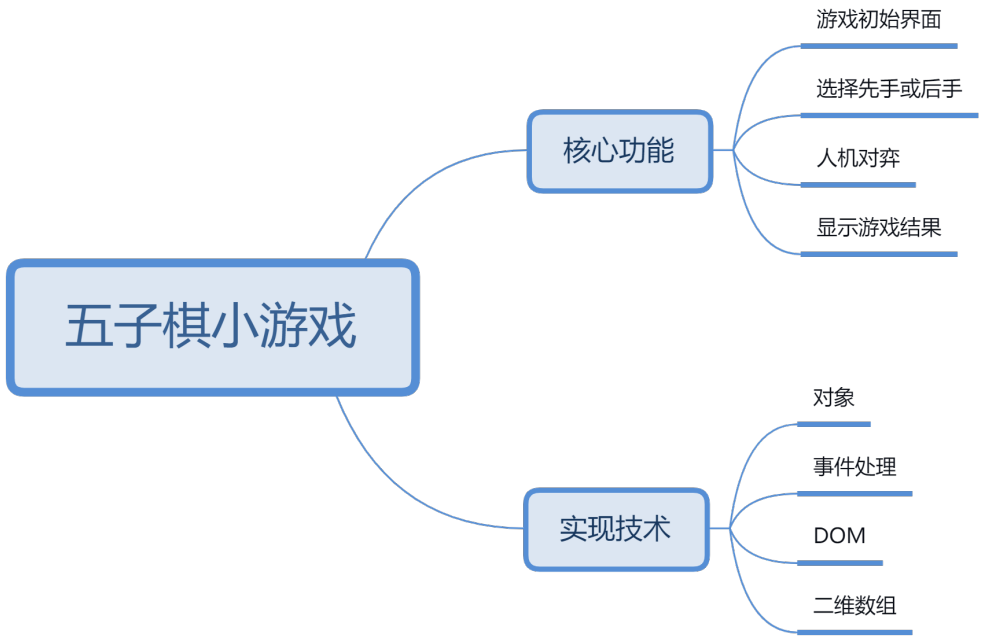
第 4 章

五子棋小游戏

——对象 + 事件处理 + DOM 文档对象模型 + 二维数组

五子棋是一款容易上手、老少皆宜的益智趣味游戏。游戏双方分别使用黑白两色的棋子，下在棋盘横线与竖线的交叉点上，先形成 5 子连线者获胜。本章将使用 JavaScript 中的对象、事件处理、DOM 文档对象模型和二维数组等技术开发五子棋小游戏。

本项目的核心功能及实现技术如下：



4.1 开 发 背 景

随着计算机和互联网技术的飞速发展，计算机游戏已经成为人们休闲娱乐的主要方式之一，而棋类游戏是某些玩家的首选。在棋类游戏中，五子棋游戏吸引着不同年龄段的人群，无论男女老少都可以玩。五子棋作为一项棋类竞技运动，不仅能增强人的思维能力、提高智力，而且富含哲理、有助于修身养性。目前的五子棋程序发展得很快，从最初的双人对战到人机对弈，再到现在的网络对战，该游戏已经受到越来越多人的喜欢和重视。

本章将使用对象、事件处理、DOM 文档对象模型和二维数组来实现一个简单的五子棋人机对弈的小游戏，其实现目标如下：

- ☑ 游戏初始界面。
- ☑ 人机对弈。
- ☑ 游戏结果界面。

4.2 系 统 设 计

4.2.1 开发环境

本项目的开发及运行环境如下：

- ☑ 操作系统：推荐 Windows 10、11 及以上，兼容 Windows 7（SP1）。
- ☑ 开发工具：WebStorm。
- ☑ 开发语言：JavaScript。

4.2.2 业务流程

在运行五子棋小游戏后，首先进入游戏初始界面。玩家选择先手还是后手之后，单击界面中的“开始”按钮开始游戏。玩家或计算机下子后需要判断玩家或计算机是否取胜，如果未取胜则继续下子，如果玩家或计算机取胜，则游戏结束，单击“开始”按钮后可以重新开始游戏。

本项目的业务流程如图 4.1 所示。

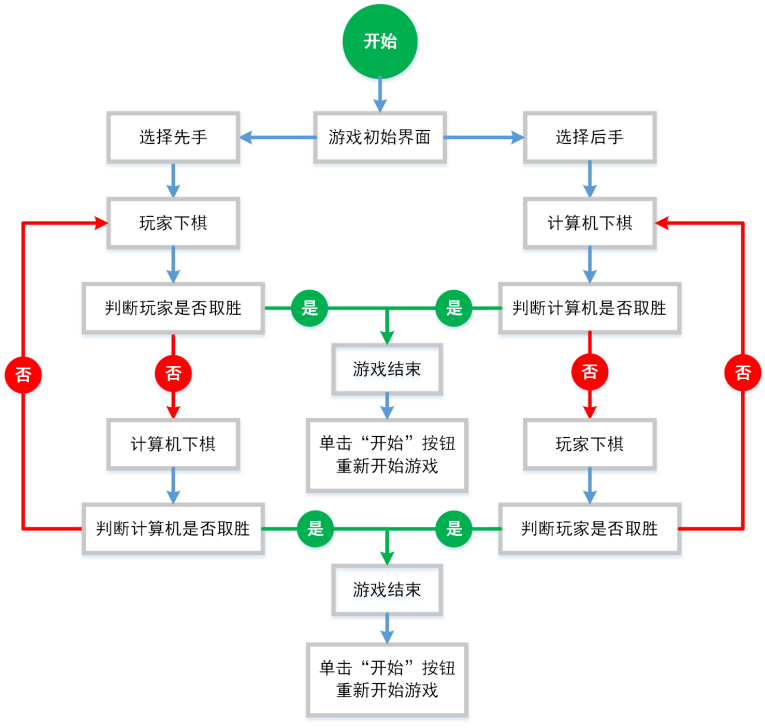


图 4.1 业务流程

4.2.3 功能结构

本项目的功能结构已经在章首页中给出。作为一个五子棋小游戏的应用，本项目实现的具体功能如下：

- ☑ 游戏初始界面：游戏初始界面主要用于显示游戏棋盘和“先手”“后手”“开始”三个按钮。
- ☑ 选择先手还是后手：单击“先手”按钮，玩家先下子，然后计算机下子。单击“后手”按钮，再单击“开始”按钮，计算机先下子，然后玩家下子。
- ☑ 实现人机对弈：玩家单击棋盘中的交叉点开始下子，玩家执黑子，计算机执白子。在游戏过程中可以随时单击“重玩”按钮重新开始游戏。
- ☑ 游戏结果界面：游戏双方先形成 5 子连线者获胜。如果玩家取胜则会显示胜利的图片效果，否则会显示失败的图片效果。

4.3 技 术 准 备

4.3.1 技术概览

在开发五子棋小游戏时应用了 JavaScript 中的对象、事件处理、DOM 文档对象模型，下面将简述本项目所用的这些核心技术点及其具体作用。

1. 对象

对象是 JavaScript 中的基本数据类型之一，是一种复合的数据类型。它将多种数据类型集中在一个数据单元中，并允许通过对象来存取这些数据的值。在 JavaScript 中，对象包含两个要素：属性和方法。通过访问或设置对象的属性，并且调用对象的方法，就可以对对象进行各种操作，从而获得需要的功能。

创建对象的语法格式如下：

```
var 对象名 = {属性名 1:属性值 1,属性名 2:属性值 2,属性名 3:属性值 3...}
```

由语法格式可以看出，在创建对象时，所有属性都放在大括号中，属性之间用逗号分隔，每个属性都由属性名和属性值两部分组成，属性名和属性值之间用冒号隔开。

例如，创建一个商品对象 goods，并设置 3 个属性，分别为 name、price 和 number，代码如下：

```
var goods = {                                                    //创建 goods 对象
    name: "戴尔笔记本电脑",
    price: 3666,
    number: 2
}
```

在创建对象时，也可以定义对象的方法，方法以函数的形式存储在属性中。在定义方法时，通常会使用 this 关键字，this 关键字表示对对象自己的引用。例如，在 goods 对象中定义一个方法 show()，该方法用于输出商品信息，代码如下：

```
var goods = {                                                    //创建 goods 对象
    name: "戴尔笔记本电脑",
    price: 3666,
    number: 2
      show:function(){
            alert("商品名称："+this.name+"\n 商品单价："+this.price+"\n 商品数量："+this.number);  //输出商品信息
      }
}
```

访问对象的方法可以使用“对象名.方法名()”的形式。例如，访问 goods 对象中定义的 show()方法，输出定义的商品信息，代码如下：

```
goods.show();                                                    //访问 show()方法
```

2. 事件处理

事件处理是对象化编程的一个很重要的环节，它可以使程序的逻辑结构更加清晰，以及使程序更具有灵活性，进而提高程序的开发效率。

事件是一些可以通过脚本响应的页面动作。当用户按下鼠标键或者提交一个表单，甚至在页面上移动鼠标时，事件就会出现。事件处理是一段 JavaScript 代码，总是与页面的特定部分以及一定的事件相关联。当与页面特定部分关联的事件发生时，事件处理器就会被调用。

在本项目中，主要应用了 JavaScript 中的 4 个常用事件，分别是 onclick、onmouseenter、onmouseleave 和 onload 事件。

onclick 事件在单击鼠标时触发。单击是指鼠标停留在对象上时按下鼠标键，并在没有移动鼠标的情况下释放鼠标键的这一完整过程。例如，单击页面中的“变换背景”按钮，将页面的背景颜色设置为蓝色。代码如下：

```
<input type="button" id="but" value="变换背景" onclick="turncolors()">
<script>
document.getElementById("but").onclick = function(){
        document.bgColor = "#0000FF";
};
</script>
```

onmouseenter 事件在鼠标指针移动到元素上时触发。onmouseleave 事件在鼠标指针移出元素时触发。例如，在 h1 标题上使用 onmouseenter 和 onmouseleave 事件，当鼠标移动到标题上时将标题颜色设置为红色，当鼠标移出标题时将标题颜色设置为黑色，代码如下：

```
<h1 id="demo">有志者事竟成</h1>
<script>
document.getElementById("demo").onmouseenter = function() {
        mouseEnter()
};
document.getElementById("demo").onmouseleave = function() {
        mouseLeave()
};
function mouseEnter() {
      document.getElementById("demo").style.color = "#FF0000";
}
function mouseLeave() {
      document.getElementById("demo").style.color = "#000000";
}
</script>
```

说明

onmouseenter 事件类似于 onmouseover 事件。唯一的区别是 onmouseenter 事件不支持事件冒泡。

onload 事件在页面或者页面中的某个元素加载完成时触发。例如，页面加载完毕后弹出一个对话框，代码如下：

```
<script type="text/javascript">
      function myFun(){
            alert("页面已加载完成");
```

```
    }
</script>
<body onload="myFun()">
    <h1>欢迎访问本网站</h1>
</body>
```

3. DOM 文档对象模型

DOM 是访问和操作 Web 页面的接口，使用该接口可以访问页面中的其他标准组件。在本项目中，主要应用了 DOM 中的 appendChild()方法、removeChild()方法和 innerHTML 属性。

appendChild()方法用于将新的子节点添加到当前节点的末尾处。appendChild()方法的语法格式如下：

```
obj.appendChild(newChild)
```

例如，创建一个 div 元素，并设置元素的宽度、高度和背景颜色，再将该 div 元素添加到页面中。代码如下：

```
var div = document.createElement("div");                    //创建 div 元素
div.style.width = "100px";                                  //设置宽度
div.style.height = "50px";                                  //设置高度
div.style.backgroundColor = "#0000FF";                      //设置背景颜色
document.body.appendChild(div);                             //向 body 中添加 div 元素
```

removeChild()方法用于删除某个子节点，语法格式如下：

```
obj. removeChild(oldChild)
```

例如，使用 removeChild()方法动态删除页面中指定的文本，代码如下：

```
<div id="di"><p>春眠不觉晓</p><p>处处闻啼鸟</p><p>夜来风雨声</p><p>花落知多少</p></div>
<button type="button" id="but">删除</button>
<script>
    document.getElementById('but').onclick = function(){
        var deleteN=document.getElementById('di');          //获取指定 id 的元素
        if(deleteN.hasChildNodes()){                        //判断是否有子节点
            deleteN.removeChild(deleteN.lastChild);         //删除节点
        }
    }

</script>
```

innerHTML 属性用于设置或获取元素含有的 HTML 文本，不包括元素本身的开始标记和结束标记。通过该属性可以使用指定的 HTML 文本替换元素的内容。

例如，通过 innerHTML 属性设置 标签的内容，代码如下：

```
<body>
<div id="date"></div>
<script type="text/javascript">
      document.getElementById("date").innerHTML="2024-<b>05</b>-26";
</script>
</body>
```

有关对象、事件处理和 DOM 文档对象模型等方面的基础知识在《JavaScript 从入门到精通（第 5 版）》中有详细的讲解，对这些知识不太熟悉的读者可以参考该书对应的内容。下面将对二维数组进行必要介绍，以确保读者可以顺利完成本项目。

4.3.2 二维数组

在 JavaScript 中，二维数组是一种特殊的数组类型。如果数组中的每一个元素也是一个数组，那么该数

组就是一个二维数组。例如，创建一个包含 3 个元素的二维数组，数组中的每个元素也是一个长度为 3 的数组，代码如下：

```
var arr = [
        [1,2,3],
        [4,5,6],
        [7,8,9]
];
```

访问二维数组可以使用“数组名[下标 1][下标 2]”的形式。使用“数组名[下标 1]”可以得到一个一维数组，使用“数组名[下标 1][下标 2]”可以得到一维数组中的元素。

例如，获取二维数组 arr 中第二个数组元素中的第三个元素，代码如下：

```
var arr = [
        [1,2,3],
        [4,5,6],
        [7,8,9]
];
document.write(arr[1][2]);
```

4.4 游戏初始界面设计

该游戏的初始界面包括一个五子棋的棋盘、一个“先手”按钮、一个“后手”按钮和一个“开始”按钮。五子棋的棋盘是由 15 条横线和 15 条竖线垂直交叉而成的。在棋盘上，横竖线交叉形成的 225 个交叉点为对弈时的落子点。初始界面的效果如图 4.2 所示。

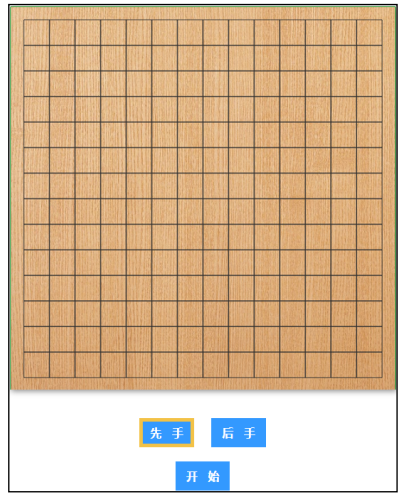


图 4.2 游戏初始界面

4.4.1 创建主页

开发五子棋小游戏需要创建主页文件 index.html，还需要为页面中的元素设置样式。主页的实现步骤如下。

（1）在 index.html 文件中定义一个 id 属性值为 wrapper 的 div 元素，在该元素中添加两个 div 元素，第一个 div 元素用于定义棋盘，在第二个 div 元素中添加 3 个超链接，分别作为游戏的“先手”“后手”和“开始”按钮。代码如下：

```
<div class="wrapper">
        <div class="chessboard" id="chessboard"></div>
        <div class="right">
                <a id="first_btn" class="btn" href="#">先   手</a>
                <a id="back_btn" class="btn" href="#">后   手</a>
                <a id="play_btn" class="btn" href="#">开   始</a>
        </div>
</div>
```

（2）新建 css 文件夹，在文件夹中创建 chess.css 文件，在文件中编写游戏界面的样式，代码如下：

```
.wrapper {
        width: 600px;                                        /*设置宽度*/
```

```
        margin:0 auto;                                              /*设置外边距*/
        position: relative;                                         /*设置相对定位*/
}
div.chessboard {
        margin: 60px auto;                                          /*设置外边距*/
        width: 542px;                                               /*设置宽度*/
        background: url(../images/chessboard.png) no-repeat;        /*设置背景图像*/
        overflow: hidden;                                           /*设置溢出内容隐藏*/
        box-shadow: 2px 2px 8px #888;                               /*设置阴影*/
}
div.chessboard div {
        float: left;                                                /*设置左浮动*/
        width: 36px;                                                /*设置宽度*/
        height: 36px;                                               /*设置高度*/
        border: 0;                                                  /*设置无边框*/
        cursor: pointer;                                            /*设置鼠标形状*/
}
div.chessboard div.black {
        background: url(../images/black.png) no-repeat 4px 4px;     /*设置背景图像*/
}
div.chessboard div.white {
        background: url(../images/white.png) no-repeat 4px 4px;     /*设置背景图像*/
}
div.chessboard div.hover {
        background: url(../images/hover.png) no-repeat 1px 1px;     /*设置背景图像*/
}
div.chessboard div.hover-up {
        background: url(../images/hover_up.png) no-repeat 1px 1px;  /*设置背景图像*/
}
div.chessboard div.hover-down {
        background: url(../images/hover_down.png) no-repeat 1px 1px;/*设置背景图像*/
}
div.chessboard div.hover-up-left {
        background: url(../images/hover_up_left.png) no-repeat 1px 1px;/*设置背景图像*/
}
div.chessboard div.hover-up-right {
        background: url(../images/hover_up_right.png) no-repeat 1px 1px;/*设置背景图像*/
}
div.chessboard div.hover-left {
        background: url(../images/hover_left.png) no-repeat 1px 1px;/*设置背景图像*/
}
div.chessboard div.hover-right {
        background: url(../images/hover_right.png) no-repeat 1px 1px;/*设置背景图像*/
}
div.chessboard div.hover-down-left {
        background: url(../images/hover_down_left.png) no-repeat 1px 1px;/*设置背景图像*/
}
div.chessboard div.hover-down-right {
        background: url(../images/hover_down_right.png) no-repeat 1px 1px;/*设置背景图像*/
}
div.chessboard div.white-last {
        background: url(../images/white_last.png) no-repeat 4px 4px;/*设置背景图像*/
}
div.chessboard div.black-last {
        background: url(../images/black_last.png) no-repeat 4px 4px;/*设置背景图像*/
}
div.right {
        position: absolute;                                         /*设置绝对定位*/
        left: 200px;                                                /*设置元素到父元素左端的距离*/
        top: 570px;                                                 /*设置元素到父元素顶端的距离*/
        width: 200px;                                               /*设置宽度*/
        text-align: center;                                         /*设置文本水平居中显示*/
```

```
}
.right a {
        display: inline-block;                          /*设置元素为行内块元素*/
        margin: 10px;                                   /*设置外边距*/
        padding: 10px 15px;                             /*设置内边距*/
        background: #3399FF;                            /*设置背景颜色*/
        text-decoration: none;                          /*设置无下画线*/
        color: #FFF;                                    /*设置文字颜色*/
        font-weight: bold;                              /*设置字体粗细*/
        font-size: 16px;                                /*设置字体大小*/
        font-family: "微软雅黑";                        /*设置字体*/
}
.right a:hover {
        background: #AACCFF;                            /*设置背景颜色*/
        text-decoration: none;                          /*设置无下画线*/
}
.right a.disable{
        cursor: default;                                /*设置鼠标形状*/
        background: #CCCCCC;                            /*设置背景颜色*/
        color: #777777;                                 /*设置文字颜色*/
}
.right a.selected {
        border: 5px solid #F3C246;                      /*设置边框*/
        padding: 5px 10px;                              /*设置内边距*/
}
#result_tips {
        color: #CE4246;                                 /*设置文字颜色*/
        font-size: 26px;                                /*设置字体大小*/
        font-family: "微软雅黑";                        /*设置字体*/
}
#result_info {
        margin-bottom: 26px;                            /*设置下外边距*/
}
.imgDiv{
        border:1px solid #FF0000;                       /*设置边框*/
        position:absolute;                              /*设置绝对定位*/
        top:0;                                          /*设置元素到父元素顶端距离*/
        z-index:10;                                     /*设置堆叠顺序*/
}
```

（3）在 index.html 文件中引入 chess.css 文件。引入 CSS 文件的代码如下：

```
<link rel="stylesheet" type="text/css" href="css/chess.css">
```

4.4.2 游戏初始化

要生成五子棋的棋盘，需要对游戏进行初始化操作。游戏初始化的实现方法如下。

（1）新建 js 文件夹，在文件夹中创建 chess.js 文件，在文件中编写 JavaScript 代码。创建一个名为 Gobang 的对象，在对象中定义多个属性和初始化棋盘的方法 init()。在 init()方法中分别定义单击“先手”按钮、单击“后手”按钮和单击“开始”按钮执行的操作。代码如下：

```
var Gobang = {
    NO_CHESS: 0,                                        //无棋位置的值
    BLACK_CHESS: -1,                                    //黑棋的值
    WHITE_CHESS: 1,                                     //白棋的值
    chessArr: [],                                       //记录棋子
    chessboardHtml: "",                                 //棋盘 HTML
    playerColor: "black",                               //玩家棋子颜色
```

```
    comColor: "white",                                          //计算机棋子颜色
    isPlayerTurn: true,                                         //是否轮到玩家下棋
    isGameStart: false,                                         //游戏是否已经开始
    isGameOver: false,                                          //游戏是否已经结束
    playerLastChess: [],                                        //玩家最后下子位置
    comLastChess: [],                                           //计算机最后下子位置
    isBack: false,
    init: function () {
        var chessboard = document.getElementById("chessboard");
        //初始化棋盘
        for(var i = 0; i < 15 * 15;i++){
            var div = document.createElement("div");            //创建 div 元素
            chessboard.appendChild(div);                        //添加 div 元素
        }
        this.chessboardHtml = chessboard.innerHTML;             //设置棋盘内容
        var _t = this;
        var firstBtn = document.getElementById("first_btn");
        var backBtn = document.getElementById("back_btn");
        var playBtn = document.getElementById("play_btn");
        firstBtn.classList.add("selected");                     //添加类
        firstBtn.onclick = function(event){
            event.preventDefault();
            if(!_t.isGameStart){
                this.classList.add("selected");                 //添加类
                backBtn.classList.remove("selected");           //移除类
                _t.isBack = false;
            }
        }
        backBtn.onclick = function(event){
            event.preventDefault();
            if(!_t.isGameStart) {
                this.classList.add("selected");                 //添加类
                firstBtn.classList.remove("selected");          //移除类
                _t.isBack = true;
            }
        }
        //单击操作按钮
        playBtn.onclick = function (event) {
            event.preventDefault();
            _t.resetChessBoard();                               //重置棋盘
            if (_t.isGameStart) {                               //如果游戏已经开始
                firstBtn.classList.remove("disable");           //移除类
                backBtn.classList.remove("disable");            //移除类
                _t.gameOver();                                  //游戏结束
            } else {
                firstBtn.classList.add("disable");              //添加类
                backBtn.classList.add("disable");               //添加类
                _t.gameStart();                                 //游戏开始
                if(_t.isBack) _t.comTurn();                     //如果选择后手，就让计算机先下子
            }
        };
        this.resetChessBoard();                                 //重置棋盘
    }
}
```

（2）当页面加载完成后，调用 Gobang 对象中的 init()方法对游戏进行初始化。代码如下：

```
window.onload = function() {
    Gobang.init();                                              //游戏初始化
};
```

4.5 实现人机对弈

在开始游戏时，玩家执黑棋，计算机执白棋。棋子只能下在棋盘中的空白交叉点上。如果选择玩家先手，则玩家先下子，计算机后下子，效果如图 4.3 所示；如果选择玩家后手，则计算机先下子，玩家后下子，效果如图 4.4 所示。

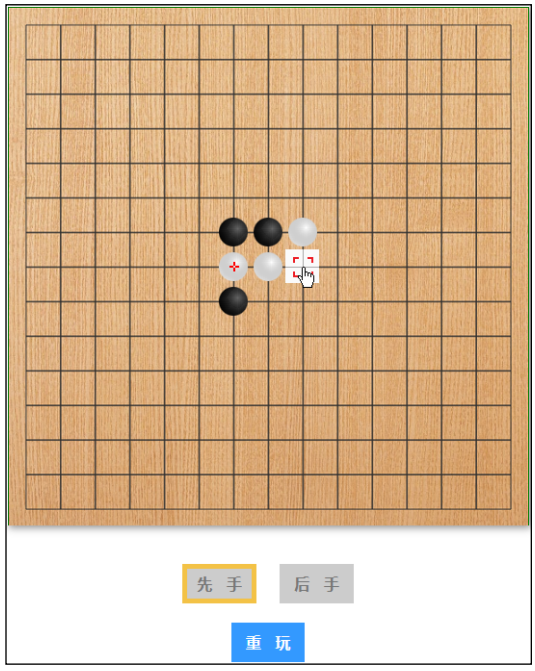


图 4.3 玩家先手

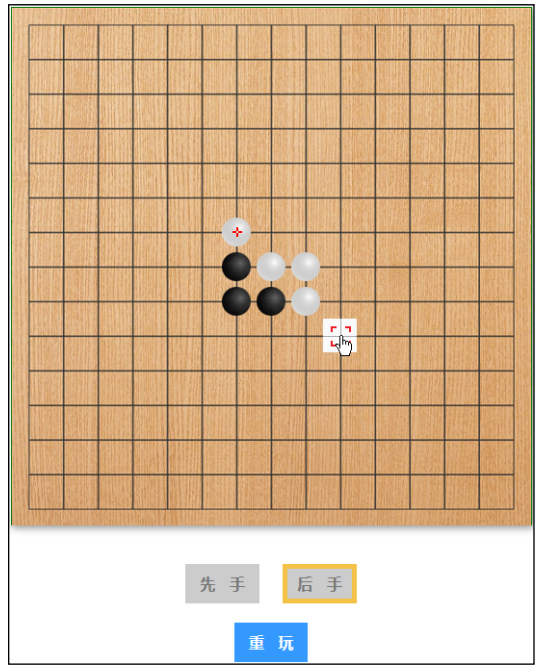


图 4.4 玩家后手

4.5.1 玩家下棋

如果选择玩家先手，玩家直接单击棋盘中的交叉点就开始下子，或者先单击“开始”按钮，再单击棋盘中的交叉点开始下子。玩家下棋的实现方法如下。

（1）在 Gobang 对象中分别定义游戏开始的方法 gameStart()、游戏结束的方法 gameOver()，以及玩家或计算机下棋的方法 playChess()，代码如下：

```
//游戏开始
gameStart: function () {
    this.isGameStart = true;
    document.getElementById("play_btn").innerHTML = "重   玩";      //设置按钮文字
},
//游戏结束
gameOver: function () {
    this.isGameStart = false;
    var firstBtn = document.getElementById("first_btn");
    var backBtn = document.getElementById("back_btn");
    firstBtn.classList.remove("disable");                                          //移除类
    backBtn.classList.remove("disable");                                           //移除类
```

```
        document.getElementById("play_btn").innerHTML = "开   始";      //设置按钮文字
    },
    //下棋
    playChess: function (i, j, color) {
        //设置该位置棋子的值
        this.chessArr[i][j] = color === "black" ? this.BLACK_CHESS : this.WHITE_CHESS;
        var chessboard = document.getElementById("chessboard");
        var divs = chessboard.getElementsByTagName("div");
        if (color === this.comColor) {
            //为计算机上一个棋子添加类
            chessboard.getElementsByClassName(color + "-last")[0]?.classList.add(color);
            //移除计算机上一个棋子的类
            chessboard.getElementsByClassName(color + "-last")[0]?.classList.remove(color + "-last");
            divs[i * 15 + j].classList.add(color + "-last");                          //为计算机新下的棋子添加类
        } else {
            divs[i * 15 + j].classList.add(color);                                    //为玩家新下的棋子添加类
        }
    }
```

（2）在 Gobang 对象中定义重置棋盘的方法 resetChessBoard()，在方法中定义单击棋盘时执行的函数，以及鼠标在棋盘上移动时执行的函数。单击棋盘时进行判断，如果当前位置没有棋子就调用 playChess()方法实现下棋。代码如下：

```
    //重置棋盘
    resetChessBoard: function () {
        var chessboard = document.getElementById("chessboard");
        chessboard.innerHTML = this.chessboardHtml;                                   //设置棋盘内容
        this.isGameOver = false;
        //初始化棋子状态
        var i, j;
        for (i = 0; i < 15; i++) {
            this.chessArr[i] = [];
            for (j = 0; j < 15; j++) {
                this.chessArr[i][j] = this.NO_CHESS;
            }
        }
        //玩家下棋
        var _t = this;
        var divs = chessboard.getElementsByTagName("div");
        for(i = 0; i < divs.length; i++){
            divs[i].index = i;
        }
        for(i = 0; i < divs.length; i++){
            divs[i].onclick = function () {
                if (!_t.isPlayerTurn || _t.isGameOver) {
                    return;
                }
                if (!_t.isGameStart && _t.isBack) {
                    return;
                }
                if (!_t.isGameStart) {
                    var firstBtn = document.getElementById("first_btn");
                    var backBtn = document.getElementById("back_btn");
                    firstBtn.classList.add("disable");                                //添加类
                    backBtn.classList.add("disable");                                 //添加类
                    _t.gameStart();                                                   //开始游戏
                }
                var index = this.index,
                    i = Math.floor(index / 15),
                    j = index % 15;
                if (_t.chessArr[i][j] === _t.NO_CHESS) {                              //如果当前位置没有棋子
```

```
                    _t.playChess(i, j, _t.playerColor);                    //下子
                    //根据单击的位置移除指定的样式
                    if (i === 0 && j === 0) {
                        this.classList.remove("hover-up-left");
                    }
                    else if (i === 0 && j === 14) {
                        this.classList.remove("hover-up-right");
                    }
                    else if (i === 14 && j === 0) {
                        this.classList.remove("hover-down-left");
                    }
                    else if (i === 14 && j === 14) {
                        this.classList.remove("hover-down-right");
                    }
                    else if (i === 0) {
                        this.classList.remove("hover-up");
                    }
                    else if (i === 14) {
                        this.classList.remove("hover-down");
                    }
                    else if (j === 0) {
                        this.classList.remove("hover-left");
                    }
                    else if (j === 14) {
                        this.classList.remove("hover-right");
                    }
                    else {
                        this.classList.remove("hover");
                    }
                    _t.playerLastChess = [i, j];
                    _t.isPlayerWin(i, j);                                  //判断玩家是否取胜
                }
            };
        }
        //鼠标在棋盘上的移动效果
        for(i = 0; i < divs.length; i++){
            divs[i].onmouseenter = function () {
                if (!_t.isPlayerTurn || _t.isGameOver) {
                    return;
                }
                if (!_t.isGameStart && _t.isBack) {
                    return;
                }
                var index = this.index,
                    i = Math.floor(index / 15),
                    j = index % 15;
                if (_t.chessArr[i][j] === _t.NO_CHESS) {                   //如果当前位置没有棋子
                    //根据鼠标指向的位置添加指定的样式
                    if (i === 0 && j === 0) {
                        this.classList.add("hover-up-left");
                    }
                    else if (i === 0 && j === 14) {
                        this.classList.add("hover-up-right");
                    }
                    else if (i === 14 && j === 0) {
                        this.classList.add("hover-down-left");
                    }
                    else if (i === 14 && j === 14) {
                        this.classList.add("hover-down-right");
                    }
                    else if (i === 0) {
                        this.classList.add("hover-up");
```

```
                    }
                    else if (i === 14) {
                        this.classList.add("hover-down");
                    }
                    else if (j === 0) {
                        this.classList.add("hover-left");
                    }
                    else if (j === 14) {
                        this.classList.add("hover-right");
                    }
                    else {
                        this.classList.add("hover");
                    }
                }
            };
            divs[i].onmouseleave = function () {
                if (!_t.isPlayerTurn || _t.isGameOver) { return; }
                var index = this.index,
                    i = Math.floor(index / 15),
                    j = index % 15;
                //根据鼠标移开的位置移除指定的样式
                if (i === 0 && j === 0) {
                    this.classList.remove("hover-up-left");
                }
                else if (i === 0 && j === 14) {
                    this.classList.remove("hover-up-right");
                }
                else if (i === 14 && j === 0) {
                    this.classList.remove("hover-down-left");
                }
                else if (i === 14 && j === 14) {
                    this.classList.remove("hover-down-right");
                }
                else if (i === 0) {
                    this.classList.remove("hover-up");
                }
                else if (i === 14) {
                    this.classList.remove("hover-down");
                }
                else if (j === 0) {
                    this.classList.remove("hover-left");
                }
                else if (j === 14) {
                    this.classList.remove("hover-right");
                }
                else {
                    this.classList.remove("hover");
                }
            };
        }
    }
```

4.5.2 判断玩家是否取胜

玩家下棋后需要判断玩家是否取胜。在 Gobang 对象中分别定义判断玩家是否获胜的方法 isPlayerWin() 和玩家获胜时调用的方法 playerWin()。在 isPlayerWin()方法中，分别在水平、垂直、左斜和右斜四个方向上计算玩家连续棋子的个数，如果个数大于或等于 5 就可以判断玩家获得胜利。代码如下：

```
//玩家是否取胜
```

```
isPlayerWin: function (i, j) {
    var nums = 1,                                          //连续棋子个数
        chessColor = this.BLACK_CHESS,
        m, n;
    //x 方向
    for (m = j - 1; m >= 0; m--) {
        if (this.chessArr[i][m] === chessColor) {
            nums++;
        } else {
            break;
        }
    }
    for (m = j + 1; m < 15; m++) {
        if (this.chessArr[i][m] === chessColor) {
            nums++;
        } else {
            break;
        }
    }
    if (nums >= 5) {
        this.playerWin();                                  //玩家胜利
        return;
    } else {
        nums = 1;
    }
    //y 方向
    for (m = i - 1; m >= 0; m--) {
        if (this.chessArr[m][j] === chessColor) {
            nums++;
        } else {
            break;
        }
    }
    for (m = i + 1; m < 15; m++) {
        if (this.chessArr[m][j] === chessColor) {
            nums++;
        } else {
            break;
        }
    }
    if (nums >= 5) {
        this.playerWin();                                  //玩家胜利
        return;
    } else {
        nums = 1;
    }
    //左斜方向
    for (m = i - 1, n = j - 1; m >= 0 && n >= 0; m--, n--) {
        if (this.chessArr[m][n] === chessColor) {
            nums++;
        } else {
            break;
        }
    }
    for (m = i + 1, n = j + 1; m < 15 && n < 15; m++, n++) {
        if (this.chessArr[m][n] === chessColor) {
            nums++;
        } else {
            break;
        }
    }
    if (nums >= 5) {
        this.playerWin();                                  //玩家胜利
        return;
```

```
        } else {
            nums = 1;
        }
        //右斜方向
        for (m = i - 1, n = j + 1; m >= 0 && n < 15; m--, n++) {
            if (this.chessArr[m][n] === chessColor) {
                nums++;
            } else {
                break;
            }
        }
        for (m = i + 1, n = j - 1; m < 15 && n >= 0; m++, n--) {
            if (this.chessArr[m][n] === chessColor) {
                nums++;
            } else {
                break;
            }
        }
        if (nums >= 5) {
            this.playerWin();                                    //玩家胜利
            return;
        }
        this.comTurn();                                          //计算机下棋
    },
    //玩家获胜
    playerWin: function () {
        this.showResult(true);                                   //显示结果
        this.gameOver();                                         //游戏结束
    }
```

4.5.3 计算机下棋

通过 isPlayerWin()方法可以判断玩家是否取胜，如果玩家未取胜，则会调用实现计算机下棋的方法 comTurn()。在 Gobang 对象中定义 comTurn()方法，在方法中调用 playChess()方法实现计算机下棋，并判断计算机是否满足获胜的条件，如果计算机获胜就显示游戏结果。代码如下：

```
//计算机下棋
    comTurn: function () {
        this.isPlayerTurn = false;
        var maxX = 0,
            maxY = 0,
            maxWeight = 0,
            i, j, tem;
        for (i = 14; i >= 0; i--) {
            for (j = 14; j >= 0; j--) {
                if (this.chessArr[i][j] !== this.NO_CHESS) {     //如果该位置有棋子
                    continue;
                }
                tem = this.comWeight(i, j);                      //调用 comWeight()方法计算权重
                if (tem > maxWeight) {
                    maxWeight = tem;                             //获取最大权重
                    maxX = i;
                    maxY = j;
                }
            }
        }
        this.playChess(maxX, maxY, this.comColor);               //计算机下棋
        this.comLastChess = [maxX, maxY];
        //计算机胜利的条件
        if ((maxWeight >= 100000 && maxWeight < 250000) || (maxWeight >= 500000)) {
            this.showResult(false);                              //显示结果
```

```
                this.gameOver();                                    //游戏结束
                this.isPlayerTurn = true;
            } else {
                this.isPlayerTurn = true;
            }
        }
```

4.5.4 棋子权重的判断

计算机在下棋时首先需要计算棋盘中各个空白交叉点的权重，权重最高的交叉点即为计算机下棋的位置。判断棋子权重的实现方法如下。

（1）在 Gobang 对象中分别定义统计水平、垂直、左斜和右斜四个方向连续棋子个数并判断连续棋子两端是否为空的方法，代码如下：

```
//统计 X 方向连子个数并判断两端是否为空
    numAndSideX: function (i, j, chessColor) {
        var m,
            nums = 1,
            side1 = false,
            side2 = false;
        for (m = j - 1; m >= 0; m--) {
            if (this.chessArr[i][m] === chessColor) {
                nums++;
            } else {
                if (this.chessArr[i][m] === this.NO_CHESS) {
                    side1 = true;
                }
                break;
            }
        }
        for (m = j + 1; m < 15; m++) {
            if (this.chessArr[i][m] === chessColor) {
                nums++;
            } else {
                if (this.chessArr[i][m] === this.NO_CHESS) {
                    side2 = true;
                }
                break;
            }
        }
        return {"nums": nums, "side1": side1, "side2": side2};
    },
    //统计 Y 方向连子个数并判断两端是否为空
    numAndSideY: function (i, j, chessColor) {
        var m,
            nums = 1,
            side1 = false,
            side2 = false;
        for (m = i - 1; m >= 0; m--) {
            if (this.chessArr[m][j] === chessColor) {
                nums++;
            } else {
                if (this.chessArr[m][j] === this.NO_CHESS) {
                    side1 = true;
                }
                break;
            }
        }
        for (m = i + 1; m < 15; m++) {
            if (this.chessArr[m][j] === chessColor) {
                nums++;
```

```
            } else {
                if (this.chessArr[m][j] === this.NO_CHESS) {
                    side2 = true;
                }
                break;
            }
        }
        return {"nums": nums, "side1": side1, "side2": side2};
    },
    //统计左斜方向连子个数并判断两端是否为空
    numAndSideXY: function (i, j, chessColor) {
        var m, n,
            nums = 1,
            side1 = false,
            side2 = false;
        for (m = i - 1, n = j - 1; m >= 0 && n >= 0; m--, n--) {
            if (this.chessArr[m][n] === chessColor) {
                nums++;
            } else {
                if (this.chessArr[m][n] === this.NO_CHESS) {
                    side1 = true;
                }
                break;
            }
        }
        for (m = i + 1, n = j + 1; m < 15 && n < 15; m++, n++) {
            if (this.chessArr[m][n] === chessColor) {
                nums++;
            } else {
                if (this.chessArr[m][n] === this.NO_CHESS) {
                    side2 = true;
                }
                break;
            }
        }
        return {"nums": nums, "side1": side1, "side2": side2};
    },
    //统计右斜方向连子个数并判断两端是否为空
    numAndSideYX: function (i, j, chessColor) {
        var m, n,
            nums = 1,
            side1 = false,
            side2 = false;
        for (m = i - 1, n = j + 1; m >= 0 && n < 15; m--, n++) {
            if (this.chessArr[m][n] === chessColor) {
                nums++;
            } else {
                if (this.chessArr[m][n] === this.NO_CHESS) {
                    side1 = true;
                }
                break;
            }
        }
        for (m = i + 1, n = j - 1; m < 15 && n >= 0; m++, n--) {
            if (this.chessArr[m][n] === chessColor) {
                nums++;
            } else {
                if (this.chessArr[m][n] === this.NO_CHESS) {
                    side2 = true;
                }
                break;
            }
        }
        return {"nums": nums, "side1": side1, "side2": side2};
```

```
    }
```

（2）在 Gobang 对象中定义 judgeWeight()方法，在方法中使用 switch 语句，在语句中根据连续棋子的个数和棋子两端是否为空来判断权重。代码如下：

```
//判断权重
    judgeWeight: function (nums, side1, side2, isCom) {
        var weight = 0;
        switch (nums) {
            case 1:
                if (side1 && side2) {
                    weight = isCom ? 15 : 10;                    //一个子两边为空
                }
                break;
            case 2:
                if (side1 && side2) {
                    weight = isCom ? 100 : 50;                   //两个连子两边为空
                }
                else if (side1 || side2) {
                    weight = isCom ? 10 : 5;                     //两个连子一边为空
                }
                break;
            case 3:
                if (side1 && side2) {
                    weight = isCom ? 500 : 200;                  //三个连子两边为空
                }
                else if (side1 || side2) {
                    weight = isCom ? 30 : 20;                    //三个连子一边为空
                }
                break;
            case 4:
                if (side1 && side2) {
                    weight = isCom ? 5000 : 2000;                //四个连子两边为空
                }
                else if (side1 || side2) {
                    weight = isCom ? 400 : 100;                  //四个连子一边为空
                }
                break;
            case 5:
                weight = isCom ? 100000 : 10000;
                break;
            default:
                weight = isCom ? 500000 : 250000;
                break;
        }
        return weight;
    }
```

（3）在 Gobang 对象中定义 comWeight()方法，在该方法中计算玩家下棋后棋盘中某个空白位置的权重。代码如下：

```
//计算下子权重
    comWeight: function (i, j) {
        var weight = 14 - (Math.abs(i - 7) + Math.abs(j - 7)),      //基于棋盘位置的权重
            pointInfo = {},                                         //存储连子个数及两端是否为空的信息
            chessColor = this.WHITE_CHESS;
        //x 方向
        pointInfo = this.numAndSideX(i, j, chessColor);
        //计算机下子权重
        weight += this.judgeWeight(pointInfo.nums, pointInfo.side1, pointInfo.side2, true);
        pointInfo = this.numAndSideX(i, j, -chessColor);
        //玩家下子权重
        weight += this.judgeWeight(pointInfo.nums, pointInfo.side1, pointInfo.side2, false);
        //y 方向
```

```
        pointInfo = this.numAndSideY(i, j, chessColor);
        //计算机下子权重
        weight += this.judgeWeight(pointInfo.nums, pointInfo.side1, pointInfo.side2, true);
        pointInfo = this.numAndSideY(i, j, -chessColor);
        //玩家下子权重
        weight += this.judgeWeight(pointInfo.nums, pointInfo.side1, pointInfo.side2, false);
        //左斜方向
        pointInfo = this.numAndSideXY(i, j, chessColor);
        //计算机下子权重
        weight += this.judgeWeight(pointInfo.nums, pointInfo.side1, pointInfo.side2, true);
        pointInfo = this.numAndSideXY(i, j, -chessColor);
        //玩家下子权重
        weight += this.judgeWeight(pointInfo.nums, pointInfo.side1, pointInfo.side2, false);
        //右斜方向
        pointInfo = this.numAndSideYX(i, j, chessColor);
        //计算机下子权重
        weight += this.judgeWeight(pointInfo.nums, pointInfo.side1, pointInfo.side2, true);
        pointInfo = this.numAndSideYX(i, j, -chessColor);
        //玩家下子权重
        weight += this.judgeWeight(pointInfo.nums, pointInfo.side1, pointInfo.side2, false);
        return weight;
    }
```

4.6 显示游戏结果

在玩家和计算机分出胜负后，页面中会显示游戏结果，获胜方形成 5 子连线的棋子会以特定的标记进行显示。玩家获胜的界面效果如图 4.5 所示。计算机获胜的界面效果如图 4.6 所示。

图 4.5 玩家获胜

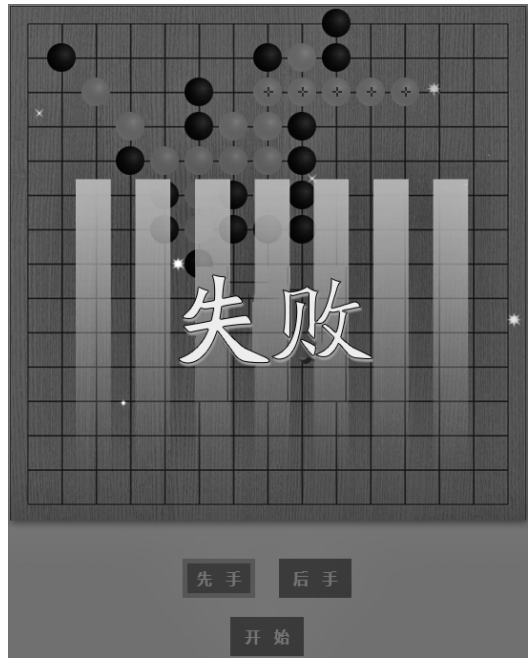


图 4.6 计算机获胜

显示游戏结果的实现方法如下。

（1）在 Gobang 对象中定义 markChess()方法，在该方法中为获胜方形成 5 子连线的棋子标记样式。代码如下：

```
//标记棋子样式
    markChess: function (pos, color) {
        var chessboard = document.getElementById("chessboard");
        var divs = chessboard.getElementsByTagName("div");
        divs[pos].classList.remove(color);                    //移除类
        divs[pos].classList.add(color + "-last");             //添加类
    }
```

（2）在 Gobang 对象中定义 markWinChesses()方法，在该方法中使用数组保存获胜方连续棋子的位置，并统计连续棋子的个数，再调用 markChess()方法标记连续棋子的样式。代码如下：

```
//标记显示获胜棋子
    markWinChesses: function (isPlayerWin) {
        var nums = 1,                                         //连续棋子个数
            lineChess = [],                                   //连续棋子位置
            i,
            j,
            k,
            chessColor,
            m, n;
        if (isPlayerWin) {
            chessColor = this.BLACK_CHESS;
            i = this.playerLastChess[0];
            j = this.playerLastChess[1];
        } else {
            chessColor = this.WHITE_CHESS;
            i = this.comLastChess[0];
            j = this.comLastChess[1];
        }
        var chessboard = document.getElementById("chessboard");
        var divs = chessboard.getElementsByTagName("div");
        //为计算机最后下子添加类
        chessboard.getElementsByClassName(this.comColor + "-last")[0]?.classList.add(this.comColor);
        //移除计算机最后下子的类
        chessboard.getElementsByClassName(this.comColor + "-last")[0]?.classList.remove(this.comColor + "-last");
        //x 方向
        lineChess[0] = [i,j];                                 //定义第一个数组元素
        for (m = j - 1; m >= 0; m--) {
            if (this.chessArr[i][m] === chessColor) {
                lineChess[nums] = [i,m];
                nums++;
            } else {
                break;
            }
        }
        for (m = j + 1; m < 15; m++) {
            if (this.chessArr[i][m] === chessColor) {
                lineChess[nums] = [i,m];
                nums++;
            } else {
                break;
            }
        }
        if (nums >= 5) {
            for (k = nums - 1; k >= 0; k--) {
                //调用 markChess()方法标记棋子样式
                this.markChess(lineChess[k][0] * 15 + lineChess[k][1], isPlayerWin ? this.playerColor : this.comColor);
            }
            return;
        }
```

```
//y 方向
nums = 1;
lineChess[0] = [i,j];
for (m = i - 1; m >= 0; m--) {
    if (this.chessArr[m][j] === chessColor) {
        lineChess[nums] = [m,j];
        nums++;
    } else {
        break;
    }
}
for (m = i + 1; m < 15; m++) {
    if (this.chessArr[m][j] === chessColor) {
        lineChess[nums] = [m,j];
        nums++;
    } else {
        break;
    }
}
if (nums >= 5) {
    for (k = nums - 1; k >= 0; k--) {
        //调用 markChess()方法标记棋子样式
        this.markChess(lineChess[k][0] * 15 + lineChess[k][1], isPlayerWin ? this.playerColor : this.comColor);
    }
    return;
}
//左斜方向
nums = 1;
lineChess[0] = [i,j];
for (m = i - 1, n = j - 1; m >= 0 && n >= 0; m--, n--) {
    if (this.chessArr[m][n] === chessColor) {
        lineChess[nums] = [m,n];
        nums++;
    } else {
        break;
    }
}
for (m = i + 1, n = j + 1; m < 15 && n < 15; m++, n++) {
    if (this.chessArr[m][n] === chessColor) {
        lineChess[nums] = [m,n];
        nums++;
    } else {
        break;
    }
}
if (nums >= 5) {
    for (k = nums - 1; k >= 0; k--) {
        //调用 markChess()方法标记棋子样式
        this.markChess(lineChess[k][0] * 15 + lineChess[k][1], isPlayerWin ? this.playerColor : this.comColor);
    }
    return;
}
//右斜方向
nums = 1;
lineChess[0] = [i,j];
for (m = i - 1, n = j + 1; m >= 0 && n < 15; m--, n++) {
    if (this.chessArr[m][n] === chessColor) {
        lineChess[nums] = [m,n];
        nums++;
    } else {
        break;
    }
```

```
        }
        for (m = i + 1, n = j - 1; m < 15 && n >= 0; m++, n--) {
            if (this.chessArr[m][n] === chessColor) {
                lineChess[nums] = [m,n];
                nums++;
            } else {
                break;
            }
        }
        if (nums >= 5) {
            for (k = nums - 1; k >= 0; k--) {
                //调用 markChess()方法标记棋子样式
                this.markChess(lineChess[k][0] * 15 + lineChess[k][1], isPlayerWin ? this.playerColor : this.comColor);
            }
        }
    }
```

（3）在 Gobang 对象中定义 showResult()方法，在该方法中创建 div 元素，如果玩家获胜，就在 div 元素中显示胜利的图片效果，3 秒后图片消失，否则就在 div 元素中显示失败的图片效果，3 秒后图片消失。代码如下：

```
//显示结果
    showResult: function(isPlayerWin) {
        var resultDiv = document.createElement("div");          //创建 div 元素
        if (isPlayerWin) {
            resultDiv.id = "success";                           //设置元素 id
            resultDiv.classList.add("imgDiv");                  //为 div 添加类
            resultDiv.innerHTML = "<img src='images/success.png' width='100%' height='100%'>";//为 div 设置图片
            document.body.appendChild(resultDiv);               //向 body 中添加 div 元素
            //3 秒后移除 div 元素
            setTimeout(function (){
                document.body.removeChild(resultDiv);
            },3000);
        } else {
            resultDiv.id = "failure";                           //设置元素 id
            resultDiv.classList.add("imgDiv");                  //为 div 添加类
            resultDiv.innerHTML = "<img src='images/failure.png' width='100%' height='100%'>";//为 div 设置图片
            document.body.appendChild(resultDiv);               //向 body 中添加 div 元素
            //3 秒后移除 div 元素
            setTimeout(function (){
                document.body.removeChild(resultDiv);
            },3000);
        }
        this.isGameOver = true;
        this.markWinChesses(isPlayerWin);                       //调用 markWinChesses()方法标记显示获胜棋子
    }
```

4.7 项 目 运 行

通过前述步骤，设计并完成了“五子棋小游戏”项目的开发。下面运行该项目，检验一下我们的开发成果。在浏览器中打开项目文件夹中的 index.html 文件即可成功运行该项目，运行后会进入游戏初始界面，效果如图 4.7 所示。如果选择玩家先手，直接单击棋盘中的交叉点，或者单击“开始”按钮后再单击棋盘中的交叉点开始下子，效果如图 4.8 所示。

如果选择玩家后手，单击“开始”按钮后计算机会先下子，效果如图 4.9 所示。

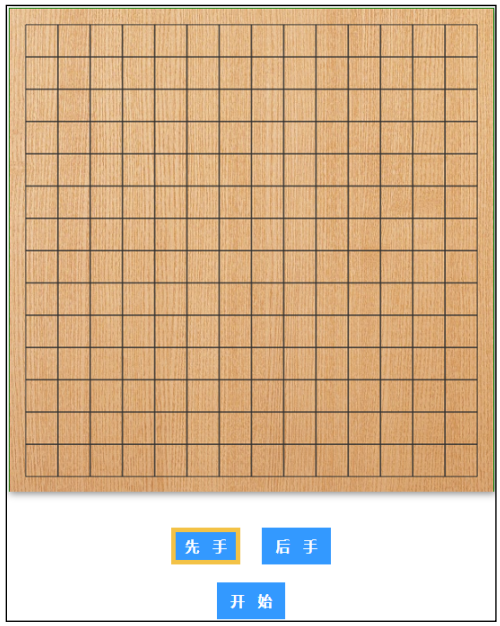


图 4.7　游戏初始界面

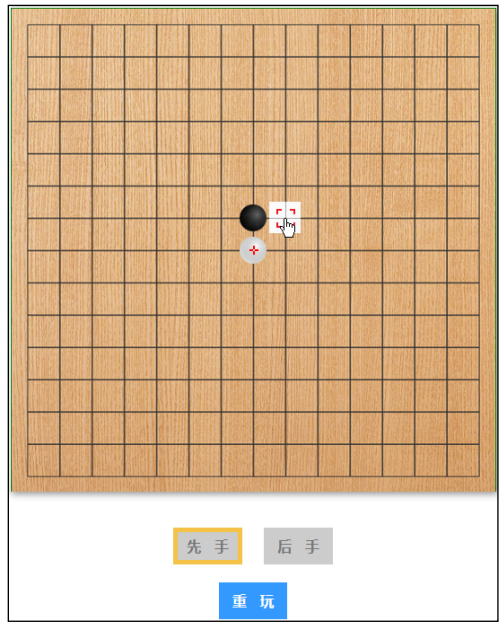


图 4.8　玩家先下子

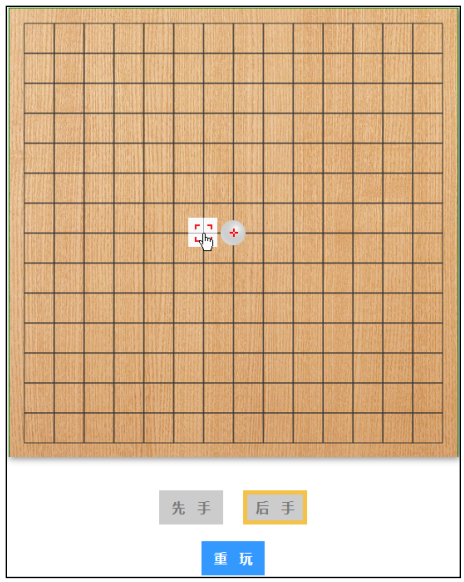


图 4.9　计算机先下子

4.8　源 码 下 载

虽然本章详细地讲解了如何编码实现“五子棋小游戏”的各个功能，但给出的代码都是代码片段。为了方便读者学习，本书提供了完整的项目源码，扫描右侧二维码即可下载。

源码下载