【译】详细讲述iOS自定义转场

2015-11-20 09:37 编辑：suiling 分类：iOS开发 来源：OBJECTIVE TOAST 8 3245

push APP转场 自定义转场 view controll

招聘信息：iOS高级开发工程师

本文由CocoaChina译者@ALEX吴浩文翻译

作者：Andrew Hershberger

原文：Custom Transitions on iOS

本文是iOS自定义视图控制器转场系列的第一篇。本文重点在于创建自定义动画(非交互式)转场。

当使用传统的iOS应用程序时，我们经常在视图间转场。过去，如果你不想用标准的转场动画，全靠你自己，但在iOS 7中苹果提供了一个新的API让我们自定义这些动画。

iOS提供了一些内置的转场类型。Navigation controllers用push和pop来有层次地导航信息，tab bar controllers用切换tabs来在各部分之间跳转，所有的视图控制器可以根据特定任务模态化地present和dismiss另一个视图控制器。

API介绍

- 每一个自定义转场涉及三个主要对象:
- from view controller (消失的那个)

热门资讯


这些 iOS 面试基础题目，你都深入了解吗?
点击量 16975


【译】17个提升iOS开发效率的必用工具
点击量 11038


iOS开发——UI组件（个人整理）
点击量 8750


Objective-C 编码建议
点击量 6074


我已经写了48年代码了，我感觉我还能写下
点击量 5886


ViewController 瘦身的另一种解决方案
点击量 5528


出大事了！A商要联合程序员们打淘宝分京东
点击量 5378


新手向：五分钟搭建App设置页面_纯代码
点击量 5274


让我们来搞崩 Cocoa 吧 （黑暗代码）
点击量 4628


来就来全套！敏捷开发知识体系笔记
点击量 4575

综合评论

这也是百度MUX翻译的文章呀。

唧唧歪歪 评论了 停止不必要的UI动效设计...

不错

呵呵呵 评论了 停止不必要的UI动效设计...

缓存起码应该有个缓存周期吧- -你这样缓存下来就不会被更新了吧 不符合实际

马宏达 评论了 iOS 数据库离线缓存思路和网络层封装...

- to view controller (出现的那个)
- 一个动画控制器

自定义转场和在自定义之前一样。对于push和pop，意味着调用UINavigationController的push-、pop-、或者set-方法来修改视图控制器的堆栈。对于切换tabs，意味着修改UITabBarController的selectedIndex或selectedViewController属性。对于modal，则意味着调用?[UIViewController presentViewController: animated: completion:]或?[UIViewController dismissViewControllerAnimated: completion:]。无论哪种情况，这个步骤都确定了“from view controller”和“to view controller”。

使用一个自定义转场，你需要一个动画控制器。对我来说这是自定义动画转场中最令人困惑的部分，因为每种转场需要的动画控制器不同。下表展示了如何为每种转场提供动画控制器。记着，委托方法总是返回动画控制器。

转场	动画控制器组成
Push and Pop	navigation controller的delegate实现 -navigationController: animationControllerForOperation: fromViewController: toViewController:
Tabs切换	tab bar controller的delegate实现 -tabBarController: animationControllerForTransitionFromViewController: toViewController:
Present	被弹出的视图控制器的transitioningDelegate实现 -animationControllerForPresentedController: presentingController: sourceController:， 同时它的-modalPresentationStyle在转场开始前要被置为UIModalPresentationCustom
Dismiss	被弹出的视图控制器的transitioningDelegate实现-animationControllerForDismissedController:， 同时它的-modalPresentationStyle在转场开始前要被置为UIModalPresentationCustom

动画控制器可以是任何遵守UIViewControllerAnimatedTransitioning协议的对象。该协议声明了两个必须要实现的方法。一个提供了动画的时间，另一个执行了动画。这些方法调用时都传递一个上下文。上下文提供了入口来访问信息和你创建自定义转场需要的对象。以下是一些重点：

- from view controller
- to view controller
- 两个视图控制器view的第一帧和最后一帧
- container view，根据这篇文档，“作为的转场中视图的父视图”

重要：上下文还实现了-completeTransition:，你必须在你自定义转场结束时调用一次。

这是关于自定义转场所有你需要知道的。让我们来看一些例子！

例子

所有这些例子都可以在GitHub找到，你可以克隆这些仓库，然后边往下看边试试这些例子。

这三个例子都直接或子类化地使用了TWTExampleViewController。它只是设置了视图的背景颜色，同时使你能够通过点击任何地方来结束例子回到主菜单。

轻弹push和pop

在这个例子中，目标是让push和pop使用flip动画而不是标准的slide动画。一开始我建立一个navigation controller并把TWTPushExampleViewController的实例当作root。TWTPushExampleViewController添加了一个叫“Push”的右按钮到导航栏。点击它时，一个新的TWTPushExampleViewController的实例被压入navigation的堆栈：

```
- (void)pushButtonTapped
{
```

```
    TWTPushExampleViewController *viewController = [[TWTPushExampleViewController alloc] init];
    viewController.delegate = self.delegate;
    [self.navigationController pushViewController:viewController animated:YES];
}
```

navigation controller的设置发生在TWTExamplesListViewController(运行demo主菜单的视图控制器)。注意，它把自己置为navigation controller的委托:

```
- (void)presentPushExample
{
    TWTPushExampleViewController *viewController = [[TWTPushExampleViewController alloc] init];
    viewController.delegate = self;

    UINavigationController *navigationController = [[UINavigationController alloc] initWithRootView
    navigationController.delegate = self;

    [self presentViewController:navigationController animated:YES completion:nil];
}
```

这意味着当navigation controller的转场即将开始时，TWTExamplesListViewController将收到委托信息，并有机会返回一个动画控制器。对于这种转场，我使用一个TWTSimpleAnimationController的实例，它是一个+[UIView transitionFromView: toView: duration: options: completion:]的封装：

```
- (id)navigationController:(UINavigationController *)navigationController
                                animationControllerForOperation:(UINavigationControllerOperation)
                                            fromViewController:(UIViewController *)fromVC
                                              toViewController:(UIViewController *)toVC
{
    TWTSimpleAnimationController *animationController = [[TWTSimpleAnimationController alloc] init]
    animationController.duration = 0.5;
    animationController.options = (  operation == UINavigationControllerOperationPush
                                   ? UIViewAnimationOptionTransitionFlipFromRight
                                   : UIViewAnimationOptionTransitionFlipFromLeft);
    return animationController;
}
```

如果转场是一个push,我使用一个从右侧的flip，否则，我使用一个从左侧的flip。

以下是TWTSimpleAnimationController的实现：

```
- (NSTimeInterval)transitionDuration:(id)transitionContext
{
    return self.duration;
}

- (void)animateTransition:(id)transitionContext
{
    UIViewController *fromViewController = [transitionContext viewControllerForKey:UITransitionCont
```

```
    UIViewController *toViewController = [transitionContext viewControllerForKey:UITransitionContex
    toViewController.view.frame = [transitionContext finalFrameForViewController:toViewController];
    [toViewController.view layoutIfNeeded];

    [UIView transitionFromView:fromViewController.view
                        toView:toViewController.view
                      duration:self.duration
                       options:self.options
                    completion:^(BOOL finished) {
                        [transitionContext completeTransition:YES];
                    }];
}
```

记着，这两个方法是UIViewControllerAnimatedTransitioning协议的一部分。在动画控制器运行自定义转场的时候，它们被UIKit调用。

这里有一些关于animateTransition:需要注意的事情：

- from view controller，to view controller，以及to view controller的最后一帧都从转场的上下文中提取。其中还有一些其他可提取的信息，但在当前情况下，并不需要所有信息。
- +[UIView transitionFromView: toView: duration: options: completion:]负责有层次地添加和删除视图。在后面的例子中，我将展示一种手动完成的情况。
- 在转场的completion代码块中，我调用[transitionContext completeTransition: YES]来告诉系统转场结束了。如果你忘了这样做，你将无法与app交互。如果出现这种情况，先检查这个原因。

以上就是全部！现在有一些值得尝试的东西：

- 改变动画的持续时间来看看它如何影响navigation bar的动画。
- 把动画选项由flip改为page curls。
- 找一个方法让navigation的堆栈中的每个视图控制器能指定自己的动画控制器。看看在本文最后的推荐模式中提出的方法。

淡入淡出切换tabs

这个例子应该很熟悉。它使用和之前相同的观点，但使用tab bar controller而不是navigation controller。

以下是TWTExamplesListViewController的设置：

```
- (void)presentTabsExample
{
    NSMutableArray *viewControllers = [[NSMutableArray alloc] init];

    for (NSUInteger i=0; i<3; i++) {
        TWTChangingTabsExampleViewController *viewController = [[TWTChangingTabsExampleViewControll
        viewController.delegate = self;
        viewController.index = i;
        [viewControllers addObject:viewController];
    }
```

```
    UITabBarController *tabBarController = [[UITabBarController alloc] init];
    [tabBarController setViewControllers:viewControllers animated:NO];
    tabBarController.delegate = self;

    [self presentViewController:tabBarController animated:YES completion:nil];
}
```

TWTChangingTabsExampleViewController的index只是一个为每个视图控制器自定义标签的方法。就像之前一样，TWTExamplesListViewController是委托，它将在切换tabs的时候提供动画控制器：

```
- (id)tabBarController:(UITabBarController *)tabBarController
           animationControllerForTransitionFromViewController:(UIViewController *)fromVC
                                             toViewController:(UIViewController *)toVC
{
    TWTSimpleAnimationController *animationController = [[TWTSimpleAnimationController alloc] init]
    animationController.duration = 0.5;
    animationController.options = UIViewAnimationOptionTransitionCrossDissolve;
    return animationController;
}
```

看着熟悉吗？这就够了。

弹出一个覆盖视图

我最喜欢自定义转场的一个用途，是它可以弹出覆盖式的视图控制器。以前，如果你想模仿一个社会化分享菜单，或任何需要呈现视图控制器的同时保持背后内容的可见，你不得不从许多不幸的选项中做出选择(直接添加视图到窗口是我见过的最通用的方法)。然而幸运的是，这不再是必要的。

能这样用的关键原因，是当被弹出视图控制器的-modalPresentationStyle被置为UIModalPresentationCustom时，弹出视图控制器就不会自动从视图层中删除。为了弹出一个覆盖视图，它仅仅如同是离开一般，这样被弹出的视图就可以显示在它上方。

以下是TWTExamplesListViewController的设置:

```
- (void)presentPresentExample
{
    TWTOverlayExampleViewController *viewController = [[TWTOverlayExampleViewController alloc] init
    viewController.delegate = self;
    viewController.modalPresentationStyle = UIModalPresentationCustom;
    viewController.transitioningDelegate = self;

    [self presentViewController:viewController animated:YES completion:nil];
}
```

这里两个要注意的事情是：modalPresentationStyle被置为UIModalPresentationCustom，被弹出视图控制器的-transitioningDelegate被置为TWTExamplesListViewController。当转场开始时，TWTExamplesListViewController收到转场的委托信息：

```
- (id)animationControllerForPresentedController:(UIViewController *)presented
```

```
                                                    presentingController:(UIViewContr
                                                        sourceController:(UIViewContr
{
    return [[TWTPresentAnimationController alloc] init];
}
```

如你所见，我创建了一个自定义动画控制器类作为示例。我将要关注-animateTransition:的实现，因为它与之前的部分并不相同。

开始前，我用transitionContext提取toViewController（被弹出视图控制器）和containerView（容器视图）。

```
UIViewController *toViewController = [transitionContext viewControllerForKey:UITransitionContextToV

UIView *containerView = [transitionContext containerView];
```

containerView在转场中是from view controller和to view controller的父视图。最初，to view controller的视图没有被添加到containerView，它的frame也未确定。我这里直接赋值frame，你也可以使用auto layout。

```
CGRect frame = containerView.bounds;
frame = UIEdgeInsetsInsetRect(frame, UIEdgeInsetsMake(40.0, 40.0, 200.0, 40.0));

toViewController.view.frame = frame;

[containerView addSubview:toViewController.view];
```

对于这种转场，我希望这个视图pop到屏幕上。要做到这一点，我用UIView的spring动画把视图的scale从0.3变化到1。

说句题外话，若动画中scale从大于0的值开始同时alpha从0到1，则可以让你动画速度比简单的scale从0到1更快。试着删除alpha动画，再和从0开始的scale动画比较。

```
toViewController.view.alpha = 0.0;
toViewController.view.transform = CGAffineTransformMakeScale(0.3, 0.3);

NSTimeInterval duration = [self transitionDuration:transitionContext];

[UIView animateWithDuration:duration / 2.0 animations:^{
    toViewController.view.alpha = 1.0;
}];

CGFloat damping = 0.55;

[UIView animateWithDuration:duration delay:0.0 usingSpringWithDamping:damping initialSpringVelocity
    toViewController.view.transform = CGAffineTransformIdentity;
} completion:^(BOOL finished) {
    [transitionContext completeTransition:YES];
}];
```

在completion块，我调用-completeTransition:，到这里就完了！现在到了弹回...

在弹回开始时，TWTExamplesListViewController收到转场的委托信息：

```
- (id)animationControllerForDismissedController:(UIViewController *)dismissed
{
    return [[TWTDismissAnimationController alloc] init];
}
```

返回另一个自定义动画控制器。让我们看一看-animateTransition:：

```
UIViewController *fromViewController = [transitionContext viewControllerForKey:UITransitionContextF

NSTimeInterval duration = [self transitionDuration:transitionContext];

[UIView animateWithDuration:3.0 * duration / 4.0
                      delay:duration / 4.0
                    options:UIViewAnimationOptionCurveEaseIn
                 animations:^{
                     fromViewController.view.alpha = 0.0;
                 }
                 completion:^(BOOL finished) {
                     [fromViewController.view removeFromSuperview];
                     [transitionContext completeTransition:YES];
                 }];

[UIView animateWithDuration:2.0 * duration
                      delay:0.0
     usingSpringWithDamping:1.0
      initialSpringVelocity:-15.0
                    options:0
                 animations:^{
                     fromViewController.view.transform = CGAffineTransformMakeScale(0.3, 0.3);
                 }
                 completion:nil];
```

如你所见，这个想法是倒转弹出动画。需要注意的重要区别是，from view controller的视图在转场结束前就从视图层中被删除了。

在继续之前，试着改变一下弹回的动画。

推荐模式

到这里，您可能已经注意到这个API在很大程度上依赖于协议和委托方法。因此很容易把自定义转场编写得混乱且无法重用。这里有一些模式帮助我保持整洁:

1.用专门的对象作动画控制器

虽然一个视图控制器可以直接代码量翻倍当作动画控制器，但几乎可以肯定这样做会减少自定义转场的可重用性。此外，视图控制器已经因为做得太多而臭名昭著。你可以创建可重用的动画控制器，只需创建NSObject子类，遵守UIViewControllerAnimatedTransitioning协议，然后在需要的时候返回它们的实例。

我们在Toast里的TWTSimpleAnimationController就是这样做的，它很容易重用和用CocoaPod集成。

2.不要让UINavigationControllerDelegate需要知道转场的细节

用navigation controller的委托返回动画控制器是好，如果你想让所有的push和pop转场都以同样的方式进行。然而在某些情况下，你可能希望只自定义一个push或pop。

一种混乱的方式是把navigation controller的委托对象暂时改变成一个知道这种转场的对象。另一种混乱的方式是使navigation controlle的委托根据特定的from view controllers和to view controllers有逻辑地去确定。显然，这些都不是好的选择。

用模式可以干净利落地解决这个问题，只要向UIViewController添加-pushAnimationController和-popAnimationController属性。然后navigation controller的委托就可以返回由被push的动画控制器和被pop视图控制器指定的动画控制器。这使得navigation controller的委托保持通用同时避免了委托对象的改变。TWTNavigationControllerDelegate实现了这种模式，它也同时包括在Toast里。

这样就结束了这个创建自定义动画视图控制器转场的介绍。一定要看看我们的在Toast里的视图控制器转场模块，并继续关注第二部分，该部分将包括自定义交互式转场。

上一篇：源码推荐(11.19)：高仿斗鱼TV，类似微信搜索联系人
下一篇：iOS系统的这些小功能会影响到你的APP哦~你测了没呢?

相关资讯

自定义 push 和 pop 实现有趣的相册翻开效果(上)
关于自定义转场动画，我都告诉你
论坛源码推荐（5月16日）：自定义push、pop和手势返
iOS开发资源：推送通知相关开源项目--PushSharp、
iPhone的Push(推送通知)功能原理浅析
源码推荐(10.10)：UITableViewCell的展开与收缩，自定义
《iOS 7 Programming Pushing the Limits》系列：你可能
应用“PUSH推送”的5个真相和5个误区
App开发者，求求你们别push了!
在Leopard中编译搭建非官方iPhone toolchain开发环境并

我来说两句

您还没有登录！请 登录 或 注册

所有评论（8）

baoyewei99 2015-11-23 07:53:23

mark

0 0 回复

sdm2010 2015-11-23 07:15:25

写得真好，有交互式转场的中文介绍吗

0 0 回复

junlinfushi 2015-11-23 02:37:20

mark

0 0 回复

低调的魅力 2015-11-23 01:54:14

mark 分享是一种快乐,评论是一种美德

0 0 回复

Eric_Scofield 2015-11-21 14:28:23

我想问下，自定义的专场动画，怎么让navigationBar也一起移动，而不是采用系统自带的那种淡化的效果?

0 0 回复

842996144 2015-11-21 13:03:56

mark

0 0 回复

垭沅嗦 2015-11-21 03:36:20

mark

0 0 回复

asdasdaa 2015-11-20 06:31:00

占个沙发哈~~~

0 0 回复