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/4.html>

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

上篇我们利用IAnimation为页面创建了Canvas元素，并且将2个自定义的组件载入到了Canvas呈现出来，接下来我们定义一个简单的棋盘，并且创建事件使得点击开始游戏即可进入棋盘界面。

棋盘必然是自定义元素控件，因此我们仍然需要打开IGraphElement.js文件追加定义棋盘，注意格式，追加内容如下。

{

ID:"Board",

Version:100.00,

Src:new Array(),

Class:function(){

//成员变量

this.LineColor;

this.LineWeigth;

this.LinexCount;

this.LineyCount;

this.Width;

this.Height;

this.Gap;

//初始化函数

this.Constructor = function(){

//对成员变量赋初值

this.LineColor = "rgb(100,100,100)";

this.LineWeigth = 1;

this.LinexCount = 15;

this.LineyCount = 15;

this.Gap = 40;

}

this.Draw = function(ctx){

ctx.beginPath();

ctx.lineWidth = this.LineWeigth;

//定义线条绘制的左上角起点，用于方便未来增加外围字母编号空隙

this.left = this.Left;

this.top = this.Top;

for(var x=0;x<this.LinexCount;x++){

ctx.moveTo(this.left+x\*this.Gap,this.top);

ctx.lineTo(this.left+x\*this.Gap,this.top+(this.LineyCount-1)\*this.Gap);

}

for(var y=0;y<this.LineyCount;y++){

ctx.moveTo(this.left,this.top+y\*this.Gap);

ctx.lineTo(this.left+(this.LinexCount-1)\*this.Gap,this.top+y\*this.Gap);

}

ctx.strokeStyle = this.LineColor;

ctx.stroke();

ctx.closePath();

return true;

}

this.Over = function(){

return false;

}

}

}

棋盘其实如果简易一点，并不会很难绘制，五子棋的棋盘是由横竖2组15根线条构成，因此我们使用两个独立的循环结构绘制直线即可。

我们回到Game.js中进行一下简单的编辑，展示一下棋盘画的效果。

在Prototype中我们追加一个成员方法：

\_CreateMainScene:function(){

this.\_MainPage = this.\_Animation.CreatePage();

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

Board.Left(10);

Board.Top(10);

Board.Visible(true);

}

并对之前所创建方法函数进行如下修改：

\_DownloadFinished:function(){

//此处外部资源已经全部加载完毕可以创建游戏场景。

this.\_CreateWelcomeScene();

//这里留有部分用于创建游戏界面

this.\_CreateMainScene();

this.\_Animation.SelectPage(this.\_MainPage);

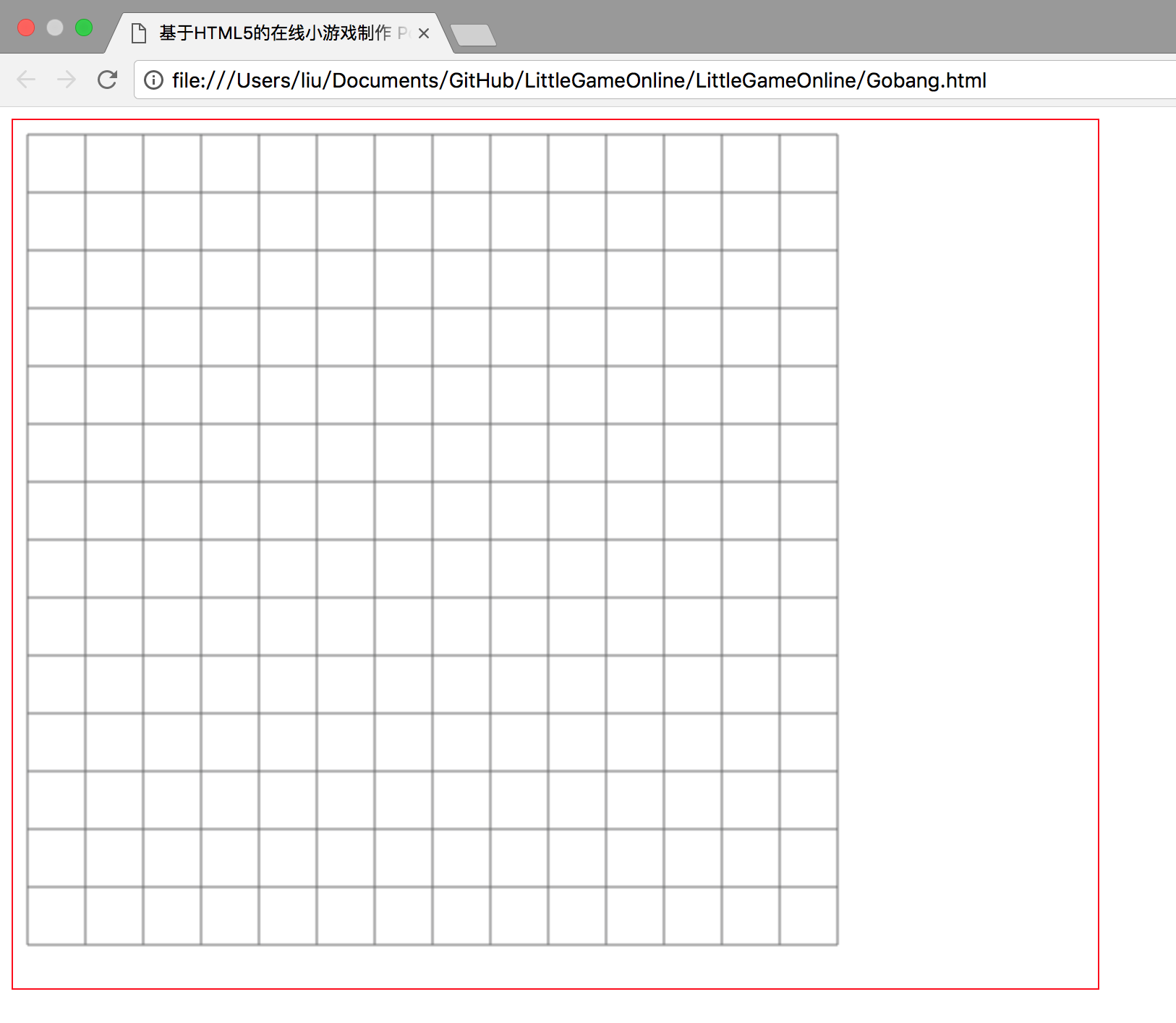
},

表示，欢迎场景创建完毕后开始创建游戏界面，最后一句this.\_Animation.SelectPage(this.\_MainPage); 用于将this.\_MainPage设置成当前渲染场景，注意，我们在上面CreateMainScene函数中AddElement（Board）了，所以不要忘记对加载项进行数组扩充：

this.\_Animation.DownloadIMG(["GobangLogo","BeginGameButton",**"Board"]**);

尽管我们目前的棋盘似乎没有外部加载资源（完全是靠线条画出来的）但作为一个好习惯，还是要写上，因为日后为了美化棋盘，不见得我们真的就不会增加外部资源。

好了，现在我们看下效果刷新一下浏览器：



是不是画的还可以？接下来，我们调整一下DIV大小，再对棋盘边缘增加编号和字母，棋盘中也会有5个位置会有加重的原点，看的更像一个棋盘。

棋盘自定义代码部分增改为：

{

ID:"Board",

Version:100.00,

Src:new Array(),

Class:function(){

//成员变量

this.LineColor;

this.LineWeigth;

this.LinexCount;

this.LineyCount;

this.Width;

this.Height;

this.Gap;

this.Numberx;

this.Numbery;

this.FontColor;

this.FontSize;

this.FontFamily;

this.Point;

//初始化函数

this.Constructor = function(){

//对成员变量赋初值

this.LineColor = "rgba(100,100,100,1)";

this.LineWeigth = 0.5;

this.LinexCount = 15;

this.LineyCount = 15;

this.Gap = 40;

this.Numberx = new Array("A","B","C","D","E","F","G","H","I","G","K","L","M","N","O");

this.Numbery = new Array(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15);

this.FontColor = this.LineColor;

this.FontSize = 16;

this.FontFamily = "宋体";

this.Point = new Array(4,4,4,12,12,4,12,12,8,8);

}

this.Draw = function(ctx){

ctx.beginPath();

ctx.lineWidth = this.LineWeigth;

//定义线条绘制的左上角起点，用于方便未来增加外围字母编号空隙

var left = this.Left+this.FontSize+15;

var top = this.Top+this.FontSize+15;

for(var x=0;x<this.LinexCount;x++){

ctx.moveTo(left+x\*this.Gap,top);

ctx.lineTo(left+x\*this.Gap,top+(this.LineyCount-1)\*this.Gap);

}

for(var y=0;y<this.LineyCount;y++){

ctx.moveTo(left,top+y\*this.Gap);

ctx.lineTo(left+(this.LinexCount-1)\*this.Gap,top+y\*this.Gap);

}

ctx.strokeStyle = this.LineColor;

ctx.stroke();

ctx.closePath();

//增加编号和字母

ctx.fillStyle = this.FontColor;

ctx.font = this.FontSize+"px "+this.FontFamily;

for(var x=0;x<this.Numberx.length;x++){

ctx.fillText(this.Numberx[x],this.Left+27+x\*this.Gap,this.Top+this.FontSize);

}

for(var y=0;y<this.Numbery.length;y++){

ctx.fillText(this.Numbery[y],this.Left+this.FontSize-15,this.Top+38+y\*this.Gap,);

}

//增加5个加重的圆点 坐标点为 4D 4L 12D 12L 8H

for(var i=0;i<this.Point.length;i++){

var x = this.Point[i++]-1;

var y = this.Point[i]-1;

ctx.beginPath();

ctx.arc(left+x\*this.Gap,top+y\*this.Gap,4,0,Math.PI\*2,true);

ctx.fillStyle = this.LineColor;

ctx.fill();

ctx.closePath();

}

return true;

}

this.Over = function(){

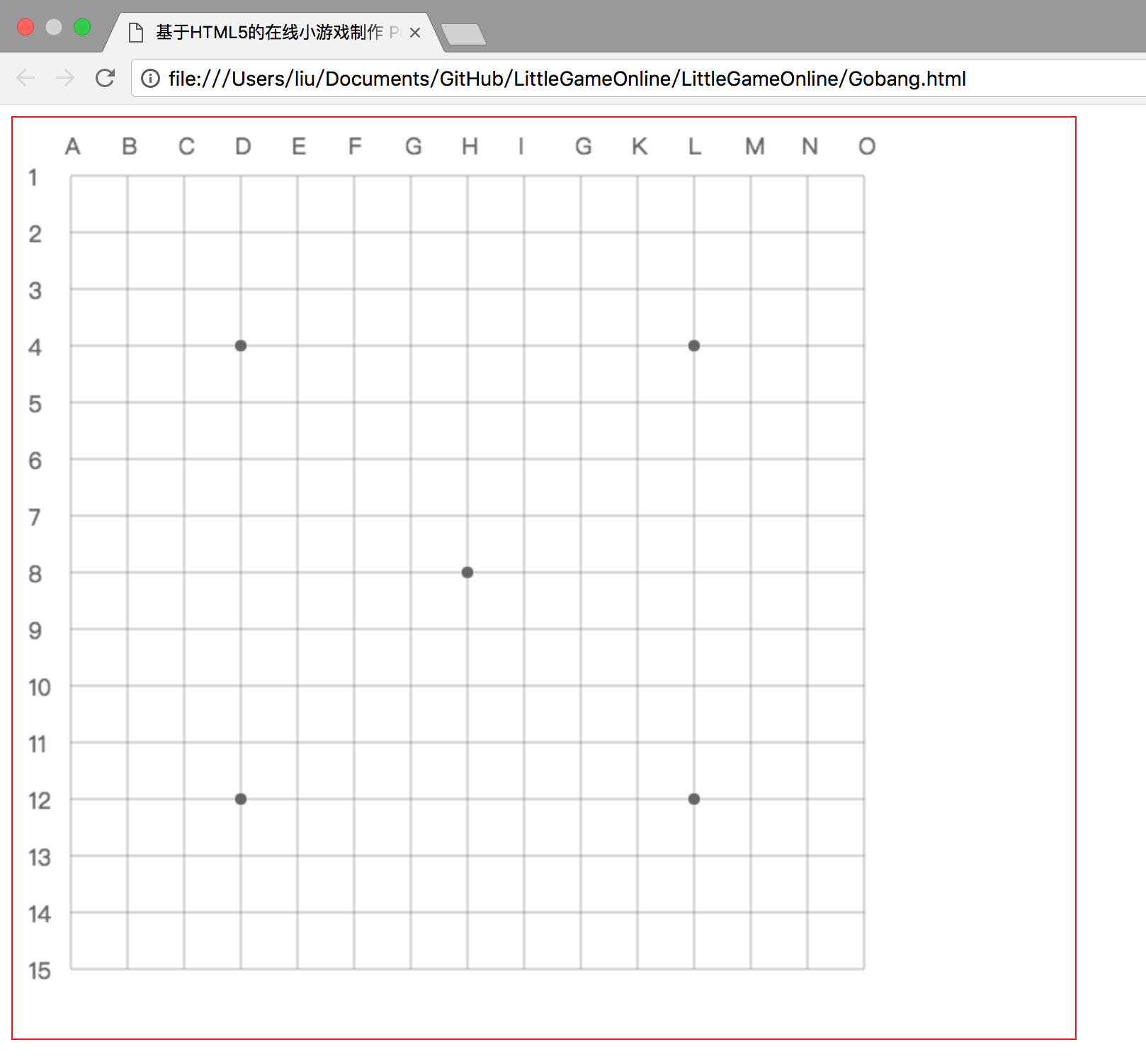
return false;

}

}

}

再看一下效果：



现在是不是很像了？

接下来就需要去制作事件相关的事情了，我们本篇就要求做到点击欢迎界面的开始游戏按钮，然后呈现棋盘界面吧。

首先IAnimation.js仅仅是对Canvas元素对象化管理插件，我并没有集成事件机制（不是不能集成，而是这不太符合我当初该插件的功能需求），于是我另单独设计有关方便管理事件监听的插件部分，需要小伙伴们同样去下载，为了方便，我上传到了我原来的IAnimation仓库中，也就是在该仓库中是可以一并下载事件相关控件。

下载地址：<https://github.com/liuxinyumocn/IAnimation>

请将该包中的ICanvasListenerB1.0.0.js文件移植到我们的项目文件夹中。

有关监听事件的这个插件实际上并不完善不过当前的应用层次已经足够，该插件支持识别鼠标动作和移动设备触屏动作，但缺点是目前还不能识别多指触屏，因为还是比较懒的……当然，我们不是去做一个MOBA游戏，没那么复杂的事件需求，所以学习足够了。

移植之后请将该文件插入到Gobang.html中，并确保该文件在Game.js之前被引用。

<script type="text/javascript" src="ICanvasListenerB1.0.0.js"></script>

<script type="text/javascript" src="Game.js"></script>

我们先注释掉，之前我们强行选择的游戏界面，

//this.\_Animation.SelectPage(this.\_MainPage);

这样刷新浏览器则会是进入欢迎页面，也就是接下来我们要做的是点击开始游戏，执行

this.\_Animation.SelectPage(this.\_MainPage);

使用事件时先要对委托器进行实例化，在Gobang构造函数中实例化委托器：

//Gobang游戏类的构造函数

this.\_Animation = new IAnimation(Container);

**this.\_Listener = new ICanvasListener(Container);**

传递给委托器的是我们的容器（DIV），因此，我们的委托器事件发生的基础是，当该DIV或该DIV内部元素被点击时，产生动作，且动作坐标是从DIV的左上角为圆点，这就方便了许多，因为我们的canvas充满DIV，所以该插件所反馈的坐标就是canvas内部的绘图坐标。

接下来需要进行事件委托，放在DownloadingFinished尾部：

//this.\_Animation.SelectPage(this.\_MainPage);

//事件委托

var t = this;

function Move(){

t.\_Move();

}

function Down(){

t.\_Down();

}

function Up(){

t.\_Up();

}

this.\_Listener.AddEventMouseMove(Move);

this.\_Listener.MouseMove(true);

this.\_Listener.AddEventMouseDown(Down);

this.\_Listener.MouseDown(true);

this.\_Listener.AddEventMouseUp(Up);

this.\_Listener.MouseUp(true);

this.\_Listener.AddEventTouchMove(Move);

this.\_Listener.TouchMove(true);

this.\_Listener.AddEventTouchDown(Down);

this.\_Listener.TouchDown(true);

this.\_Listener.AddEventTouchUp(Up);

this.\_Listener.TouchUp(true);

我们将其简单化，将Touch和Mouse事件合并成3个基本事件，移动、按下、抬起。并在prototype中提供3个回调方法用于处理回调事件。

\_Move:function(){

},

\_Up:function(){

},

\_Down:function(){

}

实际上，我们目前仅仅处理Up事件即可，当手松开时是开始游戏，则进入游戏界面。有人会问为什么不是Down，因为人们惯用习惯是松开手才发生事情，往往鼠标按下有悔意时，则鼠标移开应允许不发生按钮事件，各位可以回忆曾经经历即可。当然，单纯的判断Up还不够，更合理的时Down与Up同时满足时才应执行，我们只是简化而已。

因此在Up回调方法中应加入以下判断以及事件即可：

var Result = this.\_Animation.Over(this.\_Listener.X,this.\_Listener.Y);

if(Result == this.\_BeginGameButton){

this.\_Animation.SelectPage(this.\_MainPage);

}

注意：我们原来创建BeginGameButton的时候，使用了局部变量，在其他方法中自然是得不到的，所以需要在当初创建时增加成员变量声明：

var BeginGameButton = this.\_WelcomePage.AddElement("BeginGameButton");

this.\_BeginGameButton = BeginGameButton;

现在刷新页面，点击开始游戏，就可以进入棋盘界面了。