拖拽？

有不少用例会在本题回答 "是"：包括前面提到的进度条、底部弹出菜单 ([Bottom Sheet](#)) 或者可以通过滑动打开的弹出菜单 ([PopupMenu](#))。

### 问题 5: 该视图/控件大部分位于手势交互区域内吗？

紧接着问题 4，进一步确认该视图是否完全或大部分位于手势交互区域内。

如果您的视图放置在一个可滚动操作的容器 (如 [RecyclerView](#)) 中，那么请这么理解这个问题：该视图是否完全或大部分位于手势交互区域中？如果用户可以将视图滚动到手势交互区域之外，则应该视为没有交互冲突。

您也许已经注意到，在流程图中多图显示控件 (`ViewPager`) 在此处回答 "否"。这是因为与整个视图的宽度相比，屏幕左右侧的手势交互区域宽度相对较小 (默认为每边 20dp)。一般来说手机坚持时屏幕宽度约为 360dp，也就是说，在约为 320dp 的范围内，用户的滑动操作不受影响 (占总宽度的近 90%)。即使考虑加上了内外边距的情况，用户仍然可以正常通过滑动操作来翻看里面的图片。

### 问题 6: 该视图/控件是否和强制系统手势交互区域重叠？

最后一个问题询问该控件是否位于系统强制手势导航交互区域内。如果您读过我们之前的文章，应该还记得 "强制系统手势交互区" 是指系统手势始终被优先处理的屏幕区域。

对 Android 10 来说，强制交互区域只有一个，那就是屏幕底部。该区域内的滑动操作能让用户返回主屏或访问最近使用的其他应用。这个强制交互区域可能会在将来的平台版本中发生变化，但现在我们只需要考虑屏幕底部即可。

出现这种重叠的常见的例子：

- 非模态的底部弹出菜单，因为这种菜单常常会在屏幕底部折叠为一个较小的视图，而且还需要滑动操作。
- 屏幕底部的水平页面切换，例如软键盘里选择不同表情包的 UI。

OK，现在我已经解释了流程图中的问题，下面我们来详细说说流程图中给出的解决方案。

## 解决方案 1: 无需处理手势冲突

最简单的 "解决方案"，只需要.....什么都不做！

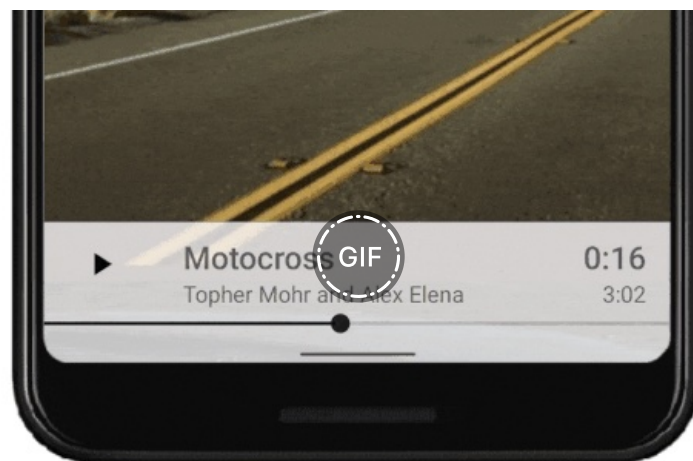
当然，也许您还可以 (参考接下来的几种解决方案) 做点优化，但在启用了手势导航的应用中，您

## 解决方案 2: 将该视图/控件移出手势交互区域

我们在上一篇文章有提到，可以用 Insets 区域来告知应用系统手势区域在屏幕中的位置。我们可以用来解决手势冲突的一种方法是，将出现冲突的视图移出手势导航交互区域。这对于屏幕底部附近的视图尤其重要，因为该区域是系统强制手势交互区域，并且应用无法在该区域使用热区切出 API。

这里让我们回到之前提到的音乐播放器示例。它包含一个位于屏幕底部的进度条，允许用户快进和快退歌曲。

但是，当用户尝试快进和快退歌曲时，会发生这种情况：

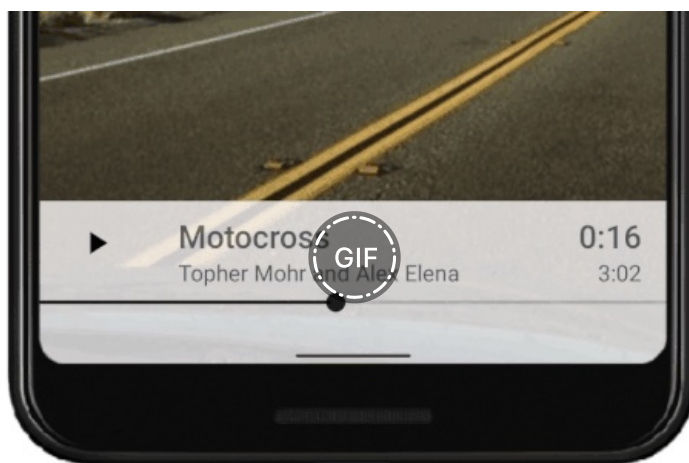


发生这种情况是因为，屏幕底部的系统手势交互区域与进度条重叠了，而在这里系统手势优先级更高。系统手势区域如下图所示：

△ 从蓝色区域向屏幕中间滑动相当于 &quot;返回&quot; 按钮；从红色区域向上滑动则是返回主屏，注意红色区域即为系统强制手势交互区域

### 简单的解法

这个问题最简单的解决方案是，添加一些内/外边距，将进度条向上推到手势区域之外。就像这样：

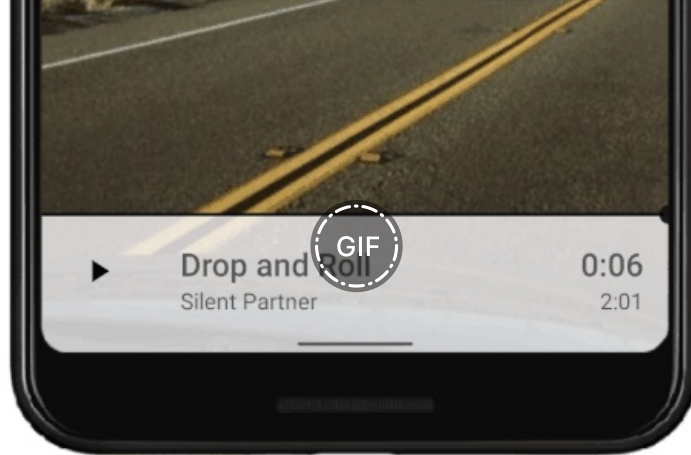


△ 进度条向上移动后不再出现冲突

为了实现这一点，我们需要使用 API 29 和 Jetpack Core 库 v1.2.0 (当前为 alpha 版) 中提供的新系统交互区 API。如下方代码，我们给进度条增加了底边距，增加的值正好是系统强制交互区的高度：

```
ViewCompat.setOnApplyWindowInsetsListener(seekBar) { view, insets ->
    // We'll set the views bottom padding to be the same
    // value as the system gesture insets bottom value
    view.updatePadding(
        bottom = insets.systemGestureInsets.bottom
    )
    insets
```





在上图中，由于进度条的播放头正好位于右侧手势区内，因此系统认为用户正在用手势执行 "返回" 操作，因此显示了 "向后" 的箭头。这时就会让用户感到困惑，因为他们可能并不想后退。出现这种冲突时，我们就可以使用上面提到的手势区域排除 API 来解决。

手势区域排除 API 通常会在两个地方被调用: 当视图被布局时 (onLayout), 或是当视图被绘制时 (onDraw)。您的视图会传入一个 List<Rect>, 其中包含应该切出 (即不响应系统手势) 的矩形区域。如前所述, 这些矩形须位于视图自己的坐标系中。

通常, 您会创建一个类似于下面的方法, 该方法会在 onLayout() 和/或 onDraw() 时被调用:

```
private val gestureExclusionRects = mutableListOf<Rect>()

private fun updateGestureExclusion() {
    // Skip this call if we're not running on Android 10+
    if (Build.VERSION.SDK_INT < 29) return

    // First, lets clear out any existing rectangles
    gestureExclusionRects.clear()

    // Now lets work out which areas should be excluded. For a SeekBar this will
    // be the bounds of the thumb drawable.
    thumb?.also { t ->
        gestureExclusionRects += t.copyBounds()
    }

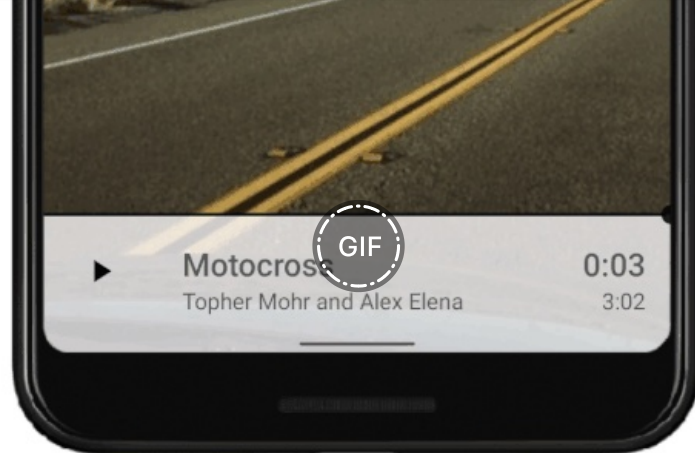
    // If we had other elements in this view near the edges, we could exclude them
    // here too, by adding their bounds to the list

    // Finally pass our updated list of rectangles to the system
    systemGestureExclusionRects = gestureExclusionRects
}
```

- 上例的完整代码[gist.github.com/chrisba...](https://gist.github.com/chrisba...)

做完这个 "切出" 操作后, 在屏幕边缘附近进行快进/快退操作就没有问题了:





注意: SeekBar 实际上会在 Android 10 中自动为您执行上述切出操作, 因此您无需在 Seekbar 中这么做。这里只是作为示例向您展示处理冲突的做法。

### 限制条件

尽管手势区域排除 API 似乎是解决所有手势冲突的完美方案, 但实际上并非如此。通过使用这个 API, 您实际上在声明应用的手势比 "返回" 等系统操作更重要。这个做法我们只建议您在没有其他解决方案时采用。

由于这个 API 会一定程度上破坏用户习惯的操作, 因此系统做出了限制: 屏幕的每个边缘最多只能被应用切除 200dp。

开发者听到这个限制时, 常会提出以下问题:

### 为什么要有限制?

我们认为, 开发者需要尽量确保用户使用一致的操作来与系统进行交互, 如从边缘向内滑动进行返回。注意是在整个设备上, 而不仅仅是在一个应用中保持一致性。这个限制看似严厉, 但如果一个应用能够让屏幕的整个边缘都不响应系统手势, 就会让用户感到困惑, 这个应用也极有可能被用户卸载。

再次强调, 系统导航必须始终保持一致性和可用性。

### 为什么是 200dp?

200dp 背后的决策逻辑非常简单。正如我们前面提到的, 手势区域排除 API 只有在万不得已的情况下才可以使用, 因此我们计算了可能需要应用这套机制的触摸对象的面积。触摸对象的最小推荐尺寸是 48dp。我们取 4 个触摸对象, 即  $4 \times 48dp = 192dp$ 。再加入一点富余量, 即为 200dp。

### 如果开发者要求在边缘上切出 200dp 以上的区域会怎样?

答案是, 系统只会兑现您的要求中位于最下方的 200dp, 如下图所示:





△ 开发者请求切出  $50 + 50 + 125 + 50$  dp 的区域，但系统只兑现最下面的总计 200dp

**我的视图不在屏幕内，是否也会受到这个限制？**

不会，系统仅计算屏幕范围内的切出矩形。同样，如果视图只有一部分显示在屏幕内，则仅计算所请求矩形的屏幕内可见部分。

**请关注下一篇连载**

读完本文您可能会问：为什么我们还没有讲流程图的右半部分？这是因为右半部分适用于那些需要全屏绘制内容的应用，我们将在下一篇手势导航连载中为您继续讲解，敬请保持关注。

请点击[这里](#)进一步了解 **Android 手势导航**

发布于 2019-12-16

[Android](#) [Android 开发](#) [Android 应用](#)

▲ 赞同 8 ▼    ● 1 条评论    ↗ 分享    ♥ 喜欢    ★ 收藏    📄 申请转载    ...



推荐阅读



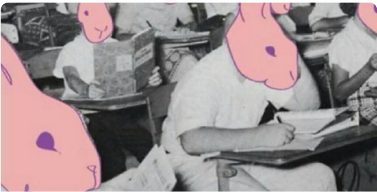
安卓学习<—>：布局与控件

vrendelle



安卓开发之  
SharedPreferences 小坑

慕课网



安卓开发：逆向的前提

Sw41n



跳过启动页广告，自由调整应用  
App 界面：3 款实用 Xposed...

少数派

发表于少数派

1 条评论

切换为时间排序

写下你的评论...



Doraemon

2019-12-16

@刘看山 @知乎小管家 快来看看，知乎的软件界面设计就有好多控件太靠下了，快改改吧[飙泪笑]

赞

赞同 8



1 条评论

分享

喜欢

收藏

申请转载

