# 动画、音频和视频

如今的手机不再像以前以发短信打电话为主。随着移动互联网的发展，手机越来越偏向娱乐化，播放音乐、视频，用手机拍照都是现代手机不可或缺的功能。

iOS设备对于多媒体的支持非常强大，多套API来支持音频视频的播放、录制，还有对麦克风和摄像头的操作。

本章将要介绍如何制作动画、播放音频和视频。

## 10.1动画的制作

动画效果是提高用户体验的关键因素。在iOS系统中，其用户界面有很多细微精美的动画，这使得应用的体验感更具有吸引力，更有动态性。适当的动画效果会是程序在传达状态，提供反馈方面更有体验感，让人们感到其操作更加可视化，体验更好。

Core Animation和UIKit是iOS中实现动画效果的主要技术。它们提供了丰富、简单的API，使得开发者可以用很简单的代码就实现酷炫的动画效果。接下来将会对UIKit中的动画进行一些讲解。

### 10.1.1 UIView的动画

在UIKit中，很多API都可以看到animated参数，表示是否动画显示，其实这是UIKit封装CoreAnimation后的结果。UIKit主要API散落在UIView+UIViewAnimationWithBlocks和UIView+UIViewKeyframeAnimations两个分类..使用UIView动画可以通过修改以下UIView视图来产生动画效果：

a、大小变化(frame)  
 b、拉伸变化(bounds)  
 c、中心位置(center)  
 d、旋转(transform)  
 e、透明度(alpha)  
 f、背景颜色(backgroundColor)  
 g、拉伸内容(contentStretch)

UIKit可以将动画集成到UIView类中，实现简单动画的创建过程。UIView类中定义了几个内在支持动画的属性声明，视图根据属性变化提供内建的动画支持。也就是说，开发者可以通过改变上述动画属性和定义一些相关属性，来设置动画效果。

UIView有都中实现动画效果的方式：UIView自带的动画、UIView的block动画、关键帧动画。

UIView的block动画

UIView的block动画的设置是相对来说比较简单的一种。创建一个新项目，模板为【Single View App】，在其ViewController.swift中编写代码。

1. import UIKit
2. class ViewController: UIViewController {
3. override func viewDidLoad() {
4. super.viewDidLoad()
5. // Do any additional setup after loading the view, typically from a nib.
6. //创建视图，颜色为黑色，并添加该视图到根视图
7. let rect = CGRect(x: 50, y: 80, width: 314, height: 240)
8. let view = UIView(frame: rect)
9. view.backgroundColor = UIColor.black
10. self.view.addSubview(view)
12. //设置执行动画的函数viewChange方法
13. func viewChange(){
14. view.bounds = CGRect(x: 0, y: 0, width: 114, height: 100)
15. view.alpha = 0.2
16. view.transform = view.transform.rotated(by: CGFloat(Double.pi))
17. }
19. //设置动画播放速度曲线类型
20. UIView.setAnimationCurve(.easeIn)
21. //设置动画时长，动画效果按照viewChange中的来执行
22. UIView.animate(withDuration: 5,delay: 1, options:.curveEaseIn, animations: viewChange)
23. }
25. override func didReceiveMemoryWarning() {
26. super.didReceiveMemoryWarning()
27. // Dispose of any resources that can be recreated.
28. }
29. }

其中第13~16行中设置视图大小的缩小，透明度的变换和旋转。

第22行使用了UIView.animate()方法来设置动画。对于该函数，其参数及说明如下表：

|  |  |
| --- | --- |
| UIView.animate()参数 | 说明 |
| withduration: TimeInterval | 动画执行时间 |
| dealy: TimeInterval | 动画延迟执行时间 |
| uisngSpringWithDamping: CGfloat | 弹簧阻力，取值范围为 0.0~1.0，数值越小“弹簧”振动效果越明显。 |
| initialSpringVelocity: CGFloat | 动画初始速度（pt/s）,数值越大初始速度越大。但要注意初始速度太大而动画时间太小时，会发生反弹情况。 |
| options: UIViewAnimationOptions | 动画播放速度曲线 |
| animations:（）-> Viod | 执行动画的函数，是本动画的核心 |
| conpletion:((Bool) -> Void)? | 动画之行结束的回调，可选性，可设为nil |

其中对动画最简单的初始化方式如下：

UIView.animate(withDuration: 3, animations: viewChange)

对于动画播放速度曲线，其类型见下表：

表 UIView的block动画曲线类型

|  |  |
| --- | --- |
| 动画曲线类型 | 说明 |
| curveEaseInOut | 动画播放速度为开始和结尾慢中间快 |
| curveEaseIn | 动画播放速度为开始慢，之后逐渐加速 |
| curveEaseOut | 动画播放速度为开始快，之后逐渐减速 |
| curveLinear | 动画播放速度不变 |

运行项目，结果如图所示。可以看到，动画的执行结果和viewChange()中设置的视图属性一样：旋转360°，尺寸逐渐收缩和透明度逐渐减小。

UIView自带的动画

把第12~22行代码改为如下所示：

1. //配置动画参数
2. UIView.beginAnimations(nil, context: nil)
3. UIView.setAnimationDuration(3)
4. UIView.setAnimationDelay(1)
5. UIView.setAnimationCurve(.easeIn)
6. UIView.setAnimationBeginsFromCurrentState(true)
8. //设置视图相关参数
9. view.bounds = CGRect(x: 0, y: 0, width: 114, height: 100)
10. view.alpha = 0.2
11. view.transform = view.transform.rotated(by: CGFloat(Double.pi))
13. UIView.commitAnimations()

这里通过UIView的类方法beginAnimations来开始配置动画参数。此方法会启动动画，但不会立即执行动画，直到调用UIView中的类方法commitAnimations。视图对象执行介于beginAnimations和commitAnimations方法之间的操作。commitAnimation方法执行后，动画才会开始播放。

第16行中设置动画播放速度曲线与block动画表示方式有所不同，但效果相同，UIView动画中播放速度曲线类型包括：easeIn、easeOut、easeInOut、linear。第17行设置动画开始播放时的状态为当前状态。

运行项目，结果如图所示。

### 10.1.2 使用UIImageView播放帧动画

帧动画即一张一张播放的动画。使用UIImageView可以存储一个UIImage数据类型的图片数组，然后根据设置好的图片序列来播放图片制作成帧动画。但是用这种方法播放动画徐注意以下两点：

1. 图片序列中的图片需要有相同的尺寸，否则会产生意料之外的动画效果。
2. 每个图片的scale属性的值（即缩放比例）要相同。

。创建一个新项目，模板为【Single View App】，在Xcode界面左侧的项目导航区打开Assets.xcassets,导入要生成动画的图片序列，如图所示。

打开ViewController.swift文件，在其中编写代码如下。

1. import UIKit
2. class ViewController: UIViewController {
3. override func viewDidLoad() {
4. super.viewDidLoad()
5. // Do any additional setup after loading the view, typically from a nib.
6. //创建UIImage类型数组，并把图片按序列添加到数组中
7. var theImage = [UIImage]()
8. for i in 1...3{
9. theImage.append(UIImage(named: "pic\(i)")!)
10. }
12. //创建图片视图
13. let theImageView = UIImageView(frame: CGRect(x: 50, y: 150, width: 314, height: 314))
14. //设置图片视图对象为theImage图像数组
15. theImageView.animationImages = theImage
16. //设置动画时间
17. theImageView.animationDuration = 3
18. //设置重复播放s次数，0表示不限次数
19. theImageView.animationRepeatCount = 0
20. //开始动画播放
21. theImageView.startAnimating()
23. //将图片视图添加到根视图中
24. self.view.addSubview(theImageView)
25. }
27. override func didReceiveMemoryWarning() {
28. super.didReceiveMemoryWarning()
29. // Dispose of any resources that can be recreated.
30. }
31. }

运行该项目，结果如图所示。