MYViewControllerTransition 封装

1. 简述：在 iOS5 和 iOS6 前，viewController 的切换主要有 4 种：push/pop、tab 点击、model、addChildViewController，一般使用 transitonFromViewController:toViewController:的 UIView 封装的 CATransition 动画块，在 block 里可实现一些简单的切换效果，但问题是：

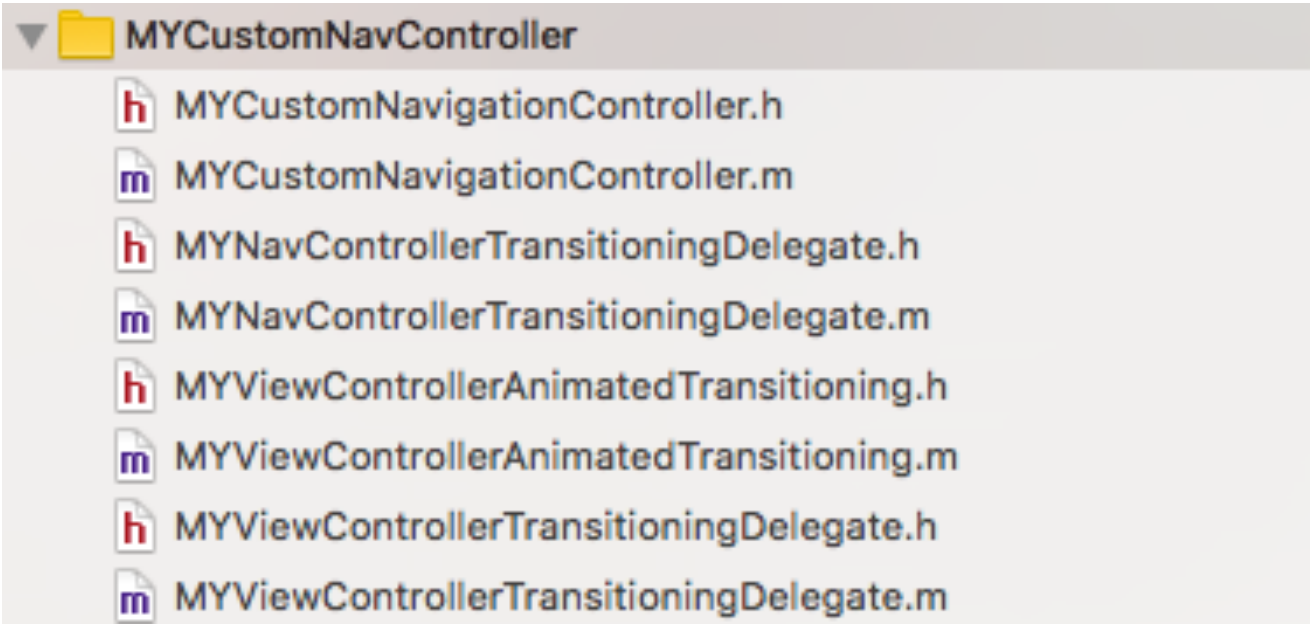
   (1) 代码高度耦和，vc 切换部分代码直接写在 controller 中，难以分离重用

   (2) 支持切换效果有限，因为只能使用 UIView 动画来切换，管理起来也略显麻烦。

   于是，苹果在 iOS7 中引入一些 API 来帮助开发者，更松耦合地定义 viewController 的转场效果，接下来以自定义 push、pop 动画(视觉上每个 viewController 的 navbar 独立)为例来讲解这些 API 的应用。

2. 实例：MYCustomNavigationController

   (1) 目录结构：



   (2) 具体封装实现：

```
/**
* 概述:
*      使用本navigationController，能在push和pop(支持手势)时在视觉上展示单独navigationBar(系统是固定、共用的)
*
*
*原理:
*      在push和pop时，通过给控制器视图添加假navigationBar并结合前后控制器视图和真navigationBar位置的移动来实现相
  应效果
*
```

```
* 实现：
*       1.实现UIViewControllerAnimatedTransitioning来定义modal或push的自定义转场动画
*       2.实现UIViewControllerTransitioningDelegate在modal动画属性为YES时触发相应方法调用UIViewControllerAnimatedTransitioning定义的自定义转场动画
*       3.实现UINavigationControllerDelegate在push或pop动画属性为YES时触发相应方法调用UIViewControllerAnimatedTransitioning定义的自定义转场动画；
*       4.在3.中给navigationController.view添加左边缘滑动手势，禁掉系统手势，在手势响应里进行popViewControllerAnimated操作并调用实现了UIViewControllerInteractiveTransitioning的UIPercentDrivenInteractiveTransition对象的相应动画进度控制方法
*       5.在3.中代理交互式转场代理方法里返回UIPercentDrivenInteractiveTransition进度转场动画进度控制对象
*       6.在本类重写push和pop方法，在push和pop前调用1.类，对当前navigationBar进行相应截屏入相应图片栈
*       7.push动画时，把pushImages栈中的最后一个元素取出(假navigationBar)，并添加到当前页面的上方，通过控制真navigationBar和后一个页面同时由屏幕右往左移动和当前页面向左移动实现视觉上单独的navigationBar效果，当然动画结束把假navigationBar移除掉
*       8.pop动画时，把popImage取出(假navigationBar)，并添加到当前页面上方；把pushImages栈中的最后一个元素取出(假navigationBar），并添加到上一个页面上方；然后把真navigationBar移到动画视图栈中两个页面的下方；通过控制当前和上一个页面往右移动（真navigationBar位置不动）实现视觉上单独的navigationBar效果，当然动画结束把两个假navigationBar移除掉，并把真navigationBar移到动画视图栈顶
*
*
* 特别注意：
        1.pop动画可能不止一个页面出栈，在pop前，需调整pushImages栈，将pop到的页面之后的假navigationBar出栈，保证
          pushImages栈顶是pop动画前一个页面的假navigationBar;
*       2.navigationBar的Translucent属性会导致页面坐标系发生变化，页面origin.y可能是64，也能是0，自定义动画需做相应处理
*       3.navigationBar的Translucent属性为YES时，在pop页面出栈数大于1时，居然会临时变成NO？这个问题虽然通过flag解决了，具体原因还得研究研究
*       4.记住UIViewControllerAnimatedTransition的动画执行在下一个toViewController的ViewDidLoad和ViewWillAppear之后，在ViewDidAppear之前
*       5.InteractiveTransitioning动画实际上是对animatedTransition相应动画进度的把控
*/
@interface MYCustomNavigationController : UINavigationController
```

```
#import "MYCustomNavigationController.h"
#import "MYNavControllerTransitioningDelegate.h"

@interface MYCustomNavigationController ()

@property (nonatomic, strong) MYNavControllerTransitioningDelegate *navDelegate;

@end

@implementation MYCustomNavigationController

- (void)viewDidLoad {
    [super viewDidLoad];
    self.navDelegate = [[MYNavControllerTransitioningDelegate alloc] initWithNavigationController:self];
    [self.navigationBar setBackgroundImage:[self imageWithColor:[UIColor brownColor] size:CGSizeMake(1, 1)] forBarMetrics:
       UIBarMetricsDefault];
    // Do any additional setup after loading the view.
}

#pragma override methods

- (void)pushViewController:(UIViewController *)viewController animated:(BOOL)animated{
    [self.navDelegate navigationControllerPreparedForPush];
    [super pushViewController:viewController animated:animated];
}

- (UIViewController *)popViewControllerAnimated:(BOOL)animated{
    NSAssert([self.viewControllers count] > 1, @"navigationcontroller.viewControllers.count must over 1 to enable popViewController");
    UIViewController *toVc = [self.viewControllers objectAtIndex:self.viewControllers.count - 2];
    [self.navDelegate navigationControllerPreparedForPop:toVc];
    return [super popViewControllerAnimated:animated];
}
```

```
- (NSArray<UIViewController *> *)popToViewController:(UIViewController *)viewController animated:(BOOL)animated{
    [self.navDelegate navigationControllerPreparedForPop:viewController];
    return [super popToViewController:viewController animated:animated];
}

- (NSArray<UIViewController *> *)popToRootViewControllerAnimated:(BOOL)animated{
    UIViewController *controller = [self.viewControllers firstObject];
    [self.navDelegate navigationControllerPreparedForPop:controller];
    return [super popToRootViewControllerAnimated:animated];
}

#pragma mark private methods
- (UIImage *)imageWithColor:(UIColor *)color size:(CGSize)size
{
    CGRect rect = CGRectMake(0, 0, size.width, size.height);
    UIGraphicsBeginImageContext(rect.size);
    CGContextRef context = UIGraphicsGetCurrentContext();
    CGContextSetFillColorWithColor(context, [color CGColor]);
    CGContextFillRect(context, rect);
    UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return image;
}
```

```
#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>

@interface MYNavControllerTransitioningDelegate : NSObject<UINavigationControllerDelegate>

- (instancetype)initWithNavigationController:(UINavigationController *)controller;

- (void)navigationControllerPreparedForPush;
- (void)navigationControllerPreparedForPop:(UIViewController *)toVc;

@end
```

```
#import "MYNavControllerTransitioningDelegate.h"
#import "MYViewControllerAnimatedTransitioning.h"

@interface MYNavControllerTransitioningDelegate ()

@property (assign, nonatomic) BOOL interActiving;

@property (strong, nonatomic) UIPercentDrivenInteractiveTransition* percentDrivenInteractiveTransition;
@property (strong, nonatomic) MYViewControllerAnimatedTransitioning *animation;

@property (strong, nonatomic) UINavigationController *navController;

@end

@implementation MYNavControllerTransitioningDelegate
```

```
- (instancetype)initWithNavigationController:(UINavigationController *)controller{
    if(self = [super init]){
        _navController = controller;
        _navController.interactivePopGestureRecognizer.enabled = NO;
        [controller setDelegate:self];

        _animation = [[MYViewControllerAnimatedTransitioning alloc] init];
        _percentDrivenInteractiveTransition = [[UIPercentDrivenInteractiveTransition alloc] init];

        UIScreenEdgePanGestureRecognizer *edgePanRecognizer = [[UIScreenEdgePanGestureRecognizer alloc] initWithTarget:self action:
            @selector(handleGesture:)];
        edgePanRecognizer.edges = UIRectEdgeLeft;
        [_navController.view addGestureRecognizer:edgePanRecognizer];
    }
    return self;
}

#pragma mark private methods

- (void)navigationControllerPreparedForPush{
    [_animation shotNavigationBarForPush:YES popToVcIndex:0];
}

- (void)navigationControllerPreparedForPop:(UIViewController *)toVc{
    NSInteger index = [self.navController.viewControllers indexOfObject:toVc];
    [_animation shotNavigationBarForPush:NO popToVcIndex:index];
}
```

```
-(void)handleGesture:(UIPanGestureRecognizer *)gesture {
    UIView* view = self.navController.view;
    CGPoint translation = [gesture translationInView:gesture.view];

    switch (gesture.state) {
        case UIGestureRecognizerStateBegan: {
            _interActiving = YES;
            if([self.navController.viewControllers count] > 1)
            [self.navController popViewControllerAnimated:YES];
            break;
        }
        case UIGestureRecognizerStateChanged: {
            CGFloat fraction = fabs(translation.x / view.bounds.size.width);
            [_percentDrivenInteractiveTransition updateInteractiveTransition:fraction];
            break;
        }
        case UIGestureRecognizerStateCancelled:
        case UIGestureRecognizerStateEnded: {
            _interActiving = NO;
            CGFloat fraction = fabs(translation.x / view.bounds.size.width);
            if (fraction < 0.5 || [gesture velocityInView:view].x < 0 || gesture.state == UIGestureRecognizerStateCancelled) {
                [_percentDrivenInteractiveTransition cancelInteractiveTransition];
            } else {
                [_percentDrivenInteractiveTransition finishInteractiveTransition];
            }
            break;
        }
        default:
            break;
    }
}
```

```
#pragma mark UINavigationControllerDelegate

- (id <UIViewControllerAnimatedTransitioning>)navigationController:(UINavigationController *)navigationController
   animationControllerForOperation:(UINavigationControllerOperation)operation fromViewController:(UIViewController *)fromVC
   toViewController:(UIViewController *)toVC {
   if (operation == UINavigationControllerOperationPop) {
      _animation.type  = UIViewControllerPresentTransitionTypePop;
      return _animation;
   }else  if (operation == UINavigationControllerOperationPush) {
      _animation.type  = UIViewControllerPresentTransitionTypePush;
      return _animation;
   }
   return nil;
}

- (id<UIViewControllerInteractiveTransitioning>)navigationController:(UINavigationController *)navigationController
   interactionControllerForAnimationController:(id<UIViewControllerAnimatedTransitioning>)animationController{
   return self.interActiving ? self.percentDrivenInteractiveTransition : nil;
}
```

```
#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>

/*
 本类仅定义了固定动画(仿系统默认动画)，可自由发挥，定义各种动画，以枚举区分即可
 */

typedef NS_ENUM(NSUInteger, UIViewControllerPresentTransitionType) {
   UIViewControllerPresentTransitionTypePresent = 0,//present动画
   UIViewControllerPresentTransitionTypeDismiss,//dismiss动画
   UIViewControllerPresentTransitionTypePush,//push动画
   UIViewControllerPresentTransitionTypePop//pop动画
};

@interface MYViewControllerAnimatedTransitioning : NSObject<UIViewControllerAnimatedTransitioning>

@property (assign, nonatomic) UIViewControllerPresentTransitionType type;

- (void)shotNavigationBarForPush:(BOOL)flag popToVcIndex:(NSInteger)index;

@end
```

```
#import "MYViewControllerAnimatedTransitioning.h"

static const float kTransitionDuration = .33;

@interface MYViewControllerAnimatedTransitioning ()

@property (strong, nonatomic) NSMutableArray *navBarImagesShotdForPush;
@property (strong, nonatomic) UIImage *navBarImageShotdForPop;

@property (assign, nonatomic) BOOL trulyIsTranslucent;//push页面后的navBar值

@end
```

```
- (instancetype)init{
    if(self = [super init]){
        _navBarImagesShotdForPush = [NSMutableArray array];
    }
    return self;
}

#pragma mark UIViewControllerAnimatedTransitioning

- (NSTimeInterval)transitionDuration:(id<UIViewControllerContextTransitioning>)transitionContext{
    return kTransitionDuration;
}

- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext{
    switch (_type) {
        case UIViewControllerPresentTransitionTypePresent:
            [self animatePresentTransition:transitionContext];
            break;
        case UIViewControllerPresentTransitionTypeDismiss:
            [self animateDissmissTransition:transitionContext];
            break;
        case UIViewControllerPresentTransitionTypePush:
            [self animatePushTransition:transitionContext];
            break;
        case UIViewControllerPresentTransitionTypePop:
            [self animatePopTransition:transitionContext];
        default:
            break;
    }
}
```

```
#pragma mark private methods

- (void)animatePresentTransition:(id<UIViewControllerContextTransitioning>)transitionContext{
    UIViewController *toViewController = [transitionContext viewControllerForKey:UITransitionContextToViewControllerKey];
    UIView *containView = [transitionContext containerView];
    [containView addSubview:toViewController.view];

    CGRect rect = [transitionContext finalFrameForViewController:toViewController];
    rect.origin.x = 0;
    rect.origin.y = rect.size.height;
    toViewController.view.frame = rect;
    [UIView animateWithDuration:[self transitionDuration:transitionContext] delay:0 options:UIViewAnimationOptionCurveEaseInOut
        animations:^{
     CGRect frame = rect;
        frame.origin.x = 0;
        frame.origin.y = 0;
        toViewController.view.frame = frame;
    } completion:^(BOOL finished) {
        BOOL isCanceled = [transitionContext transitionWasCancelled];
        [transitionContext completeTransition:!isCanceled];
    }];
}
```

```
- (void)animateDissmissTransition:(id<UIViewControllerContextTransitioning>)transitionContext{
    UIViewController *fromViewController = [transitionContext viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toViewController = [transitionContext viewControllerForKey:UITransitionContextToViewControllerKey];
    UIView *containView = [transitionContext containerView];
    [containView addSubview:toViewController.view];
    //      1.NSLog(@"%@", containView.subviews);
    [containView addSubview:fromViewController.view];
    //      2.NSLog(@"%@", containView.subviews);//两次log视图数一样，说明：不会重复添加动画视图

    CGRect rect = [transitionContext initialFrameForViewController:fromViewController];
    toViewController.view.frame = rect;
    [UIView animateWithDuration:[self transitionDuration:transitionContext] delay:0 options:UIViewAnimationOptionCurveEaseInOut
        animations:^{
        CGRect frame = rect;
        frame.origin.x = 0;
        frame.origin.y = rect.size.height;
        fromViewController.view.frame = frame;
    } completion:^(BOOL finished) {
        //      1.NSLog(@"%@", containView.subviews);
        BOOL isCanceled = [transitionContext transitionWasCancelled];
        [transitionContext completeTransition:!isCanceled];
        //      2.NSLog(@"%@", containView.subviews);//第二次log为空，说明：结束系统自动清除使用完毕的transitionContext动画视图
    }];
}
```

```
- (void)animatePushTransition:(id<UIViewControllerContextTransitioning>)transitionContext{
    UIViewController *fromViewController= [transitionContext viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toViewController = [transitionContext viewControllerForKey:UITransitionContextToViewControllerKey];
    UIView *naviBar = fromViewController.navigationController.navigationBar;
    UIView *containView = [transitionContext containerView];
    [containView addSubview:toViewController.view];
    [containView addSubview:naviBar];

    CGRect rect = [transitionContext finalFrameForViewController:toViewController];
    rect.origin.x = [[UIScreen mainScreen] bounds].size.width;
    BOOL isTranslucent = toViewController.navigationController.navigationBar.translucent;
    _trulyIsTranslucent = isTranslucent;
    CGFloat offset = isTranslucent  == NO?64:0;
    rect.origin.y = offset;
    toViewController.view.frame = rect;

    CGRect navRect = naviBar.frame;
    navRect.origin.x = [[UIScreen mainScreen] bounds].size.width;
    naviBar.frame = navRect;

    UIImageView *imageView =[[UIImageView alloc] initWithImage:[self.navBarImagesShotdForPush lastObject]];
    imageView.frame = CGRectMake(0, -offset, rect.size.width, 64);
    [fromViewController.view addSubview:imageView];

    toViewController.view.layer.shadowColor = [UIColor blackColor].CGColor;//添加阴影
    toViewController.view.layer.shadowOffset = CGSizeMake(0, -128);
    toViewController.view.layer.shadowRadius = 3;
    toViewController.view.layer.shadowOpacity = 0.8;
    toViewController.view.layer.shadowPath = [UIBezierPath bezierPathWithRect:CGRectMake(0, 0, toViewController.view.frame.size.width,
        toViewController.view.frame.size.height + 128)].CGPath;
```

```
    [UIView animateWithDuration:[self transitionDuration:transitionContext] delay:0 options:UIViewAnimationOptionCurveEaseInOut
     animations:^{
        CGRect frame = rect;
        frame.origin.x = 0;
        frame.origin.y = offset;
        toViewController.view.frame = frame;

        CGRect navRect1 = naviBar.frame;
        navRect1.origin.x = 0;
        naviBar.frame = navRect1;

        frame.origin.x = - [[UIScreen mainScreen] bounds].size.width/2;
        frame.origin.y = offset;
        fromViewController.view.frame = frame;
    } completion:^(BOOL finished) {
        BOOL isCanceled = [transitionContext transitionWasCancelled];
        [transitionContext completeTransition:!isCanceled];
        if(!isCanceled){
            [imageView removeFromSuperview];
        }
    }];
}
```

```
- (void)animatePopTransition:(id<UIViewControllerContextTransitioning>)transitionContext{
    UIViewController *toViewController = [transitionContext viewControllerForKey:UITransitionContextToViewControllerKey];
    UIView *containView = [transitionContext containerView];
    UIView *fromViewControllerView = [containView.subviews firstObject];//No:1 -> No:3
    UIView *naviBar = [containView.subviews lastObject];//No:2 -> No:1
    [containView addSubview:toViewController.view];//No:2
    [containView bringSubviewToFront:fromViewControllerView];

    CGRect rect = [transitionContext finalFrameForViewController:toViewController];
    rect.origin.x = -[[UIScreen mainScreen] bounds].size.width/2;
    toViewController.view.frame = rect;

    BOOL isTranslucent = toViewController.navigationController.navigationBar.translucent;
    CGFloat offset = isTranslucent  == NO?64:0;
    //在pop出栈数大于1时(如popToRootViewController)，translucent居然临时变成YES了？？？导致navBar截图添加位置错误！！！
    if(!isTranslucent && _trulyIsTranslucent == YES)
        offset = 0;

    UIImageView *fromViewNavBar =[[UIImageView alloc] initWithImage:self.navBarImageShotdForPop];
    fromViewNavBar.frame = CGRectMake(0, -offset, rect.size.width, 64);
    [fromViewControllerView addSubview:fromViewNavBar];

    UIImageView *toViewNavBar =[[UIImageView alloc] initWithImage:[self.navBarImagesShotdForPush lastObject]];
    toViewNavBar.frame = CGRectMake(0, -offset, rect.size.width, 64);
    [toViewController.view addSubview:toViewNavBar];

    fromViewControllerView.layer.shadowColor = [UIColor blackColor].CGColor;//(2)pop前把(1)操作取消的阴影加上
```

```
  [UIView animateWithDuration:[self transitionDuration:transitionContext] delay:0 options:UIViewAnimationOptionCurveEaseInOut
     animations:^{
     CGRect frame = rect;
     frame.origin.x = 0;
     frame.origin.y = offset;
     toViewController.view.frame = frame;

     frame.origin.x = [[UIScreen mainScreen] bounds].size.width;
     frame.origin.y = offset;
     fromViewControllerView.frame = frame;
  } completion:^(BOOL finished) {
     BOOL isCanceled = [transitionContext transitionWasCancelled];
     [transitionContext completeTransition:!isCanceled];
     if(!isCanceled){
        [containView bringSubviewToFront:naviBar];
        toViewController.view.layer.shadowColor = [UIColor clearColor].CGColor;//(1)解决pop结束阴影还在
        [fromViewNavBar removeFromSuperview];
        [toViewNavBar removeFromSuperview];
        [_navBarImagesShotdForPush removeLastObject];
     }else{//手势取消，containtView结构复位
        [fromViewNavBar removeFromSuperview];
        [toViewNavBar removeFromSuperview];
        [containView bringSubviewToFront:fromViewControllerView];
        [containView bringSubviewToFront:naviBar];
     }
  }];
}
```

```
- (void)shotNavigationBarForPush:(BOOL)flag popToVcIndex:(NSInteger)index{//index为popTo的页面
   CGSize size = [[UIScreen mainScreen] bounds].size;
   UIGraphicsBeginImageContextWithOptions(CGSizeMake(size.width, 64), YES, 0);
   CGContextRef context = UIGraphicsGetCurrentContext();
   [[[[UIApplication sharedApplication] keyWindow] layer] renderInContext:context];
   UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
   UIGraphicsEndImageContext();
   if(flag){//push前的截屏
      [self.navBarImagesShotdForPush addObject: image];
   }
   else{//pop前的截屏
      NSInteger num = _navBarImagesShotdForPush.count;
      if(num > index + 1){//只保存至当前pop页面的前一个页面及index以前的保留即可
         [_navBarImagesShotdForPush removeObjectsInRange:NSMakeRange(index + 1, num - 1)];
      }
      self.navBarImageShotdForPop = image;//截取当前即将pop的页面
   }
}
```

```
#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>

@interface MYViewControllerTransitioningDelegate : NSObject<UIViewControllerTransitioningDelegate>

@end
```

```objc
#import "MYViewControllerTransitioningDelegate.h"
#import "MYViewControllerAnimatedTransitioning.h"

@interface MYViewControllerTransitioningDelegate ()

@property (nonatomic, strong) MYViewControllerAnimatedTransitioning *animatedTransitioning;

@end

@implementation MYViewControllerTransitioningDelegate

- (instancetype)init{
    if(self = [super init]){
        self.animatedTransitioning = [[MYViewControllerAnimatedTransitioning alloc] init];
    }
    return self;
}

#pragma mark UIViewControllerTransitioningDelegate

- (id<UIViewControllerAnimatedTransitioning>)animationControllerForPresentedController:(UIViewController *)presented
    presentingController:(UIViewController *)presenting sourceController:(UIViewController *)source{
     self.animatedTransitioning.type = UIViewControllerPresentTransitionTypePresent;
    return self.animatedTransitioning;
}

- (id<UIViewControllerAnimatedTransitioning>)animationControllerForDismissedController:(UIViewController *)dismissed{
     self.animatedTransitioning.type = UIViewControllerPresentTransitionTypeDismiss;
    return self.animatedTransitioning;
}
```