Core Animation

`CABasicAnimation`

`CAKeyframeAnimation`

`CAAnimationGroup`

`CATransition`

- 隐式动画 (duration为0.25s)

```
  heLayer.opacity = 0.0;
```

- 显式动画

```
  CABasicAnimation* fadeAnim = [CABasicAnimation animationWithKeyPath:
@"opacity"];
  fadeAnim.fromValue = [NSNumber numberWithFloat:1.0];
  fadeAnim.toValue = [NSNumber numberWithFloat:0.0];
  fadeAnim.duration = 1.0;
  [theLayer addAnimation:fadeAnim forKey:@"opacity"];

  // 在动画结束后，改变theLayer的值为实际想成为的值
  theLayer.opacity = 0.0;
```

> 一个显示的动画并不会改变图层数种的真实地数据。显示的动画只是产生动画的效果。

- 创建一个跳跃的关键帧动画

```
  // 创建一个path，来构建两个跳跃的曲线
  CGMutablePathRef thePath = CGPathCreateMutable();
  CGPathMoveToPoint(thePath,NULL,74.0,74.0);
  CGPathAddCurveToPoint(thePath,NULL,74.0,500.0,
                                     320.0,500.0,
                                     320.0,74.0);
  CGPathAddCurveToPoint(thePath,NULL,320.0,500.0,
                                     566.0,500.0,
                                     566.0,74.0);

  // 创建关键帧动画对象，指定position属性按照path路径指定的进行变化

  CAKeyframeAnimation * theAnimation;

  theAnimation = [CAKeyframeAnimation animationWithKeyPath:@"position"
];
  theAnimation.path = thePath;
  theAnimation.duration = 5.0;

  // 将动画添加到图层上
  [theLayer addAnimation:theAnimation forKey:@"position"];
```

- 使用group来实现同时进行多个动画

```
  // Animation 1
  CAKeyframeAnimation* widthAnim = [CAKeyframeAnimation animationWithK
eyPath:@"borderWidth"];
  NSArray* widthValues = [NSArray arrayWithObjects:@1.0, @10.0, @5.0,
@30.0, @0.5, @15.0, @2.0, @50.0, @0.0, nil];
  widthAnim.values = widthValues;
  widthAnim.calculationMode = kCAAnimationPaced;

  // Animation 2
  CAKeyframeAnimation* colorAnim = [CAKeyframeAnimation animationWithK
eyPath:@"borderColor"];
  NSArray* colorValues = [NSArray arrayWithObjects:(id)[UIColor greenC
olor].CGColor,
              (id)[UIColor redColor].CGColor, (id)[UIColor blueColor].
CGColor,  nil];
  colorAnim.values = colorValues;
  colorAnim.calculationMode = kCAAnimationPaced;

  // Animation group
  CAAnimationGroup* group = [CAAnimationGroup animation];
  group.animations = [NSArray arrayWithObjects:colorAnim, widthAnim, n
il];
  group.duration = 5.0;

  [myLayer addAnimation:group forKey:@"BorderChanges"];
```

- 明确地使用事务来嵌套动画

> CATransaction（事务）是Core Animation的一种机制，事务的作用是为了保证多个animatable属性的变化同时进行。大部分时候，并不需要专门创建事务。 只要你对图层添加了动画操作，Core Animation 就会自动地创建一个隐式的事务，在线程的下个RunLoop开始时自动 commit这个事务。当然，你也可以显示地创建事务来精确地管理那些动画。显示地创建事务的最主要的原因就是你可以改变duration、时间函数和其他的参数。你也可以为整个事务分配一个完成之后的代码块以方便整个动画组完成后进行回调的通知。

```
  [CATransaction begin]; // 外层事务开始

  // 修改动画的duration为2s
  [CATransaction setValue:@(2.0f) forKey:kCATransactionAnimationDurati
on];
  // 修改layer的position
  theLayer.position = CGPointMake(0.0,0.0);

  [CATransaction begin]; // 内层事务开始
  // 修改动画的duration为5s
  [CATransaction setValue:@(5.0f)                  forKey:kCATransacti
onAnimationDuration];

  // Change the zPosition and opacity
  theLayer.zPosition=200.0;
  theLayer.opacity=0.0;

  [CATransaction commit]; // 内层事务提交
  [CATransaction commit]; // 外层事务提交
```

- 动画iOS视图对象中绑定的图层

```
  [UIView animateWithDuration:1.0 animations:^{
      // 隐式地改变opacity
     myView.layer.opacity = 0.0;

      // 显示地改变position
     CABasicAnimation* theAnim = [CABasicAnimation animationWithKeyPat
h:@"position"];
     theAnim.fromValue = [NSValue valueWithCGPoint:myView.layer.positi
on];
     theAnim.toValue = [NSValue valueWithCGPoint:myNewPosition];
     theAnim.duration = 3.0;
     [myView.layer addAnimation:theAnim forKey:@"AnimateFrame"];
  }];
```

- 在iOS的两个视图中执行转场动画

```
CATransition* transition = [CATransition animation];
transition.startProgress = 0;
transition.endProgress = 1.0;
transition.type = kCATransitionPush;
transition.subtype = kCATransitionFromRight;
transition.duration = 1.0;

// 为两个图层都添加转场动画
[myView1.layer addAnimation:transition forKey:@"transition"];
[myView2.layer addAnimation:transition forKey:@"transition"];

// 最后改变两个视图的hidden属性
myView1.hidden = YES;
myView2.hidden = NO;
```