关于 IOS 动画

1. 几个概念

(1) Quartz2D：是一组二维绘图和渲染 API，Core Graphic 会用到这组 API

(2) Core Graphics：是基于 C 的 API，可用于一切绘图操作，是 ios 的核心图形库

(3) Quartz Core：专指 Core Animation 用到的动画相关的库、API 和类

(4) Core Animation 和 Core Grapphics 是跨 ios 和 mac os 使用的，Core Animation 大量用到 Core Graphics 中的类，因为实现动画自然要用到图形库中的东西

(5) Core Animation：核心动画，想做动画用 CA 开头的准没错

2. 概览

在 ios 中随处可见绚丽的动画效果，实现过程并不复杂，接下来将介绍 ios 中如何使用图层精简非交互式绘图；如何通过核心动画创建基础动画、关键帧动画、动画组、转场动画；如何通过 UIView 的装饰方法对这些动画进行简化等。

(1) CALayer：CALayer 简介、CALayer 常用属性、CALayer 绘图

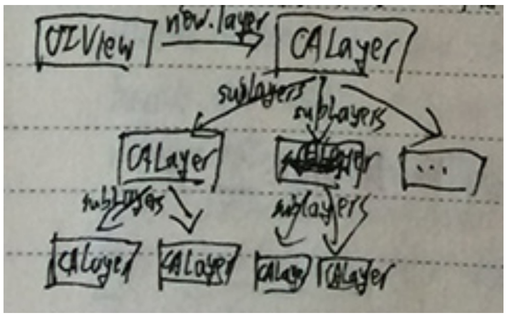
(2) Core Animation：基础动画、关键帧动画、动画组、转场动画、逐帧动画

(3) UIView 动画封装：基础动画、关键帧动画、转场动画

3. CALayer

(1) CALayer 简介

CALayer 包含在 Quartz Core 框架中。使用 Core Animation 开发动画等本质就是将 CALayer 中的内容转化为位图位图从而供硬件操作，所以要熟练掌握动画操作必须先熟悉 CALayer。Quartz 2D 绘图在 drawRect:方法绘图的本质是绘制到了 UIView 的 layer 中，而 Core Animation 操作更多的不再是 UIView 而是直接面对 CALayer，以下为 UIView 和 CALayer 的关系，在 UIView 中有一个 layer 作为根图层，根图层可放其他子图层，在 UIView 中所有能够看到的内容全部都包含在 layer 中。



(2)CALayer 常用属性

ios 中 CALayer 的设计主要为了内容展示和动画操作，CALayer 本身不包含在 UIKit 中，不能响应时间。CALayer 很多属性在修改时都能形成动画效果，这种属性称为“隐式动画属性”，但是对 UIView 的根图层更多充当容器的作用，如果属性形变形成动画效果是会直接影响子图层的。另外，UIView 的根图层创建工作由 ios 负责完成，无法重新创建，但可添加或移除子图层，下面为 CALayer 的常用属性。

1) anchorPoint：锚点，默认为(0.5,0.5)位置，支持隐式动画，永远和 position 重合

2) backgroundColor：图层背景色，支持隐式动画

3) borderColor：边框颜色，支持隐式动画

4) borderWidth：边框宽度，支持隐式动画

5) bounds：图层大小，支持隐式动画

6) content：图层显示内容，例：可将图片作为图层内容显示，支持隐式动画

7) contentRect：图层显示内容的位置和大小，支持隐式动画

8) cornerRadius：圆角半径，支持隐式动画

9) doubleSided：图层背面是否显示，默认 YES，不支持隐式动画

10) frame：图层大小和位置，不支持隐式动画，所以常用 bounds 和 position 代替

11) hidden：是否隐藏，支持隐式动画

12) mask：图层蒙版，支持隐式动画

13) maskToBounds：子图层是否剪切图层边界，默认 NO，支持隐式动画

14) opacity：透明度，类似 UIView 的 alpha，支持隐式动画

15) position：图层中心点位置，类似 UIView 的 center，支持隐式动画

16) shadowColor：阴影颜色，支持隐式动画

17) shadowOffset：阴影偏移量，支持隐式动画

18) shadowOpacity：阴影透明度，默认 0，设置阴影必须重设此值，支持隐式动画

19) shadowPath：阴影形状，支持隐式动画

20）shadowRadius：阴影模糊半径，支持隐式动画

21）subLayers：子图层，支持隐式动画

22）transform：图层形变，支持隐式动画

注：隐式动画本质是这些属性隐含了 CABasicAnimation 的动画实现

(3)CALayer 绘图

Quartz2D 绘图，在 drawRect:方法绘制图形，图像本质是在图层中绘制。而在图层中直接绘图的方式和原来基本一致，但 drawRect:方法由 UIKit 组件进行调用，因此可以使用一些 UIKit 封装的方法进行绘图，比如：UIBezierPath，而直接绘制到图层的方法非UIKit 直接调用,因此只能用原生的Core Graphics 方法绘制。

图层绘制有两种方法，不管哪种均需调用 setNeedDisplay 方法，注意是[layer setNeedDisplay];

1)使用图层代理方法 drawLayer:inContext:方法绘制

只需指定图层代理 layer.delegate = self,不用手动实现 CALayerDelegate，设置完后必须调用图层的 setNeedDisplay 方法，否则绘制内无法展示。

2)使用自定义图层绘图

只需定义一个 CALayer 子类，重写 drawInContext:即可，当然还得调用 layer 的 setNeedDisplay 方法,否者 drawInContext:不会调用。可以定义一个视图，在 initWithFrame 方法中 addSubLayer，并且 setNeedsDisplay，以及对定义 layer 一些初始化设置。若重写了UIView的drawRect:和drawLayer:inContext 两个方法，会发现 drawRect 中 UIGraphicsGetCurrentContext 得到的上下文是由 drawLayer:inContext 中传递的。UIView 在显示时根图层会创建一个 CGContextRef(CALayer 本质使用的是位图上下文)，同时调用 UIView 自身代理 drawLayerInContext:将上下文传递给这个方法(UIView 在创建时将代理自动设置为自身)，UIView 调用 drawRect:会接收此上下文，绘制完后给 layer，再交由硬件操作。

3)绘图实例

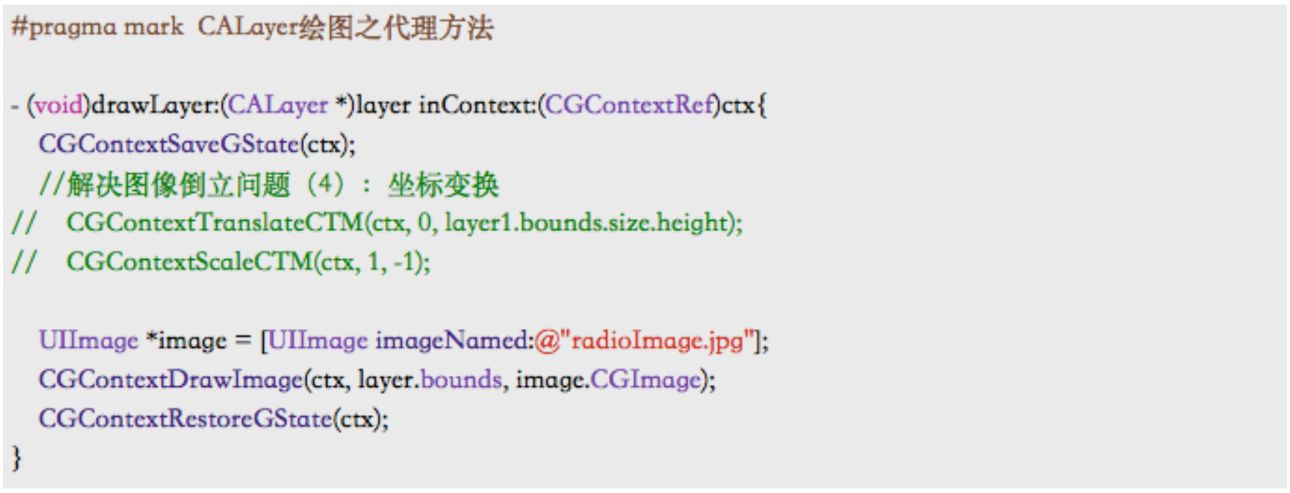
```
@implementation TestLayer

- (void)drawInContext:(CGContextRef)ctx{
    CGContextSaveGState(ctx);
    UIImage *image = [UIImage imageNamed:@"radioImage.jpg"];
    CGContextDrawImage(ctx, self.bounds, image.CGImage);
    CGContextRestoreGState(ctx);
}
```

```
@implementation ViewController_CALayer

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

```
- (void)drawDelegateLayer{
    CGSize size = self.view.bounds.size;
    CALayer *layer = [[CALayer alloc] init];
    layer.backgroundColor = [UIColor redColor].CGColor;
    layer.position = CGPointMake(size.width/2, size.height/2);
    layer.bounds = CGRectMake(0, 0, 50, 50);
    layer.cornerRadius = layer.bounds.size.width/2;
    layer.shadowColor = [UIColor grayColor].CGColor;
    layer.shadowOffset = CGSizeMake(2, 2);
    layer.shadowOpacity = .9;
    layer.masksToBounds = YES;//外边框绘制的阴影消失了，解决方法：新增底部独立阴影图层
    [self.view.layer addSublayer:layer];

    //解决图像倒立问题方法（1）：沿着x轴旋转180度
//    layer.transform = CATransform3DMakeRotation(M_PI, 1, 0, 0);

    layer.delegate = self;
    [layer setNeedsDisplay];

    //解决图像倒立问题方法（2）：设置contents
//    UIImage *image = [UIImage imageNamed:@"radioImage.jpg"];
//    [layer setContents:(id)image.CGImage];

    //解决图像倒立问题方法（3）
    [layer setValue:@M_PI forKeyPath:@"transform.rotation.x"];
}
```

```
- (void)drawDefLayer{
   CGSize size = self.view.bounds.size;
   TestLayer *layer = [[TestLayer alloc] init];//也可以新建一个view来addsublayer，则视图控制器只需负责创建视图并addsubview即可
   layer.backgroundColor = [UIColor redColor].CGColor;
   layer.position = CGPointMake(size.width/2, size.height/2);
   layer.bounds = CGRectMake(0, 0, 50, 50);
   layer.cornerRadius = layer.bounds.size.width/2;
   layer.shadowColor = [UIColor grayColor].CGColor;
   layer.shadowOffset = CGSizeMake(2, 2);
   layer.shadowOpacity = .9;
   layer.masksToBounds = YES;//外边框绘制的阴影消失了，解决方法：新增底部独立阴影图层
    [layer setNeedsDisplay];
   [self.view.layer addSublayer:layer];

   //解决图像倒立问题（3）
   [layer setValue:@M_PI forKeyPath:@"transform.rotation.x"];
}
```

```
- (void)touchesEnded:(NSSet<UITouch *> *)touches withEvent:(UIEvent *)event{
   CGPoint point = [[touches anyObject] locationInView:self.view];
   CALayer *layer = [self.view.layer.sublayers lastObject];
    CGFloat width = layer.bounds.size.width;
   layer.position = point;
   if(width == 50){
      width = 4 * width;
      layer.shadowOffset = CGSizeMake(5, 5);
   }else{
      width = 50;
      layer.shadowOffset = CGSizeMake(2, 2);
   }
   layer.bounds = CGRectMake(0, 0, width, width);
   layer.cornerRadius = layer.bounds.size.width/2;

   //隐式动画本质是基础动画
   CABasicAnimation *animation = [CABasicAnimation animationWithKeyPath:@"backgroundColor"];
   animation.duration = 0.8;
   if(width == 50){
      animation.toValue = (__bridge id _Nullable)([UIColor redColor].CGColor);
   }else{
      animation.toValue = (__bridge id _Nullable)([UIColor blueColor].CGColor);
   }
   animation.beginTime = CACurrentMediaTime();
   animation.fillMode = kCAFillModeForwards;//必须设置，否者下面一句无效，图层会被遮盖
   animation.removedOnCompletion = NO;
   [layer addAnimation:animation forKey:@"changeBackground"];
   //CALayer代理绘图
   [layer setNeedsDisplay];
}
```

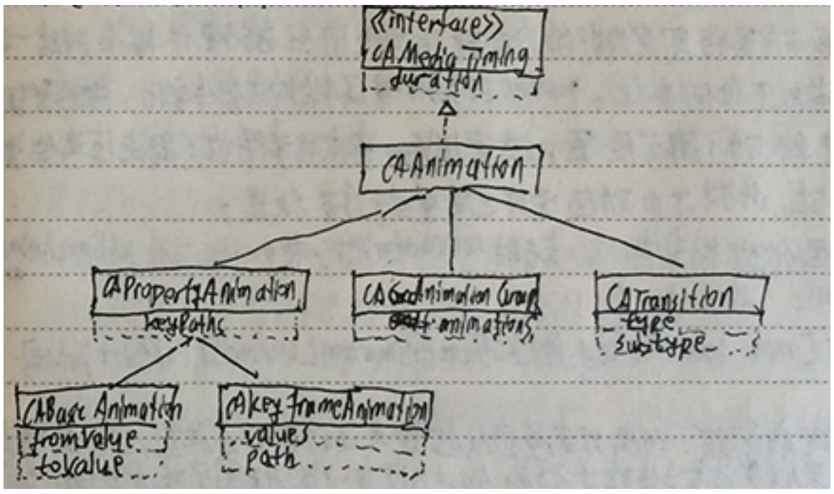


```
#pragma mark  CALayer绘图之代理方法

- (void)drawLayer:(CALayer *)layer inContext:(CGContextRef)ctx{
   CGContextSaveGState(ctx);
   //解决图像倒立问题（4）：坐标变换
//    CGContextTranslateCTM(ctx, 0, layer1.bounds.size.height);
//    CGContextScaleCTM(ctx, 1, -1);

   UIImage *image = [UIImage imageNamed:@"radioImage.jpg"];
   CGContextDrawImage(ctx, layer.bounds, image.CGImage);
   CGContextRestoreGState(ctx);
}
```

4. Core Animation

(1) 前言：

大家都知道在 ios 中实现一个动画相当简单，只需调用 UIView 块代码即可，这在其他系统开发中基本不可能实现。块代码固然方便，但具体动画如何实现就不好控制了，如动画暂停、动画组合，这就需了解核心动画 Core Animation(包含在 Quartz Core 框架中)，ios 中核心动画基本分为几类：基础动画、关键帧动画、动画组、转场动画，各个类关系大致如下：



1) CAAnimation：核心动画基础类，不能直接使用，负责动画运行时间、速度的控制，本身实现了 CAMediaTiming 协议

2) CAPropertyAnimation：属性动画的基类(通过属性进行动画设置，可动画属性)不能直接使用

3) CAAnimationGroup：动画组，是一种组合模式设计，可通过动画组来进行所有动画行为统一控制，组中所有动画效果可以并发执行

4) CATransition：转场动画，主要通过滤镜进行动画效果设置

5) CABasicAnimation：基础动画，通过属性修改进行动画参数控制，只有初始状态和结束状态

6) CAKeyframeAnimation：关键帧动画，同样是通过修改属性值产生动画效果，但可有多个状态控制

注：基础动画，关键帧动画都属属性动画，开发者只需设定状态值，补间动画由系统自动计算完成。基础动画可视为两个关键帧的关键帧动画

(2) 基础动画

在开发过程中很多情况在基础动画就可以满足开发需求，UIView 块动画实质是基础动画，ios7 中 UIView 也对关键帧动画进行了封装，只是 UIView 封装方法隐藏了很多细节，排除使用 UIView 块动画，基础动画创建一般分为如下步骤：

1) 初始化动画并设置动画属性

2) 设置动画初始值(可略)、结束值

3) 给图层添加动画

几个问题：

1) 按步骤执行完毕后动画图层回到原位置，当然使用 UIView 封装的方法没有这个问题，如何解决？

   前面所述，图层动画本质是将图层内部的内容转为位图经硬件操作形成一种动画效果，其实图层本身无任何变化。Position 和 scale 等属性均不发生变化，如果图层在一个 UIView 中，即使移动了图层位置，触发 UIView 点击事件也只能点原来位置，因为它的位置没改变过。因此，可在动画结束后重新设置位置：

```
-animationDidStop:finished:{
 _layer.postion = [[anim valueForKey:@”自定义基础动画的 key 存
 toValue      的”] CGPointValue];
}
```

2) 上面方法运行发现新问题：动画结束，重新从起始点运动到终点？

这是前面所提，非根图层的可动画属性改变时会产生隐式动画，解决方法：关闭图层隐式动画，或设置动画图层为根图层，后者不可取，因根图层可能当前作为动画背景。前者实现如下：

```
-animationDidStop:finished:{
 [CATransaction begin];
 [CATransaction setDisableActions:YES];
 _layer.postion = [[anim valueForKey:@”自定义基础动画的 key 存
 toValue     的”] CGPointValue];
 [CATransaction commit];
}
```

3) 上述方法存在瞬移现象？

可指定动画 fromValue，并在执行动画前使用动画事务将 position 移至终点位置

4) 平移动画过程控制动画暂停与继续（未理解）？

```
-animationPause{
 CALayer *layer = [self.view.layer.subLayers lastObject];
 CFTimeInterval pauseTime = [layer convertTime:CACurrentMediaTime()
 fromLayer:nil];
 [layer setTimeOffset: pauseTime];
 layer.speed = 0;
}
-animationResume{
 CALayer *layer = [self.view.layer.subLayers lastObject];
 CFTimeInterval pauseTime = layer.timeOffset;
 layer.timeOffset = 0;
 layer.speed = 1;
 CFTimeInterval interval = [layer convertTime:CACurrentMediaTime()
 fromLayer:nil] - pauseTime;
 layer.beginTime = interval;
}
```

注意：动画暂停针对的是图层而不是图层中某个动画。要做无限循环动画，动画的 removeOnCompleion 必须设为 NO 否者动画运行一次就自动销毁。

(3) 关键帧动画

分为两种形式，一种是通过设置不同的属性值尽心关键帧控制，另一种是通过绘制路径进行关键帧控制。后者优先级高于前者，若设置了 path，则 value 无效。keyTimes:@[@0.0, @0.5, @0.75, @1.0]表示：第 1 帧到第 2 帧用 0.5-0 的时间；calculationMode:动画计算模式：kCAAnimationLinear(默认)

(4) 动画组

属性动画设置时一次只能设置一个属性进行动画控制(不管是基础动画还是帧动画)，若想并发组合进行动画，统一控制，可用动画组。

1) CAAnimationGroup *animationGroup = [CAAnimationGroup animation];

2) CABasicAnimation *basicAnimation = [self rotationAnimation];

   CAKeyframeAnimation *keyAnimation = [self translationAnimation];

3) animationGroup.animations = @[basicAnimation, keyAnimation];

4) animationGroup.delegate = self;

   animationGroup.duration = 8,0f;

   animationGroup.removeOnCompletion = NO;

   animationGroup.fillMode = kCAFillModeForwards;

   animationGroup.beginTime = CACurrentMediaTime() + 5;

5) [_layer addAnimation:animationGroup forKey:nil];

(5) 转场动画

即从一个场景以动画形式过渡到另外一个场景。一般实现步骤如下：

1) 创建转场动画

2) 设置转场类型、子类型(动画方向，可选)及其他属性

3) 设置转场后的新视图并添加动画到图层

以下列出常用转场类型(私有 API 是苹果官方未公开，但目前仍可使用)

1) fade 淡出效果 kCATransitionFade 支持方向设置 公有 api

2) moveIn 新视图移到旧视图上 kCATransitionMoveIn 支持方向设置 公有 api

3) push 新视图推掉旧视图 kCATransitionPush 支持方向设置 公有 api

4) reveal 移开旧视图显示新视图 kCATransitionReveal 支持方向设置 公有 api

5) cube 立体翻转效果 支持方向设置 私有 api

6) oglFlip 翻转效果 支持方向设置 私有 api

7) suckEffect 吮吸效果 不支持方向设置 私有 api

8) rippleEffect 水滴波纹效果 不支持方向设置 私有 api

9) pageCurl 向上翻页效果 支持方向设置 私有 api

10) pageUnCurl 向下翻页效果 支持方向设置 私有 api

11) cameraIrisHollowOpen 摄像头打开效果 不支持方向设置 私有 api

12) cameraIrisHollowClose 摄像头关闭效果 不支持方向设置 私有 api

对于支持方向设置的动画类型支持以下子类型

1) kCATransitionFromRight 从右侧转场

2) kCATransitionFromLeft 从左侧转场

3) kCATransitionFromTop 从顶部转场

4) kCATransitionFromBottom 从底部转场

可用 cube 做一个循环图片轮播器，用两个 UISwipeGestureRecognizer 手势，设置 direction 分别为左和右

(6) 逐帧动画

UIImageView 的 animationImages 属性，可调用 startAnimationing 执行动画播放图片；当然，也可以使用时钟计时器 CADisplayLink 来实现逐帧动画：ios 程序在运行后就进入一个消息循环(主运行循环)，整个程序相当于进入一个死循环始终等待用户输入，将 CADisplayLink 加入主运行循环后，时钟周期和主运行循环保持一致，即屏幕刷新周期 1 秒 60 次。逐帧动画性能较低，尽量优化算法，避免循环时内存峰值过高。

1)1 个数组放入若干图片

2)定义时钟 CADisplayLink，添加到主运行循环

3)- timeDidFired:{

```
 if(_startTime == 0)

 _startTime = sender.stampTime;

 if(sender.stampTime - _startTime >= 0.11){

  UIImage *image = _images[_index];

  _layer.contents = (id)image.CGImage;//更新图片

  _index = (_index + 1) % IMAGE_COUNT;
 }

}
```

(7)UIView 动画封装

在对动画细节无特殊要求情况下，使用 UIView 对 CAAnimation 的封装 api 比较简单方便。

1)基础动画

方法一：block 方式(UIView 动画执行完动画停与终点位置，无需特殊处理)

duration：执行时间

delay：延迟时间

options：动画设置，如自动回复、匀速运动等

```
 [UIView animateWithDuration:5.0 delay:0
options:UIViewAnimationOptionCurveLinear animations:^{

 weakSelf.image.center = location;

 }completion:^(BOO finished){

 NSLog(@” Animation end”);

 }];
```

方法二：静态方法

```
 [UIView beginAnimations:@”kCABasicAnimation” context:nil];

 [UIView setAnimationDuration:5.0f];

 [UIView seAnimationRepeatAutoreverses:NO];//是否回复
```

```
[UIView setAnimationDelay:1.0f];//延时
[UIView setAnimationRepeatCount:10];//重复次数
[UIView setAnimationStartDate:(NSDate *)];//动画开始运行时间
[UIView setAnimationDelegate:self];//设置代理
[UIView setAnimationStartSelector:(SEL)];//开始动画执行方法
[UIView setAnimationDidStopSelector:(SEL)];//动画结束执行方法
_imageView.center = location;
[UIView commitAnimations];
```

2) 弹簧动画效果

```
[UIView animateWithDuration:5.0 delay:0 usingSpringWithDamping:0.1
intialSpringVelocity:1.0 options:UIViewAnimationOptionsCurveLinear
animations:^{
 weakSelf.imageView.center = location;
}completion:nil];
```

动画方法中有个 option 参数，UIViewAnimationOptions，为枚举类型，分三类，可组合使用

1. 常规动画属性设置（可同时选择多个设置）

UIViewAnimationOptionsLayoutSubViews:动画过程保证子视图跟随移动

UIViewAnimationOptionsAllowsUserIneraction:动画过程中允许用户交互

UIViewAnimationOptionsBeginFromCurrentState:所有视图当前状态开始运行

UIViewAnimationOptionsRepeat:重复运行动画

UIViewAnimationOptionsAutoreverse:动画运行结束仍折返回原点

UIViewAnimationOptionsOverrideInheritedDuration:忽略嵌套动画时间设置

UIViewAnimationOptionsOverrideInheritedCurve:忽略嵌套动画速度设置

UIViewAnimationOptionsAllowAnimatedContent:动画过程中重绘视图(转场)

UIViewAnimationOptionsShowHideTransitionViews:视图切换隐藏旧视图不移除

UIViewAnimationOptionsOverrideInHeritedOptions:不继承父动画设置或动画类型

2.动画速度控制(可从中选择一个设置)

UIViewAnimationOptionsCurveEaseInOut:动画先慢后逐渐加速

UIViewAnimationOptionsCurveEaseIn:动画逐渐变慢

UIViewAnimationOptionsCurveEaseOut:动画逐渐加速

UIViewAnimationOptionsCurveEaseLinear:匀速执行，默认值

3.转场类型(仅适转场动画，选一个设置)

UIViewAnimationOptionsTransitionNone:没有转场效果

UIViewAnimationOptionsTransitionFlipFromLeft:左侧翻转效果

UIViewAnimationOptionsTransitionFlipFromRight:右侧翻转效果

UIViewAnimationOptionsTransitionCurlUp:向后翻页效果

UIViewAnimationOptionsTransitionCurlDown:向前翻页效果

UIViewAnimationOptionsTransitionCrossDissolve:旧视图溶解消失

UIViewAnimationOptionsTransitionFlipFromTop:从上方翻转效果

UIViewAnimationOptionsTransitionFlipFromBottom:从下方翻转效果

3)关键帧动画(属性，path 动画目前 UIView 不支持)

[UIView animateKeyframesWithDuration:5.0f delay:0 options:____Linear animations:^{

//第二个关键帧位置(准确说第一个关键帧是开始位置)：从 0 秒开始，持续百分之 50 时间即 2.5s

[UIView addKeyframeWithRelativeStartTime:0.0 relativeDuration:0.5 animations:^{

```
   weakSelf.imageView.center = CGPointMake(x,y);
}];
//第三个关键帧
[UIView addKeyframeWithRelativeStartTime:0.5 relativeDuration:0.25
animations:^{
   weakSelf.imageView.center = CGPointMake(x,y);
}];
//第四个关键帧
[UIView addKeyframeWithRelativeStartTime:0.75 relativeDuration:0.25
animations:^{
   weakSelf.imageView.center = CGPointMake(x,y);
}];
}completion:^(BOO finished){
NSLog(@"Animation ended");
}];
```

4) 转场动画

```
[UIView transitionWithView:_imageView duration:1.0 options:option
animations:^{
 weakSelf.imageView.image = ;
}completion:nil];
```

如果有两个完全不同的视图，且局部复杂，两个视图转场如下：

+ (void)transitionFromView:toView:Duration:Options:completion:^{

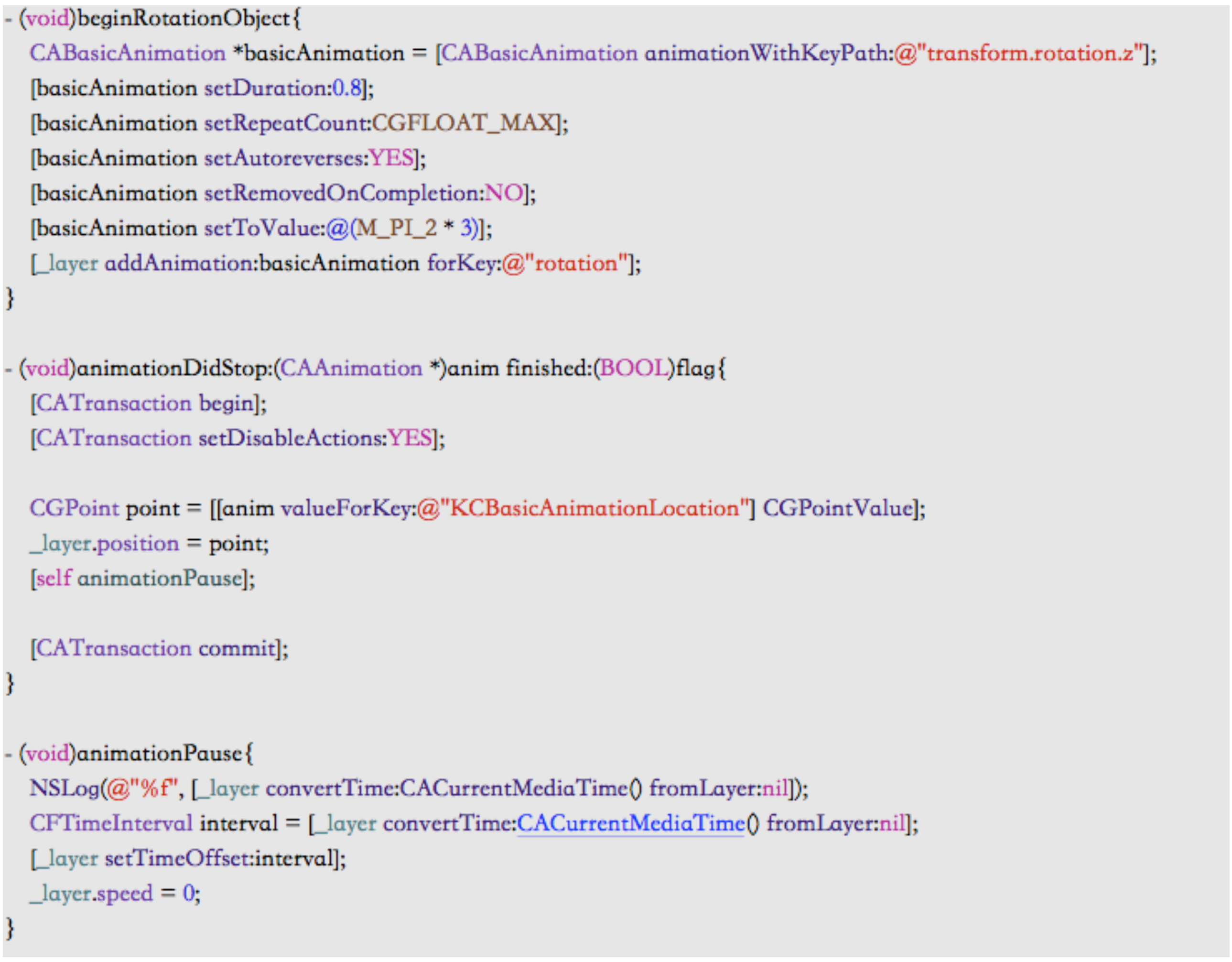
};//此方法默认情况转出的视图会从父视图移除，转入后重新添加，可通过 options = UIViewAnimationOptionsShowHideTransitionViews 设置转出视图隐藏不移除，转入后再显示，同直接使用转场动画不同的是使用 UIView 的装饰方法进行转场动画其动画效果少，因为这里无法使用私有 api。

(8) Core Animation 实例

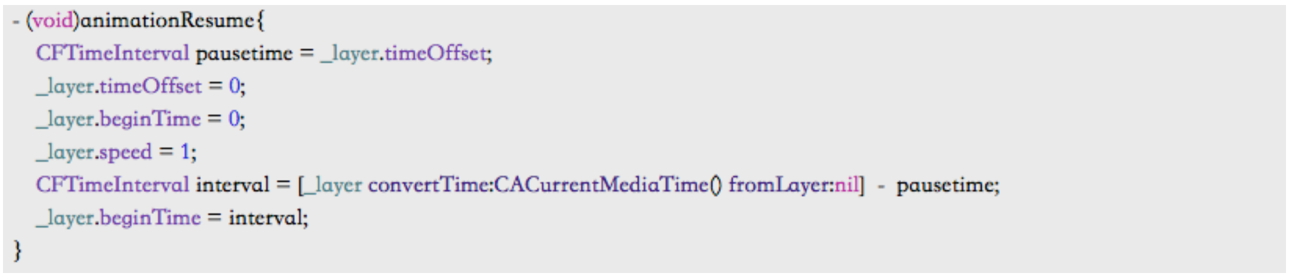
1) CABasicAnimation

```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.title = @"CABasicAnimation";
    _layer = [[CALayer alloc] init];
    _layer.bounds = CGRectMake(0, 0, 50, 50);
    _layer.position = CGPointMake(25, 100);
    _layer.allowsEdgeAntialiasing = YES;//防止边缘锯齿
    _layer.backgroundColor = [UIColor redColor].CGColor;
    [self.view.layer addSublayer:_layer];
    self.view.backgroundColor = [UIColor whiteColor];
    // Do any additional setup after loading the view.
}
```

```
- (void)touchesEnded:(NSSet<UITouch *> *)touches withEvent:(UIEvent *)event{
    UITouch *touch = [touches anyObject];
    CGPoint point = [touch locationInView:self.view];
    CAAnimation *animation = [_layer animationForKey:@"translation"];
    if(animation){
        if(_layer.speed == 1){
            [self animationPause];
        }else{
            [self animationResume];
        }
    }else{
        CABasicAnimation *basicAnimation = [CABasicAnimation animationWithKeyPath:@"position"];
        [basicAnimation setValue:[NSValue valueWithCGPoint:point] forKey:@"KCBasicAnimationLocation"];
        [basicAnimation setDuration:5.f];
        [basicAnimation setToValue:[NSValue valueWithCGPoint:point]];
        [basicAnimation setBeginTime:CACurrentMediaTime()];
        [basicAnimation setFillMode:kCAFillModeForwards];
        [basicAnimation setDelegate:self];
        [basicAnimation setRemovedOnCompletion:NO];
        [_layer addAnimation:basicAnimation forKey:@"translation"];
        [self beginRotationObject];
    }
}
```

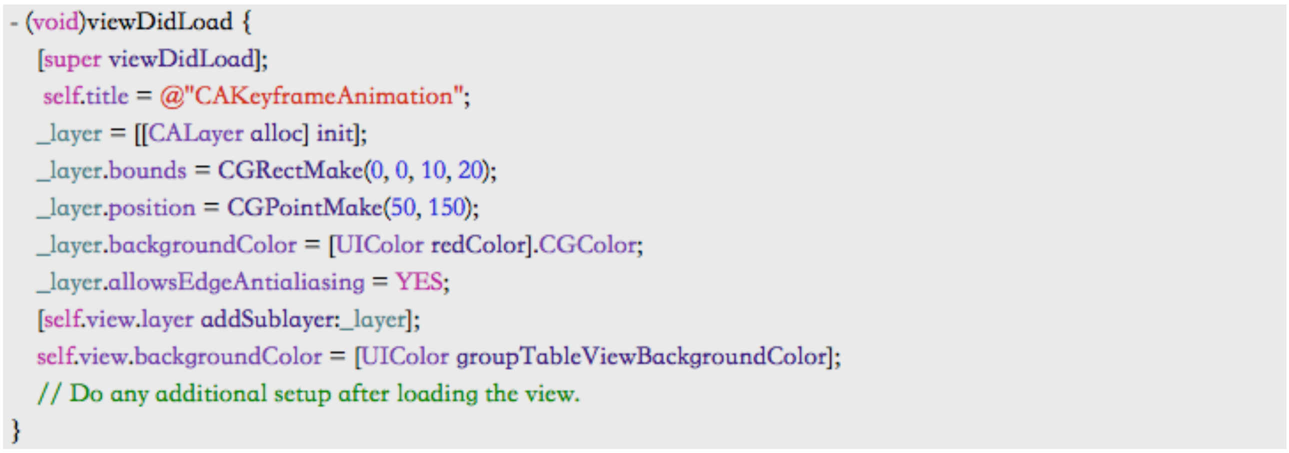


```
- (void)beginRotationObject{
   CABasicAnimation *basicAnimation = [CABasicAnimation animationWithKeyPath:@"transform.rotation.z"];
   [basicAnimation setDuration:0.8];
   [basicAnimation setRepeatCount:CGFLOAT_MAX];
   [basicAnimation setAutoreverses:YES];
   [basicAnimation setRemovedOnCompletion:NO];
   [basicAnimation setToValue:@(M_PI_2 * 3)];
   [_layer addAnimation:basicAnimation forKey:@"rotation"];
}

- (void)animationDidStop:(CAAnimation *)anim finished:(BOOL)flag{
   [CATransaction begin];
   [CATransaction setDisableActions:YES];

   CGPoint point = [[anim valueForKey:@"KCBasicAnimationLocation"] CGPointValue];
   _layer.position = point;
   [self animationPause];

   [CATransaction commit];
}

- (void)animationPause{
   NSLog(@"%f", [_layer convertTime:CACurrentMediaTime() fromLayer:nil]);
   CFTimeInterval interval = [_layer convertTime:CACurrentMediaTime() fromLayer:nil];
   [_layer setTimeOffset:interval];
   _layer.speed = 0;
}
```

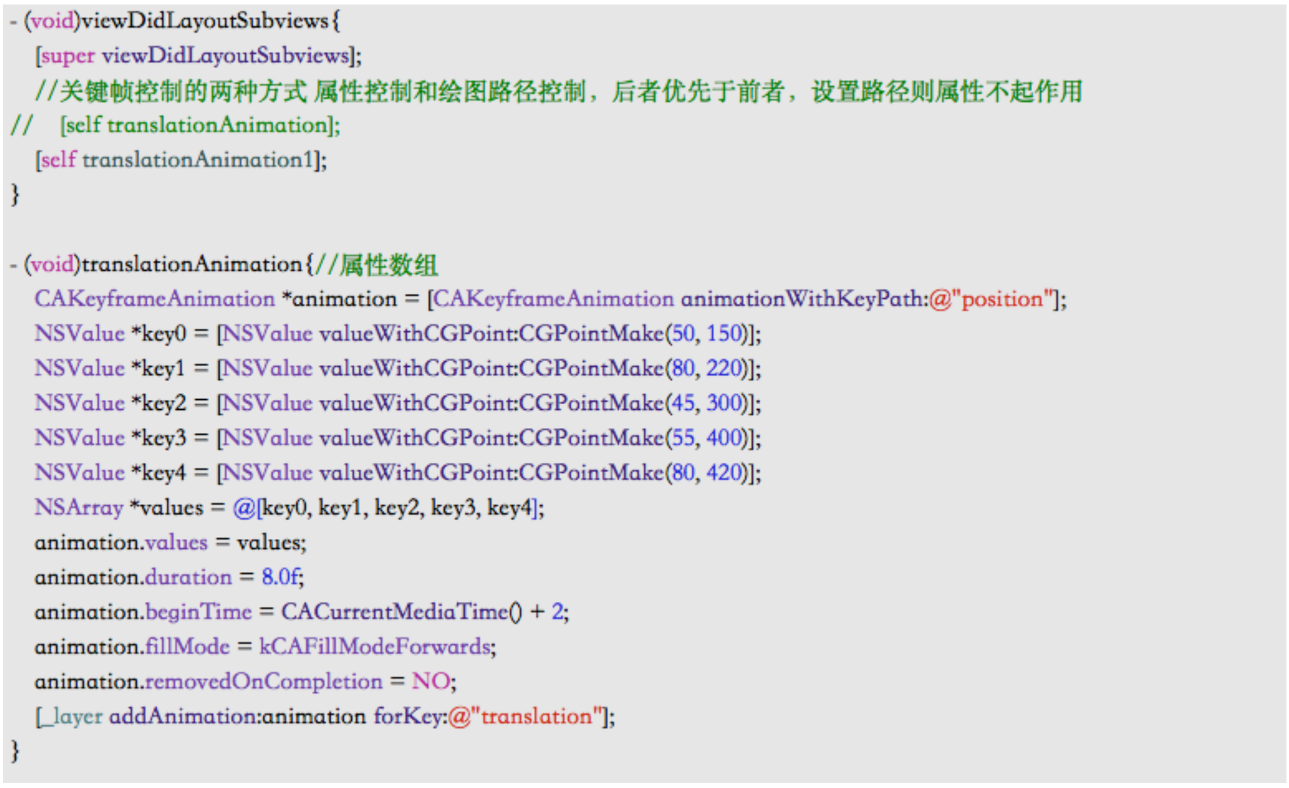


```
- (void)animationResume{
   CFTimeInterval pausetime = _layer.timeOffset;
   _layer.timeOffset = 0;
   _layer.beginTime = 0;
   _layer.speed = 1;
   CFTimeInterval interval = [_layer convertTime:CACurrentMediaTime() fromLayer:nil]  -  pausetime;
   _layer.beginTime = interval;
}
```

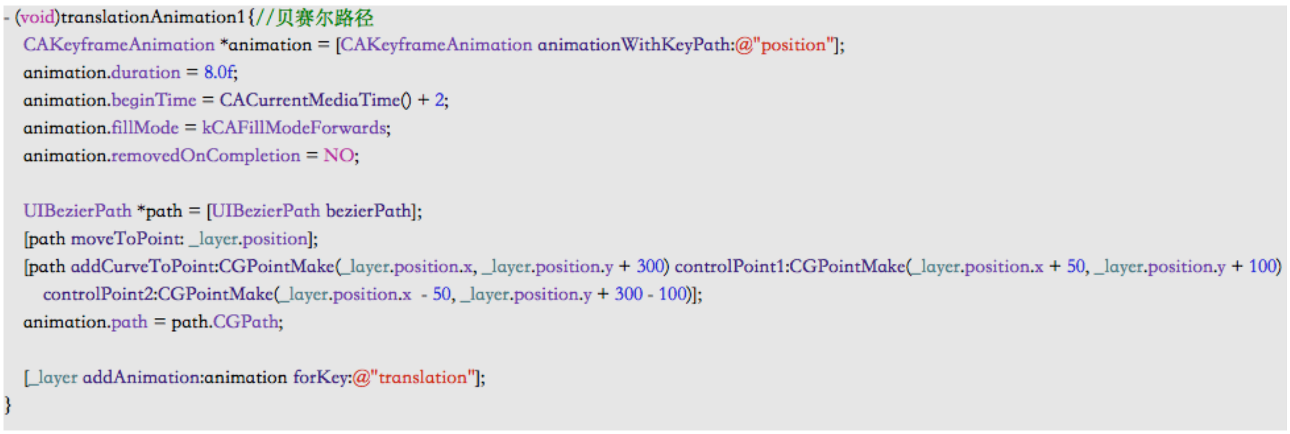
2) CAKeyframeAnimation



```
- (void)viewDidLoad {
    [super viewDidLoad];
     self.title = @"CAKeyframeAnimation";
    _layer = [[CALayer alloc] init];
    _layer.bounds = CGRectMake(0, 0, 10, 20);
    _layer.position = CGPointMake(50, 150);
    _layer.backgroundColor = [UIColor redColor].CGColor;
    _layer.allowsEdgeAntialiasing = YES;
    [self.view.layer addSublayer:_layer];
    self.view.backgroundColor = [UIColor groupTableViewBackgroundColor];
    // Do any additional setup after loading the view.
}
```

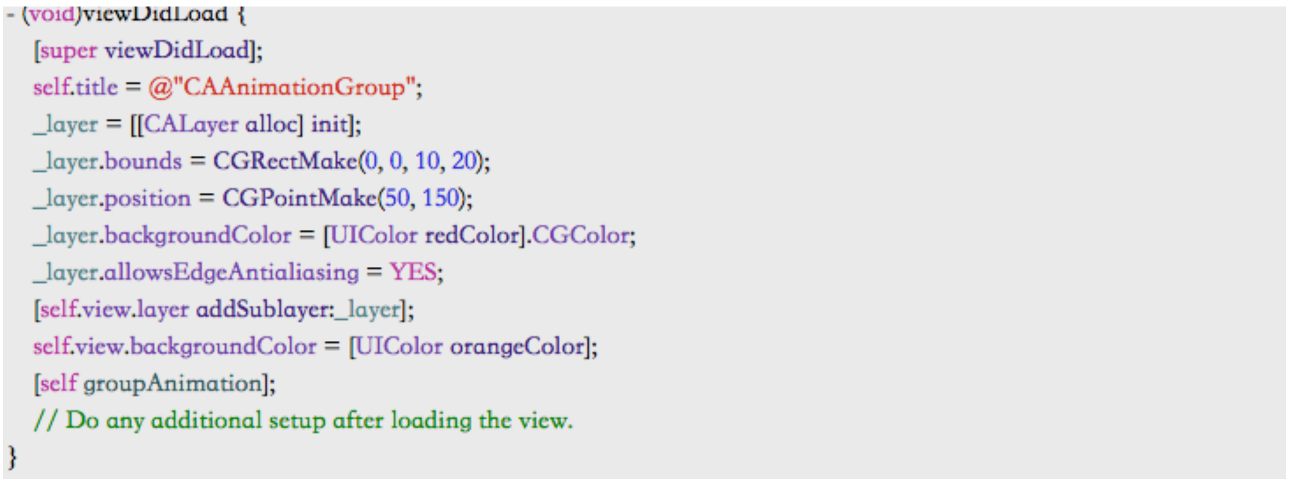


```
- (void)viewDidLayoutSubviews{
    [super viewDidLayoutSubviews];
    //关键帧控制的两种方式 属性控制和绘图路径控制，后者优先于前者，设置路径则属性不起作用
//    [self translationAnimation];
    [self translationAnimation1];
}

- (void)translationAnimation{//属性数组
    CAKeyframeAnimation *animation = [CAKeyframeAnimation animationWithKeyPath:@"position"];
    NSValue *key0 = [NSValue valueWithCGPoint:CGPointMake(50, 150)];
    NSValue *key1 = [NSValue valueWithCGPoint:CGPointMake(80, 220)];
    NSValue *key2 = [NSValue valueWithCGPoint:CGPointMake(45, 300)];
    NSValue *key3 = [NSValue valueWithCGPoint:CGPointMake(55, 400)];
    NSValue *key4 = [NSValue valueWithCGPoint:CGPointMake(80, 420)];
    NSArray *values = @[key0, key1, key2, key3, key4];
    animation.values = values;
    animation.duration = 8.0f;
    animation.beginTime = CACurrentMediaTime() + 2;
    animation.fillMode = kCAFillModeForwards;
    animation.removedOnCompletion = NO;
    [_layer addAnimation:animation forKey:@"translation"];
}
```

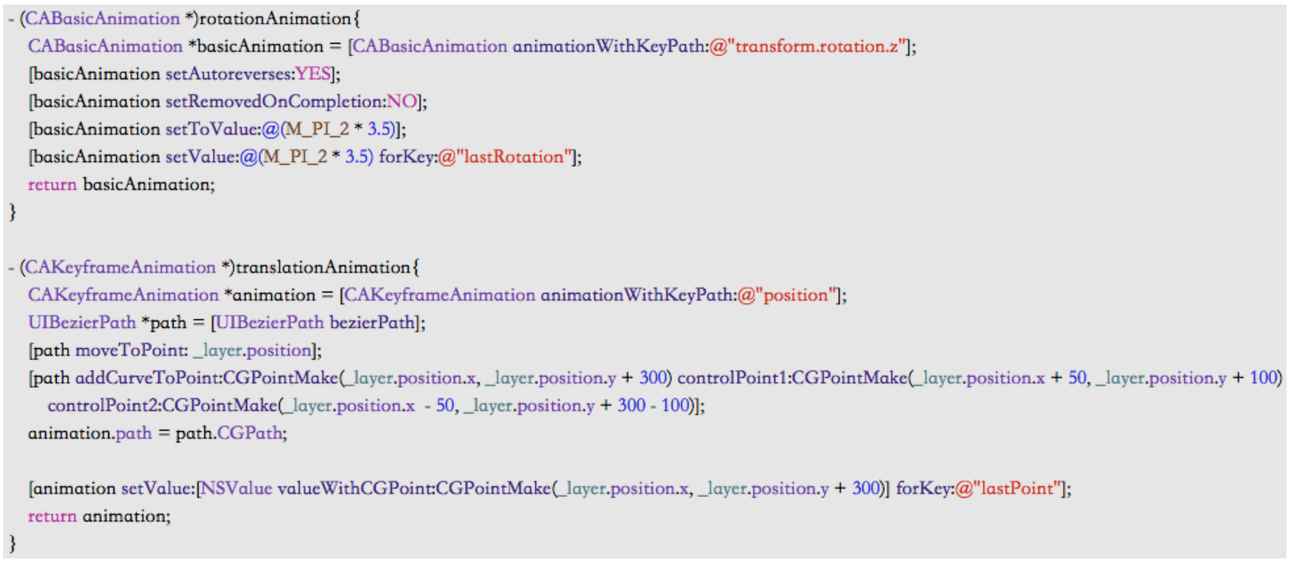


```
- (void)translationAnimation1{//贝赛尔路径
    CAKeyframeAnimation *animation = [CAKeyframeAnimation animationWithKeyPath:@"position"];
    animation.duration = 8.0f;
    animation.beginTime = CACurrentMediaTime() + 2;
    animation.fillMode = kCAFillModeForwards;
    animation.removedOnCompletion = NO;

    UIBezierPath *path = [UIBezierPath bezierPath];
    [path moveToPoint: _layer.position];
    [path addCurveToPoint:CGPointMake(_layer.position.x, _layer.position.y + 300) controlPoint1:CGPointMake(_layer.position.x + 50, _layer.position.y + 100)
      controlPoint2:CGPointMake(_layer.position.x  - 50, _layer.position.y + 300 - 100)];
    animation.path = path.CGPath;

    [_layer addAnimation:animation forKey:@"translation"];
}
```

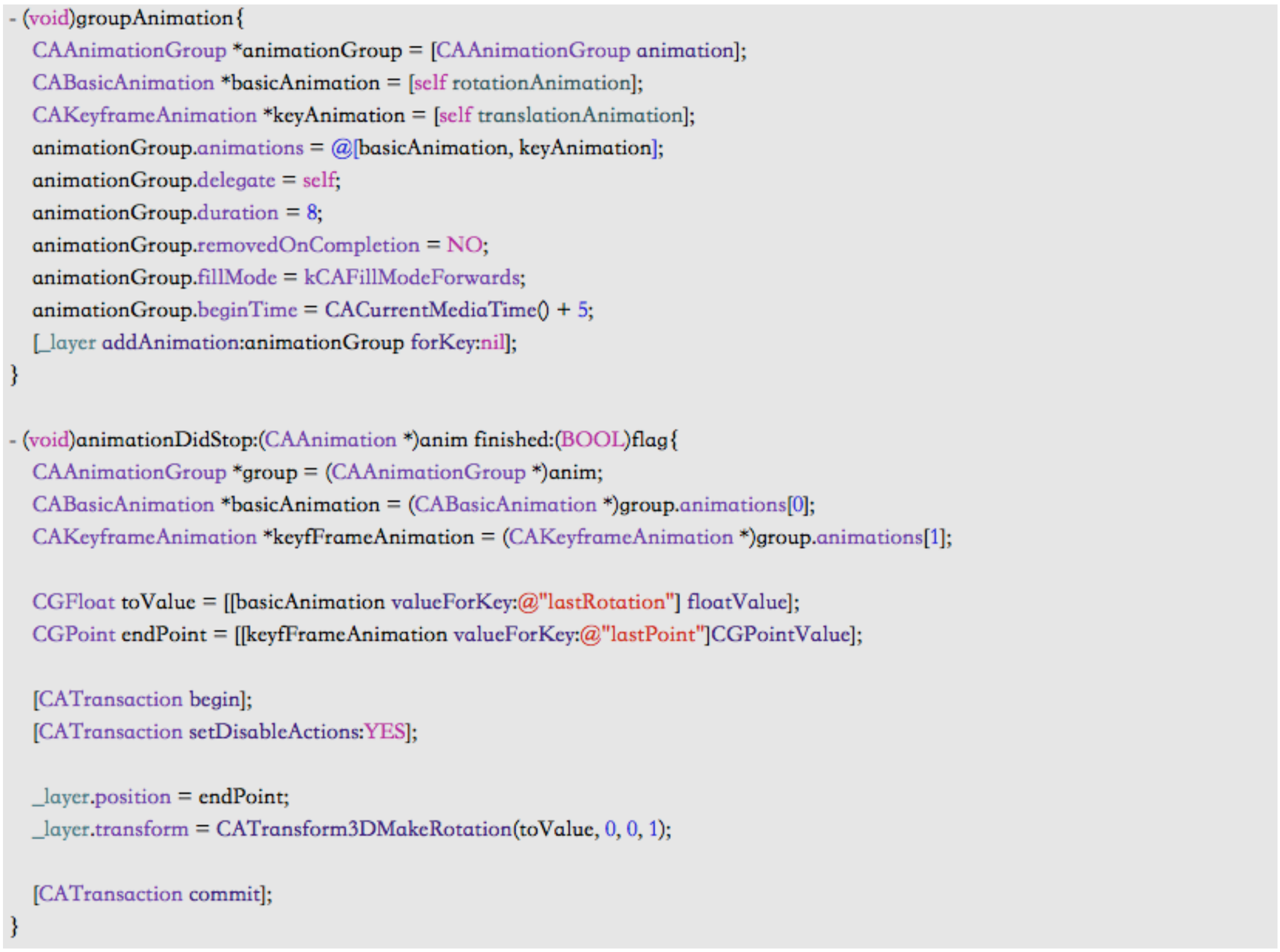
3) CAAnimationGroup



```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.title = @"CAAnimationGroup";
    _layer = [[CALayer alloc] init];
    _layer.bounds = CGRectMake(0, 0, 10, 20);
    _layer.position = CGPointMake(50, 150);
    _layer.backgroundColor = [UIColor redColor].CGColor;
    _layer.allowsEdgeAntialiasing = YES;
    [self.view.layer addSublayer:_layer];
    self.view.backgroundColor = [UIColor orangeColor];
    [self groupAnimation];
    // Do any additional setup after loading the view.
}
```

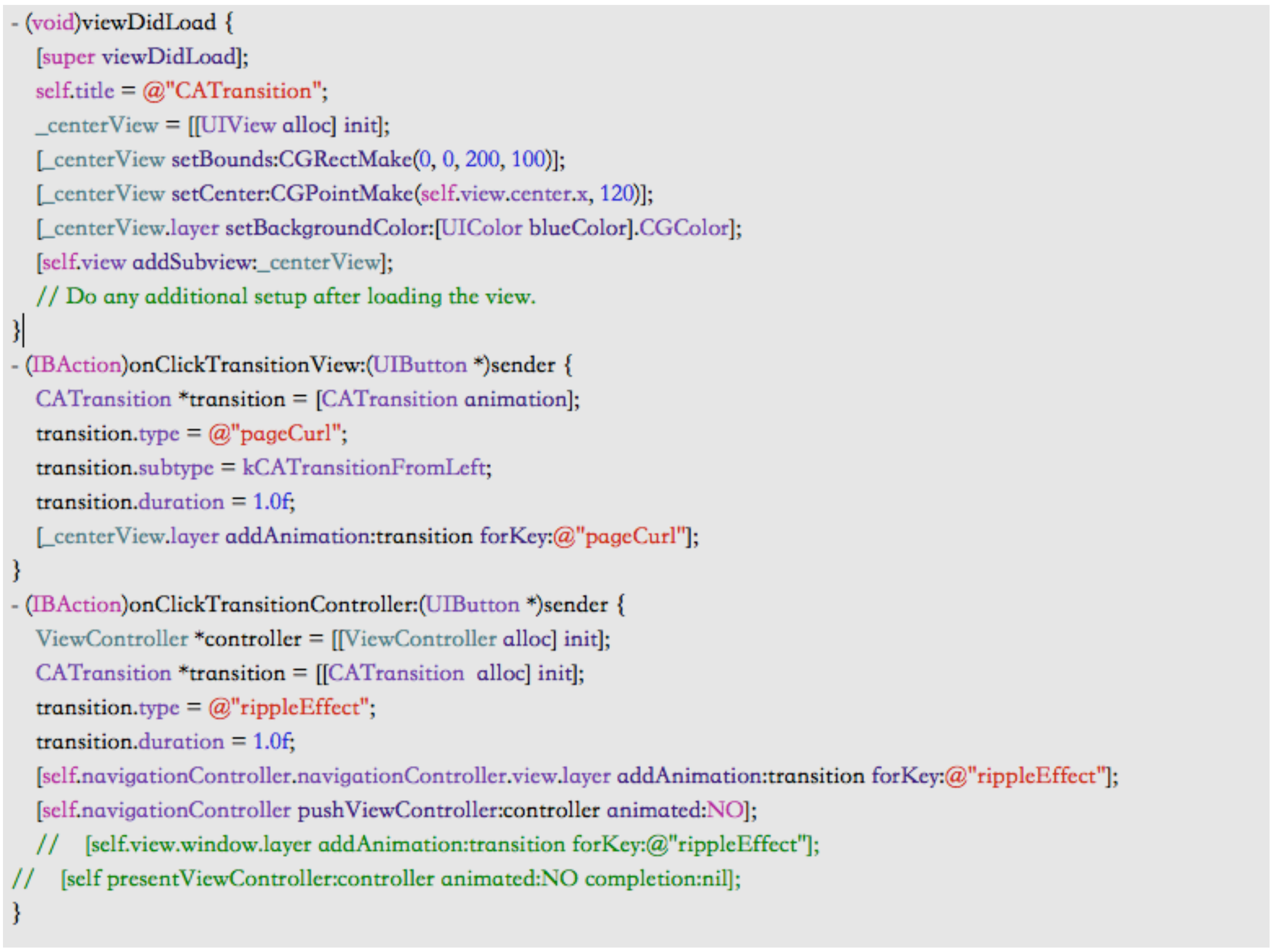


```
- (CABasicAnimation *)rotationAnimation{
   CABasicAnimation *basicAnimation = [CABasicAnimation animationWithKeyPath:@"transform.rotation.z"];
   [basicAnimation setAutoreverses:YES];
   [basicAnimation setRemovedOnCompletion:NO];
   [basicAnimation setToValue:@(M_PI_2 * 3.5)];
   [basicAnimation setValue:@(M_PI_2 * 3.5) forKey:@"lastRotation"];
   return basicAnimation;
}

- (CAKeyframeAnimation *)translationAnimation{
   CAKeyframeAnimation *animation = [CAKeyframeAnimation animationWithKeyPath:@"position"];
   UIBezierPath *path = [UIBezierPath bezierPath];
   [path moveToPoint: _layer.position];
   [path addCurveToPoint:CGPointMake(_layer.position.x, _layer.position.y + 300) controlPoint1:CGPointMake(_layer.position.x + 50, _layer.position.y + 100)
      controlPoint2:CGPointMake(_layer.position.x  - 50, _layer.position.y + 300 - 100)];
   animation.path = path.CGPath;

   [animation setValue:[NSValue valueWithCGPoint:CGPointMake(_layer.position.x, _layer.position.y + 300)] forKey:@"lastPoint"];
   return animation;
}
```



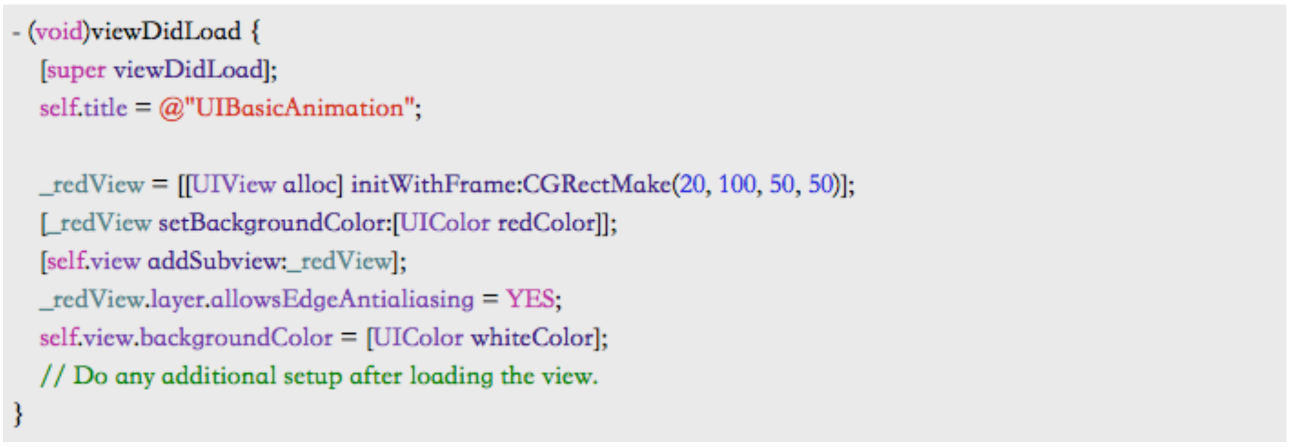
```
- (void)groupAnimation{
   CAAnimationGroup *animationGroup = [CAAnimationGroup animation];
   CABasicAnimation *basicAnimation = [self rotationAnimation];
   CAKeyframeAnimation *keyAnimation = [self translationAnimation];
   animationGroup.animations = @[basicAnimation, keyAnimation];
   animationGroup.delegate = self;
   animationGroup.duration = 8;
   animationGroup.removedOnCompletion = NO;
   animationGroup.fillMode = kCAFillModeForwards;
   animationGroup.beginTime = CACurrentMediaTime() + 5;
   [_layer addAnimation:animationGroup forKey:nil];
}

- (void)animationDidStop:(CAAnimation *)anim finished:(BOOL)flag{
   CAAnimationGroup *group = (CAAnimationGroup *)anim;
   CABasicAnimation *basicAnimation = (CABasicAnimation *)group.animations[0];
   CAKeyframeAnimation *keyfFrameAnimation = (CAKeyframeAnimation *)group.animations[1];

   CGFloat toValue = [[basicAnimation valueForKey:@"lastRotation"] floatValue];
   CGPoint endPoint = [[keyfFrameAnimation valueForKey:@"lastPoint"]CGPointValue];

   [CATransaction begin];
   [CATransaction setDisableActions:YES];

   _layer.position = endPoint;
   _layer.transform = CATransform3DMakeRotation(toValue, 0, 0, 1);

   [CATransaction commit];
}
```

4) CATransition



```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.title = @"CATransition";
    _centerView = [[UIView alloc] init];
    [_centerView setBounds:CGRectMake(0, 0, 200, 100)];
    [_centerView setCenter:CGPointMake(self.view.center.x, 120)];
    [_centerView.layer setBackgroundColor:[UIColor blueColor].CGColor];
    [self.view addSubview:_centerView];
    // Do any additional setup after loading the view.
}
- (IBAction)onClickTransitionView:(UIButton *)sender {
    CATransition *transition = [CATransition animation];
    transition.type = @"pageCurl";
    transition.subtype = kCATransitionFromLeft;
    transition.duration = 1.0f;
    [_centerView.layer addAnimation:transition forKey:@"pageCurl"];
}
- (IBAction)onClickTransitionController:(UIButton *)sender {
    ViewController *controller = [[ViewController alloc] init];
    CATransition *transition = [[CATransition  alloc] init];
    transition.type = @"rippleEffect";
    transition.duration = 1.0f;
    [self.navigationController.navigationController.view.layer addAnimation:transition forKey:@"rippleEffect"];
    [self.navigationController pushViewController:controller animated:NO];
    //    [self.view.window.layer addAnimation:transition forKey:@"rippleEffect"];
//    [self presentViewController:controller animated:NO completion:nil];
}
```

5) UIBasicAnimation



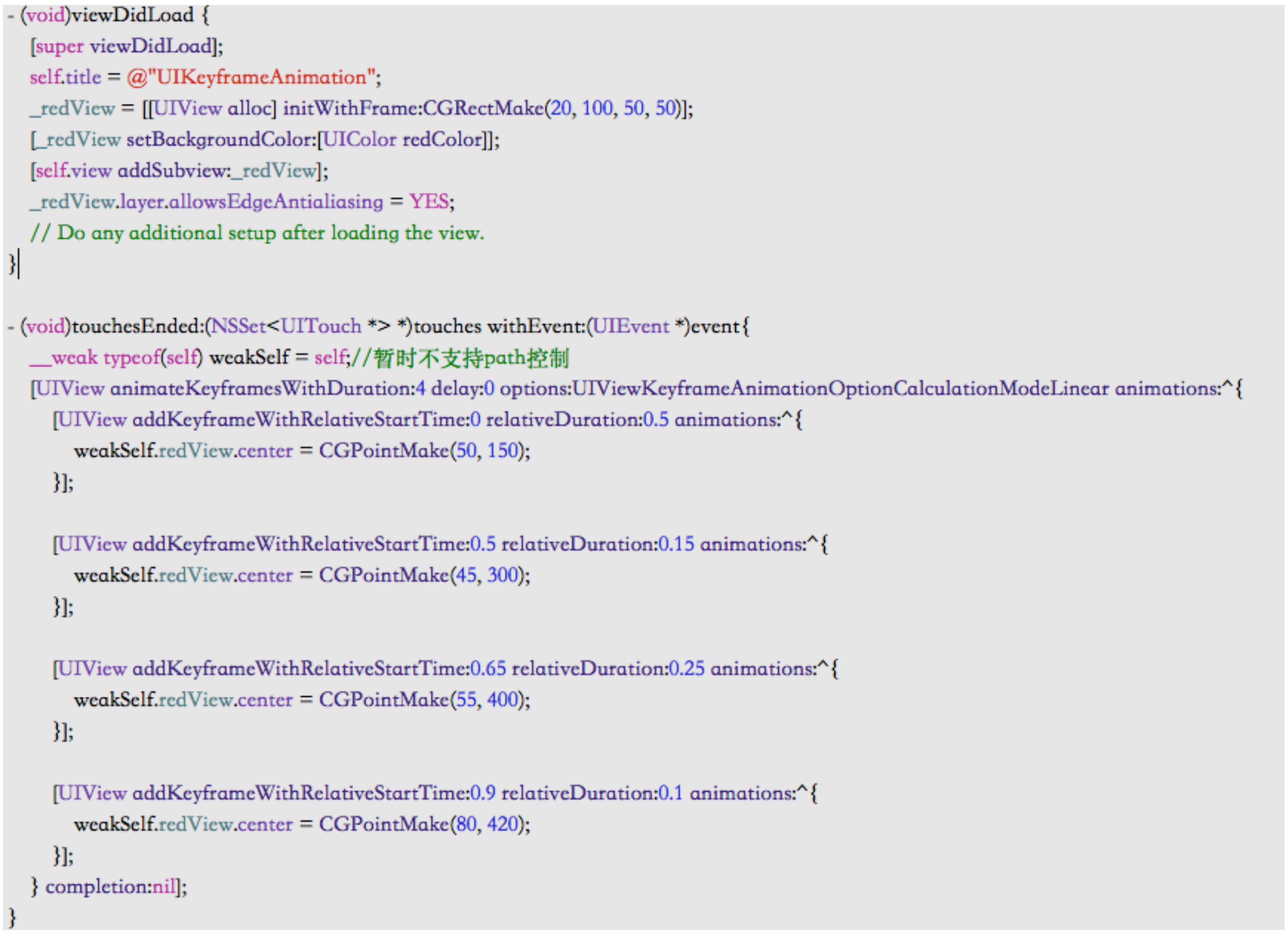
```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.title = @"UIBasicAnimation";

    _redView = [[UIView alloc] initWithFrame:CGRectMake(20, 100, 50, 50)];
    [_redView setBackgroundColor:[UIColor redColor]];
    [self.view addSubview:_redView];
    _redView.layer.allowsEdgeAntialiasing = YES;
    self.view.backgroundColor = [UIColor whiteColor];
    // Do any additional setup after loading the view.
}
```

```
- (void)touchesEnded:(NSSet<UITouch *> *)touches withEvent:(UIEvent *)event{
    UITouch *touch = [touches anyObject];
    CGPoint point = [touch locationInView:self.view];
    __weak typeof(self) weakSelf = self;
    [UIView animateWithDuration:5 delay:0 options:UIViewAnimationOptionCurveEaseOut animations:^{
        weakSelf.redView.center = point;
    } completion:nil];


    [UIView beginAnimations:@"changeBackground" context:nil];
    [UIView setAnimationDuration:5];
    [UIView setAnimationDelay:0];
    [UIView setAnimationRepeatAutoreverses:NO];
    [UIView setAnimationRepeatCount:0];
    [UIView setAnimationCurve:UIViewAnimationCurveLinear];
    int a = arc4random() % 2;
    switch (a) {
        case 0:
            [_redView setBackgroundColor:[UIColor orangeColor]];
            break;
        case 1:
            [_redView setBackgroundColor:[UIColor redColor]];
            break;
        default:
            break;
    }
    [UIView commitAnimations];
}
```

6) UISpringAnimation

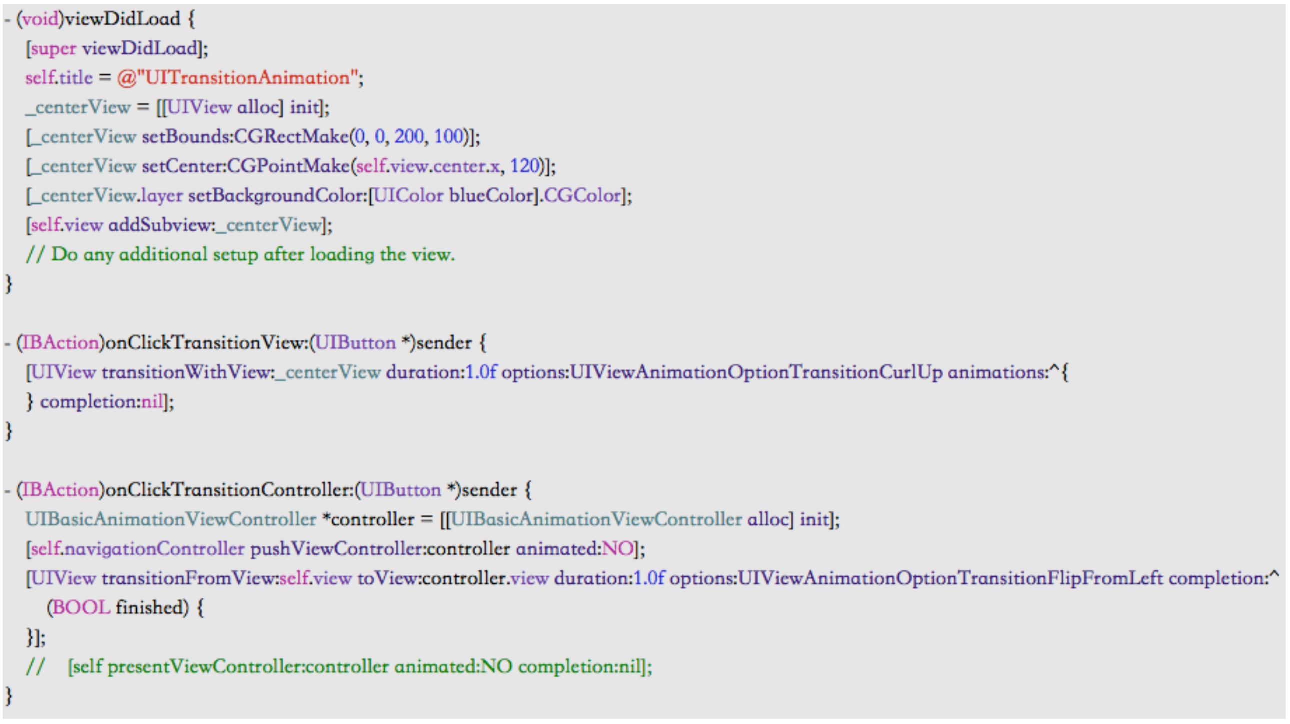
```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.title = @"UISpringAnimation";
    _redView = [[UIView alloc] initWithFrame:CGRectMake(20, 100, 50, 50)];
    [_redView setBackgroundColor:[UIColor redColor]];
    [self.view addSubview:_redView];
    _redView.layer.allowsEdgeAntialiasing = YES;
    // Do any additional setup after loading the view.
}

- (void)touchesEnded:(NSSet<UITouch *> *)touches withEvent:(UIEvent *)event{
    UITouch *touch = [touches anyObject];
    CGPoint point = [touch locationInView:self.view];
    __weak typeof(self) weakSelf = self;
    [UIView animateWithDuration:5 delay:0 usingSpringWithDamping:0.1 initialSpringVelocity:1.0f options:UIViewAnimationOptionCurveLinear
        animations:^{
        weakSelf.redView.center = point;
        int a = arc4random() % 2;
        switch (a) {//背景色也会弹性变化
            case 0:
                [_redView setBackgroundColor:[UIColor orangeColor]];
                break;
            case 1:
                [_redView setBackgroundColor:[UIColor redColor]];
                break;
            default:
                break;
        }

    } completion:nil];
}
```

7) UIKeyframeAnimation



```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.title = @"UIKeyframeAnimation";
    _redView = [[UIView alloc] initWithFrame:CGRectMake(20, 100, 50, 50)];
    [_redView setBackgroundColor:[UIColor redColor]];
    [self.view addSubview:_redView];
    _redView.layer.allowsEdgeAntialiasing = YES;
    // Do any additional setup after loading the view.
}

- (void)touchesEnded:(NSSet<UITouch *> *)touches withEvent:(UIEvent *)event{
    __weak typeof(self) weakSelf = self;//暂时不支持path控制
    [UIView animateKeyframesWithDuration:4 delay:0 options:UIViewKeyframeAnimationOptionCalculationModeLinear animations:^{
        [UIView addKeyframeWithRelativeStartTime:0 relativeDuration:0.5 animations:^{
            weakSelf.redView.center = CGPointMake(50, 150);
        }];

        [UIView addKeyframeWithRelativeStartTime:0.5 relativeDuration:0.15 animations:^{
            weakSelf.redView.center = CGPointMake(45, 300);
        }];

        [UIView addKeyframeWithRelativeStartTime:0.65 relativeDuration:0.25 animations:^{
            weakSelf.redView.center = CGPointMake(55, 400);
        }];

        [UIView addKeyframeWithRelativeStartTime:0.9 relativeDuration:0.1 animations:^{
            weakSelf.redView.center = CGPointMake(80, 420);
        }];
    } completion:nil];
}
```

8) UITransition



```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.title = @"UITransitionAnimation";
    _centerView = [[UIView alloc] init];
    [_centerView setBounds:CGRectMake(0, 0, 200, 100)];
    [_centerView setCenter:CGPointMake(self.view.center.x, 120)];
    [_centerView.layer setBackgroundColor:[UIColor blueColor].CGColor];
    [self.view addSubview:_centerView];
    // Do any additional setup after loading the view.
}

- (IBAction)onClickTransitionView:(UIButton *)sender {
    [UIView transitionWithView:_centerView duration:1.0f options:UIViewAnimationOptionTransitionCurlUp animations:^{
    } completion:nil];
}

- (IBAction)onClickTransitionController:(UIButton *)sender {
    UIBasicAnimationViewController *controller = [[UIBasicAnimationViewController alloc] init];
    [self.navigationController pushViewController:controller animated:NO];
    [UIView transitionFromView:self.view toView:controller.view duration:1.0f options:UIViewAnimationOptionTransitionFlipFromLeft completion:^
        (BOOL finished) {
    }];
    //    [self presentViewController:controller animated:NO completion:nil];
}
```