# 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。

## 13.1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。

### 13.1.1 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：

表1 UIView.animate参数

|  |  |
| --- | --- |
| 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2 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。

### 13.1.2 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请注意以下两点：

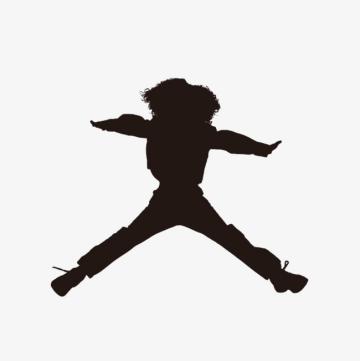
1. 图片序列中的图片需要有相同的尺寸，否则会产生意料之外的动画效果。
2. 每个图片的scale属性的值（即缩放比例）要相同。

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。

## 13.2 音频的播放

音频播放从形式上可分为音效播放和音乐播放。音效主要指通常作为点缀的音频，一般比较短暂。这类音频不需要进度控制和循环控制等。音乐指的是较长的音频，对于这类音频需要对其在音量、进度等方面有精确地控制。

在iOS中对音频的播放方式有4种：System Sound Services 、AVAudioPlayer、Audio Queue Services和Open AL技术。这四种音频播放方式各有其特点：

System Sound Services是最底层、最简单的音频播放服务，其特点为：1.限制：用于播放不超过30秒的声音。2.支持格式：CAF、AIF和使用PCM或IMA/ADPCM数据的WAV文件。3.其他：没有提供操纵声音和控制音量的功能。功能有：1、声音：播放声音文件。如果手机被设置为静音，用户什么也听不到。2、提醒：播放一个声音文件且震动，如果手机被设置为静音或震动，将通过震动提醒用户。3、震动：手机震动。

AVAudioPlayer，其功能类似于System Sound Service，但是比System Sound Service强大的多，她可以实现：1、播放任意时长的音频文件。2、播放文件中或者内存缓冲区的音频文件。单它只能播放一个指定路径的音频，如果想要播放多个路径的音频就要创建多个AVAudioPlayer。3、可控制音频播放的音量、进度，可以多个音频同时播放、支持循环播放。

Audio Queue Service可以实现对声音的完全控制。开发者可以将音频数据从文件中读取到内存缓冲区，并对声音数据进行特殊处理，如改变音频播放的速率、改变其声音的音色等。

Open AL是一个跨平台的开源音频处理接口，其设计给多通道三维位置音效的特效表现，使得它特别适合具有复杂音频使用场景的游戏开发。

### 13.2.1 System Sound Services的使用

在设置一个自定义的警告音或者消息提示音时，System Sound Services无疑是最好的选择。它比其他方法更节省资源。

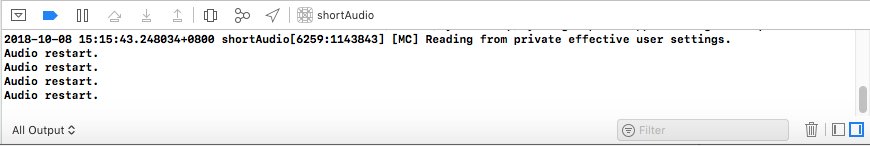
System Sound Services本身无法进行音频的循环播放，但是它可以通过其AudioServices- -AddSystemComplection方法对音频进行监听，在音频播放完后可以重新让音频播放，以此来实现音频的循环播放。接下来将使用System Sound Services重复播放一个短暂的铃声。

使用xcode新创建一个名为“shortVideo“的项目，项目模板为Single View App，创建完成后在项目名称文件夹上单机鼠标右键，选择右键菜单中的Add Files To，向项目中导入一个名为Hint、格式为mp3的音频文件，如图所示。

在左侧项目导航区打开ViewController.swift文件，在其中编辑代码如下：

1. import UIKit
2. //使用System Sound Services要导入AudioToolbox框架
3. import AudioToolbox
4. class ViewController: UIViewController {
5. override func viewDidLoad() {
6. super.viewDidLoad()
7. // Do any additional setup after loading the view, typically from a nib.
9. var \_soundId:SystemSoundID = 0
10. //获得音频文件路径
11. let path = Bundle.main.path(forResource: "Hint", ofType: "mp3")
12. //将字符格式的文件路径转换为URL路径
13. let soundURL = URL(fileURLWithPath: path!)
14. //加载指定路径的音频文件，并创建一个System sound对象
15. AudioServicesCreateSystemSoundID(soundURL as CFURL, &\_soundId)
16. //监听音频播放状态，实现c循环播放
17. AudioServicesAddSystemSoundCompletion(\_soundId, nil, nil, {(soundID,clientData)->Void in
18. print("Audio restart.")
19. AudioServicesPlaySystemSound(soundID)
20. }, nil)
21. //开始播放音频
22. AudioServicesPlaySystemSound(\_soundId)
23. }
24. }

运行该项目会循环播放音频，并且每次音频重新播放，都会在日志区输出一条语句，如图所示。



### 13.2.2 使用AVAudioPlayer播放音乐

AVAudioPlayer支持对音频播放进度、音量大小和播放速率等的控制，通过遵循AVAudioPlayerDelegate协议实现对音频播放状态的监测。如音频文件是否处于播放状态、音频播放速率大小。其支持的音频文件格式如下。

1. AAC
2. AMR (Adaptive multi-Rate，一种语音格式)
3. ALAC (Apple lossless Audio Codec)
4. iLBC (internet Low Bitrate Codec，另一种语音格式)
5. IMA4 (IMA/ADPCM)
6. linearPCM (uncompressed)
7. u-law 和 a-law
8. MP3 (MPEG-Laudio Layer 3)

AVAudioPlayer常用属性说明如下表。

表3 AVAudioPlayer常用属性

|  |  |
| --- | --- |
| 属性名称 | 属性说明 |
| volume | 播放器的音频音量，值：0.0～1.0 |
| pan | 立体声设置 设为 －1.0 则左边播放 设为 0.0 则中央播放 设为 1.0 则右边播放 |
| isPlayer | 布尔类型，表示audio player是否处于播放状态 |
| rate | audio player播放视频的速度，范围在0.5~2.0之间，其中1.0表示正常播放速率 |
| numberOfLoops | audio player循环播放的次数，-1表示无限循环播放 |
| numberOfChannles | audio player的声道数 |
| duration | audio player音频的总长度，单位为秒 |
| currentTime | audio player当前播放位置的时间点，单位为秒 |

AVAudioPlayer常用方法说明如下表。

表4 AVAudioPlayer常用方法列表

|  |  |
| --- | --- |
| init(contentsOf:URL) | 从指定路径上加载并播放音频文件 |
| player（） | 以异步的方式播放一条音频 |
| palyer(atTime:TimeInterval) | 从指定的位置开始音频文件的播放 |
| pause（） | 暂停音频的播放 |
| stop（） | 停止音频的播放 |
| prepareToPlay | 将音频文件加载到缓冲区，为其播放做准备 |
| init（data：Data） | 从内存缓冲区加载并播放音频文件 |
| averagePower（forChannle：Int） | 获得正在播放的音频指定声道的平均功率，单位分贝 |
| peakPower（forChannle：Int） | 获得正在播放音频指定声道的峰值功率，单位分贝 |
| updateMeters（） | 更新正在播放音频的所有声道的平均和峰值功率数值 |

使用Xcode新创建一个名为“musicOpen“的项目，项目模板为Single View App，创建完成后在项目名称文件夹上单机鼠标右键，选择右键菜单中的Add Files To，向项目中导入2个格式为mp3的音频文件，如图所示。

在左侧项目导航区打开ViewController.swift文件，在其中编辑代码如下：

1. import UIKit
2. //y使用AVAudioPlayer要引入AVFoundation框架
3. import AVFoundation
4. //给当前类文件添加AVAudioPlayerDelegate协议来实现对音频播放事件的监听
5. class ViewController: UIViewController ,AVAudioPlayerDelegate{
6. var audioPlayer:AVAudioPlayer = AVAudioPlayer()
7. var audioPlayer1:AVAudioPlayer = AVAudioPlayer()
8. override func viewDidLoad() {
9. super.viewDidLoad()
10. // Do any additional setup after loading the view, typically from a nib.
12. let width = UIScreen.main.bounds.width
13. //获取这2个音频文件的路径，将字符格式的文件路径转换为URL路径
14. let path1 = Bundle.main.path(forResource: "Hotei-Battle Without Honour", ofType: "mp3")
15. let path2 = Bundle.main.path(forResource: "Tobu[7obu]&Itro-Sunburst", ofType: "mp3")
16. let soundUrl = URL(fileURLWithPath: path1!)
17. let soundUrl1 = URL(fileURLWithPath: path2!)
19. //实现对这两个音频文件的初始化操作，并加载音频文件
20. do{
21. try audioPlayer = AVAudioPlayer(contentsOf: soundUrl)
22. audioPlayer.volume = 0.5
23. audioPlayer.numberOfLoops = -1
24. audioPlayer.delegate = self
25. }
26. catch{
27. print(error)
28. }
29. do{
30. try audioPlayer1 = AVAudioPlayer(contentsOf: soundUrl1)
31. audioPlayer1.volume = 0.5
32. audioPlayer1.numberOfLoops = -1
33. audioPlayer1.delegate = self
34. }
35. catch{
36. print(error)
37. }
39. //创建一个按钮，实现音频播放的暂停操作
40. let stopMusic = UIButton(frame: CGRect(x: 40, y: 80, width: width - 80, height: 44))
41. stopMusic.backgroundColor = UIColor.purple
42. stopMusic.setTitle("stop music", for: UIControlState.init(rawValue: 0))
43. stopMusic.addTarget(self, action: #selector(ViewController.pauseMusic), for: .touchUpInside)
44. self.view.addSubview(stopMusic)
45. //创建一个按钮，实现音频的切换播放
46. let changeMusic = UIButton(frame: CGRect(x: 40, y: 200, width: width - 80, height: 44))
47. changeMusic.backgroundColor = UIColor.purple
48. changeMusic.setTitle("change music", for: UIControlState.init(rawValue: 0))
49. changeMusic.addTarget(self, action: #selector(ViewController.changeMusic), for: .touchUpInside)
50. self.view.addSubview(changeMusic)
51. //创建一个按钮，实现开始音频播放和让暂停的音频播放的操作
52. let startMusic = UIButton(frame: CGRect(x: 40, y: 320, width: width - 80, height: 44))
53. startMusic.backgroundColor = UIColor.purple
54. startMusic.setTitle("start music", for: UIControlState.init(rawValue: 0))
55. startMusic.addTarget(self, action: #selector(ViewController.startMusic), for: .touchUpInside)
56. self.view.addSubview(startMusic)
57. }
59. var player1IsPause = false
60. var playerIsPause = false
61. //以下函数是对以上三个按钮操作的具体实现
62. @objc func pauseMusic(){
63. if self.audioPlayer.isPlaying || self.audioPlayer1.isPlaying{
64. if audioPlayer1.isPlaying{
65. self.audioPlayer1.pause()
66. player1IsPause = true
67. }else{
68. self.audioPlayer.pause()
69. playerIsPause = true
70. }
72. }
73. }
75. @objc func changeMusic(){
76. if self.audioPlayer.isPlaying || playerIsPause{
77. playerIsPause = false
78. self.audioPlayer.stop()
79. self.audioPlayer.currentTime = 0
80. self.audioPlayer1.play()
81. }
82. else{
83. player1IsPause = false
84. self.audioPlayer.play()
85. self.audioPlayer1.currentTime = 0
86. self.audioPlayer1.stop()
87. }
88. }
90. @objc func startMusic(){
91. if playerIsPause || player1IsPause{
92. if player1IsPause{
93. self.audioPlayer1.play()
94. player1IsPause = false
95. }else{
96. self.audioPlayer.play()
97. playerIsPause = false
98. }
99. }
100. else{
101. self.audioPlayer.play()
102. }
103. }
104. }

运行项目，模拟器界面如下。点击start music按钮开始播放音频，点击stop music按钮暂停播放音频，再次点击start music按钮暂停的音频继续播放，点击change music按钮切换音频并且播放它。

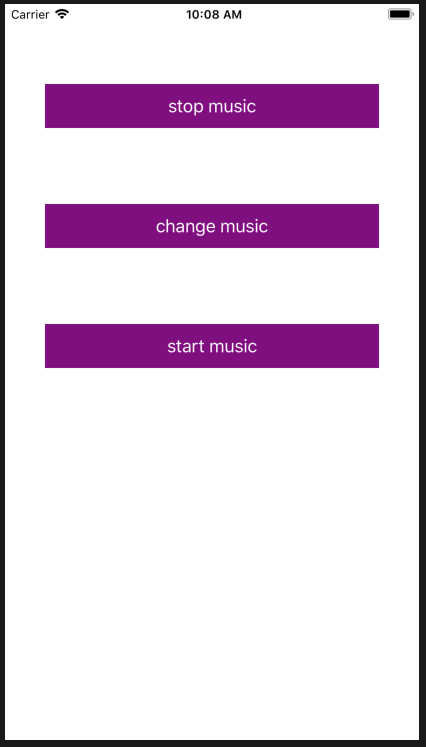


图 音频播放模拟器界面

## 13.3 视频的播放

iOS开发中播放视频一般以以下两种方式播放：MPMoviePlayerController和AVPlayer。MPMoviePlayerController是iOS9.0版本之前的方式，这种视频播放方式比AVPlayer更简单，但是灵活上缺失，而AVPlayer可以高度自定义。所以如今大多数的视频播放都是用AVPlayer。接下来将会对AVPlayer的使用进行讲解，MPMoviePlayerController不再讲解说明，感兴趣的可以在网上查看。

### 13.3.1 使用AVPlayer播放视频

使用AVPlayer需要导入AVFoundation框架。AVPlayer是用来播放基于时间的视听媒体的控制器，但只是对播放和资源时间相关信息的管理，具体的用户界面需要开发者自定义。AVPlayer本身是不可见的，如果想让其可见，需要使用AVPlayerLayer类。简单的视频播放只需创建AVPlayer视频操作对象，将该对象添加到一个AVPlayerLayer类实例上，然后将该AVPlayerLayer类实例添加到当前视图控制器的根视图的子层中。

使用Xcode新创建一个名为“movieBegin“的项目，项目模板为Single View App，创建完成后在项目名称文件夹上单机鼠标右键，选择右键菜单中的Add Files To，向项目中导入1个格式为mp3的音频文件，如图所示。

在左侧项目导航区打开ViewController.swift文件，在其中编辑代码如下：

1. import UIKit
2. //需要导入AVFoundation框架
3. import  AVFoundation
4. class ViewController: UIViewController {
5. override func viewDidLoad() {
6. super.viewDidLoad()
7. // Do any additional setup after loading the view, typically from a nib.
8. //e得到文件的URL
9. let moviePath = Bundle.main.path(forResource: "01", ofType: "m4v")
10. let movieURL = URL(fileURLWithPath: moviePath!)
11. //创建AVPlayer对象
12. let avPlayer = AVPlayer(url: movieURL as URL)
13. //将AVPlayer对象加入到AVPlayerLayer层中
14. let avPlayerLayer = AVPlayerLayer(player:avPlayer)
15. //设置层范围大小和视频播放尺寸
16. avPlayerLayer.frame = self.view.bounds
17. avPlayerLayer.videoGravity = AVLayerVideoGravity.resizeAspect
18. //加入到当前视图控制器根视图层中
19. self.view.layer.addSublayer(avPlayerLayer)
20. //播放
21. avPlayer.play()
22. }
23. }

运行项目，模拟器界面如下。



图 视频播放模拟器界面

其中代码的17行设置了VideoGravity属性，它有三个选项，对应的作用如下：

表5 AVPlayerlayer的VideoGravity属性类型

|  |  |
| --- | --- |
| 方法名称 | 说明 |
| AvPlayerVideoGravityResizeAspectFill | 保留视频的宽高比，对视频进行缩放，将其填满层的区域范围 |
| AvPlayerVideoGravityResizeAspect | 保留视频的宽高比，对视频进行缩放，在视频层完整显示 |
| AvPlayerVideoGravityResize | 将视频进行拉伸来匹配层的显示区域 |

在ViewControoler中有一个AVPlayerViewController子类，不仅可以实现视频播放，还提供了额外的视频播放控制界面。使用它需要引入AVKit框架。