梯度动画

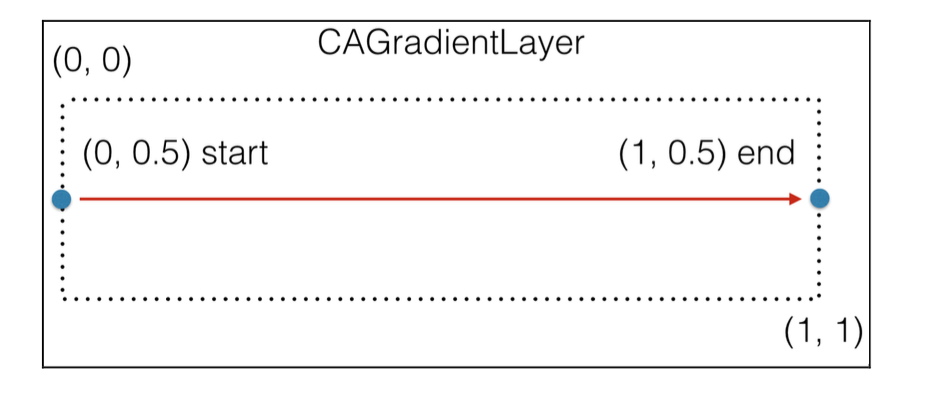
iOS8中有非常漂亮的动画，就是你看时间的时候，在屏幕下方“滑动解锁”的样式；

就模仿iPhone的解锁样式实现这个梯度动画；

首先在SB中放置一个Label，显示时间，自定义个类实现梯度动画：

先了解梯度动画：

gradientLayer.startPoint = CGPoint(x: 0.0, y: 0.5) gradientLayer.endPoint = CGPoint(x: 1.0, y: 0.5)



let colors = [ UIColor.blackColor().CGColor, UIColor.whiteColor().CGColor, UIColor.blackColor().CGColor

] gradientLayer.colors = colors

设置颜色出现的位置

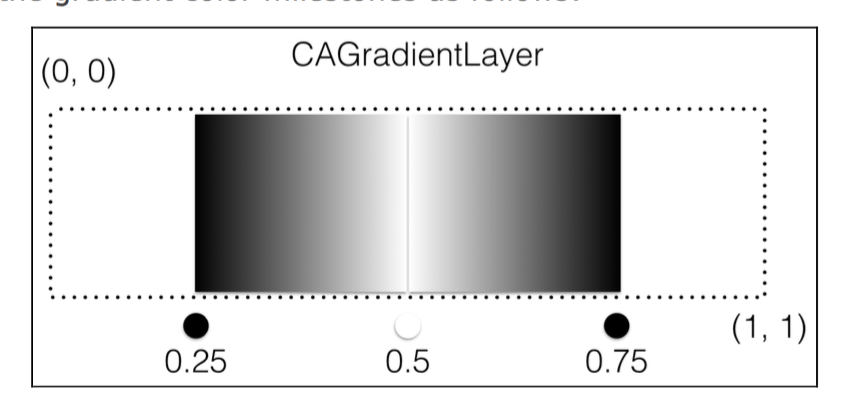
let locations = [ 0.25,

0.5,

0.75

] gradientLayer.locations = locations

如下图：



设置梯度动画的Frame

gradientLayer.frame = bounds

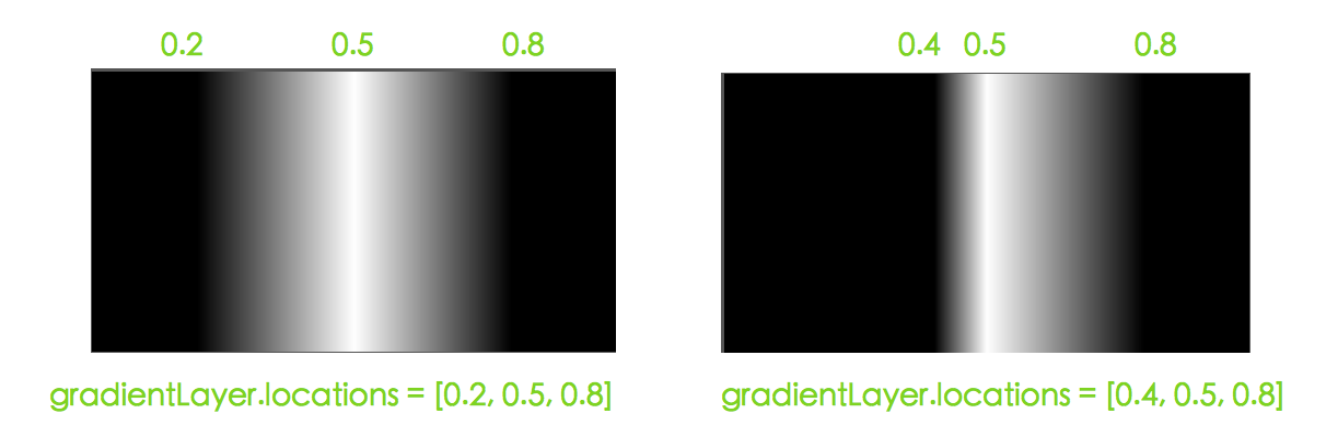
将gradientLayer添加到View.Layer上，在didMoveToWindow（）

layer.addSublayer(gradientLayer)

CAGradientLayer 提供了四个属性，都继承自CALayer；

colors ：梯度动画颜色

locations ：用图片来解释。



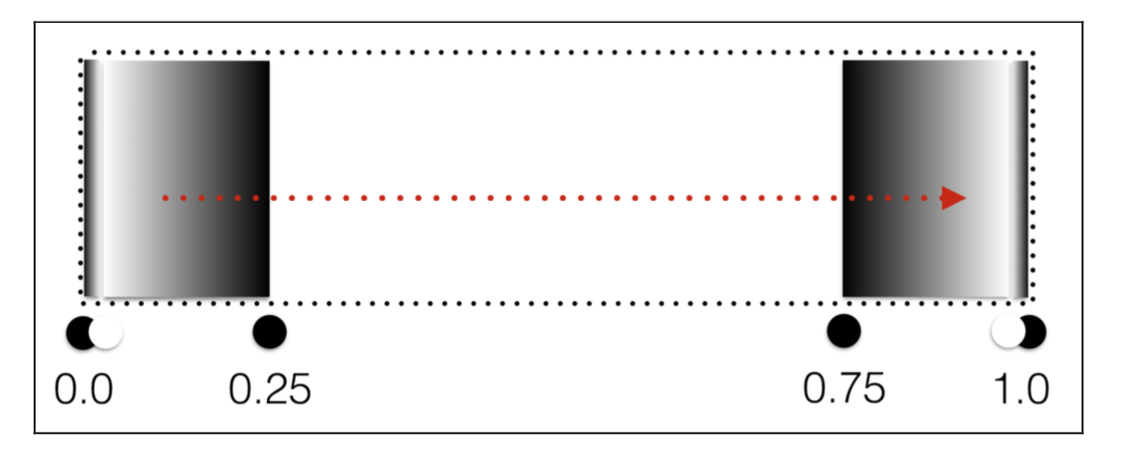
startPoint and endPoint ：梯度动画的范围；

接着就是让它移动：

let gradientAnimation = CABasicAnimation(keyPath: "locations")

gradientAnimation.fromValue = [0.0, 0.0, 0.25] gradientAnimation.toValue = [0.75, 1.0, 1.0] gradientAnimation.duration = 3.0

gradientAnimation.repeatCount = Float.infinity



**创建一个常量属性来控制text的属性：**

let textAttributes: NSDictionary = { let style = NSMutableParagraphStyle() style.alignment = .Center

return [ NSFontAttributeName:UIFont(name: "HelveticaNeue-Thin",

size: 28.0)!, NSParagraphStyleAttributeName: style

] }()

**创建一个临时的****图像上下文****（graphic context）将text渲染为image**

UIGraphicsBeginImageContextWithOptions(frame.size, false, 0)

let context = UIGraphicsGetCurrentContext()

**创建一个大小和View一样大的并且是空的graphics context，接下来将text绘制到上下文中**

text.drawInRect(bounds, withAttributes: textAttributes) let image = UIGraphicsGetImageFromCurrentImageContext() UIGraphicsEndImageContext()

drawInRect(\_ :withAttributes: ) 实现绘制，UIGraphicsGetImageFromCurrentImageContext （）匹配你所绘制的，并返回给你一个实例的image；接着就可以使用这个image在geadient layer 上创建一个修饰层；

let maskLayer = CALayer()

 maskLayer.backgroundColor = UIColor.clearColor().CGColor

maskLayer.frame = CGRectOffset(bounds, bounds.size.width, 0)

maskLayer.contents = image.CGImage

使用默认的初始化方式创建了一个空的maskLayer，设置它的Layer层的背景色为透明色，以此作为修饰层；然后设置偏移View的宽度，这样修饰层就会直接显示在梯度的中心，当你的“拉伸”梯度是目前的三倍宽的可见视图。最后，你声明一个image对象直接放着Layer的属性contents之上；

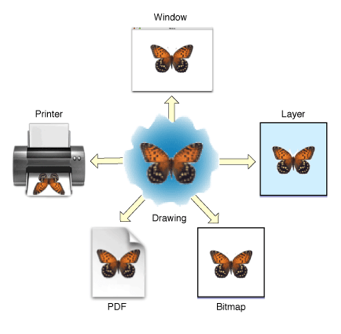
图形上下文（Graphics Context）：是一个CGContextRef类型的数据

图形上下文的作用：保存绘图信息、绘图状态

决定绘制的输出目标（绘制到什么地方去？）（输出目标可以是PDF文件、Bitmap或者显示器的窗口上）



相同的一套绘图序列，指定不同的Graphics Context，就可将相同的图像绘制到不同的目标上。



Quartz2D提供了以下几种类型的Graphics Context：

Bitmap Graphics Context

PDF Graphics Context

Window Graphics Context

Layer Graphics Context

Printer Graphics Context