CAAnimation学习

> *在应用程序中，随处可见富有绚丽的动画效果，大大提高了应用的人机交互效果。比如UIKit框架中实现的UINavigationController的Push和Pop动画，还有Modal视图的Present效果等等。动画能够使应用程序变得更加生动，富有灵性。本节我们将会学习如何使用CAAnimation实现动画效果，让你的应用程序可以自定义出更多丰富的动画效果。*

章节一 隐式动画

上节课中我们学习了CALayer，CALayer在设计之初就是为了实现动画功能，所以它并不在UIKit框架中，而是在Core Animation框架中，从命名上我们也可以看出Core Animation框架主要功能是动画。CALayer中有很多的属性可以设置，很多属性的值在修改时会默认添加动画，我们称之为隐式动画，哪些属性支持隐式动画呢，请看下面的列表：

属性	说明	是否支持隐式动画
anchorPoint	和中心点position重合的一个点，称为“锚点”，锚点的描述是相对于x、y位置比例而言的默认在图像中心点(0.5,0.5)的位置	是
backgroundColor	图层背景颜色	是
borderColor	边框颜色	是
borderWidth	边框宽度	是
bounds	图层大小	是
contents	图层显示内容，例如可以将图片作为图层内容显示	是
contentsRect	图层显示内容的大小和位置	是
cornerRadius	圆角半径	是
doubleSided	图层背面是否显示，默认为YES	否
frame	图层大小和位置，不支持隐式动画，所以CALayer中很少使用frame，通常使用bounds和position代替	否
hidden	是否隐藏	是
mask	图层蒙版	是
maskToBounds	子图层是否剪切图层边界，默认为NO	是
opacity	透明度，类似于UIView的alpha	是
position	图层中心点位置，类似于UIView的center	是
shadowColor	阴影颜色	是
shadowOffset	阴影偏移量	是
shadowOpacity	阴影透明度，注意默认为0，如果设置阴影必须设置此属性	是
shadowPath	阴影的形状	是
shadowRadius	阴影模糊半径	是
sublayers	子图层	是
sublayerTransform	子图层形变	是
transform	图层形变	是

从图中我们可以看出，frame属性是不支持隐式动画的，所以我们也不常用CALayer的frame属性

在CALayer课程中，已经讲解过如何触发隐式动画，本节将不再复述。

章节二 Core Animation

在以前的UIKit学习中，我们经常使用到如下的动画代码：

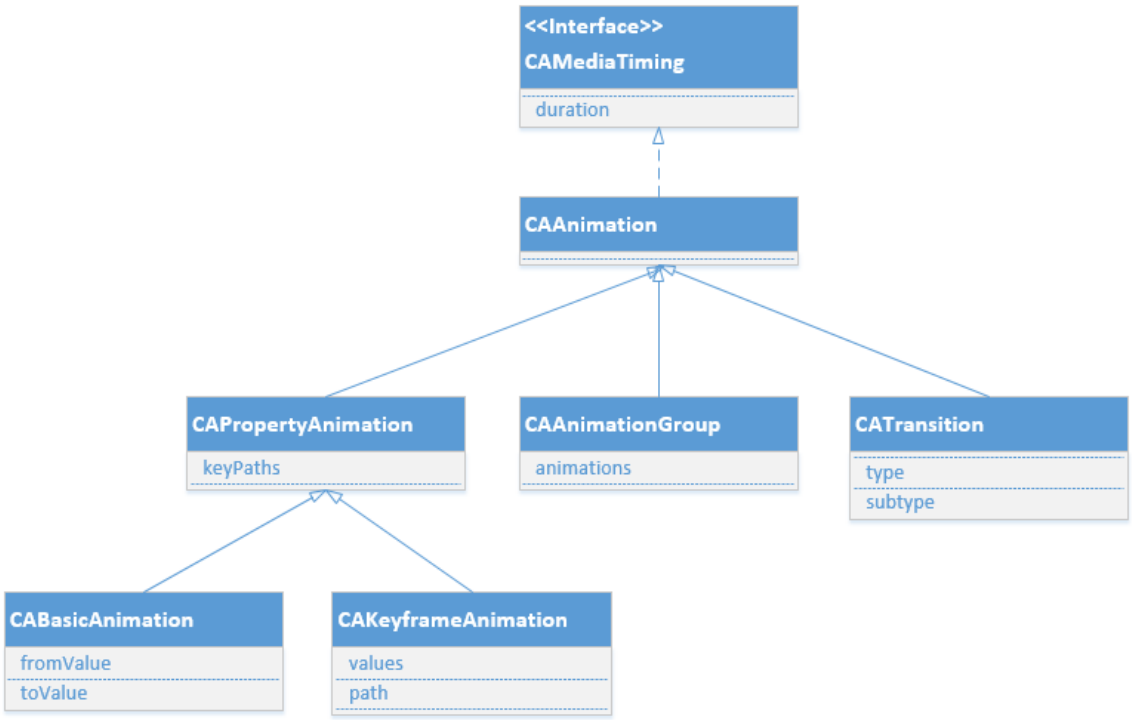
```
[UIView animateWithDuration:1 delay:2 options:UIViewAnimationOptionBeginFromCurrentState animations:^{
        imageView.frame=CGRectMake(80, 100, 160, 160);
    } completion:nil];
```

此段代码的功能是，两秒后开始一个时间为一秒的动画，动画会把imageView的frame从当前值过渡到(80, 100, 160, 160)，过渡过程就是

我们触发的动画。在iOS中使用UIKit内封装的动画功能十分简单，其他平台基本都很难做到这点。但是具体动画如何实现我们是不清楚的（笔者推断是使用CABasicAnimation实现的，后文中会学习），再有我们无法控制动画的具体过程，甚至无法暂停动画或者组合多个动画。CAAnimation就给我们提供了充分并且丰富的解决方案。

了解iOS的核心动画Core Animation（包含在Quartz Core框架中）。在iOS中核心动画分为几类：基础动画、关键帧动画、动画组、转场动画

首先我们来看结构图：



- CAAnimation：核心动画的基础类，**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。
- CAPropertyAnimation：属性动画的基类（通过属性进行动画设置，注意是可动画属性），不能直接使用。
- CAAnimationGroup：动画组，动画组是一种组合模式设计，可以通过动画组来进行所有动画行为的统一控制，组中所有动画效果可以并发执行。
- CATransition：转场动画，主要通过滤镜进行动画效果设置。
- 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。
- CAKeyframeAnimation：关键帧动画，同样是通过属性进行动画参数控制，但是同基础动画不同的是它可以有多个状态控制。

下面我们分别对他们进行学习

CAAnimation

属性说明：(加粗代表来自CAMediaTiming协议的属性)

- **duration**：动画的持续时间
- **repeatCount**：重复次数，无限循环可以设置HUGE_VALF或者MAXFLOAT
- **repeatDuration**：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。比如动画开始之前或者动画结束之后
- **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

fillMode属性 (要想fillMode有效，最好设置removeOnCompletion = NO)

```
kCAFillModeRemoved  默认值，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会恢复到动画开始前的状态

kCAFillModeForwards  当动画结束后，layer会保持动画最后的显示状态

kCAFillModeBackwards 在动画开始前，只需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的开始

kCAFillModeBoth 上面两个的合成，动画加入后开始前，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
```

速度控制函数(CAMediaTimingFunction)

```
kCAMediaTimingFunctionLinear（线性）：匀速，给你一个相对静态的感觉
kCAMediaTimingFunctionEaseIn（渐进）：动画缓慢进入，然后加速离开
kCAMediaTimingFunctionEaseOut（渐出）：动画全速进入，然后减速的到达目的地
kCAMediaTimingFunctionEaseInEaseOut（渐进渐出）：动画缓慢的进入，中间加速，然后减速的到达目的地。这个是默认的动画行为。
```

动画暂停和继续功能实现

```
// 动画的暂停
-(void)pauseLayer:(CALayer*)layer
{
    CFTimeInterval pausedTime = [layer convertTime:CACurrentMediaTime() fromLayer:nil];
    // 让CALayer的时间停止走动
      layer.speed = 0.0;
    // 让CALayer的时间停留在pausedTime这个时刻
    layer.timeOffset = pausedTime;
}


// 动画的恢复
-(void)resumeLayer:(CALayer*)layer
{
    CFTimeInterval pausedTime = layer.timeOffset;
    // 1. 让CALayer的时间继续行走
      layer.speed = 1.0;
    // 2. 取消上次记录的停留时刻
      layer.timeOffset = 0.0;
    // 3. 取消上次设置的时间
      layer.beginTime = 0.0;

    // 4. 计算暂停的时间(这里也可以用CACurrentMediaTime()-pausedTime)
    CFTimeInterval timeSincePause = [layer convertTime:CACurrentMediaTime() fromLayer:nil] - pausedTime;
    // 5. 设置相对于父坐标系的开始时间(往后退timeSincePause)
      layer.beginTime = timeSincePause;
}
```

CAPropertyAnimation

CAPropertyAnimation是CAAnimation的子类，也是个抽象类，要想创建动画对象，应该使用它的两个子类 * CABasicAnimation * CAKeyframeAnimation

属性说明：

- keyPath：通过指定CALayer的一个属性名称为keyPath（NSString类型），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，就修改CALayer的position属性的值，以达到平移的动画效果。

CABasicAnimation 基础动画

> *随着动画的进行，在长度为duration的持续时间内，keyPath相应属性的值从fromValue渐渐地变为toValue*

属性说明:

- fromValue：keyPath相应属性的初始值
- toValue：keyPath相应属性的结束值

CAKeyframeAnimation 关键帧动画

> *关键帧动画，是CAPropertyAnimation的子类，与CABasicAnimation的区别是 CABasicAnimation只能从一个数值（fromValue）变到另一个数值（toValue），而CAKeyframeAnimation会使用一个NSArray保存这些数值*
>
> ***属性说明:***
>
> - *values：上述的NSArray对象。里面的元素称为"关键帧"(keyframe)。动画对象会在指定的时间（duration）内，依次显示values数组中的每一个关键帧*
> - *path：可以设置一个CGPathRef、CGMutablePathRef，让图层按照路径轨迹移动。path只对CALayer的anchorPoint和position起作用。如果设置了path，那么values将被忽略*
> - *keyTimes：可以为对应的关键帧指定对应的时间点，其取值范围为0到1.0，keyTimes中的每一个时间值都对应values中的每一帧。如果没有设置keyTimes，各个关键帧的时间是平分的*

CAAnimationGroup 动画组

> *动画组，是CAAnimation的子类，可以保存一组动画对象，将CAAnimationGroup对象加入层后，组中所有动画对象可以同时并发运行*
>
> ***属性说明:***
>
> - *animations：用来保存一组动画对象的NSArray*

默认情况下，一组动画对象是同时运行的，也可以通过设置动画对象的beginTime属性来更改动画的开始时间

CATransition 转场动画

> *CATransition是CAAnimation的子类，用于做转场动画，能够为层提供移出屏幕和移入屏幕的动画效果。iOS比Mac OS X的转场动画效果少一点 UINavigationController就是通过CATransition实现了将控制器的视图推入屏幕的动画效果*
>
> ***属性说明:***
>
> - *type：动画过渡类型*
> - *subtype：动画过渡方向*
> - *startProgress：动画起点(在整体动画的百分比)*
> - *endProgress：动画终点(在整体动画的百分比)*

type属性说明:

动画类型	说明	对应常量	是否支持方向设置
公开API			
fade	淡出效果	kCATransitionFade	是
movein	新视图移动到旧视图上	kCATransitionMoveIn	是
push	新视图推出旧视图	kCATransitionPush	是
reveal	移开旧视图显示新视图	kCATransitionReveal	是
私有API		**私有API只能通过字符串访问**	
cube	立方体翻转效果	无	是
oglFlip	翻转效果	无	是
suckEffect	收缩效果	无	否
rippleEffect	水滴波纹效果	无	否
pageCurl	向上翻页效果	无	是
pageUnCurl	向下翻页效果	无	是
cameraIrisHollowOpen	摄像头打开效果	无	否
cameraIrisHollowClose	摄像头关闭效果	无	否

编码演示：

```
- (void)viewDidLoad {
    [super viewDidLoad];

    //定义图片控件
    _imageView=[[UIImageView alloc]init];
    _imageView.frame=[UIScreen mainScreen].applicationFrame;
    _imageView.contentMode=UIViewContentModeScaleAspectFit;
    _imageView.image=[UIImage imageNamed:@"0.jpg"];//默认图片
    [self.view addSubview:_imageView];
    //添加手势
    UISwipeGestureRecognizer *leftSwipeGesture=[[UISwipeGestureRecognizer alloc]initWithTarget:self
action:@selector(leftSwipe:)];
    leftSwipeGesture.direction=UISwipeGestureRecognizerDirectionLeft;
    [self.view addGestureRecognizer:leftSwipeGesture];

    UISwipeGestureRecognizer *rightSwipeGesture=[[UISwipeGestureRecognizer alloc]initWithTarget:self
action:@selector(rightSwipe:)];
    rightSwipeGesture.direction=UISwipeGestureRecognizerDirectionRight;
    [self.view addGestureRecognizer:rightSwipeGesture];
}

    #pragma mark 向左滑动浏览下一张图片
-(void)leftSwipe:(UISwipeGestureRecognizer *)gesture{
    [self transitionAnimation:YES];
}

    #pragma mark 向右滑动浏览上一张图片
-(void)rightSwipe:(UISwipeGestureRecognizer *)gesture{
    [self transitionAnimation:NO];
}


#pragma mark 转场动画
-(void)transitionAnimation:(BOOL)isNext{
    //1.创建转场动画对象
    CATransition *transition=[[CATransition alloc]init];

    //2.设置动画类型,注意对于苹果官方没公开的动画类型只能使用字符串，并没有对应的常量定义
    transition.type=@"cube";

    //设置子类型
    if (isNext) {
        transition.subtype=kCATransitionFromRight;
    }else{
        transition.subtype=kCATransitionFromLeft;
    }
    //设置动画时常
    transition.duration=1.0f;

    //3.设置转场后的新视图添加转场动画
    _imageView.image=[self getImage:isNext];
    [_imageView.layer addAnimation:transition forKey:@"KCTransitionAnimation"];
}

#pragma mark 取得当前图片
-(UIImage *)getImage:(BOOL)isNext{
    if (isNext) {
        _currentIndex=(_currentIndex+1)%IMAGE_COUNT;
    }else{
        _currentIndex=(_currentIndex-1+IMAGE_COUNT)%IMAGE_COUNT;
    }
    NSString *imageName=[NSString stringWithFormat:@"%i.jpg",_currentIndex];
    return [UIImage imageNamed:imageName];
}
```

通过以上简单地代码就可以实现非常绚丽的效果。

总结：

通过今天的学习，我们应该已经掌握了哪些动画功能可以通过CAAnimation实现，如果不能实现，就可能需要美工配合你来提供更丰富的UI素材来达到需求了。