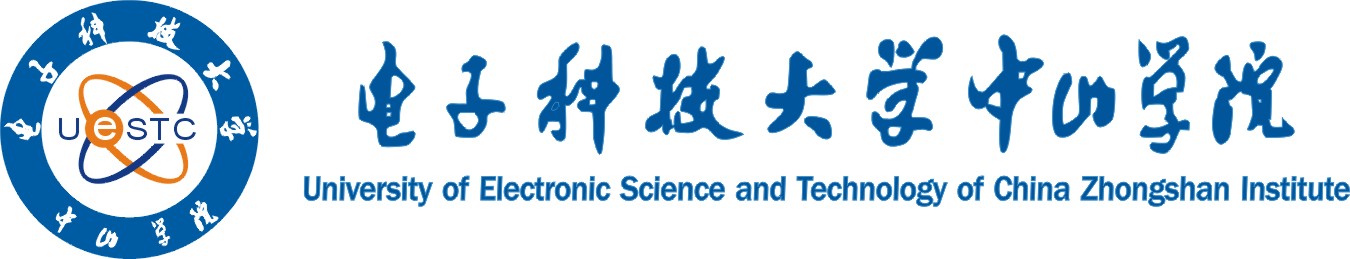
**课程设计说明书**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 学院 | | 计算机学院 | | | 班别 | 17软件B班 | | 电子照片 | |
| 姓名 | 陈淑婷 | | | | 学号 | 2017030401048 | |
| 电话 | 15362994620 | | | | 其它成员 | 邱子豪 | |
| 教师评价 | 难度 | | | 完成情况 | 答辩情况 | 课程报告 | | 综合成绩 | |
| 优（ ）  良( )  中( )  及格( )  不及格( ) | | | 优（ ）  良( )  中( )  及格( )  不及格( ) | 优（ ）  良( )  中( )  及格( )  不及格( ) | 优（ ）  良( )  中( )  及格( )  不及格( ) | | 优（ ）  良( )  中( )  及格( )  不及格( ) | |
| 主  要  工  作  内  容 | 1.负责了整理报告、总结代码等工作；  2.负责了程序界面的设计和控件的使用；  3.负责了五子棋的棋盘和棋子的绘制；  4.负责了全局变量及相关函数的设计；  5.负责了计时器的函数设计；  6.负责了鼠标移动时跟踪光标的设计；  7.参与了电脑自动落子及判断胜负的函数设计；  8.参与了玩家落子、悔棋及禁手的设计；  9.参与了总体设计。  团队分工：  组长：陈淑婷 组员：邱子豪 | | | | | | | | |
| 课程名称 | | | C#程序设计 | | | | 指导教师 | | 黄敏 |
| 课程设计题目 | | | 五子棋游戏 | | | | | | |
| 课程设计时间 | | | 第7周至第8周 | | | | | | |

**目 录**

[一、 课程设计内容 3](#_Toc12026_WPSOffice_Level1)

[二、 课程设计目的 3](#_Toc22117_WPSOffice_Level1)

[三、 工具/准备工作 3](#_Toc15835_WPSOffice_Level1)

[四、 设计步骤和方法 3](#_Toc8233_WPSOffice_Level1)

[（一） 总体设计 3](#_Toc22117_WPSOffice_Level2)

[1． 总体设计思路及设计图 3](#_Toc22117_WPSOffice_Level3)

[2． 界面设计 4](#_Toc15835_WPSOffice_Level3)

[3． 全局变量设计 4](#_Toc8233_WPSOffice_Level3)

[（二） 详细设计 5](#_Toc15835_WPSOffice_Level2)

[1． 刷新棋盘 5](#_Toc25120_WPSOffice_Level3)

[2． 绘制棋盘 5](#_Toc2251_WPSOffice_Level3)

[3． 分步计时 5](#_Toc6537_WPSOffice_Level3)

[4． 显示光标 6](#_Toc14782_WPSOffice_Level3)

[5． 判断胜负 8](#_Toc32707_WPSOffice_Level3)

[6． 判断禁手 10](#_Toc8184_WPSOffice_Level3)

[7． 电脑落子 15](#_Toc30203_WPSOffice_Level3)

[8． 玩家落子 18](#_Toc26555_WPSOffice_Level3)

[9． 开始游戏 19](#_Toc12302_WPSOffice_Level3)

[10． 退出游戏 20](#_Toc21388_WPSOffice_Level3)

[11． 重新开局 20](#_Toc13346_WPSOffice_Level3)

[12． 玩家认输 21](#_Toc12255_WPSOffice_Level3)

[13． 玩家悔棋 22](#_Toc10264_WPSOffice_Level3)

[（三） 测试 23](#_Toc8233_WPSOffice_Level2)

[1． 判断胜负模块测试 23](#_Toc29006_WPSOffice_Level3)

[2． 判断禁手模块测试 23](#_Toc25463_WPSOffice_Level3)

[五、 心得体会 24](#_Toc25120_WPSOffice_Level1)

1. **课程设计内容**

在Visual Studio 2015 平台上，开发一个“五子棋游戏”的Windows Form应用程序，题目要求：

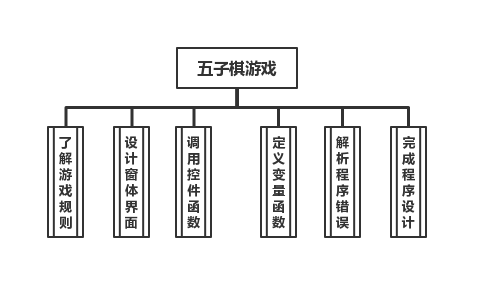
1. 整体规划，即画出棋盘和显示出“游戏开始”、“悔棋”、“退出游戏”等按钮。
2. 游戏界面的具体实现，即有绝对坐标与相对坐标、又有界面的颜色与大小和各部分所处的位置。
3. 在鼠标在棋盘上经过的时候，可以看出光标在那里；
4. 正常落棋子，不会覆盖已有棋子。
5. 记录游戏中棋子的落子过程，并且允许悔棋
6. 判断双方输赢，先走出五子连珠的一方赢
7. 先下者落子时不允许走禁手。
8. **课程设计目的**

综合运用c#.net知识，在vs 2015 平台上，进行控制台应用程序和简单Windows Form应用程序的开发；初步熟悉开发一个Windows软件的流程。

1. **工具/准备工作**

Microsoft Visual Studio 2015，PC机一台，上网查找资料，相关参考书籍。

1. **设计步骤和方法**
2. **总体设计**
3. **总体设计思路及设计图**
4. 设计思路
5. 充分了解五子棋游戏的一般规则，了解三三禁手、四四禁手和长连禁手的判定条件，以及黑白方如何落子、如何计时；
6. 结合游戏规则设计简洁美观的游戏界面，使用相关控件，注意控件之间的联系；
7. 搜集资料，了解控件有关的函数及使用方法，完善程序逻辑；
8. 根据拟定好的程序逻辑，定义相关的全局变量和自定义函数；
9. 模拟多种情况调试程序，找出隐藏的错误，完善程序；
10. 整合逻辑及相关程序，完成课程设计报告。
11. 总体设计图



1. **界面设计**

使用了splitContainer控件和panel控件将Form1分割为panel1和panel2两部分；

panel1包含groupBox控件区分“下棋顺序”和“游戏开关”，randioButton控件控制玩家先手或者电脑先手，Button控件设置了游戏的开关——“开始”、“退出”、“重开”、“悔棋”、“认输”，Timer控件计算黑白双方每下一步的时间；

panel2使用GDI+，在代码中新建了一块画布，每次加载窗体时自动绘画棋盘，再通过函数的控制，实现用户点击panel2的区域可以完成落子操作。



1. **全局变量设计**

static int first = 32, last = 452, step = 30;//棋盘数据

static int half = step / 2;//红框

static int blackTime = 0;

static int whiteTime = 0;

static int[,] chessColor;//棋子颜色

static bool isBlack = true;//黑白执棋

static bool isBegin = false;//开始游戏

static bool[,] hasChess;//是否落子

static bool computerState;

static bool isFirst;

static bool isEnd;

static Pen pen = new Pen(Color.Black, 2);//棋盘画笔黑

static Pen pen2 = new Pen(Color.Red, 2);//光标画笔红

static Point mousePosition = new Point(0, 0);//当前鼠标

static Point lastChessPosition;//上一颗玩家棋子

static Point lastChessPosition2; //上一颗电脑棋子

static Point p1, p2, p3, p4, p5, p6, p7, p8;//补充棋盘黑线

1. **详细设计**
2. **刷新棋盘**

控制相关变量的恢复初始化，点击“开始”以及“重开”按钮时调用。

public void initBoard()

{

panel2.Refresh();

button3.Enabled = button4.Enabled = button5.Enabled = false;

isBegin = false;

isBlack = true;

}

1. **绘制棋盘**

通过panel2\_Paint()调用board()函数。

public void board()

{

Graphics g = panel2.CreateGraphics();

for (int i = first; i <= last; i += step)

{

Point start = new Point(i, first);

Point end = new Point(i, last);

g.DrawLine(pen, start, end);

start = new Point(first, i);

end = new Point(last, i);

g.DrawLine(pen, start, end);

}

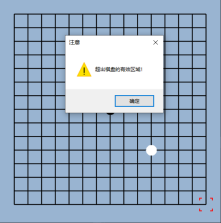
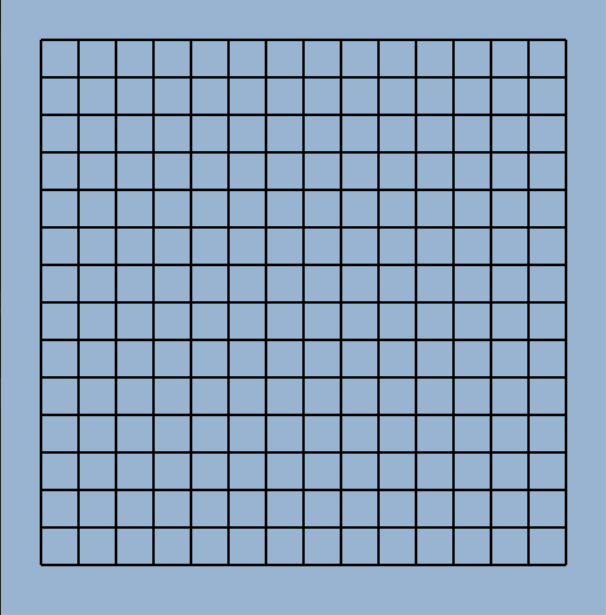
}

private void panel2\_Paint(object sender, PaintEventArgs e)

{

board();

}



1. **分步计时**

点击panel2落子，根据下一个落子的棋子颜色（黑/白）控制两个textBox的显示，每60秒交换玩家落子顺序。

private void timer1\_Tick(object sender, EventArgs e)

{

if (isEnd)

{

blackTime = 0;

whiteTime = 0;

}

else

{

if (isBlack)

{

blackTime++;

if (blackTime == 60)

{

isBlack = !isBlack;

blackTime = 0;

}

}

else

{

whiteTime++;

if (whiteTime == 60)

{

isBlack = !isBlack;

whiteTime = 0;

}

}

}

textBox1.Text = blackTime >= 10 ? blackTime.ToString() : 0.ToString() + blackTime.ToString();

textBox2.Text = whiteTime >= 10 ? whiteTime.ToString() : 0.ToString() + whiteTime.ToString();

}



1. **显示光标**

利用GDI+绘画在panel2棋盘上捕获鼠标位置（绝对坐标,panel2\_MouseMove()函数），转化为相应的棋盘坐标（相对坐标），画出红色的方框，再添加背景色的横竖线以及补充棋盘遮挡住的黑线（fillBoard()函数），最后实时更新鼠标坐标（update()函数），重复上述过程。

public void update(int x0, int y0)

{

//画出鼠标所在点的红框

Graphics g = panel2.CreateGraphics();

int cenX = first + x0 \* step;

int cenY = first + y0 \* step;

g.DrawRectangle(pen2, new Rectangle(new Point(cenX - half + 1, cenY - half + 1), new Size(step - 2, step - 2)));

//将红框修改为四个直角的边框

Pen pen3 = new Pen(panel2.BackColor, 2);

int a = 5;

p1 = new Point(cenX - half + 1 + a, cenY - half + 1); //n

p2 = new Point(cenX - half + 1 + step - 2 - a, cenY - half + 1);

p3 = new Point(cenX - half + 1, cenY - half + 1 + a); //w

p4 = new Point(cenX - half + 1, cenY - half + 1 + step - 2 - a);

p5 = new Point(cenX - half + 1 + a, cenY - half + 1 + step - 2);//s

p6 = new Point(cenX - half + 1 + step - 2 - a, cenY - half + 1 + step - 2);

p7 = new Point(cenX - half + 1 + step - 2, cenY - half + 1 + a);//e

p8 = new Point(cenX - half + 1 + step - 2, cenY - half + 1 + step - 2 - a);

g.DrawLine(pen3, p1, p2);

g.DrawLine(pen3, p3, p4);

g.DrawLine(pen3, p5, p6);

g.DrawLine(pen3, p7, p8);

//补充黑色棋盘线

fillBoard(cenX, cenY, g);

//取消红框，将红框置为背景色

cenX = first + mousePosition.X \* step;

cenY = first + mousePosition.Y \* step;

g.DrawRectangle(pen3, new Rectangle(new Point(cenX - half + 1, cenY - half + 1), new Size(step - 2, step - 2)));

//补充黑色棋盘线

fillBoard(cenX, cenY, g);

}

public void fillBoard(int x, int y, Graphics g)

{

int t = 2;

p1 = new Point(x, y - half); //n

p2 = new Point(x, y - half + t);

p3 = new Point(x, y + half); //s

p4 = new Point(x, y + half - t);

p5 = new Point(x - half, y);//w

p6 = new Point(x - half + t, y);

p7 = new Point(x + half, y);//e

p8 = new Point(x + half - t, y);

if (y != first) g.DrawLine(pen, p1, p2);

if (y != last) g.DrawLine(pen, p3, p4);

if (x != first) g.DrawLine(pen, p5, p6);

if (x != last) g.DrawLine(pen, p7, p8);

}

private void panel2\_MouseMove(object sender, MouseEventArgs e)

{

if (!isBegin) return;

int x = (e.X - first) / step;

int y = (e.Y - first) / step;

int leftX = (e.X - first) % step;

int leftY = (e.Y - first) % step;

if (leftX > step / 2)

x += 1;

if (leftY > step / 2)

y += 1;

if (e.X < first - half || e.X > last + half || e.Y < first - half || e.Y > last + half) return;

if (hasChess[x, y]) return;

if (x != mousePosition.X || y != mousePosition.Y)

{

if (!hasChess[x, y])

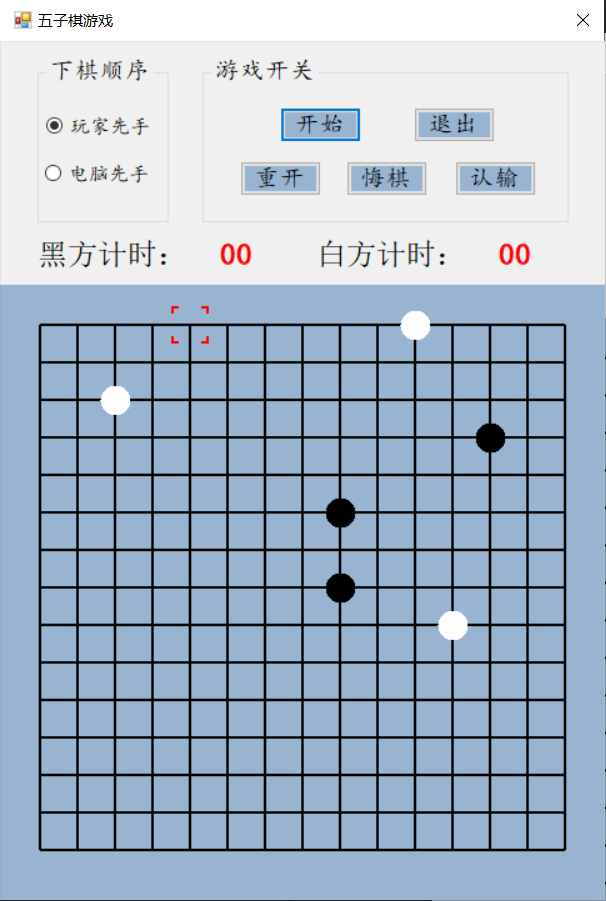
update(x, y);

mousePosition.X = x;

mousePosition.Y = y;

}

}



1. **判断胜负**

遍历当前棋子坐标的行上、列上、左斜线上、右斜线上的棋子数，若棋子数达到5颗，则胜利。

public void judge(int x, int y, int chessState)

{

int scoreCol = 0, scoreRow = 0, scoreLeftXie = 0, scoreRightXie = 0;

int i, j, k, color;

if (chessState == 0) color = 1; else color = 2;

//col

i = y;

j = 0;

while (i > 0 && chessColor[x, --i] == color) ++j;

i = y;

while (i < 14 && chessColor[x, ++i] == color) ++j;

scoreCol = j + 1;

//Row

i = x;

j = 0;

while (i > 0 && chessColor[--i, y] == color) ++j;

i = x;

while (i < 14 && chessColor[++i, y] == color) ++j;

scoreRow = j + 1;

//LeftXie

i = x;

j = y;

k = 0;

while (i > 0 && j > 0 && chessColor[--i, --j] == color) ++k;

i = x;

j = y;

while (i < 14 && j < 14 && chessColor[++i, ++j] == color) ++k;

scoreLeftXie = k + 1;

//RightXie

i = x;

j = y;

k = 0;

while (i < 14 && j > 0 && chessColor[++i, --j] == color) ++k;

i = x;

j = y;

while (i > 0 && j < 14 && chessColor[--i, ++j] == color) ++k;

scoreRightXie = k + 1;

//贪心取最高连子数

int ans = Math.Max(Math.Max(scoreCol, scoreRow), Math.Max(scoreLeftXie, scoreRightXie));

if (ans == 5)

{

String winner;

if (!isBlack) winner = "白方";

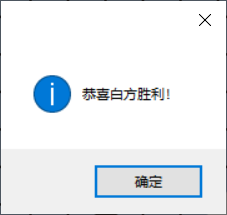
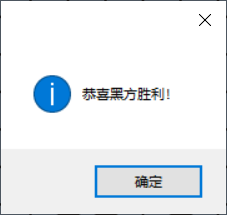
else winner = "黑方";

MessageBox.Show("恭喜" + winner + "胜利！", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

isEnd = true;

}

}



1. **判断禁手**

枚举可能出现的禁手的情况，若当前棋子所下位置为禁手，则弹框警告且不允许落子。

public bool stop(int x, int y)

{

int i, j, zj, yj, sj, xj, zxsj, zxxj, yxsj, yxxj;

int x1j, z1j, y1j,s1j, zxs1j,zxx1j,yxs1j,yxx1j; //空一格

bool z101j, y101j, s101j, x101j, zxs101j, zxx101j, yxs101j, yxx101j;

bool z10112j, z0112j, y10112j, y0112j, s10112j, s0112j, x10112j, x0112j, zxs10112j, zxs0112j, zxx10112j, zxx0112j, yxs10112j, yxs0112j, yxx10112j, yxx0112j;

//zj yj

i = x; zj = 0;yj = 0;

while (i > 0 && chessColor[--i, y] == 1) ++zj;

i = x;

while (i < 14 && chessColor[++i, y] == 1) ++yj;

//x1j

i = y + 1; x1j = 0;

while (i < 14 && chessColor[x, ++i] == 1) ++x1j;

//s1j

i = y - 1; s1j = 0;

while (i > 0 && chessColor[x, --i] == 1) ++s1j;

//z1j

i = x - 1; z1j = 0;

while (i > 0 && chessColor[--i, y] == 1) ++z1j;

//y1j

i = x + 1; y1j = 0;

while (i < 14 && chessColor[++i, y] == 1) ++y1j;

//zxs1j

i = x - 1; j = y - 1; zxs1j = 0;

while (i >0&&j>0 && chessColor[--i, --j] == 1) ++zxs1j;

//zxx1j

i = x - 1; j = y + 1; zxx1j = 0;

while (i>0&&j<14 && chessColor[--i, ++j] == 1) ++zxx1j;

//yxs1j

i = x + 1; j = y - 1; yxs1j = 0;

while (i < 14 && j > 0 && chessColor[++i, --j] == 1) ++yxs1j;

//yxx1j

i = x + 1; j = y + 1; yxx1j = 0;

while (i < 14 && j < 14 && chessColor[++i, ++j] == 1) ++yxx1j;

//sj xj

i = y;

sj = 0; xj = 0;

while (i > 0 && chessColor[x, --i] == 1) ++sj;

i = y;

while (i < 14 && chessColor[x, ++i] == 1) ++xj;

//zxsj zxxj yxsj yxxj

i = x; j = y; yxsj = 0; zxxj = 0; yxxj = 0;zxsj = 0;

while (i > 0 && j > 0 && chessColor[--i, --j] == 1) ++zxsj;

i = x;j = y;

while (i > 0 && j < 14 && chessColor[--i, ++j] == 1) ++zxxj;

i = x; j = y;

while (i < 14 && j > 0 && chessColor[++i, --j] == 1) ++yxsj;

i = x; j = y;

while (i < 14 && j < 14 && chessColor[++i, ++j] == 1) ++yxxj;

//z101j

i = x;z101j = false;

if (i>2&&chessColor[i - 1, y] == 1 && chessColor[i - 2, y] == 0 && chessColor[i - 3, y] == 1) z101j = true;

//y101j

i = x; y101j = false;

if (i<12&&chessColor[i + 1, y] == 1 && chessColor[i + 2, y] == 0 && chessColor[i + 3, y] == 1) y101j = true;

//s101j

i = y; s101j = false;

if (i>2&&chessColor[x, i-1] == 1 && chessColor[x, i-2] == 0 && chessColor[x, i-3] == 1) s101j = true;

//x101j

i = y; x101j = false;

if (i<12&&chessColor[x, i + 1] == 1 && chessColor[x, i + 2] == 0 && chessColor[x, i + 3] == 1) x101j = true;

//zxs101j

i = x; j=y; zxs101j = false;

if (i>2&&j>2&&chessColor[i-1, j-1] == 1 && chessColor[i-2, j - 2] == 0 && chessColor[i-3, j - 3] == 1) zxs101j = true;

//zxx101j

i = x; j = y; zxx101j = false;

if (i>2&&j<12&&chessColor[i - 1, j + 1] == 1 && chessColor[i - 2, j + 2] == 0 && chessColor[i - 3, j + 3] == 1) zxx101j = true;

//yxs101j

i = x;j = y;yxs101j = false;

if (i<12&&j>2&&chessColor[i + 1, j - 1] == 1 && chessColor[i + 2, j - 2] == 0 && chessColor[i + 3, j - 3] == 1) yxs101j = true;

//yxx101j

i = x; j = y; yxx101j = false;

if (x<12&&y<12&&chessColor[i + 1, j + 1] == 1 && chessColor[i + 2, j + 2] == 0 && chessColor[i + 3, j + 3] == 1) yxx101j = true;

//z10112j

i = x;z10112j = false;

if (i > 4 && chessColor[i - 1, y] == 1 && chessColor[i - 2, y] == 0 && chessColor[i - 3, y] == 1 && chessColor[i - 4, y] == 1 && chessColor[i - 5, y] == 2) z10112j = true;

//z0112j

i = x;z0112j = false;

if (i > 3 && chessColor[i - 1, y] == 0 && chessColor[i - 2, y] == 1 && chessColor[i - 3, y] == 1 && chessColor[i - 4, y] == 2) z0112j = true;

//y10112j

i = x;y10112j = false;

if (i < 10 && chessColor[i + 1, y] == 1 && chessColor[i + 2, y] == 0 && chessColor[i + 3, y] == 1 && chessColor[i + 4, y] == 1 && chessColor[i + 5, y] == 2) y10112j = true;

//y0112j

i = x;y0112j = false;

if (i < 11 && chessColor[i + 1, y] == 0 && chessColor[i + 2, y] == 1 && chessColor[i + 3, y] == 1 && chessColor[i + 4, y] == 2) y0112j = true;

//s10112j

i = y;s10112j = false;

if (i > 4 && chessColor[x, i - 1] == 1 && chessColor[x, i - 2] == 0 && chessColor[x, i - 3] == 1 && chessColor[x, i - 4] == 1 && chessColor[x, i - 5] == 2) s10112j = true;

//s0112j

i = y;s0112j = false;

if (i > 3 && chessColor[x, i - 1] == 0 && chessColor[x, i - 2] == 1 && chessColor[x, i - 3] == 1 && chessColor[x, i - 4] == 1) s0112j = true;

//x10112j

i = y;x10112j = false;

if (i < 10 && chessColor[x, i + 1] == 1 && chessColor[x, i + 2] == 0 && chessColor[x, i + 3] == 1 && chessColor[x, i + 4] == 1 && chessColor[x, y + 5] == 2) x10112j = true;

//x0112j

i = y;x0112j = false;

if (i < 11 && chessColor[x, i + 1] == 0 && chessColor[x, i + 2] == 1 && chessColor[x, i + 3] == 1 && chessColor[x, i + 4] == 2) x0112j = true;

//zxs10112j

i = x;j = y;zxs10112j = false;

if (i > 4 && j > 4 && chessColor[i - 1, j - 1] == 1 && chessColor[i - 2, j - 2] == 0 && chessColor[i - 3, j - 3] == 1 && chessColor[i - 4, j - 4] == 1 && chessColor[i - 5, j - 5] == 2) zxs10112j = true;

//zxs0112j

i = x;j = y;zxs0112j = false;

if (i > 3 && j > 3 && chessColor[i - 1, j - 1] == 0 && chessColor[i - 2, j - 2] == 1 && chessColor[i - 3, j - 3] == 1 && chessColor[i - 4, j - 4] == 2) zxs0112j = true;

//zxx10112j

i = x;j = y;zxx10112j = false;

if (i > 4 && j < 10 && chessColor[i - 1, j + 1] == 1 && chessColor[i - 2, j + 2] == 0 && chessColor[i - 3, j + 3] == 1 && chessColor[i - 4, j + 4] == 1 && chessColor[i - 5, j + 5] == 2) zxx10112j = true;

//zxx0112j

i = x; j = y; zxx0112j = false;

if (i > 4 && j < 10 && chessColor[i - 1, j + 1] == 0 && chessColor[i - 2, j + 2] == 1 && chessColor[i - 3, j + 3] == 1 && chessColor[i - 4, j + 4] == 2) zxx0112j = true;

//yxs10112j

i = x; j = y; yxs10112j = false;

if (i < 10 && j >4 && chessColor[i + 1, j - 1] == 1 && chessColor[i + 2, j - 2] == 0 && chessColor[i + 3, j - 3] == 1 && chessColor[i + 4, j - 4] == 1 && chessColor[i + 5, j - 5] == 2) yxs10112j = true;

//yxs0112j

i = x; j = y; yxs0112j = false;

if (i < 11 && j > 3 && chessColor[i + 1, j - 1] == 0 && chessColor[i + 2, j - 2] == 1 && chessColor[i + 3, j - 3] == 1 && chessColor[i + 4, j - 4] == 2) yxs0112j = true;

//yxx10112j

i = x; j = y; yxx10112j = false;

if (i < 10 && j < 10 && chessColor[i + 1, j + 1] == 1 && chessColor[i + 2, j + 2] == 0 && chessColor[i + 3, j + 3] == 1 && chessColor[i + 4, j + 4] == 1 && chessColor[i + 5, j + 5] == 2) yxx10112j = true;

//yxx0112j

i = x; j = y; yxx0112j = false;

if (i < 11 && j < 11 && chessColor[i + 1, j + 1] == 0 && chessColor[i + 2, j + 2] == 1 && chessColor[i + 3, j + 3] == 1 && chessColor[i + 4, j + 4] == 2) yxx0112j = true;

bool if1, if2, if3, if4;

//长连禁手

if1 = (zj == 2 && yj == 3) || (zj == 3 && yj == 2) || (sj == 2 && xj == 3) || (sj == 3 && xj == 2) || (zxsj == 2 && yxxj == 3) || (zxsj == 3 && yxxj == 2) || (yxsj == 2 && yxxj == 3) || (yxsj == 3 && yxxj == 2);

if (if1)

{

MessageBox.Show("长连禁手！此步不允许走！", "警告", MessageBoxButtons.OK, MessageBoxIcon.Information);

return true;

}

//三三禁手

if1 = (zj == 2 && x1j == 2) || (xj == 2 && y1j == 2) || (yj == 2 && s1j == 2) || (sj == 2 && z1j == 2);

if2 = (zxsj == 1 && yxsj == 1 && zxxj == 1 && yxxj == 1) || (zj==1&&xj==1&&yj==1&&sj==1);

if (if1 || if2)

{

MessageBox.Show("三三禁手！此步不允许走！", "警告", MessageBoxButtons.OK, MessageBoxIcon.Information);

return true;

}

//四四禁手

if1 = (sj == 1 && xj == 2 && zj == 1 && yj == 2) || (sj == 2 && xj == 1 && zj == 2 && yj == 1);

if2 = (zj == 3 && xj == 3) || (xj == 3 && yj == 3) || (yj == 3 && sj == 3) || (sj == 3 && zj == 3);

if3 = (z10112j && y0112j) || (z0112j && y10112j) || (s10112j && x0112j) || (s0112j && x10112j) && (zxs10112j && yxx0112j) || (zxs0112j && yxx10112j) || (yxs0112j && yxx10112j) || (yxs10112j && yxx0112j);

if4 = (z101j && y101j) || (s101j && x101j) || (zxs101j && yxx101j) || (zxx101j && yxs101j);

if (if1 || if2 || if3 ||if4)

{

MessageBox.Show("四四禁手！此步不允许走！", "警告", MessageBoxButtons.OK, MessageBoxIcon.Information);

return true;

}

//四三三禁手

if1 = (zj == 2 && zxxj == 2 && yxsj == 1 && yxxj == 2) || (yj == 2 && zxsj == 1 && zxxj == 2 && yxxj == 2) || (yj == 2 && zxxj == 1 && zxsj == 2 && yxsj == 2) || (zj == 2 && yxxj == 1 && zxsj == 2 && yxsj == 2);

if (if1)

{

MessageBox.Show("四三三禁手！此步不允许走！", "警告", MessageBoxButtons.OK, MessageBoxIcon.Information);

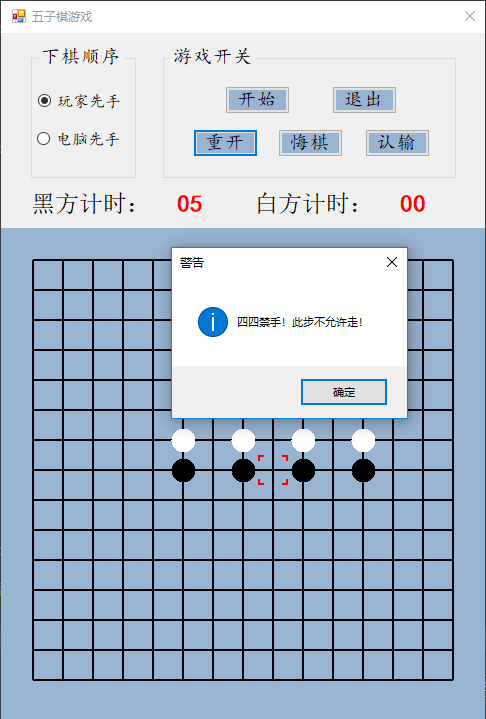
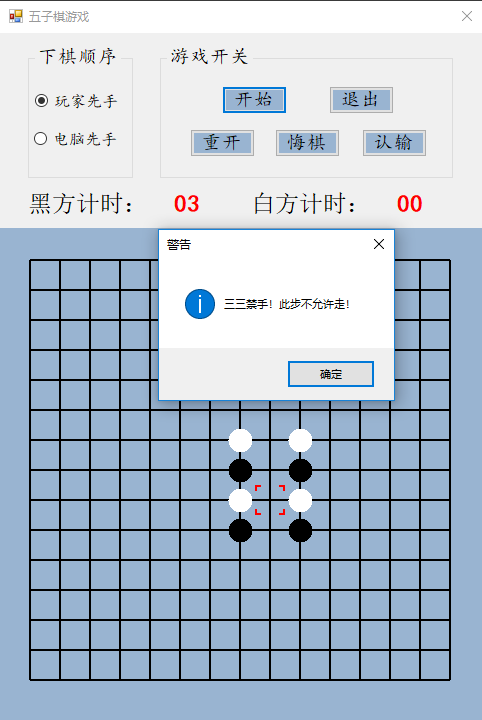
return true;

}

//位置合法，不需要禁手

return false;

}



1. **电脑落子**

先判断是否电脑先手，如果是，第一颗黑子应落在天元点上（chess(7,7)），否则，通过算法分析玩家棋子可能胜利的情况来决定电脑落子点，阻止玩家胜利。

public void where(ref int x, ref int y)

{

int scoreCol = 0, scoreRow = 0, scoreLeftXie = 0, scoreRightXie = 0;

int i, j, k;

int wx = lastChessPosition.X, wy = lastChessPosition.Y;

int color;

if (computerState) color = 1; else color = 2;

//col

i = wy;j = 0;

while (i > 0 && chessColor[wx, --i] == color) ++j;

i = wy;

while (i < 14 && chessColor[wx, ++i] == color) ++j;

scoreCol = j + 1;

//Row

i = wx;j = 0;

while (i > 0 && chessColor[--i, wy] == color) ++j;

i = wx;

while (i < 14 && chessColor[++i, wy] == color) ++j;

scoreRow = j + 1;

//LeftXie

i = wx;j = wy; k = 0;

while (i > 0 && j > 0 && chessColor[--i, --j] == color) ++k;

i = wx;j = wy;

while (i < 14 && j < 14 && chessColor[++i, ++j] == color) ++k;

scoreLeftXie = k + 1;

//RightXie

i = wx;j = wy;k = 0;

while (i < 14 && j > 0 && chessColor[++i, --j] == color) ++k;

i = wx;j = wy;

while (i > 0 && j < 14 && chessColor[--i, ++j] == color) ++k;

scoreRightXie = k + 1;

int ans = Math.Max(Math.Max(scoreCol, scoreRow), Math.Max(scoreLeftXie, scoreRightXie));

//MessageBox.Show(ans.ToString(), "ans", MessageBoxButtons.OK, MessageBoxIcon.Information);

if (ans == scoreCol)

{

if (wy > 0 && !hasChess[wx, wy - 1]) { x = wx; y = wy - 1; }

else if (wy < 14 && !hasChess[wx, wy + 1]) { x = wx; y = wy + 1; }

else

{

i = wy;

while (i > 0 && chessColor[wx, --i] == color) ;

if (hasChess[i, wy]) { i = wy; while (i < 14 && chessColor[wx, ++i] == color) ; }

if (hasChess[wx, i]) { int tx = 1, ty = 1; Random r = new Random(); while (hasChess[tx, ty]) { tx = r.Next(0, 15); ty = r.Next(0, 15); } x = tx; y = ty; }

else { x = wx; y = i; }

}

return;

}

else if (ans == scoreRow)

{

if (wx > 0 && !hasChess[wx - 1, wy]) { x = wx - 1; y = wy; }

else if (wx < 14 && !hasChess[wx + 1, wy]) { x = wx + 1; y = wy; }

else

{

i = wx;

while (i > 0 && chessColor[--i, wy] == color) ;

if (hasChess[i, wy]) { i = wx; while (i < 14 && chessColor[++i, wy] == color) ; }

if (hasChess[i, wy]) { int tx = 1, ty = 1; Random r = new Random(); while (hasChess[tx, ty]) { tx = r.Next(0, 15); ty = r.Next(0, 15); } x = tx; y = ty; }

else { x = i; y = wy; }

}

return;

}

else if (ans == scoreLeftXie)

{

if (wx > 0 && wy > 0 && !hasChess[wx - 1, wy - 1]) { x = wx - 1; y = wy - 1; }

else if (wx < 14 && wy < 14 && !hasChess[wx + 1, wy + 1]) { x = wx + 1; y = wy + 1; }

else

{

i = wx; j = wy;

while (i > 0 && j > 0 && chessColor[--i, --j] == color) ;

if (hasChess[i, wy]) { i = wx; j = wy; while (i < 14 && j < 14 && chessColor[++i, ++j] == color) ; }

if (hasChess[i, j]) { int tx = 1, ty = 1; Random r = new Random(); while (hasChess[tx, ty]) { tx = r.Next(0, 15); ty = r.Next(0, 15); } x = tx; y = ty; }

else { x = i; y = j; }

}

return;

}

else if (ans == scoreRightXie)

{

if (wx < 14 && wy > 0 && !hasChess[wx + 1, wy - 1]) { x = wx + 1; y = wy - 1; }

else if (wx > 0 && wy < 14 && !hasChess[wx - 1, wy + 1]) { x = wx - 1; y = wy + 1; }

else

{

i = wx; j = wy;

while (i < 14 && j > 0 && chessColor[++i, --j] == color) ;

if (hasChess[i, wy]) { i = wx; j = wy; while (i > 0 && j < 14 && chessColor[--i, ++j] == color) ; }

if (hasChess[i, j]) { int tx = 1, ty = 1; Random r = new Random(); while (hasChess[tx, ty]) { tx = r.Next(0, 15); ty = r.Next(0, 15); } x = tx; y = ty; }

else { x = i; y = j; }

}

return;

}

}

public void ai()

{

if (isFirst)

{

chess(7, 7);

return;

}

int x = 0, y = 0;

if(!computerState)

{

do{ where(ref x, ref y); } while (hasChess[x,y] && !stop(x, y));

}

else

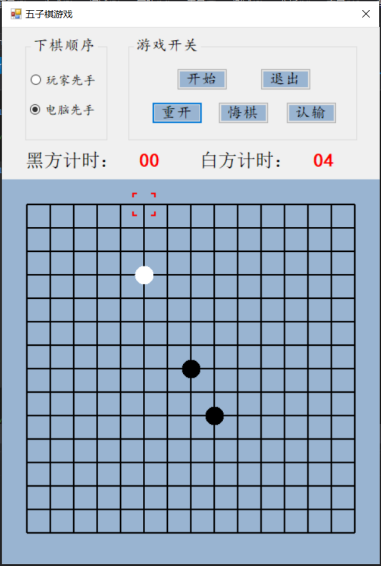
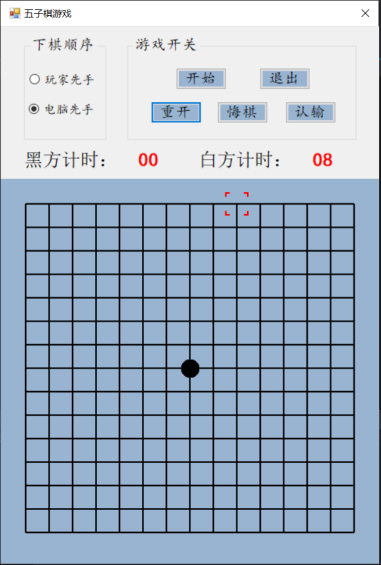
{

do{ where(ref x, ref y);} while (hasChess[x, y]);

}

chess(x, y);

}



天元点（左） 自动落子（右）

1. **玩家落子**

先判断是否玩家先手，如果是，第一颗黑子应落在天元点上（chess(7,7)），否则，在鼠标位置（panel2\_MouseClick()函数）转换成的棋盘坐标处绘制棋子（chess()函数）。

public void chess(int x, int y)

{

if (hasChess[x, y]) return;

int cenX = first + x \* step;

int cenY = first + y \* step;

SolidBrush sb;

Graphics g = panel2.CreateGraphics();

if (isBlack == true)

{

sb = new SolidBrush(Color.Black);

blackTime = 0;

}

else

{

sb = new SolidBrush(Color.White);

whiteTime = 0;

}

if (isBlack) chessColor[x, y] = 1;

else chessColor[x, y] = 2;

Rectangle r = new Rectangle(new Point(cenX - half + 3, cenY - half + 3), new Size(step - 6, step - 6));

g.FillEllipse(sb, r);

hasChess[x, y] = true;

if ((computerState && isBlack) || (!computerState && !isBlack))

lastChessPosition = new Point(x, y);

if((computerState && !isBlack) || (!computerState && isBlack))

lastChessPosition2 = new Point(x, y);

if (isBlack) judge(x, y, 0);

else judge(x, y, 1);

isBlack = !isBlack;

}

private void panel2\_MouseClick(object sender, MouseEventArgs e)

{

if (!isBegin)

{

MessageBox.Show("请先开始游戏！", "注意", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (isEnd)

{

MessageBox.Show("比赛已结束！！！", "注意", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if ((computerState && !isBlack) || (!computerState && isBlack))

return;

button3.Enabled = button4.Enabled = button5.Enabled = true;

if (e.X < first - half || e.X > last + half || e.Y < first - half || e.Y > last + half)

{

MessageBox.Show("超出棋盘的有效区域！", "注意", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

int x = (e.X - first) / step;

int y = (e.Y - first) / step;

int leftX = (e.X - first) % step;

int leftY = (e.Y - first) % step;

if (leftX > step / 2)

x += 1;

if (leftY > step / 2)

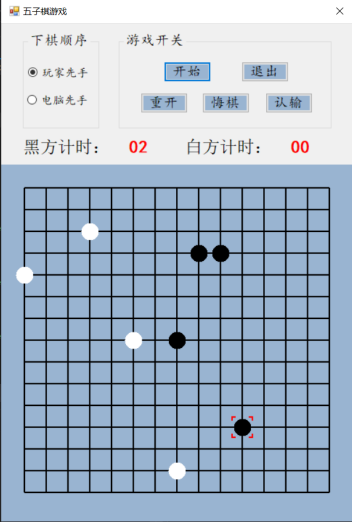
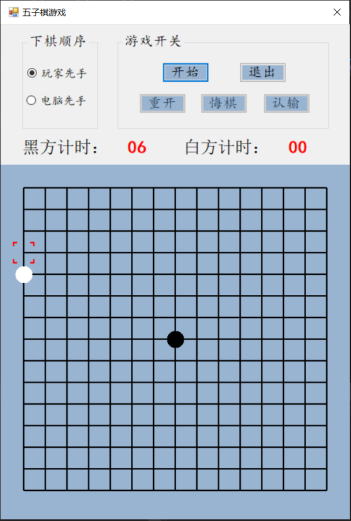
y += 1;

if (computerState && stop(x, y)) return;

chess(x, y);

if(!isEnd) ai();

}



天元点（左） 落子（右）

1. **开始游戏**

初始化部分控件的可见状态以及部分全局变量的值，判断响应状态弹出MessageBox提示消息，在天元点（chess(7,7)）绘制黑子，进行相关变量bool值的转化。

private void button1\_Click(object sender, EventArgs e)

{

if (isBegin)

{

MessageBox.Show("游戏已开始！！！", "加油", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

if (radioButton1.Checked) computerState = true;//电脑白棋

if (radioButton2.Checked) computerState = false;//电脑黑棋

isBegin = true;

isEnd = false;

MessageBox.Show("游戏开始！", "加油", MessageBoxButtons.OK, MessageBoxIcon.Information);

hasChess = new bool[15, 15];

chessColor = new int[15, 15];

isFirst = true;

//如果电脑黑棋，先下天元点

if (!computerState)

ai();

//否则玩家先下天元点，再ai下棋

else

{

chess(7, 7);

isFirst = false;

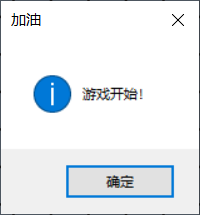
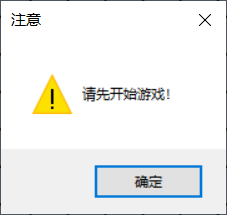
ai();

}

isFirst = false;

timer1.Start();

}



1. **退出游戏**

通过Exit()函数退出窗体运行状态。

private void button2\_Click(object sender, EventArgs e)

{

Application.Exit();

}

1. **重新开局**

通过Refresh()函数刷新panel2画布，将全局变量置为默认值，启动计时器。

private void button3\_Click(object sender, EventArgs e)

{

panel2.Refresh();

isBlack = true;

isEnd = false;

isBegin = true;

if (radioButton1.Checked) computerState = true;//电脑白棋

if (radioButton2.Checked) computerState = false;//电脑黑棋

isBegin = true;

MessageBox.Show("游戏开始！", "加油", MessageBoxButtons.OK, MessageBoxIcon.Information);

hasChess = new bool[15, 15];

chessColor = new int[15, 15];

isFirst = true;

//如果电脑黑棋，先下天元点

if (!computerState)

ai();

//否则玩家先下天元点，再ai下棋

else

{

chess(7, 7);

isFirst = false;

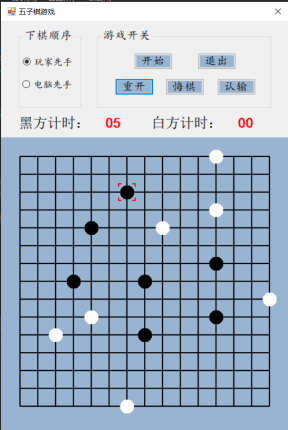
ai();

}

isFirst = false;

timer1.Start();

}



1. **玩家认输**

判定玩家棋子颜色，弹出提示框并重新加载棋盘，开始新的游戏。

private void button4\_Click(object sender, EventArgs e)

{

if (isBlack)

{

MessageBox.Show("白棋获胜！", "恭喜", MessageBoxButtons.OK, MessageBoxIcon.Information);

initBoard();

return;

}

else

{

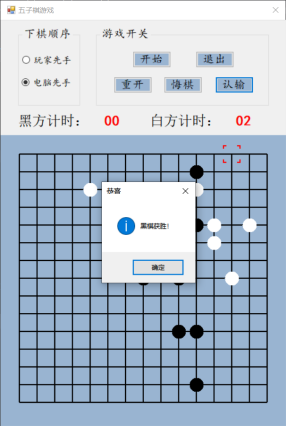
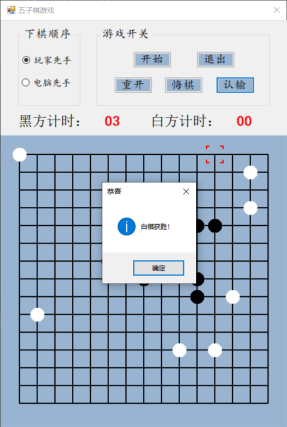
MessageBox.Show("黑棋获胜！", "恭喜", MessageBoxButtons.OK, MessageBoxIcon.Information);

initBoard();

return;

}

}



白棋电脑胜（左） 黑棋电脑胜（右）

1. **玩家悔棋**

通过设置变量存储黑白棋上一个落子点，若玩家悔棋，则在上一个落子点撤销棋子。

private void button5\_Click(object sender, EventArgs e)

{

//电脑下棋不需要悔棋/比赛结束了不允许悔棋

if (isEnd || (computerState && !isBlack) || (!computerState && isBlack)) return;

int x = lastChessPosition.X \* step + first;

int y = lastChessPosition.Y \* step + first;

int t = (step - 6) / 2;

hasChess[lastChessPosition.X, lastChessPosition.Y] = false;

chessColor[lastChessPosition.X, lastChessPosition.Y] = 0;

Rectangle r = new Rectangle(new Point(x - half + 3, y - half + 3), new Size(t \* 2, t \* 2));

SolidBrush sb = new SolidBrush(panel2.BackColor);

Graphics g = panel2.CreateGraphics();

g.FillEllipse(sb, r);

g.DrawLine(pen, x, y - t >= first ? y - t : first, x, y + t <= last ? y + t : last);

g.DrawLine(pen, x - t >= first ? x - t : first, y, x + t <= last ? x + t : last , y);

x = lastChessPosition2.X \* step + first;

y = lastChessPosition2.Y \* step + first;

t = (step - 6) / 2;

hasChess[lastChessPosition2.X, lastChessPosition2.Y] = false;

chessColor[lastChessPosition2.X, lastChessPosition2.Y] = 0;

r = new Rectangle(new Point(x - half + 3, y - half + 3), new Size(t \* 2, t \* 2));

sb = new SolidBrush(panel2.BackColor);

g = panel2.CreateGraphics();

g.FillEllipse(sb, r);

g.DrawLine(pen, x, y - t >= first ? y - t : first, x, y + t <= last ? y + t : last);

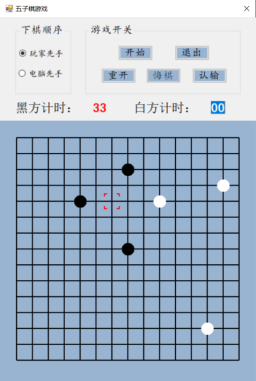
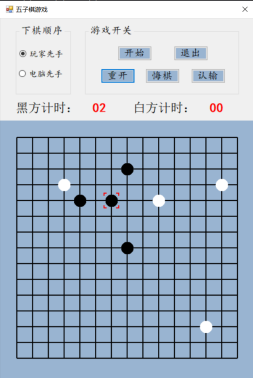
g.DrawLine(pen, x - t >= first ? x - t : first, y, x + t <= last ? x + t : last, y);

if (computerState) isBlack = true;

else isBlack = false;

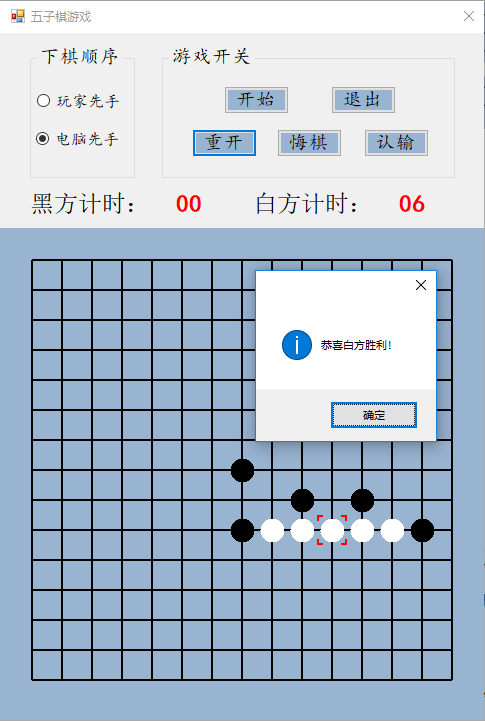
button5.Enabled = false;

}



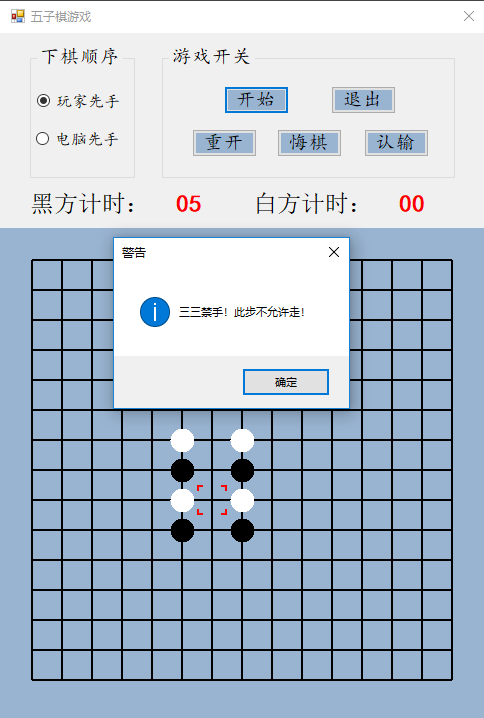
1. **测试**
2. **判断胜负模块测试**

针对算法缺陷进行落子，结果如下图，验证正确！



1. **判断禁手模块测试**

测试禁手，结果如下图，验证正确！



1. **心得体会**

本次课程设计中，切实体会了从一个项目要求出发，分析实现原理以及思维逻辑，最后实现预期效果的辛酸与快乐。一开始需要搜集资料，了解五子棋界面需要用到的控件，还要了解控件对应的函数的参数及用法，根据已有的思维逻辑确定所需函数，界面构建好之后，从简单到复杂地分析程序功能，先将简单逻辑的函数写好，例如棋盘绘制，确保运行程序后能有棋盘，点击棋盘后能落子，落子能区分黑白子，按钮能够正常弹框或启动程序，做完这些，大概的五子棋模型就出来了，再结合题目要求，增加显示光标、判断禁手、判断胜负、悔棋认输等功能，经过不断地调试修改，实现一局五子棋游戏应该具备的基本功能，最后设置个性化地提示、优化界面，一个五子棋游戏项目就做好了。

两个星期的日夜，让我亲身经历了团队合作完成项目的过程，不仅在思维上更加紧密，逻辑也更加清晰，更是收获了完成项目后的喜悦，我觉得这才是课设的意义所在。