哈尔滨工业大学

<<计算机图形学>>

大作业报告

(2018 年度春季学期)

姓名：	王陈阳
学号：	1150310609
学院：	计算机科学与技术学院
教师：	唐好选

小游戏项目：五子棋

一、 项目基本信息

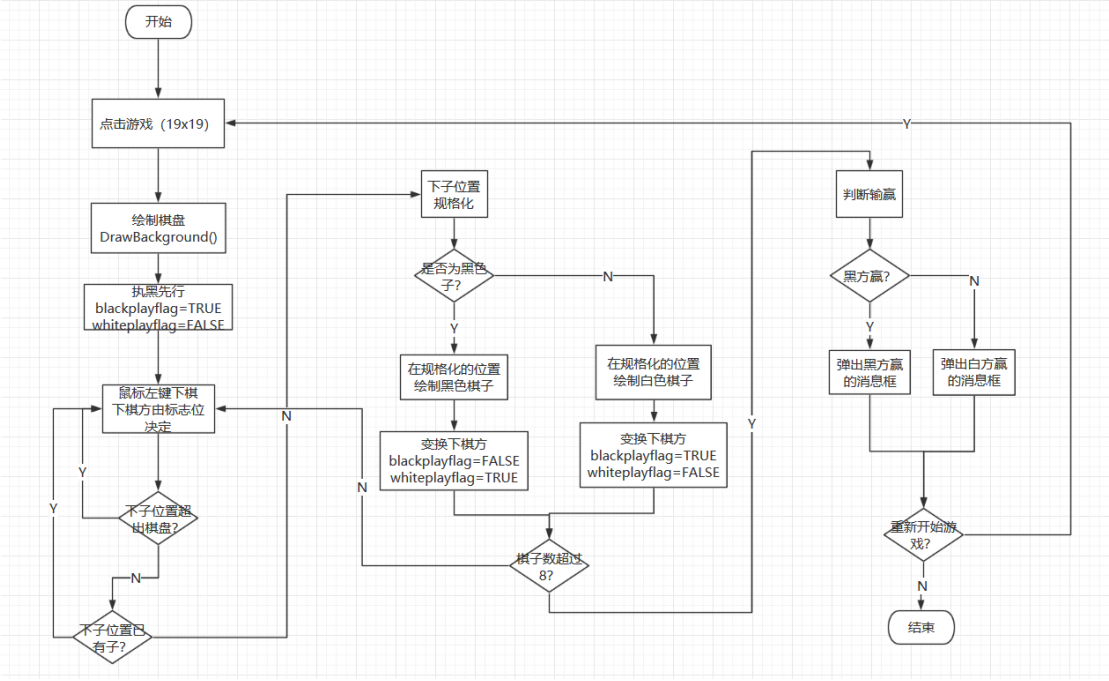
1、 项目题目：五子棋

2、 项目环境：win10 操作系统，vs2017,MFC 框架

3、 项目基本信息：通过 VS MFC 框架编程编制五子棋程序，棋盘 19*19 大小，执黑先行，黑白交替，当一方出现五个子连成一条线，即判断胜方，弹出消息框，显示胜方，可重新开始游戏。

二、 项目基本结构

程序流程图：



数据结构：

chesspoint(棋子逻辑结点)

```
struct chesspoint {

    int chesscolor;  //棋子颜色:1 黑色，-1 白色

    bool chessuseflag;   //棋子是否被占用

    int chessflag[8];//棋子标志，一个棋子周围的 8 个位置，标识当前组成的线段最大长度,0 号为左上角，顺时针计数

};
```

主要函数：

```
// 绘制棋盘背景

void DrawBackground();

// 绘制棋子,绘制成功返回 1，失败返回-1

int DrawChessPoint();

// 规格化落子位置,正常返回 1，子越界返回-1，该位置已落子返回-2

int StandardPointPosition();

// 绘制一个特定颜色的棋子

void DrawPoint(int color);

// 添加棋子（logic_x,logic_y）到棋子逻辑数组中，并动态更新棋子连线信息

void AddChessPoint(int logic_x,int logic_y);
```

程序结构主要分为以下几个部分：

1、菜单部分

（1）、游戏（子菜单：19x19）——开始游戏；

（2）、重新开始（子菜单：初始化）——初始化游戏（棋盘未绘制）

2、绘制棋盘部分（void DrawBackground()）

以（500，500）为中心绘制 19x19 的棋盘，每个棋盘结点间距为 50，共 361 个结点

3、下子部分

（1）、设置 blackplayflag、whiteplayflag 作为交替下子的标志位，一真一假，完成下子后交替互换；

（2）、设置 CPoint 类型的向量 player_black、player_white 存储鼠标左键下子的坐标值，用 black_pointNum、white_pointNum、chess_pointNum 分别记录黑子数，白子数，总棋子数。

4、规格化下子位置部分（int StandardPointPosition()）

（1）、下子位置超出棋盘边界：删除添加的相应颜色的棋子，棋子数相应减一，返回-1。

（2）、将棋子位置化成（i*50,j*50）的形式，即下子位置若在某节点为中心，50 为边长的正方形内时，下子位置更新为该节点位置（将下子位置划归到离散有限点上），返回 1。

（3）、由第二步可获得棋子的逻辑位置（i，j），使用自定义数据结构 chesspoint 的一个 19x19 的数组 chess_logic 存储棋子的逻辑信息（在 void AddChessPoint(int logic_x,int logic_y)函数中进行修改）：①：int chesscolor; //棋子颜色:1 黑色，-1 白色；②bool chessuseflag; //棋子是否被占用；③int chessflag[8];//棋子标志，一个棋子周围的 8 个位置，标识当前组成的线段最大长度,0 号为左上角，顺时针计数。若（i，j）位置的 chessuseflag 标志位为 TRUE 则表明该位置已被占用，删除添加的相应颜色的棋

子，棋子数相应减一，返回-2。

（4)、若规格化完成，将棋子逻辑逻辑位置添加到棋子逻辑数组中。

5、添加棋子到棋子逻辑数组中，并动态更新棋子连线信息（void AddChessPoint(int logic_x,int logic_y)）

设置相应 chesspoint 的结点信息；

① int chesscolor; //棋子颜色:1 黑色，-1 白色；

② bool chessuseflag; //棋子是否被占用；

③ int chessflag[8]; //棋子标志，一个棋子周围的 8 个位置，标识当前组成的线段最大长度,0 号为左上角，顺时针计数。

关键算法：动态更新每个逻辑棋子结点的 chessflag[8]，用来标识当前连线长度。

基本思想：以 0 号标志位（左上角）为例，当前棋子位置为（i，j),若（i-1.j-1)位置棋子的 chesscolor 与(i,j)一致,则 chess_logic[i][j].chessflag[0] = chess_logic[i- 1][j- 1].chessflag[0]+1(即左上方连线长度加一)，否则，置为 1，表示左上方连线长度为 1，其他方向相同。

6、绘制特定颜色的棋子

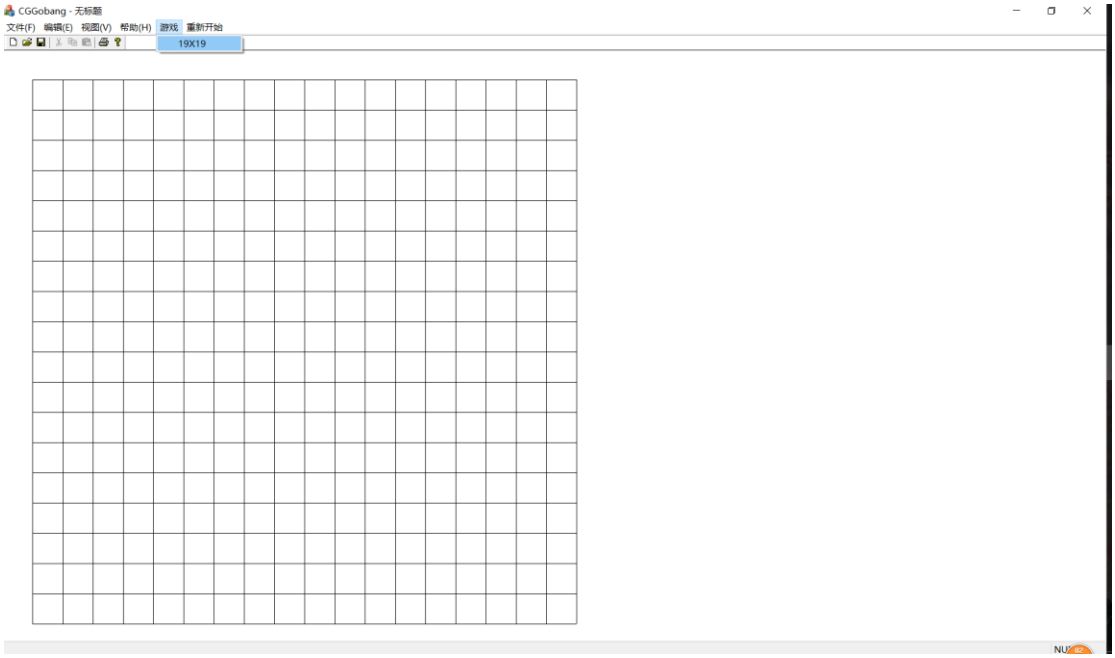
利用 MFC 绘制特定颜色圆的方法绘制棋子。

7、判断输赢部分

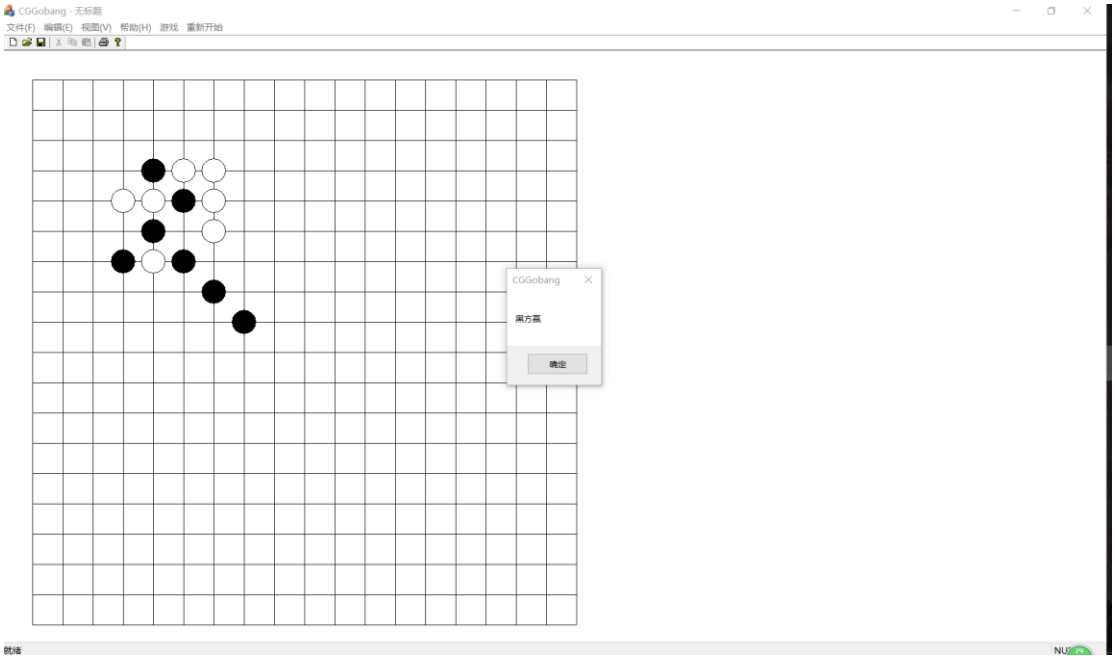
当棋子总数 chess_pointNum 大于 8 时(即棋盘上至少有五个黑子,4 个白子时)，开始对最后落下的子进行判断，检查其逻辑棋子中的 chessflag[8]部分，若有一个大于等于 5，即说明有五个颜色相同的子连成直线，由该子的颜色断定胜利的是黑方还是白方，弹出消息框。

三、 项目演示

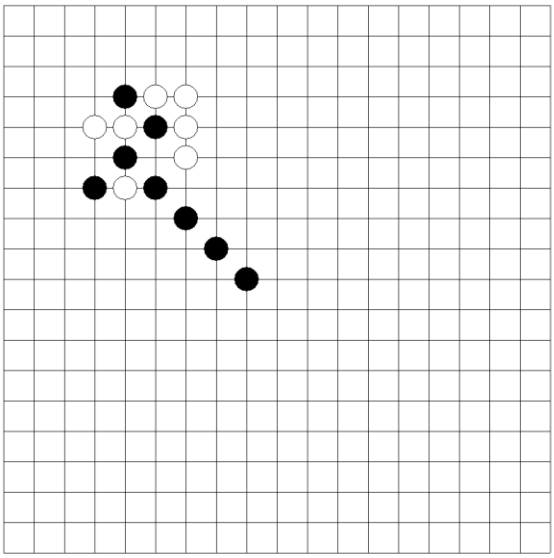
1、 选择游戏（19x19）,开始游戏；



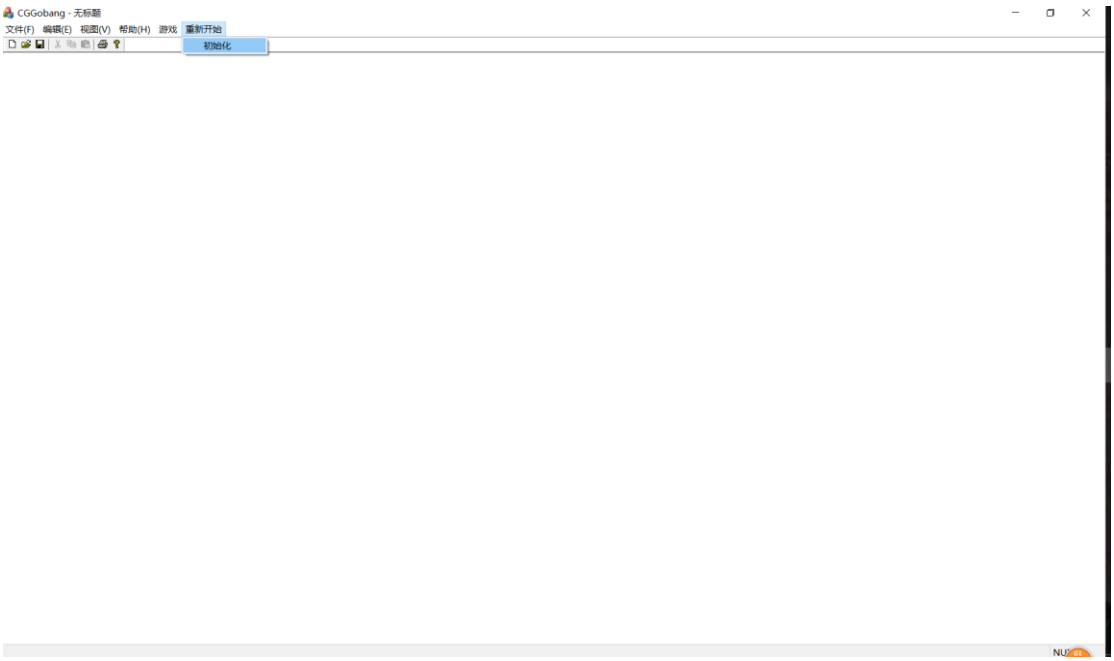
2、 开始下棋，简单演示。



点击确定后



3、　　重新开始



四、 项目心得

经过本次项目编写，收获如下：

1、 对 MFC 的框架编程更加熟练了，自己的编程能力也有了提升。

2、 在添加黑白棋子，绘制棋子的过程中发现了很多逻辑上的问题，通过不断检查调试，发现在重复绘制棋子的过程中绘制完成后没有返回，导致之前绘制的被覆盖，出现了比较大的问题，通过解决这个问题，对于 MFC 的绘图有了比较深刻的理解。

3、 该项目的核心部分是判断哪一方胜利的算法的设计，最开始使用的是遍历算法，通过遍历每一条线，寻找相邻的五个颜色相同的棋子，发现算法思想简单，但时间空间复杂度较高，于是开始算法的改造，设计了特殊的数据结构 chesspoint，通过动态更新棋子标志位的方法将棋子的连线信息存储在标志位中，使得遍历寻找的过程变成了查看当前棋子的标志位是否有等于 5 的判断，使得算法的时间复杂度下降，理解起来也比较容易，这样的思考过程对自己的提升也很大。