上一章

6.9 为游戏添加声效和背景音乐

本节，我们将为游戏添加声音效果和触控反馈功能，这两个功能在游戏类应用中是非常常见的，因为任何一个成功的游戏都离不开精美的画面和动人的音效。

在奇妙水果机应用中，我们先来添加一些音效和背景音乐。

在Helpers文件夹中创建一个Swift文件，将其命名为PlaySound。直接修改代码如下。

```
import AVFoundation

var audioPlayer: AVAudioPlayer?

func playSound(sound: String, type: String) {
  if let path = Bundle.main.path(forResource: sound, ofType: type) {
    do {
      audioPlayer = try AVAudioPlayer(contentsOf: URL(fileURLWithPath: path))
      audioPlayer?.play()
    } catch {
      print("错误：无法找到文件并播放该声效文件！")
    }
  }
}
```

在PlaySound文件中，我们创建了一个函数playSound()，利用它就可以在项目中任何需要的地方播放声音了。其中，大部分的代码我们之前都见过，AVFoundation框架适用于处理音频素材、控制摄像头，以及配置系统音频的交互等。该框架涵盖音频的捕获、处理、合成、控制、导入和导出几个任务。

playSound()函数包含两个参数，其中，sound参数代表音频文件名称，type参数代表音频文件类型（例如mp3、acc、wav等）。我们先通过Bundle类获取音频文件的路径path，然后通过**do-try-catch**形式来处理可能发生的错误。在**do**闭包中，我们将path作为参数初始化一个AVAudioPlayer类型的对象，并利用play()方法直接播放该声音。

需要注意的是，在初始化AVAudioPlayer类型的对象时，前面加上了**try**关键字，在初始化该类型对象的时候，如果发生错误（音频文件名或格式错误）则会抛出异常（**throws**），而被**do**定义的闭包中一旦抛出异常，就会执行catch闭包中的代码，从而防止程序莫名其妙地崩溃。目前在**catch**闭包中，我们会在控制台中输出一行文本信息。如果一切正常，则会播放指定的音频文件。

接下来，我们需要在几个地方调用playSound()方法。在玩家单击拉杆按钮的时候需要播放spin音效。

```
// MARK: - 拉杆操作
func spinReels() {
  reels = reels.map { _ in
    Int.random(in: 0...symbols.count - 1)
  }
  playSound(sound: "spin", type: "mp3")
}
```

当玩家赢得一局游戏的时候，播放win音效。

```
// MARK: - 检测是否赢得一局游戏
func checkWinning() {
  if reels[0] == reels[1] && reels[1] == reels[2] {
    // 赢得一局游戏
    playerWins()
    // 新的高分记录
    if coins > highScore {
      newHighScore()
    } else {
      playSound(sound: "win", type: "mp3")
    }
  } else {
    // 输掉一局游戏
    playerLoses()
  }
}
```

当玩家打破最高分记录的时候，播放high-score音效。

```
func newHighScore() {
  highScore = coins
  UserDefaults.standard.set(highScore, forKey: "HighScore")
  playSound(sound: "high-score", type: "mp3")
}
```

当玩家切换分值的时候，播放change音效。

```
func activate10() {
  coinsAmount = 10
  isActive10 = true
  playSound(sound: "change", type: "mp3")
}

func activate20() {
  coinsAmount = 20
  isActive10 = false
  playSound(sound: "change", type: "mp3")
}
```

当游戏结束的时候，播放game-over音效。

```
// 检测游戏是否结束

 func isGameOver() {
   if coins <= 0 {
     // 呈现弹出窗口
     showingModal = true
     playSound(sound: "game-over", type: "mp3")
   }
 }
```

当玩家重置游戏最高分记录的时候，播放reset音效。

```
// 重置游戏最高分记录
func resetGame() {
  UserDefaults.standard.set(0, forKey: "HighScore")
  highScore = 0
  coins = 100
  activate10()
  playSound(sound: "reset", type: "mp3")
}
```

在添加完这些音效以后，我们可以在模拟器或者真机中进行音效测试。

我们还需要为InfoView添加一个背景音乐，只要玩家点开Info视图就会听到背景音乐，一旦玩家关闭Info视图，背景音乐就会停止播放。

在InfoView的Body部分，为顶级的VStack容器添加onAppear修饰器。

```
//MARK: - Body
var body: some View {
  ……
}
……
.overlay(
  Button(action: {
    // 关闭信息视图页面
    audioPlayer?.stop()
    self.presentationMode.wrappedValue.dismiss()
  }) {
    Image(systemName: "xmark.circle")
      .font(.title)
  }
  .padding(.top, 30)
  .padding(.trailing, 20)
  .accentColor(.secondary)
, alignment: .topTrailing)
.onAppear(){
  playSound(sound: "background", type: "mp3")
}
```

在模拟器中运行游戏，在玩家点开Info视图以后，会播放背景音乐，关闭Info视图则背景音乐停止。

我们还需要为游戏添加触控反馈特性。在ContentView的Properties部分添加一个常量属性。

```
let haptics = UINotificationFeedbackGenerator()
```

然后在spinReels()、activate10()和activate20()三个方法的最后添加如下代码。

```
haptics.notificationOccurred(.success)
```

对于触控反馈的测试，我们必须在真机上才可以完成，你会发现在不经意间就制作了一款不可思议的游戏！

通过本章的学习，我们制作了一个奇妙水果机游戏，该游戏本身的逻辑并不复杂。我们学会了使用UserDefaults存储和读取最高分记录；创建了比较复杂的游戏界面；通过编写代码实现了简单的游戏逻辑；在Swift语言中生成了随机数；在程序项目中添加了音效；利用微动画增加了程序的吸引力，继续巩固了在项目中创建自定义视图修饰器的方法。

下一章