#=====================================================================

# ChessClient.html

#=====================================================================

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>五子棋</title>

<**script** src="**socket.io.js**"></script>

<**script** src="**connect.js**"></script>

</head>

<body>

<canvas id="Chess" width="500" height="500"></canvas>

ip<input id="serverIP" type="text" value="127.0.0.1"/>

Name<input id="name" type="text" value="abc123"/>

<button value="test" onclick="Register()">連線</button>

<br>

<font id="test"></font>

</body>

<**script** src="**setting.js**"></script>

<script>

function otherData(x,y){

console.log("對手下"+x+","+y);

}

//random

function randoms(){

var ranok = true;

var maxNumX = (chessX-1);

var maxNumY = (chessY-1);

var ran = {x:0,y:0};

while(ranok){

ran.x = Math.floor(Math.random() \* (maxNumX + 1));

ran.y = Math.floor(Math.random() \* (maxNumX + 1));

if(cheesspace[ran.x][ran.y] == 2){

ranok = false;

}

}

return ran;

}

CreateChessBoard();

</script>

</html>

#=====================================================================

#=====================================================================

# connect.js

#=====================================================================

var socket = null;

//送出資料

function **SendData**(x,y){

// socket.emit('SendData',x,y,fightID);

socket.emit('setPiece',x,y);

}

//Send

function **Register**(){

var address = "**http://**"+document.getElementById("serverIP").value+":3000";

**socket = io.connect**(address);

console.log("connecting....");

//初始連線

**socket.on**('connect', function(){

console.log("reg");

socket.**emit**('reg', document.getElementById("name").value);

});

//註冊成功

socket.on('regFinish', function(){

console.log("註冊成功:");

});

//開始對弈

socket.on('fightStart', function(){

CreateChessBoard();

isFirst = false;

});

//接收資料

socket.on('getPiece', function(x,y){

//console.log("對手下子:"+x+","+y);

otherData(x,y);

if(x==-1 && y==-1)

{

isFirst = true;

player = true;

autorun();

return;

}

// 假設沒有棋可以下了

if(maxChess<=0)

{

fightResult = 2;

}

maxChess--;

cheesspace[x][y] = 1;

replychess();

player = true;

if(!gameover)

autorun();

});

//結束對弈

socket.on('fightEnd', function(){

gameover = true;

//console.log("對弈結束");

});

//接收資料

socket.on('start', function(){

//console.log("start");

});

//**接收資料**

socket.on('**ReceiveData**', function(x,y){

**cheesspace**[x][y] = **1**;

**replychess**();

**player** = true;

if(!gameover)

**autorun**();

});

}

#=====================================================================

#=====================================================================

# setting.js

#=====================================================================

var chessX = 11;

var chessY = 11;

var chessSize = 40;

var player = true;

var online = true;

var cheesspace = new Array(chessX);

var gameover = false;

var maxChess = 0; // 最大可下棋子數

var fightResult = -1;

var fightRecord = null;

var isFirst = false;

var stepNumberL=0;

var stepNumberV=0;

var c = document.getElementById("Chess");

var ctx = c.getContext("2d");

fightRecord = new Array();

//初始化棋盤

function CreateChessBoard(){

//初始化棋盤陣列

for(var i=0;i<chessX;i++){

cheesspace[i] = new Array(chessY);

for(var j=0;j<chessY;j++){

cheesspace[i][j] = 2;

maxChess++;

}

}

fightResult = -1;

gameover = false;

// 清空畫布

ctx.clearRect(0, 0, c.width, c.height);

// c.removeEventListener("mousedown", mouseDownHandler);

//開始畫棋盤(初始化)

ctx.beginPath();

//X線

for(var i = 1;i<chessY+1;i++){

ctx.moveTo(chessSize, i\*chessSize);

ctx.lineTo(chessX\*chessSize, i\*chessSize);

}

//Y線

for(var i = 1;i<chessX+1;i++){

ctx.moveTo(i\*chessSize, chessSize);

ctx.lineTo(i\*chessSize, chessY\*chessSize);

}

//基本框

for(var i = 0;i<chessX;i++){

for(var j = 0;j<chessY;j++){

ctx.fillStyle = "#FFF";

**ctx.fillRect((chessSize\*0.75)+(i\*chessSize),(chessSize\*0.75)+(j\*chessSize),chessSize/2,chessSize/2);**

}

}

ctx.stroke();

// c.addEventListener('mousedown', mouseDownHandler, false);

}

var tt = "";

//繪製目前的棋盤

function replychess(){

for(var i = 0;i<chessX;i++){

for(var j = 0;j<chessY;j++){

ctx.beginPath();

if(cheesspace[i][j] == **0**){

ctx.arc(chessSize+(i\*chessSize),chessSize+(j\*chessSize),chessSize/3,0,2\*Math.PI);

ctx.fillStyle = '**white**';

ctx.fill();

}

else if(cheesspace[i][j] == **1**){

ctx.arc(chessSize+(i\*chessSize),chessSize+(j\*chessSize),chessSize/3,0,2\*Math.PI);

ctx.fillStyle = '**black**';

ctx.fill();

}

ctx.stroke();

}

}

}

//自動下棋

function autorun(){

var ran = **randoms**();

ctx.beginPath();

**cheesspace**[ran.x][ran.y] = **0**;

ctx.arc(chessSize+(ran.x\*chessSize),chessSize+(ran.y\*chessSize),chessSize/3,0,2\*Math.PI);

ctx.fillStyle = '**white**';

ctx.fill();

ctx.stroke();

**SendData**(ran.x,ran.y);

**player** = !player;

**maxChess--**;

}

#=====================================================================

#=====================================================================

socket.io.js: Socket io v2.1.0

1. JavaScript library for Web api
2. Based on WebSocket protocal
3. Object-oriented, Realtime, Event driven
4. bi-direction communication between client and server
5. Client: browser, Server: Node.js

#=====================================================================