

[iOS]调和 pop 手势导致 AVPlayer 播放卡顿



NewPan (/u/e2f2d779c022) [+ 关注](#)

2017.07.15 14:24* 字数 3212 阅读 1572 评论 14 喜欢 31

(/u/e2f2d779c022)

声明：我为这个框架写了四篇文章：

第一篇：[iOS] UINavigationController全屏pop之为每个控制器自定义 UINavigationController (https://www.jianshu.com/p/88bc827f0692)

第二篇：[iOS] UINavigationController全屏pop之为每个控制器添加底部联动视图 (https://www.jianshu.com/p/3ed21414551a)

第三篇：[iOS] UINavigationController全屏pop之为控制器添加左滑push (https://www.jianshu.com/p/ff68b5e646fc)

第四篇：[iOS]调和 pop 手势导致 AVPlayer 播放卡顿 (https://www.jianshu.com/p/be02059b9e6a)



JPNavigationController

框架特性

- ✓ 全屏 pop 手势支持
- ✓ 全屏 push 到绑定的控制器支持
- ✓ 为每个控制器定制 UINavigationController 支持(包括设置颜色和透明度等)
- ✓ 为每个控制器添加底部联动视图支持
- ✓ 自定义 pop 手势范围支持(从屏幕最左侧开始计算宽度)
- ✓ 为单个控制器关闭 pop 手势支持
- ✓ 为所有控制器关闭 pop 手势支持
- ♥ 当当前控制器使用 AVPlayer 播放视频的时候, 使用自定义的 pop 动画以保证 AVPlayer 流畅播放.



01.真有这么回事？

做过视频的朋友都知道系统的 pop 手势会导致视频画面卡顿，没做过朋友都不敢相信，这绝对不是苹果的风格，居然留了这么一个坑。如果碰到这个问题，尝试去网上搜关键词 pop 手势 AVPlayer 卡顿，你搜不到太多有价值的解决方案。

因为我之前写了一个导航控制器的轮子，还写了一个视频播放器的轮子，所以理所当然，我必须趟平这个坑。下面我们花几分钟一起来看一下我是怎么做的。

02.思路分析

pop 手势就是为了在大屏下能获得更好的用户体验设计的。有了 pop 手势，返回的时候不用非要按一下返回按钮，只需优雅的右滑就能返回。但是系统的播放器会和 pop 手势冲突，对于有追求的程序员来说，这样做太影响用户体验了。

如果不做任何处理，系统在执行 pop 动画的时候，视频声音仍然播放正常，但是画面会阻塞会卡顿，等你取消 pop 手势仍然回到当前页面的时候，你会惊喜的发现，系统也知道画面出问题了，所以飞快的向后查找当前需要播放的那帧画面，但是很遗憾，系统也找不到了，所以最后播放的时候，声音和画面对不上，或者画面根本就不更新了，就卡在那里，然后声音一直在播放。

为了应对这个系统的 bug，开发者心里一般是默念一句...（此处略去三个字），然后在 `-viewWillDisappear:` 里写下一行：

```
[self.player pause];
```

可是别人的 APP 都没这个问题啊，你看看腾讯视频、哔哩哔哩、爱奇艺...

为了说明这个问题，我前段时间在公司内部分享上讲了这个事情，这里我简单说一下。如果你自己对比一下这些实现了 pop 手势不导致画面卡顿的 APP，你会发现他们的 pop 动画和系统默认的似乎有些不一样，至于究竟有哪些不一样，请诸位各位自己去自己观察。

有了这样的观察以后，我们的思路似乎变得清晰起来，没错，就是自己实现 pop 手势。

03.动手实现

思路有了，赶紧来验证一下我们的思路吧。

我在 [iOS] UINavigationController全屏pop之为控制器添加左滑push (https://www.jianshu.com/p/ff68b5e646fc) 这篇文章里详细的说了如何实现 push 动画，虽然现在 UINavigationController 2.0 的具体实现已经全部重新写过了，但是大致思路还是一样的。为了保证内容不重复，我这里就不再讲一遍一样的知识点了，如果你不知道如何实现，你去看那篇文章就好了。

我们的动画结构仍然是在动画容器上面添加我们当前要 pop 的 view 以及要 pop 到的元素的 view，然后用一个 `UIPercentDrivenInteractiveTransition` 百分比手势来驱动整个动画过程。按照这个思路实现以后，然后在要 pop 的页面上添加了一个 AVPlayer 播放视频，日了狗了，发现和系统的居然是一样的卡顿。

这样就比较郁闷了，瞬间感觉自己方向错了，有一种柯洁面对 AlphaGo 的赶脚。

但是从别的 APP 分析得到的启发就是要自己实现这个 pop 动画，这一点肯定没错。仔细想一下，pop 动画整个过程有以下几个部分：

- 手势：自己定义的 `UIPanGestureRecognizer`。



- 动画元素：自己添加的.
- 百分比驱动：系统的.
- 动画容器：系统的.

从上面的分析可以知道，我们只是自己定义了 手势 和 动画元素，但是 百分比手势驱动 和 动画容器 都是系统的，所以问题只有可能出在 百分比手势驱动 和 动画容器 上面。我想找到问题所在，所以逐个排除。

04.如何实现自己的百分比手势驱动类？

我们先来看 CAMediaTiming 协议下的一个属性

```
/* Additional offset in active local time. i.e. to convert from parent
 * time tp to active local time t: t = (tp - begin) * speed + offset.
 * One use of this is to "pause" a layer by setting `speed' to zero and
 * `offset' to a suitable value. Defaults to 0. */

@property CFTimeInterval timeOffset;
```

这里说了动画时间的计算方法 $t = (tp - \text{beginTime}) * \text{speed} + \text{timeOffset}$ ，比方说我们约定一个动画在 0.25 秒内执行完成，系统默认 $\text{beginTime} = 0$, $\text{speed} = 1$, $\text{timeOffset} = 0$ ，这样处理以后，这个计算式就变成了 $t = tp$ 。当动画开始，tp 开始从 0 增长到 0.25，那么动画执行的进度 $t = tp$ ，也是从 0 增长到 0.25。

这里还有一句 **"One use of this is to "pause" a layer by setting speed to zero and offset to a suitable value."**也就是说可以通过设置 $\text{speed} = 0$ 的方式来实现动画的技术性暂停。

```
@interface CALayer : NSObject <NSCoding, CAMediaTiming>
```

从 CALayer 的头文件，我们可以看到 CALayer 是遵守了 CAMediaTiming 协议的。所以我们可以写一个 demo 来模仿一下。



```

#import "ViewController.h"

@interface ViewController ()

@property (nonatomic, weak) IBOutlet UIView *containerView;
@property (nonatomic, weak) IBOutlet UISlider *speedSlider;
@property (nonatomic, weak) IBOutlet UISlider *timeOffsetSlider;
@property (weak, nonatomic) IBOutlet UILabel *speedLabel;
@property (weak, nonatomic) IBOutlet UILabel *timeOffsetLabel;

@property(nonatomic, strong) UIView *animateView;

@end

@implementation ViewController

- (void)viewDidLoad{
    [super viewDidLoad];

    self.animateView = ({
        UIView *view = [UIView new];
        view.frame = self.containerView.bounds;
        view.backgroundColor = [UIColor redColor];
        [self.containerView addSubview:view];

        view;
    });

- (IBAction)updateSliders{
    self.speedLabel.text = [NSString stringWithFormat:@"%0.2f", self.speedSlider.value];
    self.timeOffsetLabel.text = [NSString stringWithFormat:@"%0.2f", self.timeOffsetSlider.value];

    CFTimeInterval timeOffset = self.timeOffsetSlider.value;
    self.animateView.layer.timeOffset = timeOffset * 0.25;
}

- (IBAction)play{
    CGRect rect = self.animateView.frame;
    rect.origin.x = rect.size.width;

    [UIView animateWithDuration:0.25 animations:^(

        self.animateView.frame = rect;

    )];
    self.animateView.layer.speed = self.speedSlider.value;
}

@end

```

我们先把动画速度 speed 设置为 1，timeOffset 设为 0，很简单的动画，就是一个 x 轴平移，来看下效果。



接下来我们把 speed 设置为 0 timeOffset 设为 0，再开始动画。

没有做动画，对吧？因为我们已经把 speed 设置为 0 了，那么 $t = (tp - beginTime) * speed + timeOffset$ 这个方法的结果恒等于 0，所以不会有任何动画。接下来我们移动一下 offsetTime 滑条，更改一下上面公式的 timeOffset 的值，再看一下效果：

是不是和系统的 pop 手势有点像，这里是用滑条的值（0 到 1）来驱动动画的进度，系统是用手势的位置的百分比来驱动 pop 动画的进度，为此，系统专门抽出一个 `UIPercentDrivenInteractiveTransition` 来负责这个用手势来驱动动画的功能，叫做 百分比手势驱动。我们了解了这个知识点以后，就可以动手实现一个自己的 `PercentDrivenInteractiveTransition` 了。但是由于篇幅原因我不带大家实现了，这里只负责授人以渔。如果你感兴趣，想要一探究竟，可以去看一下这篇文章 [Interactive Custom Container View Controller Transitions \(https://link.jianshu.com?t=http://www.iosnomad.com/blog/2014/5/12/interactive-custom-container-view-controller-transitions\)](https://link.jianshu.com?t=http://www.iosnomad.com/blog/2014/5/12/interactive-custom-container-view-controller-transitions)。



我自己实现了这个类，然后把这个类用到我们的 JPNavigationController 项目中来，但是并没有能够解决我们播放视频卡顿的问题。至此 pop 动画四个组成部分，我们排除了三个。

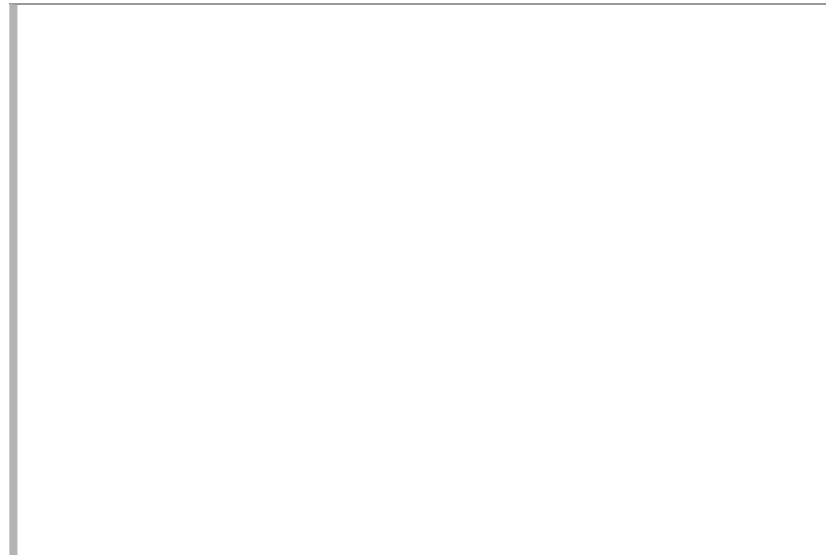
05.凶手真的是动画容器？

虽然没有成功实现我们的目标，但是我们知道了问题可能就出在系统提供的动画容器上，事实上，当我们自己代理系统的 transition 动画的时候，遵守 UINavigationControllerContextTransitioning 协议的动画上下文都会有一个 containerView 的属性：

```
- (void)animateTransition:(id <UINavigationControllerContextTransitioning>)transitionContext {
    _containerView = [transitionContext containerView];
}
```

通过断点拦截，我们可以看一下这个 containerView 是个什么东西。

```
Printing description of self->_containerView:
<UINavigationControllerWrapperView: 0x7fc40fe17810; frame = (0 0; 375 667); autoresize = YES>
```



系统有一个私有类 UINavigationControllerWrapperView，每个控制器（UIViewController 或者其子类，但是 UINavigationController 和 UITabBarController 除外）在渲染到屏幕上的时候都被一个 UINavigationControllerWrapperView 包裹。

通过我的测试，我使用 UIView 写了一个进度条，通过更新 frame.size.width 方式，在执行 pop 手势的时候添加定时器来更新这个宽度，进而达到进度条的效果，这里进度条更新没有问题。同样的我把这个进度条的更新放到 AVPlayer 的播放进度回调中，再执行 pop 手势，这个时候进度条就不更新了。

```
[player addPeriodicTimeObserverForInterval:CMTimeMake(1.0, 10.0) queue:dispatch_get_main_queue()]{
    float current = CMTimeGetSeconds(time);
    float total = CMTimeGetSeconds(sItem.currentPlayerItem.duration);
    if (current && progress) {
        progress(current / total);
    }
};
```

由此我们排除了动画容器的嫌疑，同时引出了凶手的另一个人选 AVPlayer。

06.万流归宗

从一开始怀疑是动画的过程有问题，到用排除法排除了所有的已知选项，最后一路顺藤摸瓜找到 AVPlayer，这一切并不容易，而且似乎有一种禅宗的 为万流，皆归宗 的感觉。



到现在为止，我们所做的只是把矛头指向 AVPlayer，但是这个类在系统执行 pop 手势的时候，里面究竟发生了什么，还是未解之谜。

看到这里，诸君各位可能要骂我没找到原因也敢写文章，而且还起了这么一个浮夸的标题。是的，这个骂名我担了，确实没有找到问题所在，但是我的标题也算比较谨慎，我用了一个 调和，并不敢在标题里用 解决 这个字眼。而且诸君不要担心，虽然没找出原因，但是我已经找到了一个可实践的应对这个问题的方法。

我们来分析一下这个 view 的层次和结构，从 UIViewControllerWrapperView 开始，下面有三个等级相当的 view，依次是 UIImageView、JPTransitionShadowView、当前控制的 view。可以看到使用 JPNavigationController 来应对这个有视频播放的控制器的 pop 的时候，我会创建这三层 view 用来做动画。

- 最下面的 UIImageView 里装的是上个界面的截屏。
- 中间的 JPTransitionShadowView 装的是一个模拟系统的阴影图片，事实上系统的动画还会更加细腻，在上一幅 3d 图中你可以找到一个 _UIParallaxDimmingView，顾名思义，这个 view 是用来模拟渐变色的。但是我还没有把这一点做进去。
- 最上面一层是当前控制器的 view。

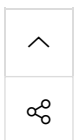
有了这三层以后，我会用手势来驱动这三层进行动画，以模拟系统的 pop 手势效果。代码太长了，我已经放在 GitHub 上了，这里就不贴了。

07.最后

至此，向诸君交了一份 60 分的考卷，GitHub 地址在这里 JPNavigationController (<https://link.jianshu.com?t=https://github.com/newyjp/JPNavigationController>)。谢谢大家。

08.注意

注意: tabBar 的 translucent 默认为 YES, 使用 JPNavigationCotroller 不能修改 tabBar 的透明属性. 这是因为 Xcode 9 以后, 苹果对导航控制器内部做了一些修改, 一旦将 tabBar 设为不透明, 当前架构下的 UI 就会错乱, 设置 tabBar 的 backgroundImage 为不透明图片, 或者设置 backgroundColor 为不透明的颜色值也是一样的会出错.



我的文章集合

下面这个链接是我所有文章的一个集合目录。这些文章凡是涉及实现的，每篇文章中都有 Github (https://link.jianshu.com?t=https://github.com/newyjp) 地址，Github (https://link.jianshu.com?t=https://github.com/newyjp) 上都有源码。如果某篇文章刚好在你的实际开发中帮到你，又或者提供一种不同的实现思路，让你觉得有用，那就看看这句话“坚持每天点赞的人，99%都是帅哥美女，再也不用单身了”。

我的文章集合索引 (https://www.jianshu.com/p/e03cd37db0d5)

你还可以关注我自己维护的简书专题
(https://www.jianshu.com/users/e2f2d779c022/latest_articles)iOS开发心得
(https://www.jianshu.com/collection/72d7b853d415)。这个专题的文章都是实打实的干货。

如果你有问题，除了在文章最后留言，还可以在微博 @盼盼_HKbuy
(https://link.jianshu.com?t=http://weibo.com/u/5590458451/home?wvr=5)上给我留言，以及访问我的 Github (https://link.jianshu.com?t=https://github.com/newyjp)。

iOS开发心得 (/nb/4374579) 举报文章 © 著作权归作者所有



NewPan (/u/e2f2d779c022) ♂

写了 82755 字，被 1973 人关注，获得了 1919 个喜欢 (/u/e2f2d779c022)

+ 关注

iOS 菜鸟工程师。

喜欢 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-like-button) 31







更多分享

(http://cwb.assets.jianshu.io/notes/images/1438750



(/apps/download?utm_source=nbc)

被以下专题收入，发现更多相似内容

-  好东西 (/c/ea7dca305051?utm_source=desktop&utm_medium=notes-included-collection)
-  小知识点 (/c/24491a80859c?utm_source=desktop&utm_medium=notes-included-collection)
-  iOS开发心得 (/c/72d7b853d415?utm_source=desktop&utm_medium=notes-included-collection)
-  iOS学习专题 (/c/cff595cc7e2b?utm_source=desktop&utm_medium=notes-included-collection)

^

🔗



上海恩美路演 (/c/277af95bf636?utm_source=desktop&utm_medium=notes-included-collection)



学无止境 (/c/55cc4ffafb26?utm_source=desktop&utm_medium=notes-included-collection)



iOS进阶之路 (/c/e25b25208315?utm_source=desktop&utm_medium=notes-included-collection)

展开更多 ▾

