```
1  using UnityEngine;
2  using UnityEngine.UI;
3  using System.Collections;
4
5  public class GameController : MonoBehaviour
6  {
7      // UI引用
8      public Text playerChoiceTxt;
9      public Text computerChoiceTxt;
10     public Text resultTxt;
11     public Text scoreTxt;
12     public Button confirmBtn;
13
14     private int playerScore = 0;
15     private int computerScore = 0;
16
17     private int? playerSelectedChoice = null;
18     private int computerChoice;
19     private bool roundEnded = true;
20
21     void Start()
22     {
23         UpdateScoreDisplay();
24         confirmBtn.interactable = false;
25         ResetDisplay();
26     }
27
28     // 玩家选择方法
29     public void SetPlayerChoice(int choice)
30     {
31         if (roundEnded) return;
32
33         playerSelectedChoice = choice;
34         playerChoiceTxt.text = GetChoiceName(choice);
35         confirmBtn.interactable = true;
36     }
37
38     // 确认按钮点击
39     public void OnConfirmButtonClick()
40     {
41         if (!playerSelectedChoice.HasValue) return;
42
43         GenerateComputerChoice();
44         CompareChoices();
45         StartCoroutine(ShowResultAndReset()); // 使用协程控制流程
46     }
47
48     void GenerateComputerChoice()
49     {
50         computerChoice = Random.Range(0, 3);
51         computerChoiceTxt.text = GetChoiceName(computerChoice); // 立即显示电脑选择
52     }
53
54     void CompareChoices()
55     {
56         int playerChoice = playerSelectedChoice.Value;
```

```
57
58          if (playerChoice == computerChoice)
59          {
60              resultTxt.text = "平局！";
61              return;
62          }
63
64          int result = (playerChoice - computerChoice + 3) % 3;
65
66          if (result == 1)
67          {
68              resultTxt.text = "你输了！";
69              computerScore++;
70          }
71          else
72          {
73              resultTxt.text = "你赢了！";
74              playerScore++;
75          }
76
77          UpdateScoreDisplay();
78      }
79
80      // 新增协程控制显示流程
81      IEnumerator ShowResultAndReset()
82      {
83          confirmBtn.interactable = false; // 禁用确认按钮
84          roundEnded = true;
85
86          // 保持结果显示2秒
87          yield return new WaitForSeconds(2f);
88
89          ResetDisplay();
90      }
91
92      void ResetDisplay()
93      {
94          playerSelectedChoice = null;
95          playerChoiceTxt.text = "请选择";
96          computerChoiceTxt.text = "???";
97          resultTxt.text = "等待选择...";
98          roundEnded = false; // 开启新回合
99      }
100
101     string GetChoiceName(int choice)
102     {
103         return choice switch
104         {
105             0 => "✊ 石头",
106             1 => "✌ 剪刀",
107             2 => "✋ 布",
108             _ => "未知"
109         };
110     }
111
112     void UpdateScoreDisplay()
```

```
113     {
114         scoreTxt.text = $"玩家: {playerScore} - 电脑: {computerScore}";
115     }
116 
117     public void ResetGame()
118     {
119         playerScore = 0;
120         computerScore = 0;
121         UpdateScoreDisplay();
122         ResetDisplay();
123     }
124 
125     // 原有字段保持不变...
126 
127     [Header("Animation Settings")]
128     public float shuffleDuration = 0.5f;
129     public Vector2 shuffleAreaMin = new Vector2(-200, -100);
130     public Vector2 shuffleAreaMax = new Vector2(200, 100);
131 
132     // 必须使用SerializeField保持私有字段的编辑器可见性
133     [SerializeField] private ButtonAnimator rockBtnAnimator;
134     [SerializeField] private ButtonAnimator scissorsBtnAnimator;
135     [SerializeField] private ButtonAnimator paperBtnAnimator;
136 
137     private IEnumerator ShowResultAndReset()
138     {
139         confirmBtn.interactable = false;
140         roundEnded = true;
141 
142         yield return new WaitForSeconds(2f);
143 
144         // 启动洗牌动画协程
145         yield return StartCoroutine(PlayShuffleAnimation());
146 
147         ResetDisplay();
148     }
149 
150     // 修改为协程方法
151     private IEnumerator PlayShuffleAnimation()
152     {
153         // 生成三个不重叠的位置
154         Vector2[] positions = new Vector2[3];
155         positions[0] = GetValidPosition(Vector2.zero);
156         positions[1] = GetValidPosition(positions[0]);
157         positions[2] = GetValidPosition(positions[0], positions[1]);
158 
159         // 并行执行动画
160         bool[] doneFlags = new bool[3];
161         rockBtnAnimator.PlayShuffleAnimation(positions[0], shuffleDuration, () =>
              doneFlags[0] = true);
162         scissorsBtnAnimator.PlayShuffleAnimation(positions[1], shuffleDuration, () =>
              doneFlags[1] = true);
163         paperBtnAnimator.PlayShuffleAnimation(positions[2], shuffleDuration, () =>
              doneFlags[2] = true);
164 
165         // 等待所有动画完成
```

```
166         yield return new WaitUntil(() => doneFlags[0] && doneFlags[1] && doneFlags
              [2]);
167     }
168
169     private Vector2 GetValidPosition(params Vector2[] existingPositions)
170     {
171         const float minDistance = 80f;
172         Vector2 newPos;
173         bool isValid;
174
175         do
176         {
177             isValid = true;
178             newPos = new Vector2(
179                 Random.Range(shuffleAreaMin.x, shuffleAreaMax.x),
180                 Random.Range(shuffleAreaMin.y, shuffleAreaMax.y)
181             );
182
183             foreach (var pos in existingPositions)
184             {
185                 if (Vector2.Distance(newPos, pos) < minDistance)
186                 {
187                     isValid = false;
188                     break;
189                 }
190             }
191         } while (!isValid);
192
193         return newPos;
194     }
195
196
197 }
```