# CALayer图层

> *本节课我们将深度学习UIView的绘制原理，最终你将会有一个完整的屏幕绘制逻辑。为下一章节“Core Animation”做一下铺垫工作。*

## 章节一 UIView和CALayer的关系

### UIView

#### UIView的继承结构为: UIResponder : NSObject

UIResponder 官方解释

> *The UIResponder class defines an interface for objects that respond to and handle events. It is the superclass of UIApplication, UIView and its subclasses (which include UIWindow). Instances of these classes are sometimes referred to as responder objects or, simply, responders.*

UIView 官方解释

> *The UIView class defines a rectangular area on the screen and the interfaces for managing the content in that area. At runtime, a view object handles the rendering of any content in its area and also handles any interactions with that content.*

可见 UIResponder是用来响应事件的，也就是UIView可以响应用户事件。UIView还有第二个功能就是用来管理显示的内容。

#### 所属框架UIKit

> *The UIKit framework provides the classes needed to construct and manage an application’s user interface for iOS. It provides an application object, event handling, drawing model, windows, views, and controls specifically designed for*

*a touch screen interface.*

UIKit主要是用来构建用户界面，并且可以响应事件的 (*详见UIResponder章节*)

## CALayer

### CALayer的继承结构为： NSObject

> *The CALayer class is the model class for layer-tree objects. It encapsulates the position, size, and transform of a layer, which defines its coordinate system.*

CALayer是图层树的模型类，封装了图层的位置、大小、形变等，定义了图层的坐标系统

我们了解到，UIView 和CALayer的基本信息和主要负责处理的事情。

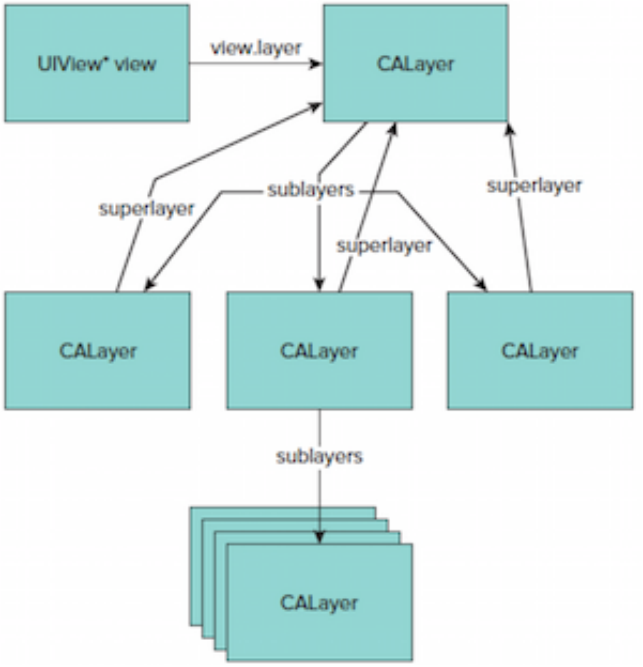
### CALayer所属框架QuartzCore

CALayer作为一个低级的，可以承载绘制内容的底层对象出现在该框架中。

> *Core Animation doesn't provide a means for actually displaying layers in a window, they must be hosted by a view. When paired with a view, the view must provide event-handling for the underlying layers, while the layers provide display of the content.*
> *The view system in iOS is built directly on top of Core Animation layers. Every instance of UIView automatically creates an instance of a CALayer class and sets it as the value of the view's layer property. You can add sublayers to the view's layer as needed.*

UIView侧重于对显示内容的管理，CALayer侧重于对内容的绘制。创建人和UIView对象时，会自动生成一个CALayer，并且作为View的layer属性。我们可以在这个view的layer上添加子图层。

CALayer具有和UIView相似的属性和结构




- 相似的树形结构
- 都有绘制内容的方法和相关属性
- 可以布局约束

小结： **UIView和CALayer并不是两套重复的绘图系统，它们是互相依赖的，CALayer依靠UIView提供的内容来绘制。没有CALayer，UIView是无法绘制视图的。UIView来自CALayer，高于CALayer，是CALayer高层实现与封装。UIView的所有界面特性来源于CALayer支持，响应特性来源于父类UIResponder。**

> *在使用CALayer前，应该引入QuartzCore框架。因为UIView的头文件中使用的是@class类声明*
> *在使用时避免使用UIColor等UIKit提供的类或函数，因为QuartzCore是跨平台的框架。Mac OS X中没有UIKit框架。*

## 章节二 CALayer的常用属性

**修改某些属性时，会自动带有动画， 在文档中搜索“Animatable” 即可查找到可动画的属性**

禁用动画,使用方法`[CATransaction setDisableActions:YES];`

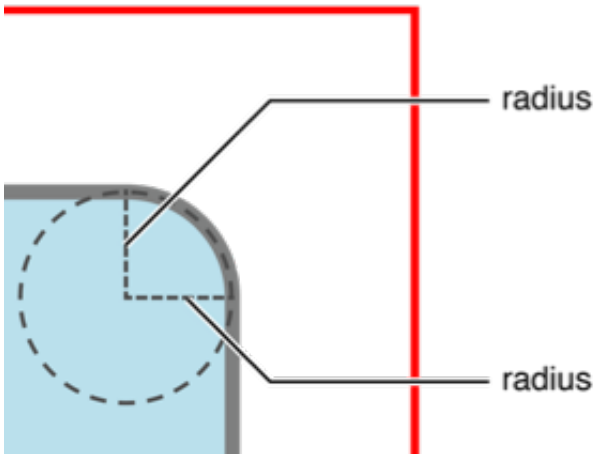
### 0.基本属性

```
CALayer *myLayer = [CALayer layer];

//将图层添加到视图的图层上
[self.view.layer addSublayer:myLayer];

//2.设置属性
//<1> 设置尺寸
[myLayer setBounds:CGRectMake(0, 0, 200, 200)];
//<2>设置背景颜色
myLayer.backgroundColor = [UIColor redColor].CGColor;
//<3>中心点(默认对应UIView的center)
myLayer.position = CGPointMake(100, 100);
//<4>锚点，定位点 (x\y的取值范围0-1),锚点显示在positon点的位置
myLayer.anchorPoint = CGPointMake(0, 0);
```

## 1.圆角




```
//1.设置圆角
view.layer.cornerRadius = 50;
```

## 2.边框

```
//3.边框
view.layer.borderColor = [UIColor whiteColor].CGColor;
view.layer.borderWidth = 3.0f;
```

### 3.阴影

```
//2.阴影
//阴影的颜色
view.layer.shadowColor = [UIColor lightGrayColor].CGColor;
//阴影的偏移量
view.layer.shadowOffset = CGSizeMake(0, 10);
//阴影的透明度
view.layer.shadowOpacity = 1;
```

### 4.内容contents

```
UIImage *img = [UIImage imageNamed:@"2012100413195471481.jpg"];
myLayer.contents = (id)img.CGImage;
```

### 5.3D形变

CALayer的形变属性为 `CATransform3D transform` 与UIView的仿射变换属性 `CGAffineTransform transform` 类似，但是功能会更加强大，能够在使二维的坐标系统模拟出3D的显示效果。

UIView仿射变换的最终实现就是CALayer的transform，所以CATransform3D可以实现所有CGAffineTransform的功能。

CGAffineTransform结构体中定义了16个变量，从m11到m44。分别表示矩阵中的不同转换倍数或者偏移量。可以通过函数的方式快速构建CGAffineTransform结构体变量，得到我们需要的变换矩阵。

#### 1.位移

`CATransform3DMakeTranslation()` 等

#### 2.旋转

`CATransform3DMakeRotation()` 等

#### 3.缩放

`CATransform3DMakeScale()` 等

**详见代码示例**

## 章节三 CALayer的代理方法

UIView有一个layerClass方法，返回主layer所使用的类，UIView的子类，可以通过重载这个方法，来让UIView使用不同的CALayer来显示，例如通过

```
- (class) layerClass {

  return ([CAEAGLLayer class]);

}
```

使某个UIView的子类使用GL来进行绘制。

**要在CALayer上绘图，有两种方法：**

1. 创建一个CALayer的子类，然后覆盖drawInContext:方法，可以使用Quartz2D API在其中进行绘图
2. 设置CALayer的delegate，然后让delegate实现drawLayer:inContext:方法进行绘图

UIView的根layer默认把delegate设置为了视图本身，不需要我们重新设置

`- (void)drawLayer:(CALayer *)layer inContext:(CGContextRef)ctx;` **详见代码示例**