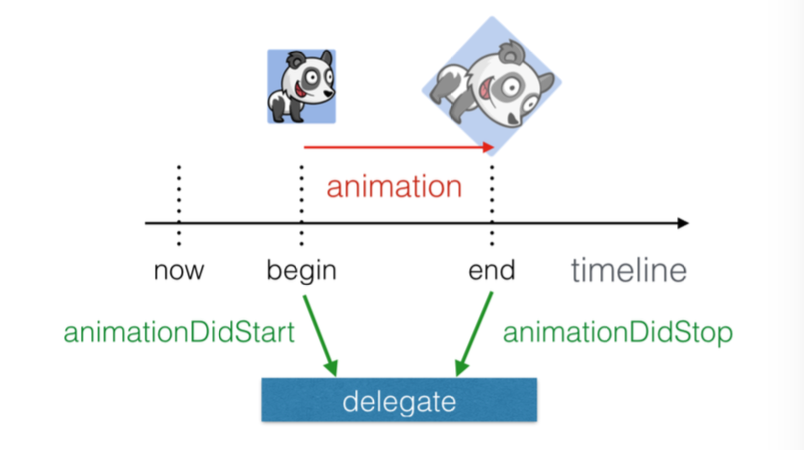
Animation Keys and Delegateses

棘手的部分关于UIKit关闭动画和相应的语法是一旦你创建并运行一个视图动画不能暂停,停止,或以任何方式访问它。

对于Core Animation可以很容易地检查动画运行在一层,如果你需要停止他们。

此外,你甚至可以设置一个委托对象动画和动画事件的反应。相比完成块你看过UIView动画,您可以收到委托回调时动画开始和结束(或中断)。

动画代理



CAAnimation及其子类 CABasicAnimation实现委托模式,让你对动画事件作出回应。CAAnimation没有一个特定的协议意味着他的协议必须实现;相反,每个实例NSObject已经实现了以下两个动画委托方法,你可以根据需要覆盖:

func animationDidStart(anim: CAAnimation!)

func animationDidStop(anim: CAAnimation!, finished flag: Bool)

UIKit or Cocoa通常不这样处理,但在这种情况下只可以这样使用。检查CoreAnimation模块的头文件,你会看到,它定义了两个NSObject委托方法。

设置自己为代理，并实现两个协议：

flyRight.delegate = self

func animationDidStart(anim: CAAnimation!)

func animationDidStop(anim: CAAnimation!, finished flag: Bool)

CAAnimation 类与它的子类是遵循键值编码协议的，意味着你可以像操作字典一样在运行时给他们添加新的属性

可以利用这个机制给flyRight声明一个名称，这样你就可以在其他的动画中依据这个id来查找！

在viewdidApper()中添加：

flyRight.setValue("form", forKey: "name") flyRight.setValue(heading.layer, forKey: "layer")

现在你在flyRight上创建了一个key并且设置它的表单，你可以在代理中依据这个ID来获得你设置的表单；

Switching on key values

在animationDidStop

中添加

let layer = anim.valueForKey("layer") as? CALayer anim.setValue(nil, forKey: "layer")

let pulse = CABasicAnimation(keyPath: "transform.scale") pulse.fromValue = 1.25

pulse.toValue = 1.0

pulse.duration = 0.25

layer?.addAnimation(pulse, forKey: nil)

setValue 设置为nil，意味着移除以原始的Layer为参照，然后重新设置一个新的Layer层的比例动画，扩大1.25倍之后恢复原样，

你可能已经注意到,addAnimation(\_:key:)有两个参数;到目前为止,只使用第一个参数。key参数是一个字符串标识符,允许访问和控制已经开始的动画。接下来将创建另一个层动画,学习如何在同一时间运行不止一个动画、如何使用键来控制运行动画。

添加一个由右至左的label，

let info = UILabel()

初始化

info.frame = CGRect(x: 0.0, y: loginButton.center.y + 60.0, width: view.frame.size.width, height: 30)

info.backgroundColor = UIColor.clearColor()

info.font = UIFont(name: "HelveticaNeue", size: 12.0) info.textAlignment = .Center

info.textColor = UIColor.whiteColor()

info.text = "Tap on a field and enter username and password"

view.insertSubview(info, belowSubview: loginButton)

在Viewdidapper()中添加代码

let flyLeft = CABasicAnimation(keyPath: "position.x") flyLeft.fromValue = info.layer.position.x +

view.frame.size.width flyLeft.toValue = info.layer.position.x

flyLeft.duration = 5.0 info.layer.addAnimation(flyLeft, forKey: "infoappear")

这样的标签就像你的表单字段,只有标签下滑会从屏幕的右边,飞到左边。注意设置动画的key :infoappear;将使用这个值来寻找,然后停止动画当用户开始在用户名或密码输入文本字段。

添加第二层动画

let fadeLabelIn = CABasicAnimation(keyPath: "opacity")

fadeLabelIn.fromValue = 0.2

 fadeLabelIn.toValue = 1.0

 fadeLabelIn.duration = 4.5

info.layer.addAnimation(fadeLabelIn, forKey: "fadein")

label从右至左的动画加上透明度由0.2至1.0的动画.

点击输入框触发事件

username.delegate = self

password.delegate = self

extension ViewController: UITextFieldDelegate { func textFieldDidBeginEditing(textField: UITextField) {

println(info.layer.animationKeys()) }

打印结果：

Optional(["fadein"])

为何没有打印出infoappear（从右至左的key）,因为这个动画执行完了，即便没有执行完，也不可以触发输入框，Layer层没有交互性。

接下来添加复杂一点的动画：

func animateCloud(layer: CALayer) {

//1

let cloudSpeed = 60.0 / Double(view.layer.frame.size.width)

let duration: NSTimeInterval = Double(view.layer.frame.size.width - layer.frame.origin.x) \* cloudSpeed



//2

let cloudMove = CABasicAnimation(keyPath: "position.x") cloudMove.duration = duration cloudMove.toValue = self.view.bounds.size.width + layer.bounds.width/2 cloudMove.delegate = self

cloudMove.setValue("cloud", forKey: "name") cloudMove.setValue(layer, forKey: "layer")

layer.addAnimation(cloudMove, forKey: nil) }

这个方法也有类似的实现使用UIKit变化的云。事实上,上面的部分的前几行/ / 1 UIKit实现几乎是一样的。

上面的第二部分方法创建一个实例的CABasicAnimation一层一层运动和设置一个名称和动画。

调用方法

animateCloud(cloud1.layer)

animateCloud(cloud2.layer)

animateCloud(cloud3.layer)

animateCloud(cloud4.layer)