第一节：View Animations

首先学习UIKit中的关于动画的API，这些特殊的API可以让你很容易实现视图的动画,同时可以避免使用复杂的Core Animation。UIKit 提供了大量的灵活的API以方便使用，基本可以实现你的需求。

你可以让任何继承自UIView的对象动起来，包括UILabel、UIImageVIew、UIbutton与任何自定义的视图。

主要学的的内容：

1、开始学习动画：包括移动、按比例缩放、淡入淡出等，使用基础的UIkit/APIs创建不同类型的动画；

2、关于弹性运动：创建一个基于直线的动画的概念和使用动画创建更吸引眼球的效果。

创建新工程，使用sizeclass，拖入label/button/textfield,;在viewwillapper中添加

tilteLab.frame.origin.x -= view.frame.width

userTF.frame.origin.x -= view.frame.width

passwordTF.frame.origin.x -= view.frame.width

在viewDidAppear(): 添加

UIView.animateWithDuration(0.5) { () -> Void in

self.tilteLab.frame.origin.x += self.view.frame.width

}

此次添加userLab/passwordLab的动画效果；运行之后会发现空间从左至右实现动画。

对于这个类方法中的几个参数：

1、duration：动画持续的时间；

2、delay:在动画开始之前的等待时间；

3、options:可以定制，需要学习相关知识，现在可以设置为【】,代表为没有；

4、Animation：闭包中实现你需求的代码；

5、completion：在这里可以扩展完成动画后的功能，这个参数通常方便当你想要执行一些最后的清理任务或链接另外一个动画。

视图的动画属性。

**位置和大小**

* bounds：改变视图内容的位置和尺寸大小的属性。
* frame：改变视图的位置和尺寸大小的属性。
* center：改变视图位置的属性。

**外观**

* backgroundColor：改变背景色时，UIKit会线性的从原始颜色转变为目标颜色。
* alpha：改变透明度，UIKit会创建淡入淡出的效果。

**转换**

transform属性的类型为CGAffineTransform，它是一个结构体，CoreGraphics中有若干方法可生成不同的CGAffineTransform结构，使视图旋转、按比例缩放、翻转等，我们来看看它如何使用。在viewDidAppear()方法中添加如下代码：

**let** rotation **=** CGAffineTransformMakeRotation(CGFloat(M\_PI))

UIView.animateWithDuration(1, **animations**: {

self.sun.transform **=** rotation

})

首先创建了一个旋转的结构，参数是一个CGFloat类型的角度，这里我们使用预定义好的常量比如M\_PI代表3.14...，也就是旋转一周、M\_PI\_2代表1.57...，也就是旋转半周等。

**动画选项**

animationWithDuration(\_:delay:options:animations:completion:)方法，其中的options当时没有详细的讲述，这节会向大家说明该属性。options选项可以使你自定义让UIKit如何创建你的动画。该属性需要一个或多个UIAnimationOptions枚举类型，让我们来看看都有哪些动画选项吧。

**重复类**

* .Repeat：该属性可以使你的动画永远重复的运行。

.Autoreverse：该属性可以使你的动画当运行结束后按照相反的行为继续运行回去。该属性只能和.Repeat属性组合使用。

**动画缓冲**

在现实生活中，几乎没有什么东西可以突然开始运动，然后突然停止一动不动。可以运动的物体基本都是以较慢的速度启动，逐渐加速，达到一个稳定的速度，然后当要停止时，会逐渐减速，最后停止。所以要使动画更加逼真，也可以采用这种方式，那就是**ease-in**和**ease-out**。

* .CurveLinear ：该属性既不会使动画加速也不会使动画减速，只是做以线性运动。
* .CurveEaseIn：该属性使动画在开始时加速运行。
* .CurveEaseOut：该属性使动画在结束时减速运行。

.CurveEaseInOut：该属性结合了上述两种情况，使动画在开始时加速，在结束时减速。

UIView Animation – Spring

让视图在一次动画中的运动轨迹像弹簧一样，有多次不同方向的运动，最后停止在终点：

UIView.animateWithDuration(1, **delay**: 0.5, **usingSpringWithDamping**: 0.5, **initialSpringVelocity**: 0, **options**: .AllowUserInteraction, **animations**: {

self.loginButton.center.y **-=** 30

self.loginButton.alpha **=** 1

}, **completion**: nil)

* usingSpringWithDamping：弹簧动画的阻尼值，也就是相当于摩擦力的大小，该属性的值从0.0到1.0之间，越靠近0，阻尼越小，弹动的幅度越大，反之阻尼越大，弹动的幅度越小，如果大道一定程度，会出现弹不动的情况。

initialSpringVelocity：弹簧动画的速率，或者说是动力。值越小弹簧的动力越小，弹簧拉伸的幅度越小，反之动力越大，弹簧拉伸的幅度越大。这里需要注意的是，如果设置为0，表示忽略该属性，由动画持续时间和阻尼计算动画的效果。

当usingSpringWithDamping属性值为0.1时，表示阻尼很小，虽然没有动力因素的影响，但登录按钮弹动的幅度依然比较大，相当于在冰面滑行一样。当该属性为1时，表示阻尼非常大，可以看到登录按钮几乎是没有什么弹动的幅度。这就是阻尼的效果。