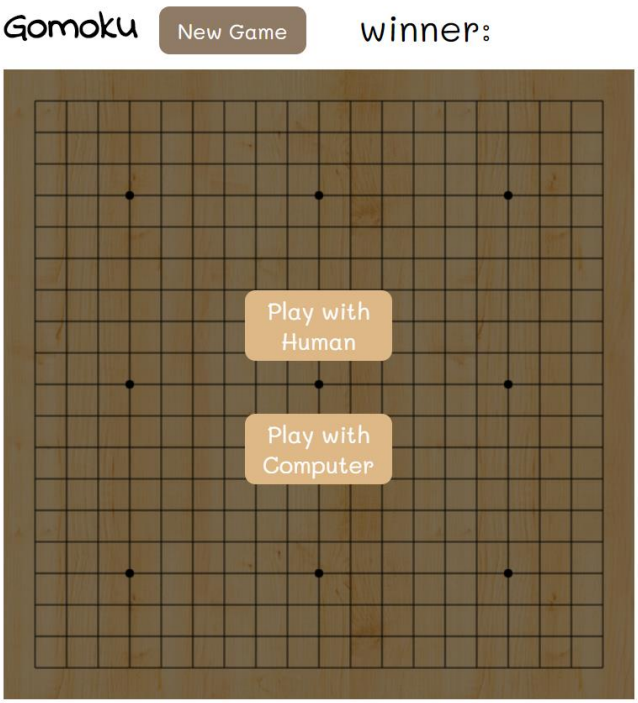
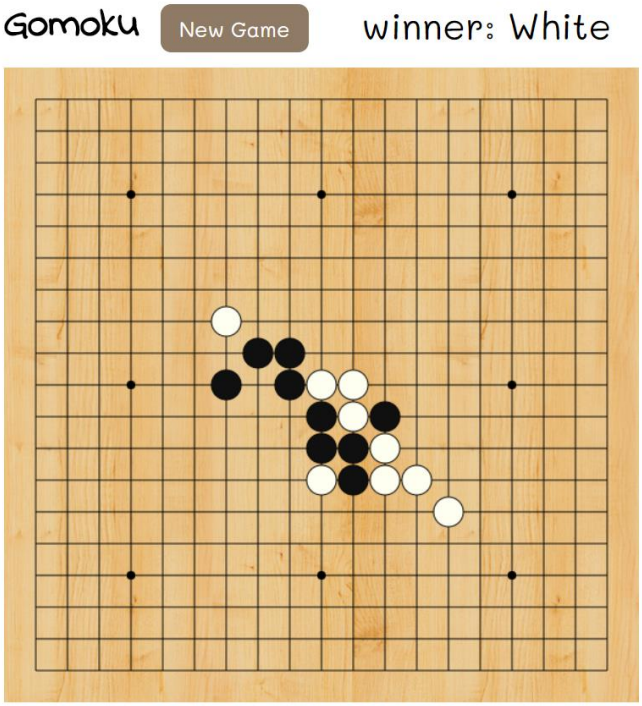
(一)系統功能/摘要

五子棋的對弈系統。主要有人對人的玩法以及人對電腦的玩法。系統主要用 html、css 以及 javascript(有使用 jquery)

遊戲網址：

https://zd3vfm5xzdlkfw5zry40vw-on.drv.tw/gomoku/gomoku.html





(二)系統開發平台

系統同上面所述，沒有使用資料庫。IDE 使用 VScode，網站是架設在 www drive，用第三方軟體把 google drive 當作 server。

(三)程式說明

程式主要分成三部分說明：主要架構、人對人對弈、人對電腦對弈

(1)主要架構：header 的部分：點取 new game 按鈕的時候會呼叫 newgame()，這個 newgame()會把畫布清乾淨，然後執行初始化。初始化的時候會執行 startGame()，使用 jQuery 增加一個半透明的 div(包含兩個按鈕)覆蓋在 canvas 上面。Body 的部分：使用 canvas 畫

出棋盤，DrawLine()畫出 19*19 的棋盤線，DrawPoint()畫出棋盤上的 9 個點，(棋盤為 600px*600px)

```
118 function DrawLine(){
119     ctx.beginPath();
120     for(var i=1;i<20;i++){
121         ctx.moveTo(30*i,30);
122         ctx.lineTo(30*i,570);
123         ctx.moveTo(30,30*i);
124         ctx.lineTo(570,30*i);
125     }
126     ctx.stroke();
127 }
```

```
133 function point(x,y){
134     ctx.beginPath();
135     ctx.arc(x,y,4,0,2*Math.PI,true);
136     ctx.fillStyle="black";
137     ctx.fill();
138 }
139
140 function DrawPoint(){
141     for(var i=0;i<=360;i+=180){
142         for(var j=0;j<=360;j+=180){
143             point(120+i,120+j);
144         }
145     }
146 }
```

(二)人對人對弈：在一開始 gamestart 的時候如果選擇跟人對弈，那麼 human 的參數就會被設成 1，之後每次點擊落子時就會執行下面這段程式碼(此程式碼在 addEventListener("click",function(e){});裡面)

```
157     if(human && enable){  //if play human v.s. human
158         var px = Math.floor((e.offsetX+15)/30)-1;
159         var py = Math.floor((e.offsetY+15)/30)-1;
160         if((px+1)*30==0||(px+1)*30==600||(py+1)*30==0||(py+1)*30==600)
161             return;
162         if(checkerboard[px][py]==0){
163             DrawChess((px+1)*30,(py+1)*30,ChessColor[step%2]);
164             checkerboard[px][py]=ChessColor[step%2];
165             //console.log(px,py,color);
166             setTimeout(CheckWin,100,px,py,ChessColor[step%2],mode[0]);
167             setTimeout(CheckWin,100,px,py,ChessColor[step%2],mode[1]);
168             setTimeout(CheckWin,100,px,py,ChessColor[step%2],mode[2]);
169             setTimeout(CheckWin,100,px,py,ChessColor[step%2],mode[3]);
170             //setTimeout(CheckWinY,100,px,py,ChessColor[step%2]);
171             step++;
172         }
173         //setTimeout(CheckWin,100,px,py,ChessColor[step%2]);
174         //CheckWin(px,py,ChessColor[step%2]);
175     }
```

第 158、159 行的功用是將點擊的座標轉換成 0~18、0~18 的座標(棋盤為 19*19)；第 160 行是當點擊的座標超過棋盤格線範圍，就無效。第 162 行的 checkerboard 是存放落子的二維陣列，如果該點已

經被下過，就不會是零，避免重複落子。另外黑白輪流下的方法是使用 step 計數，每下一步就加一，然後 mod 2 來找對應的顏色。以下說明判斷勝利的演算法(只在人對人的時候使用這種判斷法)。

```
284 function CheckWin(x,y,color,mode){
285     //console.log(x,y,color);
286     var count=0;
287     for(var i=1;i<=5;i++){
288         if(checkerboard[x+i*mode[0]]){
289             if(checkerboard[x+i*mode[0]][y+i*mode[1]]==color)
290                 count++;
291             else break;
292         }
293     }
294     for(var i=1;i<=5;i++){
295         if(checkerboard[x-i*mode[0]]){
296             if(checkerboard[x-i*mode[0]][y-i*mode[1]]==color)
297                 count++;
298             else break;
299         }
300     }
301     //console.log("水平方向有",count+1,"個",color);
302     if(count>=4){
303         // alert(color+"win");
304         enable=!enable;
305         EndGame(color);
306     }
```

首先傳入的 mode 總共會有四種([1,0],[0,1],[1,1],[1,-1])，分別對應判斷水平、垂直、右下左上、右上左下，然後有兩個 for 迴圈(287 行、294 行)，分別檢查落子的兩端，如果落子的兩端相加有超過或等於 4 顆(302 行)，加上落子本身就大於等於 5，遊戲就結束。(enable 的目的是使遊戲結束後不能再繼續下)

(三)人對電腦對弈：人對電腦的勝利條件判斷並非使用上述的演算法，而是從電腦應該如何落子的演算法延伸出來做判斷的。以下說明電腦如何判斷落子位置。首先，我需要建立一個三維陣列(wins[i][j][count])，這個陣列會儲存所有 19*19 的棋盤格上連五顆子的贏法，總共有 1020 種。在 winsINIT 的時候會算出所有可能並初始

化，這邊以橫的贏法(水平)作為例子：wins[0][0][0]、wins[0][1][0]、wins[0][2][0]、wins[0][3][0]、wins[0][4][0]，代表第 0 種贏法，最左上角往右連五顆棋的贏法(第一個陣列代表縱坐標，第二個陣列代表橫坐標，第三個陣列代表第幾種贏法)

```
function winsINIT(){
    // 3 dimension array
    for(var i=0;i<19;i++){
        wins[i]=[];
        for(var j=0;j<19;j++){
            wins[i][j]=[];
        }
    }
    //橫的
    for(var i=0;i<19;i++){
        for(var j=0;j<15;j++){
            for(var k=0;k<5;k++)
                wins[i][j+k][count]=true;
            count++
        }
    }
    //直的
    for(var i=0;i<19;i++){
        for(var j=0;j<15;j++){
            for(var k=0;k<5;k++)
                wins[j+k][i][count]=true;
            count++;
        }
```

建立完所有可能的贏法之後，要再建立兩個陣列(humanWin[]跟computerWin[])，來儲存第幾種贏法已經有幾個子了，舉例來說，如果 humanWin[4]=3，代表人類的第四種贏法已經有三顆子，只要人類再下兩顆子就贏了(也因為如此，可以從這裡判斷輸贏，就不用使用上面的那個演算法)。

最後來說明電腦判斷落子的方式，也就是 computerAI()：執行的時候還要再創立兩個陣列，來存對人類來說下在某個點的分數以及對電腦來說下在某個點的分數。那問題來了，落在某點的分數要如何評

斷呢？就是使用剛剛上面提到的 humanWin[]跟 computerWin[]的數字來當作權重，也就是說如果某種贏法有越多顆子(代表快贏)，就越重要。(參考第 224 行到 239 行)

```
219     for(var i=0;i<19;i++){
220         for(var j=0;j<19;j++){
221             if(checkerboard[i][j]==0){
222                 for(var k=0;k<count;k++){
223                     if(wins[i][j][k]){
224                         if(humanWin[k]==1)  //black
225                             humanScore[i][j]+=200;
226                         else if(humanWin[k]==2)
227                             humanScore[i][j]+=400;
228                         else if(humanWin[k]==3)
229                             humanScore[i][j]+=2000;
230                         else if(humanWin[k]==4)
231                             humanScore[i][j]+=10000;
232                         if(computerWin[k]==1)  //white
233                             computerScore[i][j]+=220;
234                         else if(computerWin[k]==2)
235                             computerScore[i][j]+=420;
236                         else if(computerWin[k]==3)
237                             computerScore[i][j]+=2200;
238                         else if(computerWin[k]==4)
239                             computerScore[i][j]+=20000;
240                     }
241                 }
```

算好權重之後就要開始進行比較，比較方式如下：如果人類落在 i,j 的分數大於目前的分數，那就調整使電腦落在 i,j 的位置上；一樣則比較電腦原本想下的位置跟下 i,j 的位置分數哪個比較高。同理，如果電腦落在 i,j 的分數大於目前的分數，那就調整使電腦落在 i,j 的位置上；一樣則比較如果人類下在電腦原本想下的位置跟下 i,j 的位置分數哪個比較高。

```
if(humanScore[i][j]>best){
    best=humanScore[i][j];
    u=i;
    v=j;
}
else if(humanScore[i][j]==best){
    if(computerScore[i][j]>computerScore[u][v]){
        u=i;
        v=j;
    }
}
if(computerScore[i][j]>best){
    best=computerScore[i][j];
    u=i;
    v=j;
}
else if(computerScore[i][j]==best){
    if(humanScore[i][j]>humanScore[u][v]){
        u=i;
        v=j;
    }
}
```

以上就是判斷落子的演算法大概

(四)結論與心得

這次的期末報告拖太晚才做(自己的時間分配很有問題)，導致上台報告前幾天才生出一個簡單的五子棋遊戲，那時候介面很簡陋，連重新開始遊戲的按鈕都沒有，AI 對弈當然也沒有，當然報告的時候就被電很慘。報告完之後就利用剩下的一個禮拜，添加了 AI，這個 AI 嚴格說起來真的沒有很強很強，期望當然是能用機器學習來做 AI，但以目前我的功力可能還要再練練吧。之後我是希望能夠在讓這個演算法在更快一點，降低時間複雜度(目前是 n^3 吧?)，因為每次落子都要整個盤面重新計算一次，好像有點沒效率。另外這次專題卡在 css 排版其實滿久的，不太熟悉 css 的操作，上網查了很多東西，可能之前用 bootstrap 用習慣就沒甚麼會 css，總結 javascript 比起其他程式語言感覺學起來更有成就感，但做網頁還是很需要美術天分阿