一片枫叶

追求卓越，成功就会在不经意间追上你

跟随自己的节奏学习，思考，总结，找到自己，别人才会找到你

下一篇:基于**NSBundle**的工程结构

IOS：屏幕旋转与**Transform**

IOS：屏幕旋转与Transform

iTouch，iPhone，iPad设置都是支持旋转的，如果我们的程序能够根据不同的方向做出不同的布局，体验会更好。

如何设置程序支持旋转呢，通常我们会在程序的info.plist中进行设置*Supported interface orientations*，添加我们程序要支持的方向，而且程序里面每个viewController也有方法

supportedInterfaceOrientations(6.0及以后)

shouldAutorotateToInterfaceOrientation(6.0之前的系统)

通过viewController的这些方法，我们可以做到更小粒度的旋转控制，如程序中仅仅允许个别界面旋转。

一、屏幕旋转背后到底做了什么呢?

下面我们看个简单的例子，用xcode新建一个默认的单视图工程，然后在对应viewController的响应旋转后的函数中输出一下当前view的信息，代码如下：

```
//
//  SvRotateViewController.m
//  SvRotateByTransform
//
//  Created by  maple on 4/21/13.
//  Copyright (c) 2013 maple. All rights reserved.
//

#import "SvRotateViewController.h"

@interface SvRotateViewController ()

@end

@implementation SvRotateViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a
nib.

    self.view.backgroundColor = [UIColor grayColor];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (BOOL)shouldAutorotateToInterfaceOrientation:
(UIInterfaceOrientation)interfaceOrientation
```


导航

联系 管理

< 2013年4月 >

日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

统计

随笔 - 46 文章 - 3 评论 - 450

搜索

找找看

谷歌搜索

随笔分类(44)

iOS(36)

数据结构与算法

杂谈(8)

随笔档案(46)

2015年1月 (1)

2014年11月 (1)

2014年10月 (2)

2014年8月 (1)

2014年6月 (1)

2014年5月 (2)

2013年11月 (1)

2013年8月 (1)

2013年7月 (1)

2013年6月 (2)

2013年5月 (3)

2013年4月 (4)

2013年3月 (1)

2013年2月 (1)

2012年12月 (3)

2012年11月 (2)

2012年10月 (1)

2012年8月 (1)

2012年7月 (2)

2012年6月 (3)

2012年5月 (4)

2012年3月 (6)

2011年12月 (2)

积分与排名

积分 - 121926

排名 - 1287

最新评论

1. Re:UITableView学习笔记

@QYMa_HAZA恩，我是西安人，不过还从来没有在西安工作过呢...

--一片-枫叶

2. Re:UITableView学习笔记

今天刚看到博主的博客，看到里面的示例图片，猜测楼

```
{
    return YES;
}

- (BOOL)shouldAutorotate
{
    return YES;
}

- (NSUInteger)supportedInterfaceOrientations
{
    return UIInterfaceOrientationMaskAll;
}

- (void)willRotateToInterfaceOrientation:
(UIInterfaceOrientation)toInterfaceOrientation duration:
(NSTimeInterval)duration
{
    NSLog(@"UIViewController will rotate to Orientation: %d",
toInterfaceOrientation);
}

- (void)didRotateFromInterfaceOrientation:
(UIInterfaceOrientation)fromInterfaceOrientation
{
    NSLog(@"did rotated to new Orientation, view Information %@",
self.view);
}

@end
```

查看代码我们可以发现，我们的viewController支持四个方向，然后在旋转完成的didRotateFromInterfaceOrientation函数中打印了self.view的信息，旋转一圈我们可以看到如下输出：

```
SvRotateByTransform[822:c07] UIViewController will rotate to Orientation: 4
SvRotateByTransform[822:c07] <UIView: 0x71574f0; frame = (20 0; 300 480); transform = [0, -1, 1, 0, 0, 0]; autoresize = W+H; layer = <CALayer: 0x7156ad0>>
SvRotateByTransform[822:c07] UIViewController will rotate to Orientation: 2
SvRotateByTransform[822:c07] <UIView: 0x71574f0; frame = (0 0; 320 460); transform = [-1, 0, -0, -1, 0, 0]; autoresize = W+H; layer = <CALayer: 0x7156ad0>>
SvRotateByTransform[822:c07] UIViewController will rotate to Orientation: 3
SvRotateByTransform[822:c07] <UIView: 0x71574f0; frame = (0 0; 300 480); transform = [0, 1, -1, 0, 0, 0]; autoresize = W+H; layer = <CALayer: 0x7156ad0>>
SvRotateByTransform[822:c07] UIViewController will rotate to Orientation: 1
SvRotateByTransform[822:c07] <UIView: 0x71574f0; frame = (0 20; 320 460); autoresize = W+H; layer = <CALayer: 0x7156ad0>>
```

设备的初始方向是UIInterfaceOrientationPortrait的，然后顺时针依次经过LandscapeLeft，PortraitUpsideDown，LandscapeRight，最后再回到UIInterfaceOrientationPortrait方向。仔细看的话我们会发现在旋转的过程中，除了frame之外，Transform也在一直变化。观察frame发现，它的变化应该是由于系统的状态栏引起的。于是将系统状态栏隐藏掉，在输出发现frame果然不再变化。因此我们可以怀疑屏幕旋转是通过变化Transform实现的。

二、什么是Transform

Transform(变化矩阵)是一种3×3的矩阵，如下图所示：

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

通过这个矩阵我们可以对一个坐标系统进行缩放，平移，旋转以及这两者的任意组着操作。而且矩阵的操作不具备交换律，即矩阵的操作的顺序不同会导致不同的结果。UIView有个transform的属性，通过设置该属性，我们可以实现调整该view在其superView中的大小和位置。

矩阵实现坐标变化背后的数学知识：

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \times \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix} \tag{1}$$

$$\begin{aligned} x' &= ax + cy + t_x \\ y' &= bx + dy + t_y \end{aligned} \tag{2}$$

设x，y分别代表在原坐标系统中的位置，x'，y'代表通过矩阵变化以后在新的系统中的位置。其中式1就是矩阵变化的公式，对式1进行展开以后就可以得到式2。从式2我们可以清楚的看到（x，y）到（x'，y'）的变化关系。

1）当c，b，tx，ty都为零时，x' = ax，y' = by；即a，d就分别代表代表x，y方向上放大的比例；当a，d都为1时，x' = x，y' = y；这个时候这个矩阵也就是传说中的CGAffineTransformIdentity(标准矩阵)。

2）当a，d为1，c，b为零的时候，x' = x + tx，y' = y + ty；即tx，ty分别代表x，y方向上的平移距离。

3）前面两种情况就可以实现缩放和平移了，那么旋转如何表示呢?

假设不做平移和缩放操作，那么从原坐标系中的一点(x，y)旋转α°以后到了新的坐标系中的一点(x'，y')，那么旋转矩阵如下：

$$\begin{bmatrix} \cos a & \sin a & 0 \\ -\sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

展开以后就是x' = xcosα - ysinα，y' = xsinα + ycosα；

实际应用中，我们将这些变化综合起来，即可完成所有二维的矩阵变化。现在我们在回过头来看看前面设备旋转时的输出，当设备位于Portrait的时候由于矩阵是标准矩阵，所以没有进行打印。当转到UIInterfaceOrientationLandscapeLeft方向的时候，我们的设备是顺时针转了90°(逆时针为正，顺时针为负)，这个时候矩阵应该是（cos-90°，sin-90°，-sin-90°，cos-90°，tx，ty），由于未进行平移操作所以tx，ty都为0，刚好可以跟我们控制台输出：**"<UIView: 0x8075390; frame = (0 0; 320 480); transform = [0, -1, 1, 0, 0, 0]; autoresize = W+H; layer = <CALayer: 0x8074980>>"**一致。观察其他两个方向的输出，发现结果均和分析一致。

由此可以发现屏幕旋转其实就是通过view的矩阵变化实现，当设备监测到旋转的时候，会通知当前程序，当前程序再通知程序中的window，window会通知它的rootViewController的，rootViewController对其view的transform进行设置，最终完成旋转。

如果我们直接将一个view添加到window上，系统将不会帮助我们完成旋操作，这个时候我们就需要自己设置该view的transform来实现旋转了。这种情况虽然比较少，但是也存在的，例如现在很多App做的利用状态栏进行消息提示的功能就是利用自己创建window并且自己设置transform来完成旋转支持的，下一篇博客会介绍如何实现这种消息通知。

注：转载请注明出处！欢迎大家加我QQ 1592232964，一起讨论共同进步。

分类：iOS