CATransition详解（转场动画）

**iOS开发UI篇—核心动画简介**

**一、简单介绍**

Core Animation，中文翻译为核心动画，它是一组非常强大的动画处理API，使用它能做出非常炫丽的动画效果，而且往往是事半功倍。也就是说，使用少量的代码就可以实现非常强大的功能。

Core Animation是跨平台的，可以用在Mac OS X和iOS平台。

Core Animation的动画执行过程都是在后台操作的，不会阻塞主线程。不阻塞主线程，可以理解为在执行动画的时候还能点击（按钮）。

要注意的是，Core Animation是直接作用在CALayer上的，并非UIView。

**二、Core Animation的使用步骤**

1.使用它需要先添加QuartzCore.framework框架和引入主头文件<QuartzCore/QuartzCore.h>(iOS7不需要)

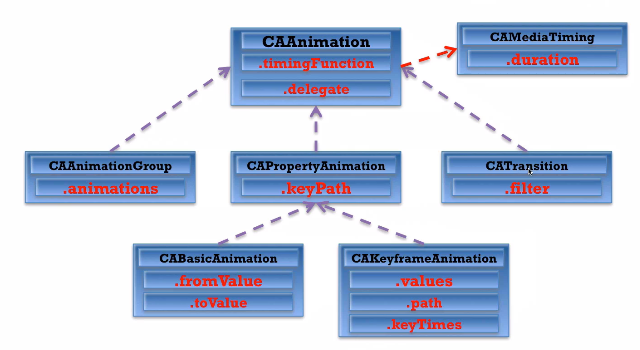
2.初始化一个CAAnimation对象，并设置一些动画相关属性

3.通过调用CALayer的addAnimation:forKey:方法增加CAAnimation对象到CALayer中，这样就能开始执行动画了

4.通过调用CALayer的removeAnimationForKey:方法可以停止CALayer中的动画

**三、CAAnimation**

类的继承结构图



CAAnimation是所有动画类的父类，但是它不能直接使用，应该使用它的子类。

 常见属性有：

duration：动画的持续时间

repeatCount：动画的重复次数

timingFunction：控制动画运行的节奏

说明：（1）能用的动画类只有4个子类：CABasicAnimation、CAKeyframeAnimation、CATransition、CAAnimationGroup

　　   （2）CAMediaTiming是一个协议(protocol)。

CAPropertyAnimation是CAAnimation的子类，但是不能直接使用，要想创建动画对象，应该使用它的两个子类：CABasicAnimation和CAKeyframeAnimation

它有个NSString类型的keyPath属性，你可以指定CALayer的某个属性名 为keyPath，并且对CALayer的这个属性的值进行修改，达到相应的动画效果。比如，指定@"position"为keyPath，就会修改 CALayer的position属性的值，以达到平移的动画效果

**四、补充说明**

所有动画对象的父类，负责控制动画的持续时间和速度，是个抽象类，不能直接使用，应该使用它具体的子类

属性解析：(红色代表来自CAMediaTiming协议的属性)

**duration**：动画的持续时间

**repeatCount**：动画的重复次数

**repeatDuration**：动画的重复时间

**removedOnCompletion**：默认为YES，代表动画执行完毕后就从图层上移除，图形会恢复到动画执行前的状态。如果想让图层保持显示动画执行后的状态，那就设置为NO，不过还要设置fillMode为kCAFillModeForwards

**fillMode**：决定当前对象在非active时间段的行为.比如动画开始之前,动画结束之后

**beginTime**：可以用来设置动画延迟执行时间，若想延迟2s，就设置为CACurrentMediaTime()+2，CACurrentMediaTime()为图层的当前时间

**timingFunction**：速度控制函数，控制动画运行的节奏

**delegate**：动画代理

**iOS 中key和keypath的区别**

在iOS开发中经常碰到重载方法有传key和keypath两种，但是一直搞不清楚他们两的区别，最近研究了一下官网KVC编程对key和keypath的区别有了具体的认识：

例如：

KVC编程时setValue(value: AnyObject?, key: String)和setValue(value: AnyObject?, keypath: String)

key：只能接受当前类所具有的属性，不能是自己的，还是从父类继承过来的，如view.setValue(CGRectZero(), key: "frame")；

keypath: 出了能接受当前类的属性，还能接受当前类属性的属性，即可以接受关系链，如view.setValue(5, keypath: "layer.cornerRadius")

/// 转换类型

1. Type

kCATransitionFade /// 交叉淡化过度

kCATransitionMoveIn /// 新视图移到旧视图上面

kCATransitionPush /// 新视图把旧视图推出去

kCATransitionReveal /// 将旧视图移开，显示下面的新视图

2. 字符串表示

pageCurl /// 向上翻一页

pageUnCurl     /// 向下翻一页

rippleEffect /// 滴水效果

suckEffect /// 收缩效果，如一块布被抽走

cube /// 立方体效果

oglFlip /// 上下翻转效果

3. CAMediaTimingFunction 动画时间变化

kCAMediaTimingFunctionLinear 线性（匀速）|  
    kCAMediaTimingFunctionEaseIn 先慢|  
    kCAMediaTimingFunctionEaseOut 后慢|  
    kCAMediaTimingFunctionEaseInEaseOut 先慢 后慢 中间快|  
    kCAMediaTimingFunctionDefault 默认|

4. CAAnimation 动画

设置removedOnCompletion ＝ false （动画完成不移除动画）

fillMode 决定 非actiive状态下行为

**kCAFillModeRemoved** 这个是默认值,也就是说当动画开始前和动画结束后,动画对layer都没有影响,动画结束后,layer会恢复到之前的状态   
**kCAFillModeForwards** 当动画结束后,layer会一直保持着动画最后的状态   
**kCAFillModeBackwards** 这 个和kCAFillModeForwards是相对的,就是在动画开始前,你只要将动画加入了一个layer,layer便立即进入动画的初始状态并等待 动画开始.你可以这样设定测试代码,将一个动画加入一个layer的时候延迟5秒执行.然后就会发现在动画没有开始的时候,只要动画被加入了 layer,layer便处于动画初始状态   
**kCAFillModeBoth** 理解了上面两个,这个就很好理解了,这个其实就是上面两个的合成.动画加入后开始之前,layer便处于动画初始状态,动画结束后layer保持动画最后的状态.