# 图层复制思想

1. 概述：

使用核心动画库时，我们难免要和图层打交道，举个例子，实现一个音乐播放的 loading 动画，某一帧如下：

 现在，我们要让这四根方柱循环交错伸缩，实现音乐正在播放的动画效果。

当然，帧动画完全可以做到。但既然学习了核心动画库，那么我们当然必需首先想到的实现方式就是图层动画。Ok，常规的想法，可能是使用四个柱形图层，分别添加图层动画，这没问题。那么，问题在哪？假如现在要求 10 个方柱呢？

这时候，就得来聊聊核心动画库一个非常重要的角色了——CAReplicatorLayer

2. CAReplicatorLayer 介绍

(1) 概述：

CAReplicatorLayer 可以将自己的子图层复制指定的次数,并且复制过程保持被复制图层的各种基础属性以及动画

(2) 基本属性

1) instanceCount

拷贝图层的次数,包括其所有的子图层,默认值是 1,也就是没有任何子图层被复制

2) instanceDelay

在短时间内的复制延时,一般用在动画上(支持动画的延时)

3) instanceTransform

复制图层在被创建时产生的和上一个复制图层的位移(位移的锚点时 CAReplicatorlayer 的中心点)

4) preservesDepth

如果设置为 YES,图层将保持于 CATransformLayer 类似的性质和相同的限制

5) instanceColor

设置多个复制图层的颜色,默认位白色

6) instanceRedOffset

设置每个复制图层相对上一个复制图层的红色偏移量

7) instanceGreenOffset

设置每个复制图层相对上一个复制图层的绿色偏移

8) instanceBlueOffset

设置每个复制图层相对上一个复制图层的蓝色偏移量

9) instanceAlphaOffset

设置每个复制图层相对上一个复制图层的透明度偏移量

3. CAReplicatorLayer 应用

(1) 对 CAReplicatorLayer 有了一定的了解了后，我们来看一个示例：

如图所示，上图是很常见的加载动画，整体感觉就是旋转(其实是单个的缩放使得整体产生的效果)

下图也是通过缩放产生的整体平移效果，具体可以参照项目演示，下面是具体实现，参照着项目演示来理解下面代码，
CAReplicatorLayer 的用法，大概也就熟悉了

(2) 项目代码：

1) 环形进度动画

```
#pragma mark - 控件get方法

- (CALayer *)animationView{
    if(!_animationView){
        _animationView = [CALayer layer];
        _animationView.backgroundColor = [UIColor grayColor].CGColor;
        _animationView.masksToBounds = YES;
        _animationView.allowsEdgeAntialiasing = YES;
        _animationView.transform = CATransform3DMakeScale(0., 0., 0.);
    }
    return _animationView;
}
```

```
- (CAReplicatorLayer *)replicatorLayer{
    if(!_replicatorLayer){
        _replicatorLayer = [CAReplicatorLayer layer];
        _replicatorLayer.instanceCount = 10;
        _replicatorLayer.instanceDelay = .08f;
        _replicatorLayer.instanceAlphaOffset = -0.06f;
        _replicatorLayer.instanceTransform = CATransform3DMakeRotation((2 * M_PI / 10), 0, 0, 1);
    }
    return _replicatorLayer;
}


- (void)startAnimation{
    [self.layer setSublayers:nil];

    [self.layer addSublayer:self.replicatorLayer];
    self.replicatorLayer.bounds = self.layer.bounds;
    self.replicatorLayer.position = CGPointMake(CGRectGetMidX(self.bounds), CGRectGetMidY(self.bounds));

    [self.replicatorLayer addSublayer:self.animationView];
    self.animationView.bounds = CGRectMake(0, 0, self.bounds.size.width / 10, self.bounds.size.width / 10);
    self.animationView.cornerRadius = self.animationView.bounds.size.width / 2;
    self.animationView.position = CGPointMake(CGRectGetMidX(self.layer.bounds), CGRectGetMaxY(self.layer.bounds) - 10);

    CABasicAnimation *animation = [CABasicAnimation animationWithKeyPath:@"transform.scale"];
    animation.duration = 0.8f;
    animation.fromValue = @1.1;
    animation.toValue = @0.6;
    animation.repeatCount = CGFLOAT_MAX;
    [self.animationView addAnimation:animation forKey:@"animation"];
}
```

## 2）直线型进度动画

```
#pragma mark - 控件get方法

- (CALayer *)animationView{
    if(!_animationView){
        _animationView = [CALayer layer];
        _animationView.backgroundColor = [UIColor colorWithRed:201/255.0 green:8/255.0 blue:19/255.0 alpha:1].CGColor;
        _animationView.masksToBounds = YES;
        _animationView.allowsEdgeAntialiasing = YES;
        _animationView.transform = CATransform3DMakeScale(0.3, 0.3, 0.3);
    }
    return _animationView;
}

- (CAReplicatorLayer *)replicatorLayer{
    if(!_replicatorLayer){
        _replicatorLayer = [CAReplicatorLayer layer];
        _replicatorLayer.instanceCount = 3;
        _replicatorLayer.instanceDelay = .2f;
        _replicatorLayer.instanceAlphaOffset = .3f;
    }
    return _replicatorLayer;
}


- (void)startAnimation{
    [self.layer setSublayers:nil];

    [self.layer addSublayer:self.replicatorLayer];
    self.replicatorLayer.bounds = self.layer.bounds;
    self.replicatorLayer.position = CGPointMake(CGRectGetMidX(self.bounds), CGRectGetMidY(self.bounds));

    [self.replicatorLayer addSublayer:self.animationView];
    self.animationView.bounds = CGRectMake(0, 0, self.bounds.size.width / 6, self.bounds.size.width / 6);
    self.animationView.cornerRadius = self.animationView.bounds.size.width / 2;
    self.animationView.position = CGPointMake(CGRectGetMidX(self.bounds) - self.bounds.size.width / 4, CGRectGetMidY(self.layer.bounds));
    self.replicatorLayer.instanceTransform = CATransform3DMakeTranslation(self.bounds.size.width / 4, 0, 0);

    CABasicAnimation *animation = [CABasicAnimation animationWithKeyPath:@"transform.scale"];
    animation.duration = 0.6f;
    animation.fromValue = @0.3;
    animation.toValue = @1.;
    animation.autoreverses = YES;
    animation.timingFunction = [CAMediaTimingFunction functionWithName:kCAMediaTimingFunctionEaseIn];
    animation.repeatCount = CGFLOAT_MAX;
    [self.animationView addAnimation:animation forKey:@"animation"];
}
```