浙江大学

本科实验报告

课程名称：	B/S 体系软件设计
姓　　名：	秦昇
学　　院：	计算机科学与技术学院
系：	计算机科学与技术
专　　业：	计算机科学与技术
学　　号：	3120000060
指导教师：	胡晓军

2015 年　7 月　3 日

浙江大学实验报告

课程名称：B/S 体系软件设计 实验类型：综合型

实验项目名称：HTML5 实现中国象棋对战游戏

学生姓名：秦昇 专业：计算机科学与技术 学号：3120000060

同组学生姓名： 指导老师：胡晓军

实验地点： 实验日期：2015 年 6 月 28 日

1 实验目的

用 HTML 5 实现一个网页对战游戏。本实验设计实现了一个基于 HTML5 的网页象棋对战游戏。

2 实验要求

需要实现的基本功能如下：

1、 用 HTML 5 实现一个网页游戏，题材不限

2、 用户需要注册后进入游戏，游戏方式可以是两人或多人对战

3、 每个参与游戏的客户端能够快速响应其他客户端的行动，具有较好的并发能力

4、 游戏结果在服务器保存并提供查询界面（如排行榜）

3 设计文档

在中期提交的设计文档的基础上有所修改。

3.1 系统开发说明

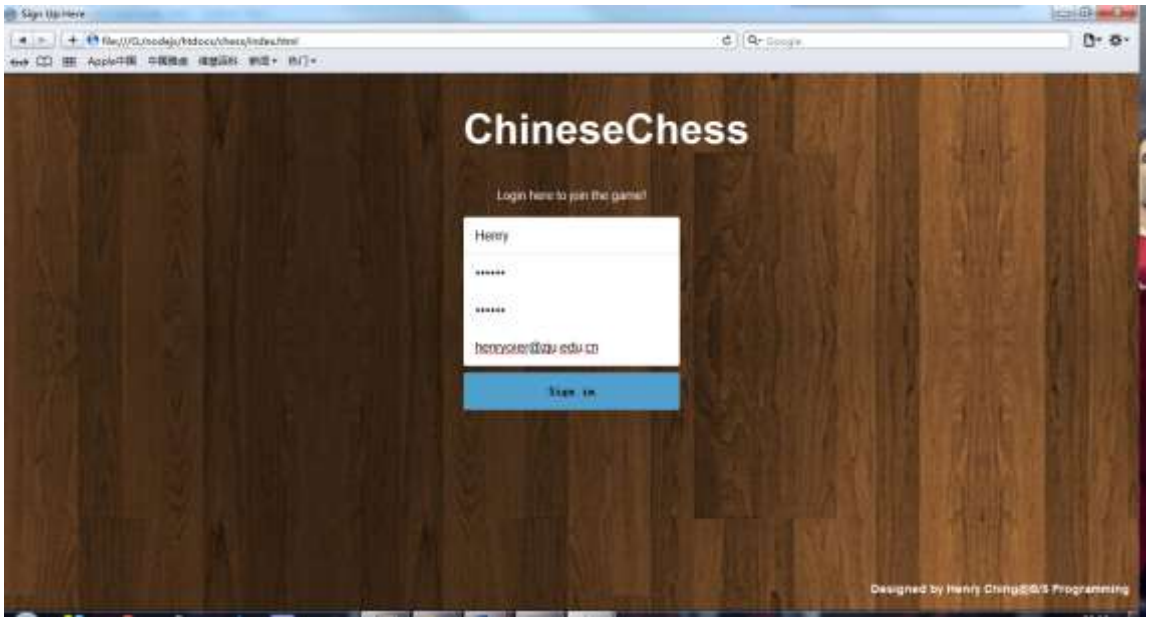
本系统旨在实现一个网页对战形式的中国象棋游戏，游戏内容和传统中国 象棋相同，能够实现两人在线实时对战的功能，前端将使用 html5 网页，嵌入 JavaScript 脚本以实现游戏功能；后台利用 node.js，建立服务器并且和用户的浏览器端口建

立 websocket 通信，实现数据的交互。

与中期报告相比，最大的变化是后台处理改用 node.js，而不是开始计划使用的 php，原因在于为了提高游戏的实时性，采用 socket 通信的方式更为简单，而若使用 php，则需要通过数据库来进行数据的同步，需要用 ajax 的方式来进行实时数据的同步。

3.2 具体实现模块

3.2.1 注册登陆模块



用户直接注册后登陆游戏进入 index.html 界面，分别设置用户名、密码、email 等系列注册信息，利用 JS 脚本对注册信息进行部分限制，诸如不能为空等。由 js 建立与服务器的 socket 通信并将表单内容发予服务器，服务器端实现数据插入数据库的操作。

代码如下，send 为给服务器发送插入数据的消息。

```
function submit()
{

    var username=document.getElementById("username");
    var pass=document.getElementById("password");
    var confirm = document.getElementById("confirm");
    var email = document.getElementById("email");

    if(username.value=="")
    {
```

```
        alert("Please input username");
        username.focus();
        return false;
    }
    if(pass.value=="" || confirm.value=="" )
    {
        alert("Please input password");
        return false;
    }
    if(pass.value != confirm.value)
    {
        alert("Confirm password is wrong");
        return false;
    }
    if(email.value=="")
    {
        alert("Please input email address");
        return false;
    }

    send('login,'+username.value+','+pass.value +','+email.value);

    window.location.href = 'maingame.html?name='+username.value;
}
```

3.2.2游戏主界面

游戏主界面分为两个部分，中间靠左的部分是棋盘，也是进行游戏对弈的区域；右边为聊天室，可以实现实时多人聊天。

游戏部分使用 HTML5 的 canvas 标签

```
<canvas  id="chess">对不起，您的浏览器不支持 HTML5，请升级浏览器至 IE9、firefox 或者谷歌浏览器！</canvas>
```

在 canvas 标签基础上添加棋盘图片，棋子等等。
在加载浏览器窗口后开始执行如下代码，包括对背景和一系列变量的初始化

```
window.onload = function(){

    com.bg=new com.class.Bg();
    com.dot = new com.class.Dot();
    com.pane=new com.class.Pane();
    com.pane.isShow=false;

    com.childList=[com.bg,com.dot,com.pane];
    com.mans   ={};      //棋子集合
    com.createMans(com.initMap)    ;   //生成棋子
    com.bg.show();

    com.get("bnBox").style.display = "block";
    //play.init();

    com.get("tyroPlay").addEventListener("click", function(e) {
        if (confirm("确认开始游戏？")){
            play.isPlay=true ;
            play.depth = 3;
            play.init();
        }
    })
}
```

其中要调用到 com.init 函数来初始化界面

```
com.init = function (stype){

    com.nowStype = "stype2";
    var stype = com.stype[com.nowStype];
    com.width        =   stype.width;     //画布宽度
    com.height          =   stype.height;        //画布高度
    com.spaceX          =   stype.spaceX;    //着点 X 跨度
    com.spaceY          =   stype.spaceY;    //着点 Y 跨度
```

```
    com.pointStartX     =   stype.pointStartX;  //第一个着点 X 坐标;
    com.pointStartY     =   stype.pointStartY;  //第一个着点 Y 坐标;
    com.page         =   stype.page;         //图片目录

    com.get("box").style.width = com.width+130+"px";

    com.canvas          =   document.getElementById("chess"); //画布
    com.ct           =   com.canvas.getContext("2d") ;
    com.canvas.width =   com.width;
    com.canvas.height   =   com.height;

    com.childList    =   com.childList||[];

    com.loadImages(com.page);     //载入图片/图片目录
    //z(com.initMap.join())
}
```

点击开始游戏以后则会执行以下代码：

```
play.init = function (){

    play.my             =   1;            //玩家方
    play.map         =   com.arr2Clone (com.initMap);      //初始化棋
盘
    play.nowManKey      =   false;        //现在要操作的棋子
    play.pace        =   [];           //记录每一步
    play.isPlay      =   true ;        //是否能走棋
    play.mans        =   com.mans;
    play.bylaw          =   com.bylaw;
    play.show        =   com.show;
    play.showPane       =   com.showPane;
    play.isOffensive =   true;         //是否先手
    play.depth          =   play.depth || 3;           //搜索深度

    play.isFoul         =   false; //是否犯规长将

    audience = false;



    com.pane.isShow     =    false;          //隐藏方块

    //初始化棋子
    for (var i=0; i<play.map.length; i++){
       for (var n=0; n<play.map[i].length; n++){
```

```
            var key = play.map[i][n];
            if (key){
                com.mans[key].x=n;
                com.mans[key].y=i;
                com.mans[key].isShow = true;
            }
        }
    }
    play.show();


    //绑定点击事件
    if(playernum < 2)
        com.canvas.addEventListener("click",play.clickCanvas)


}
```

在游戏过程中实质上是增加了一个 canvas 的 addEventListenner()，来监听是否有点击棋盘或棋子被点击。其中包括点击到棋子、棋盘和普通点三种状态，分别对应下列三个函数。

```
//点击棋盘事件
play.clickCanvas = function (e){
    if (!play.isPlay) return false;
    var key = play.getClickMan(e);
    var point = play.getClickPoint(e);

    var x = point.x;
    var y = point.y;

    if (key){
        play.clickMan(key,x,y);
    }else {
        play.clickPoint(x,y);
    }
    play.isFoul = play.checkFoul();//检测是不是长将
}

//点击棋子，两种情况，选中或者吃子
play.clickMan = function (key,x,y){
    var man = com.mans[key];
    //吃子
    if (play.nowManKey&&play.nowManKey != key && man.my != com.mans[play.nowManKey ].my){
```

```
       //man 为被吃掉的棋子
       if (play.indexOfPs(com.mans[play.nowManKey].ps,[x,y])){

          man.isShow = false;
          var
pace=com.mans[play.nowManKey].x+","+com.mans[play.nowManKey].y+',
'+x+','+y;
          //z(bill.createMove(play.map,man.x,man.y,x,y))
          delete
play.map[com.mans[play.nowManKey].y][com.mans[play.nowManKey].x];
          play.map[y][x] = play.nowManKey;

    com.showPane(com.mans[play.nowManKey].x  ,com.mans[play.nowManK
ey].y,x,y)
          com.mans[play.nowManKey].x = x;
          com.mans[play.nowManKey].y = y;
          com.mans[play.nowManKey].alpha = 1

          play.nowManKey = false;
          com.pane.isShow = false;
          com.dot.dots = [];
          com.show()
          com.get("clickAudio").play();
          if (key == "j0") play.showWin (-1);
          if (key == "J0") play.showWin (1);
          play.isPlay = false;
          send(pace);
       }
    // 选中棋子
    }else{
       if (man.my===1){

          if                              (com.mans[play.nowManKey])
com.mans[play.nowManKey].alpha = 1 ;
          man.alpha = 0.6;
          com.pane.isShow = false;
          play.nowManKey = key;
          com.mans[key].ps = com.mans[key].bl(); //获得所有能着点
          com.dot.dots = com.mans[key].ps
          com.show();
          com.get("selectAudio").play();

       }
    }
```

```
}

//点击着点
play.clickPoint = function (x,y){
    var key=play.nowManKey;
    var man=com.mans[key];
    if (play.nowManKey){
        if (play.indexOfPs(com.mans[key].ps,[x,y])){
            var pace=man.x+","+man.y+','+x+','+y;

            delete play.map[man.y][man.x];
            play.map[y][x] = key;
            com.showPane(man.x ,man.y,x,y)
            man.x = x;
            man.y = y;
            man.alpha = 1;
            play.pace.push(pace+x+y);
            play.nowManKey = false;
            com.dot.dots = [];
            com.show();
            com.get("clickAudio").play();
            play.isPlay = false;
            send(pace);
        }else{
            alert("不能这么走哦！")
        }
    }

}
```

主界面右边是聊天界面。

```
<div id="chat-box" style="border:1px solid #cccccc; background-color:#ffffff; width:400px; height:400px; overflow:scroll;"></div>
```

在聊天的时候实质上改变的是其 innerHTML 的内容。

3.2.3重点难点——游戏中实时数据交互

在中期之后我学习了一些不同的后台处理技术，在此基础上我选择了 node.js 而非之前的 php，原因很简单，node.js 能够提供和 html5 进行 websocket 的实时通信，具有轻量级实时性高的特点，而利用 ajax 和 php 结合的通信方式不如该方法简答，并且 node.js 的语法和 JavaScript 较为相似，在编程的时候也会更加简单。

首先要设置一个监听服务器 server.js

```
var conns = new Array();

var ws = require('websocket-server');
var server = ws.createServer();
var map = [];

var mysql = require('mysql');

var TEST_DATABASE = 'chat';

//创建连接
var client = mysql.createConnection({
  user: 'root',
  password: '',
});

client.connect();
client.query('USE '+ TEST_DATABASE);

server.addListener('connection', function(conn){

    conns.push(conn);

    conn.addListener('message',function(msg){

        console.log(msg);
        msgs = msg.split(',');

        if(msgs[0] == 'con'){
            for(var i=0; i<conns.length; i++){
                if(conn == conns[i])
                    conns[i].send(msgs[0] + ',' +i);
                    console.log(msgs[0] + ',' +i);
            }
        }else
        if(msgs[0] == 'login'){
            client.query("INSERT  INTO  user  VALUES('"  +  msgs[1]
+"','"+msgs[2]+"','"+msgs[3]+"')");
```

```
        }
        if(msgs[0] == 'query'){
            client.query(  "SELECT * FROM rank",
                function selectCb(err, results, fields) {
                if (err) {
                    throw err;
                }

                if(results)
                  {
                      for(var i = 0; i < results.length; i++)
                      {
                          console.log(results[i].name    +    ','    +
results[i].win);
                          conn.send(results[i].name     +     ','     +
results[i].win);
                      }
                  }
                }
            );
        }
        else{
            for(var i=0; i<conns.length; i++){
                if(conn != conns[i])
                    conns[i].send(msg);
            }
        }
    });
});

server.listen(8888);
console.log('running......');
```

在上述代码中，实现监听的部分主要是 addListener()函数，通过自定义 message 的格式，可以对传来的 message 可以进行区分并且分开执行相应的操作。

而在游戏的客户端，我们也需要建立与 server 的 socket 通信，这里采用如下代码

```
var host = '127.0.0.1';
    var para = getPar();
    name = para['name'];
    var port = 8888;
    var url = 'ws://'+host+':'+port+'/';
```

```
    w = new WebSocket(url);
```

当需要发送消息的时候，用 w.send()即可，而 w.onmessage() 函数用来监听服务器发过来的消息。

例如我们将游戏过程中客户端与服务器之间的消息进行显示：

可以看到从登陆到走出第一步已经有了这么多的数据通信，其中 con 开头的为自定义的连接信息，login 为登陆信息，而(7,7,4,7)四元组则是棋盘上棋子的移动信息，另外 chat 开头的是聊天信息等，通过这些信息的实时交互我们能够在第一时间将信息同步更新，已达到实时游戏和聊天的目的。

3.2.4重点难点——后台数据管理与显示

其实在这次游戏设计中后台的数据量非常的少，整个数据库较为重要的就两张表，一张是 user 记录用户的注册信息，一张为 rank，则是记录用户胜场数的信息。由于 html 前端无法直接与数据库交互，因此还是需要 node.js 的后台进行，在上面 server.js 的代码中可以看到，我们在服务器中建立了一个 mysql 的 client 连接，通过 client.query()来执行 sql 语句，对后台数据进行管理。

而在实现排行榜界面的时候，需要动态更新数据，因此我们可以通过 JavaScript 的函数来实现。

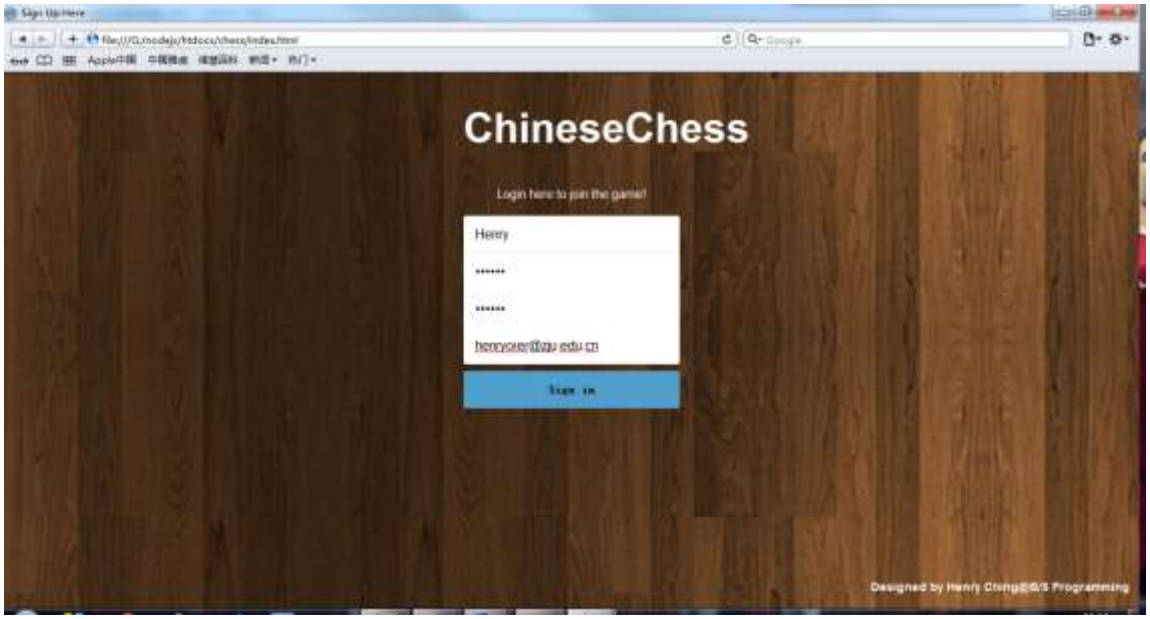
```
w.onmessage = function(e){
       var msg = e.data;

       mes = e.data.split(",");

       document.getElementById("add").innerHTML +=
                 '<th>'+ mes[0]+'</th><th>'+ mes[1] + '</th>';

   }
```

大致就是在界面中建立好显示的表格，当数据到来的时候，再追加写到表格的 innerHTML 中，同样的技术也应用到了聊天室的消息显示中。

4 测试报告

4.1 注册并登陆

在浏览器栏键入首页的路径，进入游戏登陆界面（index.html），填入注册数据
可以看到在此过程中会有 JS 对输入的检查



在登陆之前查看数据库中 user 表，内容为空

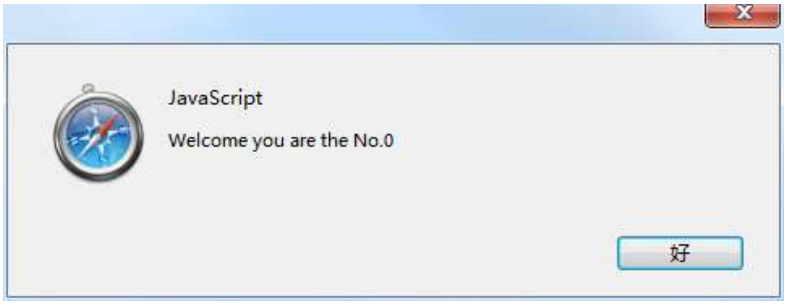
点击登陆以后，进入游戏主界面，在地址栏可以看到当前用户名 Henry，用 GET 形式在 html 页面间传递参数。

此时再查看数据库

可以看到数据已经成功插入 user 表中。

4.2 游戏主界面设置

进入游戏主界面之后，首先会弹出以下窗口



这是服务器向你发送你在房间内的登陆顺序编号，0 号表示你是第一个进入房间的，受限于监听端口，我们暂时只能实现在一个房间内双人对战，多人观看和聊天的功能。

游戏界面主要分为两个部分，中间靠左为棋盘，也是进行对弈的主要位置，右边是聊天室，可以实现多人实时聊天功能。
点击开始游戏，此时进行游戏的是 0 号 Henry 和 1 号 Jerry，观战者是 2 号 Axiu。

Henry 选择了当头炮

在 Jerry 的页面实时显示了 Henry 的下法

在下完第一步以后 Henry 将处于等待回应的状态，此时他无法再选定自己的棋子，只能等待对方下完以后自己才能再下。

这时 Jerry 选择了马二进三

我们可以看到 Henry 的页面上也显示了 Jerry 的下法

而在这期间我们可以看到 Axiu 的界面上一直在显示两人的下法，因为她是本房间的第三个用户，处于观战模式，而在观战模式下是无法进行任何棋子的移动的。

Henry 和 Jerry 继续进行对弈，其中包括各种吃子，并且在游戏过程中两人开始聊天

而此时观战者也可以在聊天室里发言，其中红色显示为他人发言，蓝色为自己发言

同时我们可以看到服务器和客户端之间的通信

```
G:\nodejs\htdocs\chess>node server.js
running......
con,Henry
con,Jerry
con,Axiu
7,7,4,7
7,9,6,7
4,7,4,3
chat,Henry,Nice!
chat,Jerry,You are excellent!
chat,Axiu,Good game!Haha!
```

此时 Henry 选择了将军，他获胜了

在 Jerry 的界面则显示他输掉了这一盘

4.3 查看排行榜

此时点击查看排行榜按钮，进入查询页面

与后台数据保持一致。至此游戏的主要功能和界面已经测试完毕。

Current selection does not contain a unique column. Grid edit, ch

正在显示第 0 - 3 行（共 4 行，查询花费 0.0010 秒。）

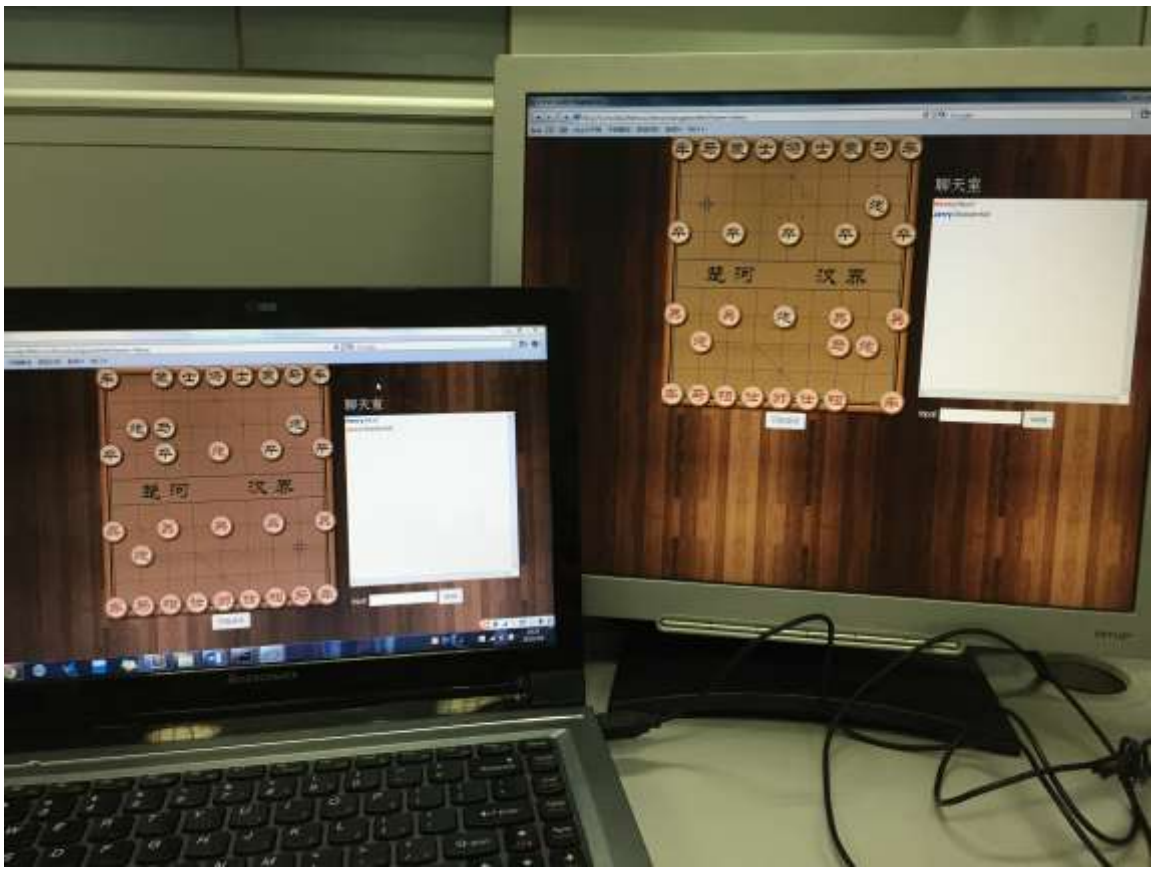
```
SELECT * FROM `rank`
```

行数： 25 过滤行： 在表中搜索

+ 选项

name	win
Henry	9
Axiu	7
Jerry	8
Jack	5

在测试的时候我使用了本地搭建的服务器，为了方便观看，将两个对弈的页面分开到了两个显示屏上。



5 用户使用手册

5.1 运行环境

本游戏前端由 HTML5 内嵌 JavaScript，后台由 node.js 实现，连接 mysql 数据库。

5.2 运行步骤

在运行前需先安装 node.js，然后在游戏根目录下，安装 web-socket 包(游戏源代码中已含有)，同时键入

```
npm install mysql
```

安装 nodejs 连接 mysql 的包，开启 mysql。
至此环境搭建完毕。

在游戏根目录下，用命令行键入

```
node server.js
```

运行后台服务器，然后即可进入页面开始游戏。

6 开发体会

这是我实现的第一个网页对战游戏，在这个过程中有过很多尝试和失败，但是收获的东西更多。尽管以前接触过网页编程以及服务器搭建之类的实验，但是真正深入去做一个游戏还是很少。总结起来收获有以下几点

- **项目设计是关键**

  其实在动手写之前，做得最多的工作就是设计，包括中期需要提交设计报告等等，都是为了在实现之前作好充分的准备和计划，然而尽管这样，还是出现了在设计好了之后更改方案的情况。这学期同时也修了软件工程这门课，在面对多人合作完成的大项目的时候，设计更是尤为重要，涉及到分工合作等等，就需要我们能够严格把关才能保证项目的按时进行。
- **代码能力是基础**

  尽管有时候我们会有很多很好的点子或者奇思妙想，但是如果要真正去实现，还需要坚实的代码能力。庆幸的是一直以来我在写代码的时候都能够保持一个高的效率和准确性，这样也无形中缩短了开发时间，有更多时间去优化项目。
- **前端设计并非与程序员无关**

  很多时候人们会认为前端的设计与程序员并无关系，程序员只用把写好的代码和页面像雕塑一样拿给前端的设计师去，他们予以着色即可。但其实，程序员很有必要对一些基本的前端设计有所了解，比如这次前段我使用了一些 bootstrap 的元素，尽管没有系统地去学习，但是基本的了解可以让我在以后

设计网页的时候有所优化，效果会比没有任何装饰的页面要好很多。

7 小结

总的来说这个项目培养了我的动手能力、并且让我有机会接触到了诸如 HTML，JavaScript，PHP, Node.js 以及 CSS 样式等许多和网页设计相关的东西，并且让我了解了服务器的搭建等等，一学期的学习下来，可谓受益匪浅。