本次课题：制作HTML5在线五子棋对弈游戏。

预计开发周期：还没想好看心情，先预计7天完成。

备注：最近做学校课题有关数据挖掘的，有时候搞得没得头绪，做个小游戏和大家分享一下，并将小程序开源到Github，欢迎下载。

接上期。传送门：

<http://liuxinyumo.cn/index.php?s=/Home/Index/content/id/5.html>

项目仓库地址：<https://github.com/liuxinyumocn/LittleGame>

到目前为止我们完成了对棋盘的设计、欢迎界面以及从欢迎界面通过点击事件进入游戏界面的工作，在此之后我为了“美化”游戏，做了一个图片放置于DIV背景，并简单调整图片与线条的颜色，接下来我们继续完成课题设计。



本次篇幅我们将设计黑白双方棋子，通过点击事件拾取落点，并独立化落子功能，希望通过提供坐标以达到下棋目的，其作用在于未来联机时，不仅可以通过屏幕落子，并可以对来自网络数据给对手落子。

第一步我们需要绘制两种颜色的棋子元素。

代码编辑器打开IGraphElement.js文件，追加棋子定义：

{

ID:"Piece",

Version:100.00,

Src:new Array(),

Class:function(){

this.Color;

//初始化函数

this.SetWhite = function(){ //将棋子变成白色

this.Color = "rgb(205,205,205)";

}

this.Constructor = function(){

this.Color = "rgb(50,50,50)";//默认为黑色

}

this.Draw = function(ctx){

ctx.beginPath();

ctx.arc(this.Left,this.Top,18,0,Math.PI\*2,true);

ctx.fillStyle = this.Color;

ctx.fill();

ctx.closePath();

return true;

}

this.Over = function(){

return false;

}

}

}

其中我们使用了自定义成员函数 SetWhite() 稍后我会介绍自定义函数如何使用。

我们定义完了一个棋子的图像，但我们现在并不知道，棋子怎么落才为规范，于是我们现在需要独立定义一个工具类，用于管理落点，并能够通过提供偏移值修正落点（因为我们将来的棋盘位置可能随时发生调整）。

独立创建新的管理工具类，该类的具体作用是：提供棋盘中的编号坐标（虽然有字母，但我们统一使用数字），给出实际落点坐标；

或给出实际落点坐标（该坐标可能存在距离误差）我们给出误差范围内最近落点编号坐标，（为什么会有误差？因为人手落点位置不太容易精准点到棋盘落子点，误差范围内我们都认为点到了）

var PiecePostionKit = {

//棋子落点工具

//需要给出棋盘左上角的落点坐标（而非棋盘左上角坐标，因为有编号）

//允许的误差半径

//因为是工具类，设计成静态类即可

Displacement:{

//棋盘偏移 我是一点点试出来因为比较快 你创建一个棋子看看什么位置能放在左上角即可

x:52,

y:52

},

Radius:20,

BoardPostion:function(x,y){ //提供棋盘坐标 给出屏幕落点坐标

var X,Y;

X = this.Displacement.x + (x-1)\*40;

Y = this.Displacement.y + (y-1)\*40;

return {x:X,y:Y};

},

ScreenPostion:function(x,y){ //提供屏幕坐标 给出棋盘落点坐标

var X,Y;

X = x-this.Displacement.x+this.Radius+40;

Y = y-this.Displacement.y+this.Radius+40;

var X = parseInt(X/40);

var Y = parseInt(Y/40);

if(X>0 && X<16 && Y > 0 && Y <16)

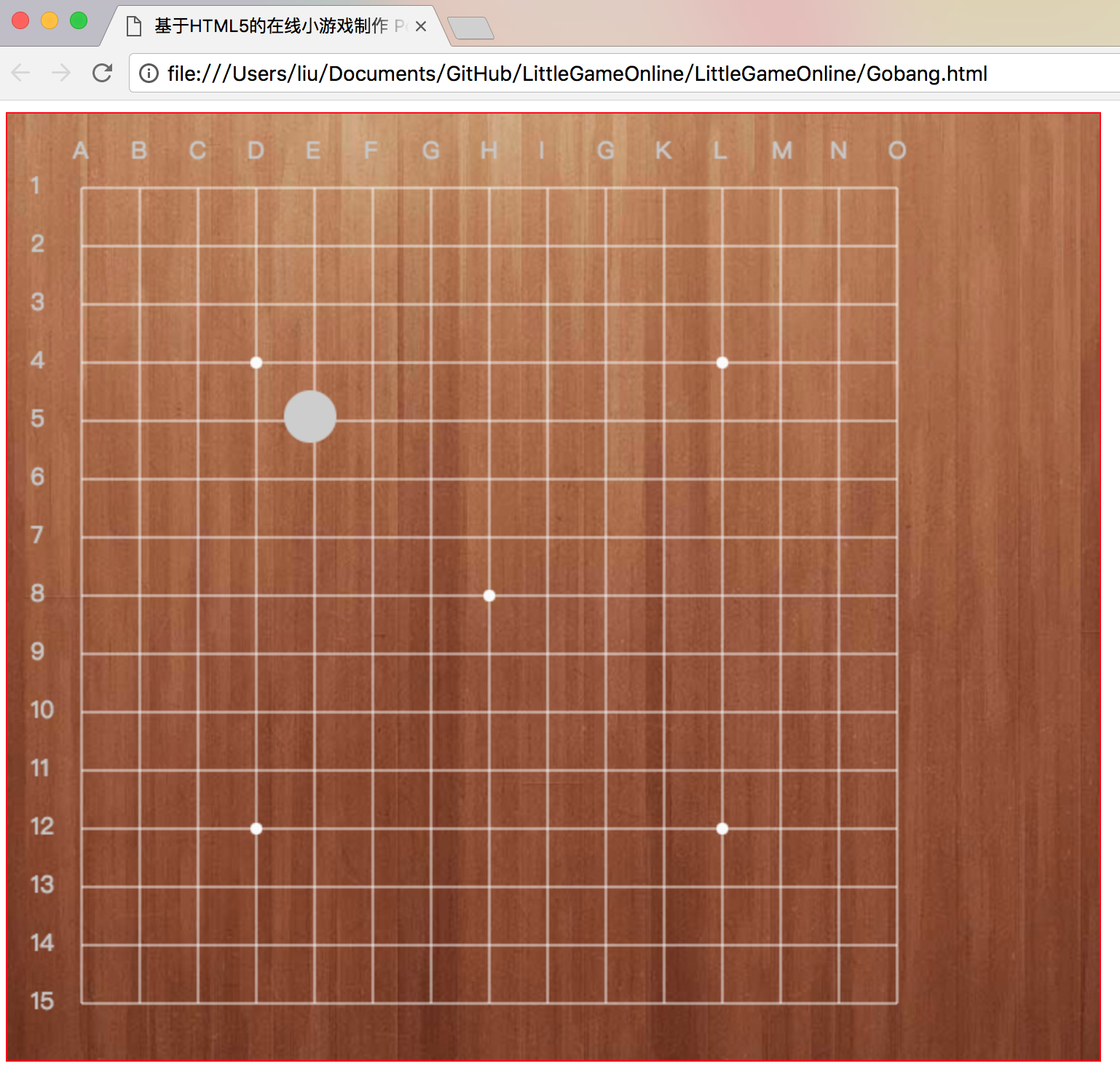
return {x:X,y:Y};

return {x:-1,y:-1};

}

}

经过直接给棋盘坐标可以看见，能够准确给出屏幕坐标位置：



为了更好的游戏体验，我们增加落点定位器来辅助确定位置，同时也测试一下由给屏幕坐标是否能够准确给出棋盘坐标。

打开IGraphElement.js，创建一个定位辅助器。

{

ID:"Sight",

Version:100.00,

Src:new Array(),

Class:function(){

this.Color;

this.Radius;

this.LineWeigth;

this.Length;

//初始化函数

this.Constructor = function(){

this.Color = "rgb(200,200,200)";//默认为黑色

this.Radius = 20;

this.LineWeigth = 1;

this.Length = 5;

}

this.Draw = function(ctx){

ctx.beginPath();

ctx.lineWidth = this.LineWeigth;

var x1,x2,y1,y2;

x1 = this.Left-this.Radius;

x2 = this.Left+this.Radius;

y1 = this.Top-this.Radius;

y2 = this.Top+this.Radius;

ctx.moveTo(x1+this.Length,y1);

ctx.lineTo(x1,y1);

ctx.lineTo(x1,y1+this.Length);

ctx.moveTo(x2-this.Length,y2);

ctx.lineTo(x2,y2);

ctx.lineTo(x2,y2-this.Length);

ctx.moveTo(x1+this.Length,y2);

ctx.lineTo(x1,y2);

ctx.lineTo(x1,y2-this.Length);

ctx.moveTo(x2-this.Length,y1);

ctx.lineTo(x2,y1);

ctx.lineTo(x2,y1+this.Length);

ctx.strokeStyle = this.Color;

ctx.stroke();

ctx.closePath();

return true;

}

this.Over = function(){

return false;

}

}

}

给出一个测试点位：

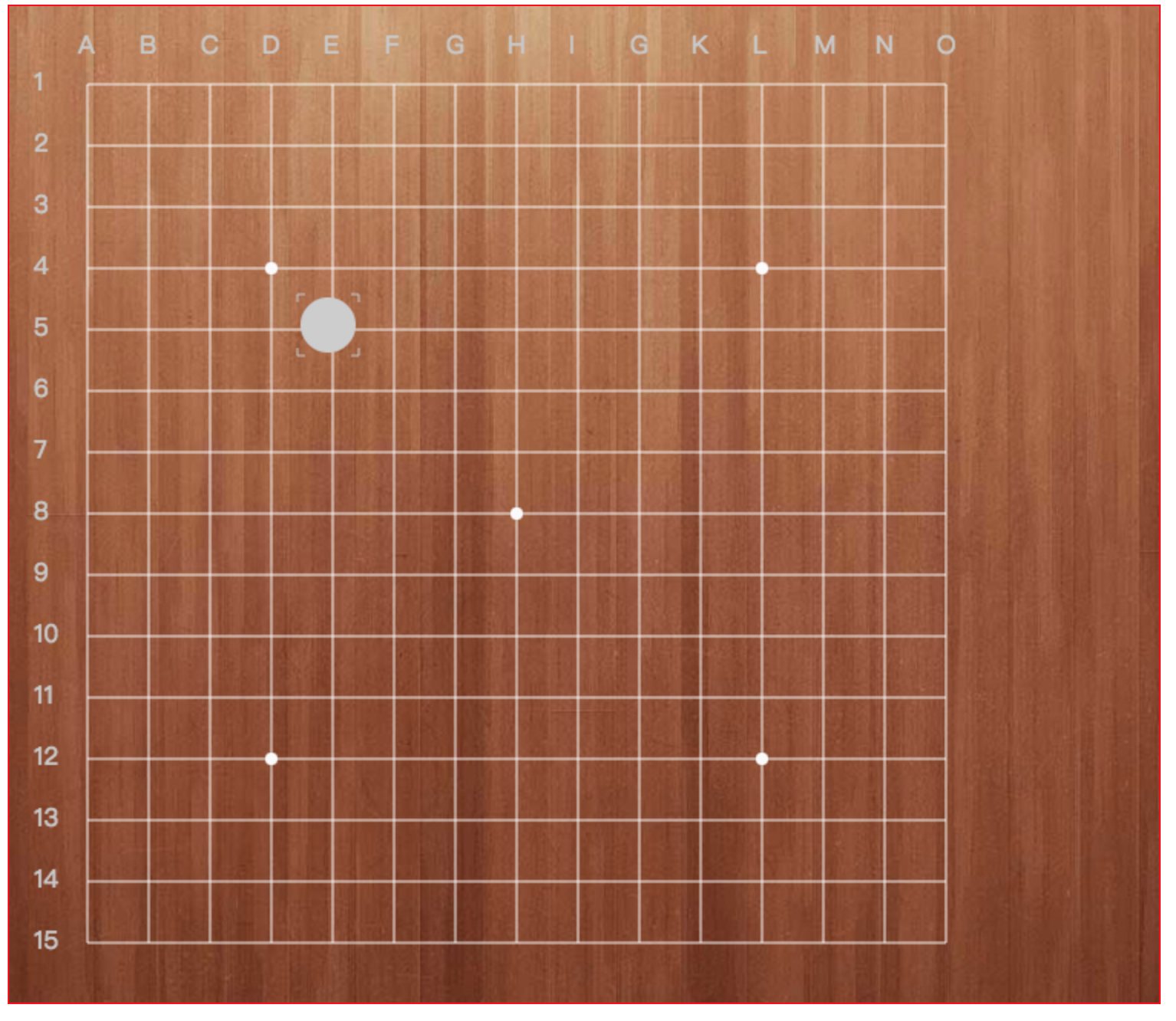
var Sight = this.\_MainPage.AddElement("Sight");

var ps = PiecePostionKit.ScreenPostion(200,160);

p = PiecePostionKit.BoardPostion(ps.x,ps.y);

Sight.Left(p.x);

Sight.Top(p.y);

Sight.Visible(true);

效果满意。

现在我们增加Move事件，保证在棋盘内，辅助器可以跟随鼠标进行落点定位。

if(this.\_Animation.CurrentPage == this.\_MainPage){

var x = this.\_Listener.X;

var y = this.\_Listener.Y;

var ps = PiecePostionKit.ScreenPostion(x,y);

if(ps.x != -1){

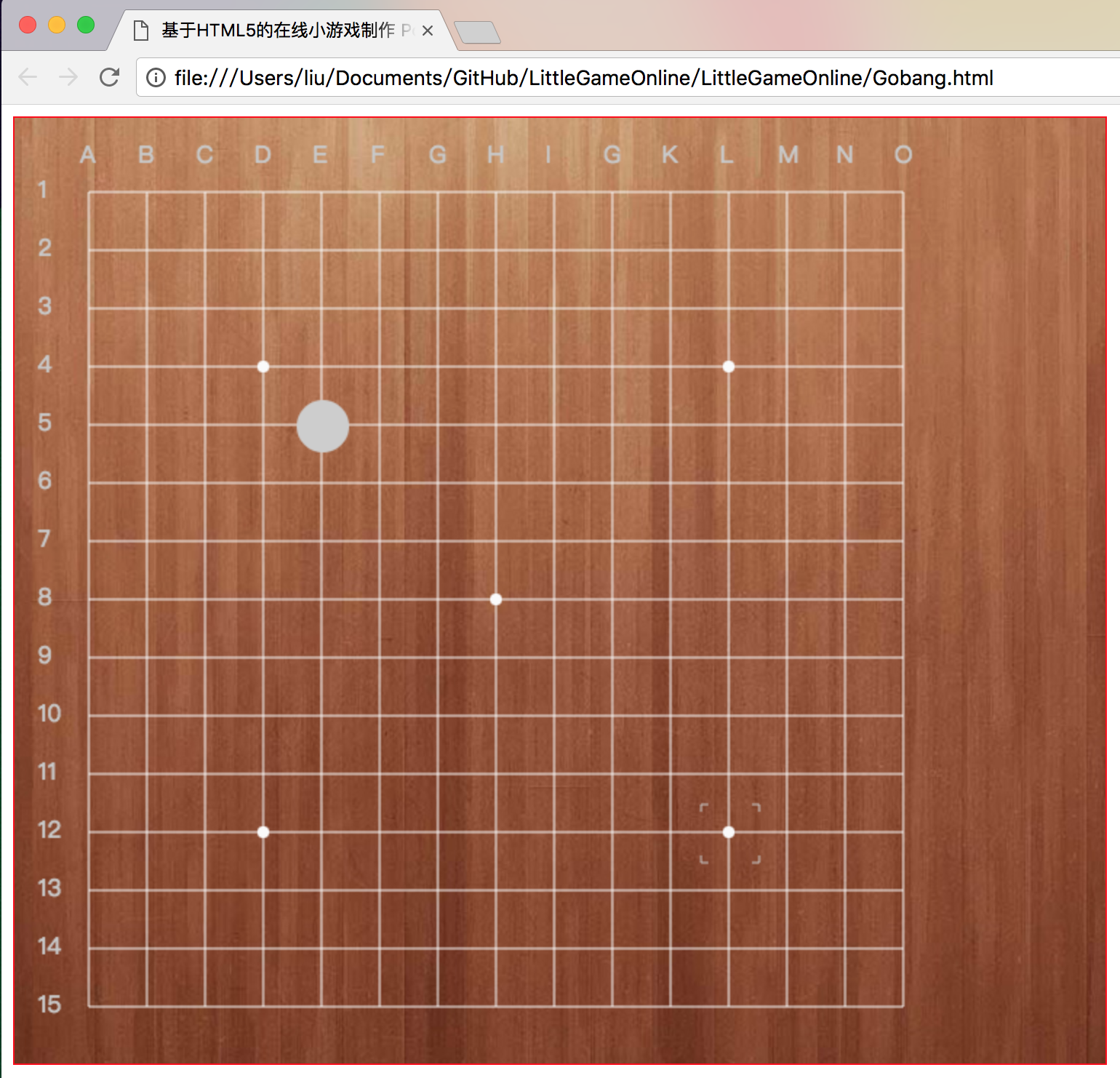
var p = PiecePostionKit.BoardPostion(ps.x,ps.y);

this.\_Sight.Left(p.x);

this.\_Sight.Top(p.y);

}

}



这样，辅助器就可以跟随鼠标了，而且还能检验坐标工具坐标计算的是否正确。

今天就到这里了……