

[Handy's](#)

Stay hungry, stay foolish.

[Blog Archives](#)

动画的创建和执行过程

Oct 11th, 2015 | [Comments](#)

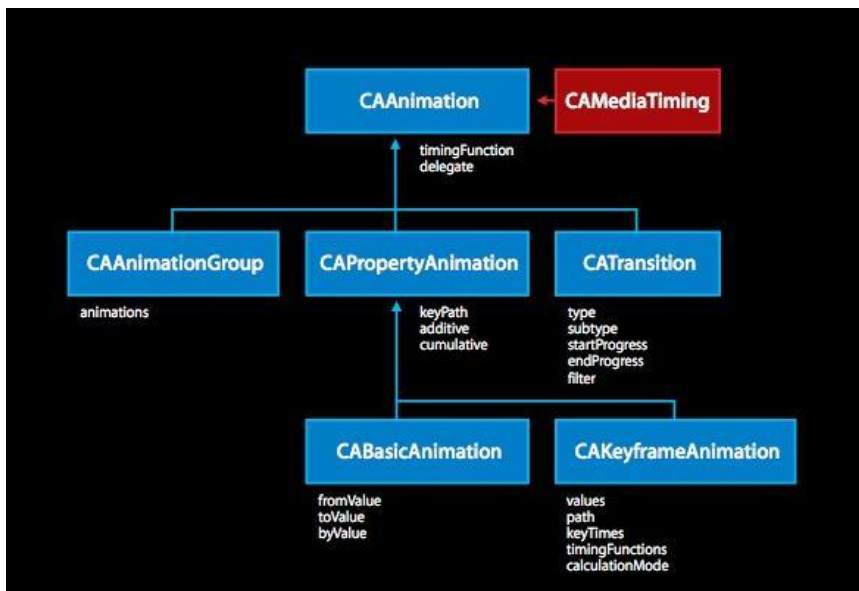
基本概念介绍

本文假设您已有RunLoop和CAAnimation的相关知识，所以这里不对[RunLoop](#)和CAAnimation的细节进行介绍。但是，这里需要提及几个重要知识点：CFRunLoopActivity、CAAction、CAAnimation、CATransaction

RunLoop的6个CFRunLoopActivity

```
1 typedef CF_OPTIONS(CFOptionFlags, CFRunLoopActivity) {  
2     kCFRunLoopEntry = (1UL << 0), //即将进入RunLoop  
3     kCFRunLoopBeforeTimers = (1UL << 1), //Timer即将触发  
4     kCFRunLoopBeforeSources = (1UL << 2), //Source0即将触发  
5     kCFRunLoopBeforeWaiting = (1UL << 5), //RunLoop即将休眠  
6     kCFRunLoopAfterWaiting = (1UL << 6), //RunLoop已被唤醒  
7     kCFRunLoopExit = (1UL << 7), //RunLoop已结束  
8     kCFRunLoopAllActivities = 0xFFFFFFFFUL  
9 };
```

CAAnimation的继承结构



CAAction

CAAction即动画行为，它是一个protocol(CAAnimation实现了此protocol)，即定义一个动画要做的事情。无论是否在Animaiton block里修改View的属性时，都会触发Core Animation回调CALayer，CALayer再回调UIView来获取CAAction(如：CABasicAnimation)；如果不在Animaiton block里修改View的属性时，CALayer再回调UIView时获取到的是NSNull null；而在Animaiton block里修改View的属性时，CALayer再回调UIView时获取到的是CAAction的一个实现类(如CABasicAnimation等，后面有流程图来说明这个过程)。

CATransaction

CATransaction是一个与动画相关的概念，它负责把多个对Layer或View的可动画属性的修改集中在一起一次性提交并执行，所以Animation应该需要被包含在CATransaction中的。CATransaction分为隐式和显示。

如下，这就是一个显示的CATransaction代码片断，即由开发人员来begin和commit。

```
1 [CATransaction begin];  
2 [CATransaction setValue:@(NO) forKey:kCATransactionDisableActions];  
3 [CATransaction setValue:@(0.5) forKey:kCATransactionAnimationDuration];  
4 [CATransaction setValue:^(void) {NSLog(@"Completion...");} forKey:kCATransactionCompletionBlock];  
5 _animLayer.position = CGPointMake(_animLayer.position.x, _animLayer.position.y - 10);  
6 [CATransaction commit];
```

从代码片断中可见，虽然只是修改了layer的position，但是最终动画的duration、动画回调都可能通过CATransaction来提供。当不是简单的修改position，而是给layer加一个CAAnimation时，最终动画的duration等参数则是以CAAnimation的内容为主。另外，CATransaction还支持嵌套，以最外层commit为准。这里还需要关注一下对kCATransactionDisableActions的修改，kCATransactionDisableActions表示是否禁用CAAction的检索：若为YES则禁用，即无论是否在Animation Block里对UI属性进行修改都不会有动画效果，因为此时actionForKey:方法不会回调；若为NO则开启CAAction的检索。

关于隐式的CATransaction，有[资料](#)说是

```
1 在大多数情况下，我们并不需要去创建自己的transaction。
2 当我们给layer添加一个显式或者隐式的Animation的时候，core animation会自动的为我们创建一个隐式的
3 transaction。
```

也有[资料](#)说是

```
1 CATransaction也分两类，显式的和隐式的，当在某次RunLoop中设置一个animatable属性的时候，
2 如果发现当前没有事务，则会自动创建一个CA事务
```

但是，无论是在非Animation Block里修改Layer的position等属性就有隐式动画效果，还是在Animation Block里修改View的center等属性就有显式动画效果，在断点+[CATransaction begin]、CA::Transaction::create()方法时，这两个方法都没有被调用，说明没有新建CATransaction。

```
1 //隐式动画 - 在非Animation Block里修改Layer的position等属性就有隐式动画效果
2 _animLayer.position = CGPointMake(_animLayer.position.x, _animLayer.position.y - 10);
3
4 //显式动画 - 在Animation Block里修改View的center等属性就有显式动画效果
5 [UIView animateWithDuration:0.5 animations:^(
6     _animView.center = CGPointMake(_animView.center.x, _animView.center.y - 10);
7 }]);
```

前面我们提到Animation是要依附于CATransaction的commit才得以执行的，而且我们在实验隐式动画和显式动画时，当CAAction被查找(后面会图示)完成后CA::Transaction:commit会被调用到，所以RunLoop回调CoreAnimation进行创建动画的过程时应该事先已创建了一个根Transaction，再次通过断点CA::Transaction::create方法也证明了Main RunLoop通过回调CFRUNLOOP_IS_CALLING_OUT_TO_A_SOURCE0_PERFORM_FUNCTION只调用了一次CA::Transaction::create方法来创建了一个根Transaction，所以这就能解释为什么隐式动画和显式动画时Core Animation内部没有新建CATransaction而只是在检索CAAction返回后调用CA::Transaction:commit就会有动画效果了。所以，我理解的隐式CATransaction是指这里我实验分析出来的根Transaction。

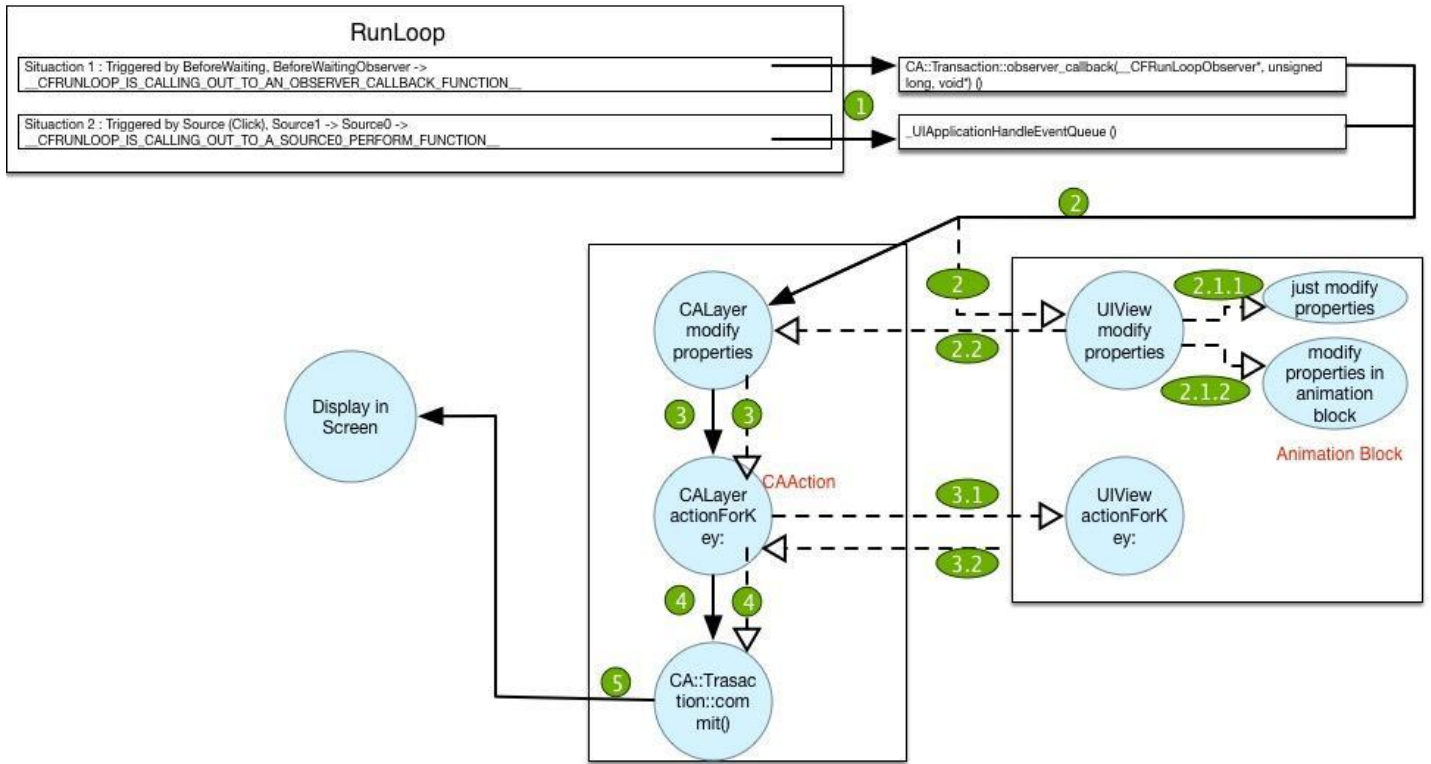
动画的创建、执行过程

在了解了以上知识后，我们再来看UIView或CALayer中通过或不通过Animation Block来修改样式属性时的动画创建和执行过程。

先看一段代码，如下：

```
1 代码一：隐式动画
2 _animLayer = [IDCAAnimationTestLayer new];
3 _animLayer.position = CGPointMake(_animLayer.position.x, _animLayer.position.y - 10);
4
5 代码二：无动画效果
6 _animView.layer.position = CGPointMake(_animView.layer.position.x, _animView.layer.position.y - 10);
7
8 代码三：无动画效果
9 [CATransaction begin];
10 [CATransaction setValue:@(NO) forKey:kCATransactionDisableActions];
11 [CATransaction setValue:@(0.5) forKey:kCATransactionAnimationDuration];
12 _animView.layer.position = CGPointMake(_animView.layer.position.x, _animView.layer.position.y - 10);
13 [CATransaction commit];
14
15 代码四：显式动画
16 [UIView animateWithDuration:0.5 animations:^(
17     _animView.center = CGPointMake(_animView.center.x, _animView.center.y - 10);
18 }]);
19
20 代码五：无动画效果
21 _animView.center = CGPointMake(_animView.center.x, _animView.center.y - 10);
```

从以上代码中可以看到：为什么同样是对CALayer或UIView的相同属性作修改，而有些有动画而有些没有动画呢？这个疑问先放一下，我们接下来看下面这张图。



如上图，我来分别梳理一个修改CALayer和UIView属性时的动画创建过程(没错，其实不在动画块里修改属性也有动画创建过程存在的)。

CALayer的动画创建和执行过程(如图中黑实线箭头及序号)

- RunLoop在两种情况下会触发与动画创建相关的回调：BeforeWaiting、点击事件。(ExitRunLoop是否会触发待研究) TODO:这里的细节还有待调研
- RunLoop回调到开发人员写的修改CALayer属性的代码
- 当修改CALayer的animatable属性时会触发CALayer自己的actionForKey:方法来查找相应的CAAction，CALayer的actionForKey:方法的默认实现会返回一个实现了CAAction protocol的CAAnimation，如CABasicAnimation
- 开发人员书写的修改CALayer属性的代码执行完毕后，在RunLoop的回调函数的后续逻辑会调用CA::Transaction:commit()来提交之前修改CALayer属性时而获取到的CAAction。
- 最后相应的动画效果显示到屏幕上(TODO:这里的细节还有待调研)

UIView的动画创建和执行过程(如图中虚线箭头及序号)

- RunLoop在两种情况下会触发与动画创建相关的回调：BeforeWaiting、点击事件。(ExitRunLoop是否会触发待研究) TODO:这里的细节还有待调研
- RunLoop回调事件回调到开发人员写的修改UIView几何或透明度等UI属性的代码，这里修改有两种情况如下：

“`objective c

- 不在动画代码块里修改UIView属性（修改UIView的几何、透明度等UI属性是通过CALayer的对应属性修改来体现的，修改UIView的center是通过CALayer的position来体现的）
- 在动画代码块里修改UIView属性

如A所说，无论是否在动画代码块里修改UIView的几何、透明度等UI属性，实际上是通过CALayer修改对应属性来完成的，这是因为UIView与CALayer的关系是平行的，且UIView是CALayer的Delegate。“`

- 与CALayer的动画创建和执行过程的第3步流程一致。但是需要注意的是：在2.A情况下，那么返回的是[NSNull null]，即后续不会有动画(表示停止对CAAction的检索)；在2.B情况下，那么返回的结果与CALayer的动画创建和执行过程第3步一样，即返回实现了CAAction protocol的CAAnimation子类。从这两种情况我们就不难明白，为什么不在Animation Block里修改UIView的UI属性后续就没有动画效果，反之有动画效果，这都是由actionForKey:是否有确切的CAAction值决定的。顺带讲一下，actionForKey:有三种返回值情况：id - 确切的动画、nil - 没有任何动画行为、[NSNull null] - 停止对CAAction的检索。
- 同CALayer的动画创建和执行过程的第4步流程一致
- 同CALayer的动画创建和执行过程的第5步流程一致

讲到这里，我们再来回顾前面的问题“为什么同样是对CALayer或UIView的相同属性作修改，而有些有动画而有些没有动画呢？”，我在下面作出了解释：

```

1  代码一：隐式动画
2  _animLayer = [IDCAAnimationTestLayer new];
3  _animLayer.position = CGPointMake(_animLayer.position.x, _animLayer.position.y - 10);
4  解释：这里首先解释一个名词叫Root Layer和非Root Layer。很简单，Root Layer就是指有对应UIView的Layer，
5  非Root Layer就是指没有对应UIView的Layer，在CALayer里只有对非Root Layer的UI属性修改才会有隐式动画效果。
6  其实，不难理解，因为Root Layer有UIView，且修改UI属性的代码没有写到Animation Block里，
7  所以CAAction的返回被UIView的actionForLayer:forKey:方法返回为[Null null]，进一步回调到CALayer里actionForKey:的返回值为nil。
8  但是，如果在动画代码块里修改Root Layer的UI属性是会有动画效果的。
9  显然，_animLayer是一个非Root Layer，修改position属性后会按照上面的“CALayer的动画创建和执行过程”来执行，所以有动画效果。
10
11
12 代码二：无动画效果
13 _animView.layer.position = CGPointMake(_animView.layer.position.x, _animView.layer.position.y - 10);
14 解释：显然_animView.layer是一个Root Layer，而且没有写在动画代码块里，所以不会有隐式动画。
15
16
17 代码三：无动画效果
18 [CATransaction begin];
19 [CATransaction setValue:@(NO) forKey:kCATransactionDisableActions];
20 [CATransaction setValue:@(0.5) forKey:kCATransactionAnimationDuration];
21 _animView.layer.position = CGPointMake(_animView.layer.position.x, _animView.layer.position.y - 10);
22 [CATransaction commit];
23 解释：同**代码二**
24
25 代码四：显式动画
26 [UIView animateWithDuration:0.5 animations:^(
27     _animView.center = CGPointMake(_animView.center.x, _animView.center.y - 10);
28 }]);
29 解释：参见**UIView的动画创建和执行过程**及其中的2.1.2点，所以有动画效果。
30
31 代码五：无动画效果
    _animView.center = CGPointMake(_animView.center.x, _animView.center.y - 10);
    解释：参见**UIView的动画创建和执行过程**及其中的2.1.1点，所以没有动画效果。

```

以上内容就是本章的全部内容。

参考

- [RunLoop学习笔记\(一\) 基本原理介绍](#)
- [Core Animation 高级动画技巧](#)
- [谈谈iOS Animation](#)
- [iOS开发基础知识：Core Animation\(核心动画\)](#)
- [iOS 事件处理机制与图像渲染过程](#)
- 参考以上资料的过程一定要做实验

Posted by Handy.Wang and File under [Runtime](#)

发推

[« 界面渲染续之CALayer的显示流程 2015年我都干嘛了 »](#)

Comments

0 Comments Handy's

 Login ▾

 Recommend 1  Share

Sort by Best ▾



Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name

Be the first to comment.

ALSO ON HANDY'S


UIImage stretch - Handy's




1 comment • a year ago

 handywang — Hello

浅谈内存布局(Memory Layout)

2 comments • a year ago

 handywang — 不知道你期望什么样的更新，我们可以一起讨论。

 Subscribe  Add Disqus to your siteAdd DisqusAdd  Privacy

Google搜索

最新文章

- [UIImage stretch](#)
- [UITableView & UITextField的九宫格对齐方式](#)
- [SDWebImage支持URL不变时更新图片内容](#)
- [iOS Crash快速分析实战](#)
- [软件架构模式 \(译\)](#)

Copyright © 2016 – Handy.Wang – Powered by [Octopress](#) on [GitHubPages](#) – Theme by [Cho](#)