


沉浸模式 | 手势导航连载 (四)



谷歌开发者 
已认证的官方帐号

已关注

4 人赞同了该文章

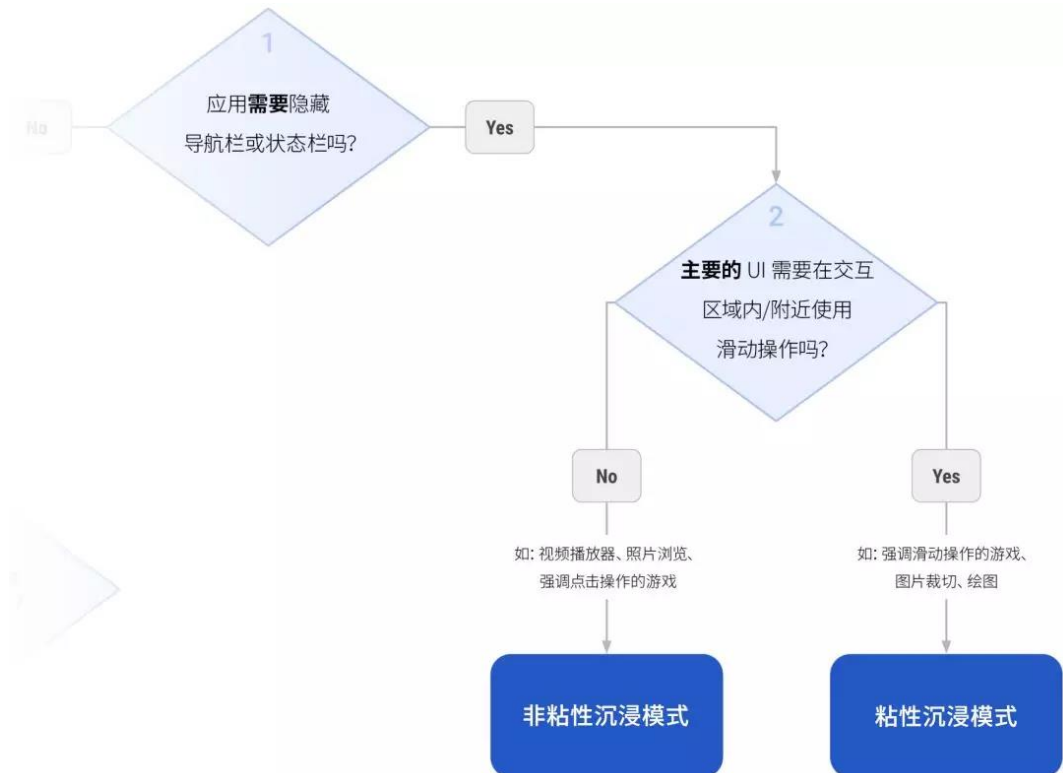


作者 / Chris Banes, Android 开发者关系团队工程师

本文是手势导航连载的第四篇文章，如果您希望了解其他手势导航的话题，请查看本系列的其他文章：

- [开启全面屏体验 | 手势导航 \(一\)](#)
- [处理视觉冲突 | 手势导航 \(二\)](#)
- [如何处理手势冲突 | 手势导航 \(三\)](#)

本文我们将为大家介绍的是手势交互和冲突在全屏应用 (系统栏也被隐藏) 下的情况和注意事项。让我们紧接[上一篇文章](#)，给大家讲讲流程图右侧的两种情况。



△ 请点击图片放大查看

右侧的两个解决方案都是 Android 平台为应用提供的沉浸模式 (immersive mode)。那问题来了：什么是沉浸模式？

什么是沉浸模式？



沉浸模式是一种让内容全屏呈现的方式，用来隐藏系统栏，从而确保应用拥有最大的屏幕空间。此外，它还提供了防误操作的功能 (比如意外使用手势离开应用)，特别适合在游戏中采用。

沉浸模式分为两种：

- 1. **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。
- 2. **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。在经过一小段时间后 (只有几秒) 会重新自动回到沉浸模式。

这两种模式都有两种状态：

- 1. **系统栏隐藏:** 在此状态下，返回主屏幕手势和后退手势均被禁用。用户必须首先从边缘向内侧滑动才能让系统栏显示。
- 2. **系统栏显示:** 在此状态下，返回主屏幕手势和后退手势可以正常工作。

现在，我们已经了解了沉浸模式的基础知识，下面介绍这两种不同模式的细节。

非粘性沉浸模式

大家在上面的流程图中可能已经看到，非粘性 (non-sticky) 沉浸模式非常适合需要全屏显示但不需要在屏幕边缘附近使用精确滑动手势的 UI。常见的例子包括全屏视频播放和照片浏览等。

就手势导航而言，非粘性沉浸模式与其在早期版本的 Android 上的工作方式一致。在 Android 10 或以上系统中运行时，应用可以使用我们在上一篇文章中介绍的手势区域排除 API。在此模式下，无论系统栏是否可见，每个边缘能排除的区域高度仍旧限制为 200dp。

如果您的应用正在使用非粘性沉浸模式，我们建议您回顾一下第三篇文章，避免在屏幕边缘出现的视图与系统手势出现冲突。

粘性沉浸模式

粘性 (sticky) 沉浸模式适合那些强烈需要使用整个屏幕，并要求在整个屏幕区域内进行触摸输入的 UI。常见的例子是绘图应用，以及使用滑动操作的游戏。

我们来看一下运行在 Android 10 上，且使用手势导航的 Markers 绘图应用：



如上图所示，一旦用户开始在屏幕边缘附近滑动（绘制），就会触发后退手势，这会打断用户当前的操作。

使用粘性沉浸模式的应用会有很强的交互性，因此手势区域排除 API 的限制会被移除，但仅限于系统栏隐藏的时候。这意味着应用可以根据需要完全占用屏幕左 / 右边缘。

但是，在系统栏可见时，系统则会忽略所有排除的手势区域，让用户可以返回，而不会受到来自应用的干扰。在粘性沉浸模式下，系统栏仅在短时间内可见，因此不会影响应用的正常交互。

屏幕底部的主屏手势区域依旧会正常存在，是无法排除的“强制”手势区域。处于粘性沉浸模式的应用可能会占用两个垂直边缘的整个长度，因此屏幕底部的主手势区域可能是用户呼出系统栏并退

接下来我们来看一下绘图应用的改进版本，整个垂直边缘都被应用占用：



可以看到，用户现在可以在屏幕边缘附近自由绘制，后退手势不会再干扰他们。如果用户想要退出应用，则可以从屏幕底部向上滑动呼出系统栏，进行后退或返回主屏的操作。

在实现方面，此处使用的代码大体沿用自第三篇文章中的 "使用手势区域排除 API" 部分，不同之处在于，我们希望视图能够知道它自身是否处于沉浸模式之中：

```
private val exclusionRects = ArrayList<Rect>()
```

```
override fun onLayout(changed: Boolean, left: Int, top: Int, right: Int, bottom: Int) {
```

```

        updateGestureExclusionRects()
    }
}

override fun onWindowSystemUiVisibilityChanged(visibility: Int) {
    super.onWindowSystemUiVisibilityChanged(visibility)

    // Update our gesture exclusions rects if we're
    // running on Android 10+
    if (Build.VERSION.SDK_INT >= 29) {
        updateGestureExclusionRects()
    }
}

private fun updateGestureExclusionRects() {
    // If the navigation bar is hidden, let's exclude any vertical edges so
    // that the user can draw freely
    if ((windowSystemUiVisibility and SYSTEM_UI_FLAG_HIDE_NAVIGATION) != 0) {
        // Root window insets are null, which happens if this is called
        // before we're attached and laid out. Ignore the call for now.
        val rootWindowInsets = rootWindowInsets ?: return

        val gestureInsets = rootWindowInsets.systemGestureInsets

        exclusionRects.clear()
        // Add an exclusion rect for the left gesture edge
        exclusionRects += Rect(0, 0, gestureInsets.left, height)
        // Add an exclusion rect for the right gesture edge
        exclusionRects += Rect(width - gestureInsets.right, 0, width, height)

        systemGestureExclusionRects = exclusionRects
    } else {
        // If the navigation bar is showing, we don't want to exclude any edge:
        systemGestureExclusionRects = emptyList()
    }
}
}

```

您可以在 GitHub 上阅读 Makers 应用更新的完整代码。

- 在 Android 10 上使用手势区域排除 [github.com/chrisbanes/m...](https://github.com/chrisbanes/makers)

总结对比: 非粘性与粘性

呼，一口气看到这里可能有点记不住。这里我为大家了提供一张表格，它总结出了非粘性和粘性沉浸模式之间的差异。

如何处理手势交互中的冲突就讲到这里。我也希望您已经对手势交互有了更深的理解，并将这些理解完美落实到应用的开发与更新中去。

本系列的第五篇文章 (也是最后一篇文章) 将介绍您可能会在应用中采用的一些常见 UI 模式，以及如何在手势导航中支持它们。

[点击这里](#)进一步了解 Android 手势导航

编辑于 2019-12-30

[Android](#) [Android 开发](#)

推荐阅读



Android SystemUI 介绍

转角一只喵 发表于Andro...



Android 仿应用宝下载进度条

Larry 发表于码力全开工...



Android 国际货币格式化的一个小知识点

Hevin

Android面试题----如何提高后台进程存活率

在Android系统中，应用进程停止运行有以下几个原因：1.用户主动退出； 2.Crash异常退出； 3.系统通过杀掉进程回收内存。其中，用户主动退出是合理行为；进程发生Crash后需要重新启动应用(...
priva... 发表于Andro...

还没有评论

写下你的评论...

