Stroke and Path Animations

本次主要结合自定义刷新控件和动画：

第一步创建一个圆形；

ovalShapeLayer.strokeColor = UIColor.whiteColor().CGColor

ovalShapeLayer.fillColor = UIColor.clearColor().CGColor ovalShapeLayer.lineWidth = 4.0 ovalShapeLayer.lineDashPattern = [2, 3]

let refreshRadius = frame.size.height/2 \* 0.8

ovalShapeLayer.path = UIBezierPath(ovalInRect: CGRect( x: frame.size.width/2 - refreshRadius, y: frame.size.height/2 - refreshRadius, width: 2 \* refreshRadius,

height: 2 \* refreshRadius) ).CGPath

layer.addSublayer(ovalShapeLayer)

ovalShapeLayer 是CAShapeLayer 类型，将高度设置为View的80%，确保合适，

lineDashPattern ：设置虚线,默认为nil;

在redrawFromProgress()

中添加ovalShapeLayer.strokeEnd = progress

进度从0-1，圆形的形状顺时针形成；

最后你会发现，当拉到最下侧之后，会形成一个圆，接下来加个更好的效果，在init(frame:scrollView:)

中添加：

let airplaneImage = UIImage(named: "airplane.png")! airplaneLayer.contents = airplaneImage.CGImage airplaneLayer.bounds = CGRect(x: 0.0, y: 0.0, width: airplaneImage.size.width, height: airplaneImage.size.height) airplaneLayer.position = CGPoint(x: frame.size.width/2 + frame.size.height/2 \* 0.8, y: frame.size.height/2)

layer.addSublayer(airplaneLayer)

在Layer层添加一个image，将它放在3点钟方向，在此运行会发现它已经在制定位置，将透明度设置为0当用户下拉的时候，随着用户的交互而改变airplan的透明度；

接下来设置strokeStart、strokeEnd 属性；确保动画同时循环执行，

let strokeStartAnimation = CABasicAnimation( keyPath: "strokeStart")

strokeStartAnimation.fromValue = -0.5 strokeStartAnimation.toValue = 1.0

let strokeEndAnimation = CABasicAnimation(keyPath: "strokeEnd") strokeEndAnimation.fromValue = 0.0 strokeEndAnimation.toValue = 1.0

这段代码创建两个动画：

第一个动画strokeStart

从-0.5 -- 1.0，这是个简单的动画窍门，当value从-0.5到0.0什么都不发生，负值意味着不可见，

第二个动画：strokeEnd 画了一个小的形状,直到strokeStart赶上strokeEnd动画。

let strokeAnimationGroup = CAAnimationGroup() strokeAnimationGroup.duration = 1.5 strokeAnimationGroup.repeatDuration = 5.0 strokeAnimationGroup.animations = [strokeStartAnimation, strokeEndAnimation] ovalShapeLayer.addAnimation(strokeAnimationGroup, forKey: nil)

接下来创建一个动画群组，并且重复五次，这样确保刷新的时候使这个圆可见。

创建关键帧动画：

let flightAnimation = CAKeyframeAnimation(keyPath: "position") flightAnimation.path = ovalShapeLayer.path flightAnimation.calculationMode = kCAAnimationPaced

你创建一个新的CAKeyframeAnimation,像之前一样,位置设置动画属性。但这一次你赋值路径;在这种情况下你可以重用ovalShapeLayer的循环路径。

最后设置动画calculationMode节奏模式——这将确保层沿路径的顺利。

现在需要创建一个组的动画和运行它在飞机上层;

let flightAnimationGroup = CAAnimationGroup() flightAnimationGroup.duration = 1.5 flightAnimationGroup.repeatDuration = 5.0 flightAnimationGroup.animations = [flightAnimation] airplaneLayer.addAnimation(flightAnimationGroup, forKey: nil)

在上面的代码中,创建组和给它的持续时间和重复计算进度条中风动画。然后添加flightAnimation作为唯一项组和组添加到飞机层。

CAKeyframeAnimation有着特殊的属性rotationMode,当设置为kCAAnimationRotateAuto将自动向图层的方向移动。然而,你将在这一章,手动创建这种效果,这是一个很简单的任务简单,循环路径。

let airplaneOrientationAnimation = CABasicAnimation( keyPath: "transform.rotation")

airplaneOrientationAnimation.fromValue = 0 airplaneOrientationAnimation.toValue = 2 \* M\_PI

airplaneOrientationAnimation层旋转360度,从0到2π,通过变换。换句话说,飞机图像完整绕其中心旋转,而且因为你移动飞机层沿圆形轨道同时,飞行将会看起来自然。