iOS (http://lib.csdn.net/base/ios)

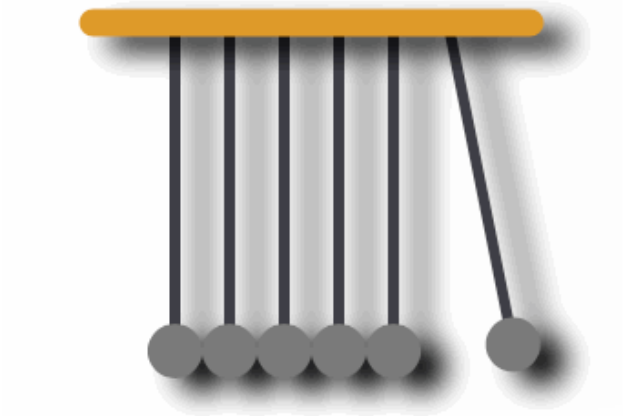
iOS (http://lib.csdn.net/base/ios) - 企业级开发 (http://lib.csdn.net/ios/node/686) - 按需加载 (http://lib.csdn.net/ios/knowledge/1495)

iOS开发 ----- 加载动画之牛顿摆的实现

作者：xiao333ma (http://my.csdn.net/xiao333ma)

牛顿摆动画

自己看动画有一段时间了,但是还是不是很能理解其中的一些属性方法之类的东西,琢磨了一下午写了一个牛顿摆的动画,这里记录一下,一遍以后查看先上图



先说下思路

说下牛顿摆的大致运动过程

根据牛顿摆的原理,中间是不动得,只有两边在动
两边运动是一个以这条线的上方位原点,长为半径,然后做半圆运动
运动模式是先快后慢
当左边的摆下来的时候,右边的开始向上摆动,右边的摆下来的时候,左边的开始向上摆动,一直循环下去

这样的话,我们用CAShapeLayer来进行画图,然后用CAAnimation来实现上述的运动过程

下边说流程

整体用CAShaepLayer + CAAnimation实现上述效果
图形全是画出来
划中间的四条线
划下边的四个圆
划左边的线
划左边的圆
划右边的线

划右边的圆
最后划上边的横线
加阴影
做动画

一步一来

1. 画线

1.1 全局变量

做成全局变量,方便后边使用

由于上边的大横线是不用动得,所以可以位局部变量

还有一个问题就是,如果直接用[self.layer subLayers]来取值的话,会取到多一些其他的layer,之前自己添加的layer是subLyaer的第一个,现在貌似是第三个,默认多了两个,这个具体原因不详,自己创建一个数组,来存放所有用到的layer,动画结束后,移除他们

动画结束后,需要回调一个block来做一些事情,下边会说到

```
    //自身的宽高
    CGFloat _height;
    CGFloat _width;

    //左边的竖线,左边的圆,左边的旋转路径
    CAShapeLayer * _leftLine;
    CAShapeLayer * _leftCircle;
    CGMutablePathRef  _leftPath;

    //右边的竖线,右边的圆,右边的旋转路径
    CAShapeLayer * _rightLine;
    CAShapeLayer * _rightCircle;
    CGMutablePathRef  _rightPath;

    //左边的动画
    CABasicAnimation * _leftBaseAnimation;
    CABasicAnimation * _rightBaseAnimation;

    //右边的动画
    CAKeyframeAnimation * _leftKeyframeAnimation;
    CAKeyframeAnimation * _rightKeyframeAnimation;

    //动画结束调用的block
    void(^animationFinishBlock)(CAAnimation * animation);

    //存放所有图层的数组
    NSMutableArray  * _array;
```

1.2 初始化

初始化宽,高,数组

```
- (instancetype)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {

        //初始化
        _height = self.frame.size.height;
        _width = self.frame.size.width;

        _array = [[NSMutableArray alloc]init];


    }
    return self;
}
```

1.3 创建中间的四个横线和圆

因为在初始化的时候设置的宽高都是100,所以,循环创建中间者四个视图,使他们的位置依次排列,然后放在中间

然后添加到self.layer上

同样也添加到数组中

至于怎么算,额..数学不太好,自己琢磨琢磨把

```
-(void)creatLayer
{
    for (int i = 0; i < 6; i++) {

        if (i >=1 && i<=4 )
        {
            CAShapeLayer * layer = [self creatFourLineX:i*10+25 andY:10];
            CAShapeLayer * layer2 = [self creatRoundLayerX:i*10+25 andY:70];
            [self.layer addSublayer:layer];
            [self.layer addSublayer:layer2];
            [_array addObject:layer];
            [_array addObject:layer2];
        }


    }
}
```

1.4 创建四个线

```
1
2  -(CAShapeLayer *)creatFourLineX:(CGFloat)x andY:(CGFloat)y
3  {
4      CAShapeLayer * layer = [CAShapeLayer layer];
5      //首先,根据传递过来的参数,布局,然后设置宽位2 高为70
6      layer.frame = CGRectMake(x, y, 2, 70);
7      //创建路径
8      CGMutablePathRef path = CGPathCreateMutable();
9      //移动到 (0,0)的位置
10     CGPathMoveToPoint(path, nil, 0, 0);
11     //然后话一条60长度的线
12     CGPathAddLineToPoint(path, nil, 0, 60);
13     //设置layer的路径位划的路径
14     layer.path = path;
15     //填充颜色,这里的颜色要转化位CGColor
16     layer.strokeColor = [UIColor colorWithRed:0.188  green:0.188  blue:0.216 alpha
17     //设置线宽
18     layer.lineWidth = 2;
19     //设置lineCape(不知道怎么说了)就是那个线的端点的样式,这里是圆形,
20     layer.lineCap = kCALineCapRound;
21     //然后设置下阴影
22     [self setShadow:layer];
23     //返回layer
24     return layer;
25 }
```

1.5 创建四个圆

```
-(CAShapeLayer *)creatRoundLayerX:(CGFloat)x andY:(CGFloat)y
{
    CAShapeLayer * layer = [CAShapeLayer layer];
    //设置位置,我们的圆是半径位5的圆,所以宽度是10就够了
    layer.frame = CGRectMake(x, y, 10, 10);

    //然后绘制路径
    CGMutablePathRef path = CGPathCreateMutable();
    //参数依次是
    //1. 路径
    //2. 变换
    //3. 圆心的x
    //4. 圆心的y
    //5. 起始角度
    //6. 结束角度
    //7. 是否顺时针
    //关于这个,大家自己体会下就知道,画图嘛,画出来什么样子看看是最清楚的
    CGPathAddArc(path, nil, 0, 0, 5, 0, M_PI*2, YES);
    //然后设置路径
    layer.path = path;

    //然后填充颜色,这里和上边的`layer.strokeColor`不一样,上边的`layer.strokeColor`这是是这
    //而这个`layer.fillColor`则是填充的颜色

    layer.fillColor = [UIColor colorWithRed:0.404  green:0.404  blue:0.404 alpha:1
    //然后设置下阴影
    [self setShadow:layer];

    return layer;
}
```

1.6 画左边的线

这里大致说下anchorPoint 这个是锚点,所谓锚点就是类似你把一张纸,用图钉固定在了墙上,当不太紧的时候,纸是可以旋转的,旋转的中心就是锚点

锚点和position都可以改变这个layer的位置,具体细节大家可以去这里(https://developer.apple.com/library/prerelease/ios/documentation/Cocoa/Conceptual/CoreAnimation_CH2-SW3)查看

由于我们在动画的时候,会对左边的线进行旋转,而且是围绕者顶部开始旋转的,所以我们把锚点设为(0,0),这样的话,我们旋转的时候,就以(0,0)为中心点,进行旋转

根据位置不同,我们这是了position的anchorPoint,然后和上边一样,画一条60长的线,同样设置一下相关的属性

```
-(void)creatLeftLine
{

    _leftLine = [CAShapeLayer layer];
    _leftLine.frame = CGRectMake(25, 10, 100, 100);

    _leftLine.position = CGPointMake(25, 10);
    _leftLine.anchorPoint = CGPointMake(0,0);
    CGMutablePathRef path = CGPathCreateMutable();
    CGPathMoveToPoint(path, nil, 0, 0);
    CGPathAddLineToPoint(path, nil, 0, 60);
    _leftLine.strokeColor = [UIColor colorWithRed:0.188  green:0.188  blue:0.216 a
    _leftLine.lineWidth = 2;
    _leftLine.lineCap = kCALineCapRound;
    _leftLine.path = path;
    [self setShadow:_leftLine];
    [self.layer addSublayer:_leftLine];
    [_array addObject:_leftLine];


}
```

1.7 画左边的圆

和上边类似,我们也要画一个圆,这里我们设置一下frame和position,使我们的圆的中心,就在线的下边,这样的话,我们在做动画的时候,从视觉效果来说,是一起的

→_→ 其实是两个

```
-(void)creatLeftRound
{


    _leftCircle = [CAShapeLayer layer];
    _leftCircle.position = CGPointMake(25, 70);
    _leftCircle.frame = CGRectMake(20, 65, 10, 10);
    CGMutablePathRef path = CGPathCreateMutable();

    CGPathAddArc(path, nil, 5, 5, 5, 0, M_PI*2, YES);

    _leftCircle.path = path;
    _leftCircle.fillColor = [UIColor colorWithRed:0.404  green:0.404  blue:0.404 a
    [self setShadow:_leftCircle];


    [self.layer addSublayer:_leftCircle];
    [_array addObject:_leftCircle];


}
```

1.8 画右边的线

同样和上边类似,要围绕上边进行旋转,所以,要设置下锚点,然后和相关属性

锚点很重要,锚点很重要,锚点很重要,重要的事要说三遍,说三遍,三遍,遍

```
-(void)creatRightLine
{

    _rightLine = [CAShapeLayer layer];
    _rightLine.frame = CGRectMake(75, 10, 100, 100);

    _rightLine.position = CGPointMake(75, 10);
    _rightLine.anchorPoint = CGPointMake(0,0);
    CGMutablePathRef path = CGPathCreateMutable();
    CGPathMoveToPoint(path, nil, 0, 0);
    CGPathAddLineToPoint(path, nil, 0, 60);
    _rightLine.strokeColor = [UIColor colorWithRed:0.188  green:0.188  blue:0.216
    _rightLine.lineWidth = 2;
    _rightLine.lineCap = kCALineCapRound;
    _rightLine.path = path;
    [self setShadow:_rightLine];
    [self.layer addSublayer:_rightLine];
    [_array addObject:_rightLine];


}
```

1.9 画右边的圆

和上边是一样的,要是再多的话,我就封装啦,别逼我发自拍

```
-(void)creatRightRound
{

    _rightCircle = [CAShapeLayer layer];
    _rightCircle.position = CGPointMake(75, 70);
    _rightCircle.frame = CGRectMake(70, 65, 10, 10);

    CGMutablePathRef path = CGPathCreateMutable();

    CGPathAddArc(path, nil, 5, 5, 5, 0, M_PI*2, YES);

    _rightCircle.path = path;
    _rightCircle.fillColor = [UIColor colorWithRed:0.404  green:0.404  blue:0.404

    [self setShadow:_rightCircle];
    [self.layer addSublayer:_rightCircle];
    [_array addObject:_rightCircle];


}
```

1.10 设置阴影

调用了这么多次,终于出现了,设置下圆角,阴影的颜色,偏移量和 ... 这个不知道怎么说,自己体会一下吧

```
-(void)setShadow:(CALayer *)layer
{
    layer.cornerRadius = 5;
    layer.shadowColor = [UIColor blackColor].CGColor;
    layer.shadowOffset = CGSizeMake(5, 3);
    layer.shadowOpacity = 3.0f;
}

```

1.11 画最上边的线

类似,类似,类似 →_→

```
-(void)reatTopLineLayer
{
    CAShapeLayer * topLine = [CAShapeLayer layer];


    CGMutablePathRef path = CGPathCreateMutable();

    CGPathMoveToPoint(path, nil, 10, 10);
    CGPathAddLineToPoint(path, nil, 90, 10);
    topLine.path = path;
    topLine.strokeColor = [UIColor colorWithRed:0.831  green:0.529  blue:0.086 alp
    topLine.lineWidth = 5;
    topLine.lineCap = kCALineCapRound;
    [self setShadow:topLine];
    [self.layer addSublayer:topLine];
    [_array addObject:topLine];

}
```

!!!!!终于,终于画完了,封装,封装,不然会累死

2 开始动画

2.1 左边的动画

终于开始动画了,先来大致说一下,CAAnimation中,有CABasicAnimation,有CAKeyframeAnimation,还有CAGroupAnimation

一般这几个够用了,他们都有keyPath属性

当是在看的时候,发现这是个字符串对象,尼玛,字符串,我知道这是个毛啊,网上扒了几篇博客,也没发现什么规律

后来,后来,终于得到了一本秘籍,可以拯救世界的秘籍,然后我就基本上知道了这货应该怎么填

其实在CAAnimation,几乎所有的属性都是可以动画的,位置,颜色,等等,都可以改变,想怎么动,动什么属性,就写什么属性

比如这里的左边的线,我们要旋转,Z 轴的旋转,那就写呗transform.rotation.z,嗯,就是这货

然后就是持续时间,这里是0.4s

旋转的角度呢,这里有fromeValue和toValue,开始,结束,想怎么写,怎么写

这里从0转到到,π/8的位置, π是180°,π/2是90° π/4是45°,π/8是 22.5°,嗯,体育老师教的数学看来还够用

_leftBaseAnimation.timingFunction 这个货是设置运动的模式的,是先快后慢,还是先慢后快,还是一开始慢后来加速,然后在减速…这个曲线可以自定义

这里用系统的,因为是向上摆动,所以一开始比较快,然后减速到0

然后_leftBaseAnimation.autoreverses这个是设置是否完成动画后反向在执行一遍,我们还要回来啊,妥妥的YES

_leftBaseAnimation.fillMod 这个无所谓了,我们最终还要回到起点

然后就是把这个动画添加到_leftLine上 ,后边的key可以不设置,不影响

```
-(void)leftAnimation
{

    //leftLine

    _leftBaseAnimation = [CABasicAnimation animation];
    _leftBaseAnimation.keyPath = @"transform.rotation.z";
    _leftBaseAnimation.duration = 0.4;
    _leftBaseAnimation.fromValue = [NSNumber numberWithFloat:0];
    _leftBaseAnimation.toValue = [NSNumber numberWithFloat:M_PI_4/2];
    _leftBaseAnimation.timingFunction =[CAMediaTimingFunction functionWithName: kC
    _leftBaseAnimation.autoreverses = YES;
    _leftBaseAnimation.delegate = self;
    _leftBaseAnimation.fillMode = kCAFillModeForwards;
    [_leftLine addAnimation:_leftBaseAnimation forKey:@"leftBaseAnimation"];




    //leftCircle

    //因为这里要使圆球,按照一个曲线与运动, CAKeyframeAnimation正好满足我们的需求
    //先创建一个路径,画一个22.5°的圆弧
    _leftPath = CGPathCreateMutable();
    CGPathAddArc(_leftPath, nil, 25, 10, 60, M_PI_2,M_PI_2+M_PI_4/2, NO);
    _leftKeyframeAnimation = [CAKeyframeAnimation animation];
    //自己本身要运动,所以肯定是position了,还记得上边设置的时候,position的位置要设为竖线的一端
    _leftKeyframeAnimation.keyPath = @"position";
    //计算模式,可以不写,对我们的动画没有影响
    _leftKeyframeAnimation.calculationMode = kCAAnimationCubic;
    //设置动画的路径,然后小球就会跟着动
    _leftKeyframeAnimation.path = _leftPath;
    //持续时间是0.4s
    _leftKeyframeAnimation.duration = 0.4f;
    //运动模式,先快后慢
    _leftKeyframeAnimation.timingFunction = [CAMediaTimingFunction functionWithNam
    //结束之后,在反过来继续运行
    _leftKeyframeAnimation.autoreverses = YES;
    //基本没什么卵用
    _leftKeyframeAnimation.fillMode = kCAFillModeForwards;
    //设置代理,监听动画结束
    _leftKeyframeAnimation.delegate = self;
    //这里设置一下value方便动画结束之后可以检测到,是这个动画
    [_leftKeyframeAnimation setValue:@"left" forKey:@"left"];
    //添加动画
    [_leftCircle addAnimation:_leftKeyframeAnimation forKey:@"leftKeyframeAnimatio



}
```

2.2 右边的动画

基本上是一样的,就是旋转的角度不一样,一个向左,一个向右,参照上边的注释即可

```
-(void)rightAnimation
{

    //RightLine

    _rightBaseAnimation = [CABasicAnimation animation];
    _rightBaseAnimation.keyPath = @"transform.rotation.z";
    _rightBaseAnimation.duration = 0.4;
    _rightBaseAnimation.fromValue = [NSNumber numberWithFloat:0];
    _rightBaseAnimation.toValue = [NSNumber numberWithFloat:-M_PI_4/2];
    _rightBaseAnimation.timingFunction =[CAMediaTimingFunction functionWithName: k
    _rightBaseAnimation.autoreverses = YES;
    _rightBaseAnimation.fillMode = kCAFillModeForwards;
    _rightBaseAnimation.delegate = self;
    [_rightLine addAnimation:_rightBaseAnimation forKey:@"rightBaseAnimation"];









    //RightCircle

    _rightPath = CGPathCreateMutable();
    CGPathAddArc(_rightPath, nil, 75, 10, 60, M_PI_2,M_PI_2-M_PI_4/2, YES);
    _rightKeyframeAnimation = [CAKeyframeAnimation animation];
    _rightKeyframeAnimation.keyPath = @"position";
    _rightKeyframeAnimation.calculationMode = kCAAnimationCubic;
    _rightKeyframeAnimation.path = _rightPath;
    _rightKeyframeAnimation.duration = 0.4f;
    _rightKeyframeAnimation.timingFunction = [CAMediaTimingFunction functionWithNa
    _rightKeyframeAnimation.autoreverses = YES;
    _rightKeyframeAnimation.fillMode = kCAFillModeForwards;
    _rightKeyframeAnimation.delegate = self;
    [_rightKeyframeAnimation setValue:@"right" forKey:@"right"];
    [_rightCircle addAnimation:_rightKeyframeAnimation forKey:@"rightKeyframeAnima
}

```

3. 控制动画的运行

> 大致流程是这样的

我们应该这样处理动画
先开始左边的动画
左边的动画完成之后,也就是摆上去之后,在摆下来
开始右边的动画
右边的动画摆上去,在摆下来之后
在开始左边的动画

3.1 现在在.h文件中写两个方法

> 一个开始动画,一个结束动画
>
> 名字写的不好,随便吧

```
#import <UIKit/UIKit.h>

@interface LoaddingAnimation : UIView


-(void)showAnimation;

-(void)hideAnimation;

@end
```

3.2 开始动画

我们在开始动画的方法中,创建所有必须得layer,然后开始做动画

还记得我们一开始的时候,定义的那个block么,现在就要排上用场了

我们会这么做,一开始就是一个空白的视图,我们调用showAnimation的时候,创建,然后开始动画

结束的时候,我们把所有layer全部移除

```
1
2  -(void)showAnimation
3  {
4
5
6      [self creatLeftLine];
7      [self creatLeftRound];
8      [self creatLayer];
9
10     [self creatRightLine];
11     [self creatRightRound];
12     [self reatTopLineLayer];
13
14     [self leftAnimation];
15
16
17     //为了防止Block中循环引用,我们要这么处理
18
19     //    [_rightKeyframeAnimation setValue:@"right" forKey:@"right"];
20      //    [_leftKeyframeAnimation setValue:@"left" forKey:@"left"];
21      //还记得我们上边这两句么,这样的话,我们就可以监听到到底是那个动画完成了
22      //因为我们是在动画结束之后调用的,所以按照上边的逻辑,我们就在检测到左边完成的时候
23      //让右边去动画
24      //同样,右边完成之后,让左边去动画
25     __weak LoaddingAnimation * load = self;
26
27     animationFinishBlock = ^(CAAnimation * animation){
28
29         if ([[animation valueForKey:@"left"] isEqualToString:@"left"]) {
30                 //检测到左边结束后,开始右边的动画
31             [load rightAnimation];
32         }else if([[animation valueForKey:@"right"] isEqualToString:@"right"])
33         {
34                 //检测到右边动画的时候,开始左边的动画
35             [load leftAnimation];
36         }
37     };
38
39 }
40
41
42
```

3.3 结束动画

当结束动画的时候,block什么都不干

还记得我们一开始声明的数组么,我们把所有的layer都添加进去了

现在我们就可以在动画结束之后,把他们移除了

```
-(void)hideAnimation
{
    NSLog(@"结束动画");

    animationFinishBlock = ^(CAAnimation * animation){



    };

    for (CALayer * layer in _array) {

        [layer removeFromSuperlayer];
    }



}

```

3.4 动画结束的代理方法

```

-(void)animationDidStop:(CAAnimation *)anim finished:(BOOL)flag
{


    animationFinishBlock(anim);


}
```

动画结束后,我们调用这个block就行了,其实相当于下边的两种情况

3.4.1 显示动画的时候

```
-(void)animationDidStop:(CAAnimation *)anim finished:(BOOL)flag
{


     if ([[anim valueForKey:@"left"] isEqualToString:@"left"]) {

            [load rightAnimation];
        }else if([[anim valueForKey:@"right"] isEqualToString:@"right"])
        {
            [load leftAnimation];
        }


}
```

3.4.2 隐藏动画的时候

```
-(void)animationDidStop:(CAAnimation *)anim finished:(BOOL)flag
{



}
```

这个应该不难理解,按照上边的逻辑,就应该是这个样子的

好了,基本上就完成了,但离目标还有一点,牛顿摆嘛,人家那是小球,这货完全没有立体感啊,只有一个阴影,忽悠谁啊,这个还有待进一步研究

这里是Demo (https://github.com/xiao333ma/iOSAnimation_NewtonCradle.git)

秘籍传送门 (https://www.gitbook.com/book/zsisme/ios-/details)

查看原文>> (http://blog.csdn.net/xiao333ma/article/details/49496043)

1

看过本文的人也看了：

- iOS知识结构图 (http://lib.csdn.net/base/ios/structure)
- iOS网络编程(三) 异步加载及缓存图片... (http://lib.csdn.net/article/ios/36941)
- IOS URL加载系统 UIWebView URL拦... (http://lib.csdn.net/article/ios/36935)
- UITableView的优化技巧 (http://lib.csdn.net/article/ios/44684)
- iOS加载本地js、css等框架 (http://lib.csdn.net/article/ios/38653)
- iOS UITableViewCell 多线程 网络+沙... (http://lib.csdn.net/article/ios/36940)

发表评论

输入评论内容

发表

1个评论

 (http://my.csdn.net/qq_36262342)

qq_36262342 (http://my.csdn.net/qq_36262342)

up

2016-09-28 11:25:59 回复