上一章

6.8 为游戏添加动画效果

本节我们将为程序添加必要的动画效果来增强游戏体验。我们一共要添加3个动画效果，第1个是每次拉杆后槽位中水果切换的动画，第2个是分值切换的动画，第3个是消息窗口的动画。

我们先来实现槽位中水果切换的动画效果。

在Properties部分添加一个新的变量。

```
@State private var animatingSymbol = false
```

然后，对3个槽位的Image同时添加下面几个修饰器，我们以槽位1中的Image为例。

```
//MARK: - 槽位 #1
ZStack {
  ReelView()
  Image(symbols[reels[0]])
    .resizable()
    .modifier(ImageModifier())
    .opacity(animatingSymbol ? 1 : 0)
    .offset(y: animatingSymbol ? 0 : -50)
    .animation(.easeOut)
    .onAppear{
      self.animatingSymbol.toggle()
    }
} //: ZStack
```

通过onAppear修饰器，当水果图片出现在屏幕上的时候，改变animatingSymbol变量的值。如果该值由false变为true，则让Image的透明度由0变为1，垂直方向的坐标从-50向上移动50点到0，也就是当前位置，动画方式为easeOut。在预览窗口中启动Live模式，我们可以发现槽位中的水果会以动画的方式呈现，但目前的效果还比较生硬。接下来我们会为三个Image的easeOut动画方式添加duration参数，修改槽位1的animation修饰器如下。

```
.animation(.easeOut(duration: Double.random(in: 0.5...0.7)))
```

继续修改槽位2的animation修饰器如下。

```
.animation(.easeOut(duration: Double.random(in: 0.7...0.9)))
```

最后修改槽位3的animation修饰器如下。

```
.animation(.easeOut(duration: Double.random(in: 0.9...1.1)))
```

这里为每个槽位的水果图片的呈现都设定了动画时长，其中槽位1的图片呈现0.5～0.7s，槽位2的图片呈现0.7～0.9s，槽位3的图片呈现0.9～1.1s。在预览窗口中我们可以启动Live模式查看动画效果。

接下来，我们要实现在用户单击拉杆按钮后变换水果的动画效果。修改拉杆按钮的action参数的闭包代码如下。

```
//MARK: - 拉杆
Button(action: {
  // 设置无动画状态
  withAnimation(
    self.animatingSymbol = false
  )

  // 拉杆操作
  self.spinReels()

  // 设置动画状态
  withAnimation(
    self.animatingSymbol = true
  )

  // 检测是否赢得一局
  self.checkWinning()

  // 检测游戏是否结束
  self.isGameOver()
}, label: {
  ……
}
```

在action闭包中，我们先将animationSymbol的值设置为false，因为水果图片出现在屏幕上的时候，该变量的值已经被切换为true，所以这里必须先将其还原为false。在执行完spinReels()方法后，我们再通过动画方式将该变量设置为true，这样动画就会继续发生。此时，我们可以在预览窗口中启动Live模式，单击拉杆按钮后查看水果切换的动画效果。

现在，我们要实现分值切换的动画效果，修改每局分值为10和分值为20的Image修饰器。

```
//MARK: -分值为20
HStack(alignment: .center, spacing: 10) {
  ……
  Image("钱币")
    .resizable()
    .opacity(isActive10 == false ? 1 : 0)
    .offset(x: isActive10 == false ? 0 : 20)
    .modifier(CoinImageModifier())
}

//MARK: -分值为10
HStack(alignment: .center, spacing: 10) {
  Image("钱币")
    .resizable()
    .opacity(isActive10 == true ? 1 : 0)
    .offset(x: isActive10 == true ? 0 : -20)
    .modifier(CoinImageModifier())
  ……
}
```

当玩家单击分值为10的按钮时，我们让目前分值为10的钱币向左移动20点并消失。当玩家单击分值为10的按钮时，则让目前分值为20的钱币向右移动20点并消失。此时，我们可以在预览窗口中启动Live模式，通过单击不同分值来查看动画效果。

接下来，我们要实现弹出消息窗口的动画。还是需要先在Properties部分添加一个属性。

```
@State private var animatingModal = false
```

然后为消息窗口的VStack容器添加下面几个修饰器。

```
//MARK: - Popup
if $showingModal.wrappedValue {
  ZStack {
    Color("ColorTransparentBlack").edgesIgnoringSafeArea(.all)

    VStack(spacing: 0) {
      ……
    } //: VStack
    .frame(minWidth: 280, idealWidth: 280, maxWidth: 320, minHeight: 260, idealHeight: 280, maxHeight: 320, alignment: .center)
    .background(Color.white)
    .cornerRadius(20)
    .shadow(color: Color("ColorTransparentBlack"), radius: 6, x: 0, y: 8)
    .opacity(animatingModal ? 1 : 0)
    .offset(y: animatingModal ? 0 : -100)
    .animation(Animation.spring(response: 0.6,
                                dampingFraction: 1.0, blendDuration: 1.0))
    .onAppear{
      self.animatingModal = true
    }
  } //: ZStack
} //: IfEnd
```

当游戏结束弹出消息按钮时，我们通过onAppear修饰器让animatingModal变量的值为true，这样就会激活该消息窗口的动画效果。其中包括透明度的变化、窗口位置从上向下移动，动画方式为弹簧效果。

另外，我们需要在“新游戏”按钮的action参数闭包中添加两行代码，这样才能保证下次出现消息窗口的时候继续产生动画效果。

```
Button(action: {
  self.showingModal = false
  self.animatingModal = false
  self.activate10()
  self.coins = 100
}, label: {
  Text("新游戏")
  ……
}
```

在玩家单击“新游戏”按钮后，animatingModal的值被修改回false，并将每局的分值设置为10。

下一章