南京邮电大学

2018 / 2019 学年第 1 学期

《Windows 编程》课程大作业报告

课题代号（单选）☐课题 1　　☐课题 2　　☐课题 3　　■课题 8

课题名称：_____五子棋对弈游戏程序_____

小组成员人数（单选）☐1 人　　☐2 人　　■3 人　　☐4 人

课题完成时间：_____2018/12/26_____

==========以下部分为教师填写区，请勿填写==========

成绩评定：

软件基本功能_____　软件提高功能_____　软件部分成绩_____

文档结构_____　文档撰写_____　文档部分成绩_____

总成绩_____

备注_____

目录

五子棋对弈游戏程序

1. 课题内容和要求

课题名称：五子棋对弈游戏程序

内容和要求：实现五子棋对弈游戏程序，要求能实现按照规则的开局、走棋、吃子以及胜负判定功能，包括相应的图形界面以及声音效果。实现网络游戏对战功能，能过支持双人或多人进行在线游戏，如果有简单的人机对战功能可作为加分项。

问题分析：在五子棋对弈游戏程序进行模块化处理后，我们小组决定从以下几个方面对程序进行设计和编码：

① 棋盘绘制，可视化界面设计

② 五子棋下棋规则（开局、走棋、胜负判定）

③ 双人对战模式

④ 人机对战模式

⑤ 机器 AI 智力设置（弱智、初级、中级、高级）

小组分工：鉴于以上我们对于问题的肢解，我们小组三个人决定进行如下分工以高效率的完成五子棋对弈游戏程序的设计：

陈慧（组长）：五子棋棋局设置及可视化界面开发，系统设计和程序总体框架的搭建，各模块连接和相关数据的保存，最终完成本小组的五子棋对弈游戏程序；

胡雪然：机器 AI 设置模块，包括弱智、初级、中级、高级四个等级，即机器走棋的不同规则设置；

贺颖婷：五子棋下棋输赢判断模块及悔棋模块设置。

由此，在本实验报告中，机器 AI 设置和五子棋下棋输赢判断将不会详细阐述。本实验报告将会从棋局绘制，可视化界面创建，数据保存，双人对战和人机对战这四个方面对本小组的五子棋对弈游戏程序进行介绍。

2. 需求分析

五子棋是起源于中国古代的传统黑白棋种之一。现代五子棋日文称之为“连珠”，英 译为“Renju”，英文称之为“Gobang”或“FIR ”(Five in a Row 的缩写)，亦有“连五子“、“五子连”、“串珠”、“五目”、“五目碰”、“五格”等多种称谓。为了能够很好的理解本五子棋程序，整个文件系统用“Gobang_FiveChess”进行命名。

五子棋的棋子分为黑白两色，棋盘为 15*15,棋子放置于棋盘线交叉点上。两人对局或者人机对战，双方各执一色，执黑先行，轮流下一子，先将横、竖或斜线的 5 个或 5 个以上同色棋子连城不间断的一排者为胜。基于以上五子棋的游戏规则，可以将该游戏系统分为以下功能模块：

（1） 绘制五子棋游戏棋盘

（2） 可视化界面设置，棋盘按钮控件布局

（3） 棋局功能实现（点击下棋，黑白棋轮流执子，悔棋）

（4） 人机对战，机器 AI 设计，游戏难度等级设置

（5） 双人对战模式设计

（6） 五子棋输赢判断

（7） 数据存储

对于整个五子棋游戏程序，用户打开可执行文件即可进行游戏，系统默认为中级难度下的人机交互模式。用户可以通过界面上的“模式设置”按钮进行游戏难度和游戏模式设置，机器智力包括“弱智、初级、中级、高级”，对战模式包括“人机对战、人人对战”，点击“确定”即开始该种模式的游戏。直接使用鼠标点击界面网格十字交叉位置，即可下棋。黑棋先行，黑白棋轮流执子。当出现某一方连成五个棋子时，系统自动弹出提示框“黑棋赢”或“白棋赢”。点击主界面“开始游戏”按钮即清空棋盘，进行新一轮的游戏。在游戏进行过程中，黑白方各有一次悔棋的机会，再悔第二次，系统发出警告。在对话框上边界右击，要能够出现本项目完成的基本信息，即“关于…”设置。至此，完成以上内容的设计和编码，该五子棋对弈游戏程序即成功完成。

3. 概要设计

分析整个五子棋对弈游戏程序，编码的流程分为以下几个模块：

（1）程序基础布局设置：创建五子棋主界面对话框，创建五子棋系统“关于…”对话框，将“关于…”对话框加入主界面对话框，设置主对话框图标即程序图标，添加最小化对话框，添加屏幕缩放功能。

（2）棋盘设置：设置背景，绘制棋盘，绘制棋子，给新下的棋子添加标记，绘制棋盘上的天元 5 个点，鼠标在棋盘区域为十字型标志。

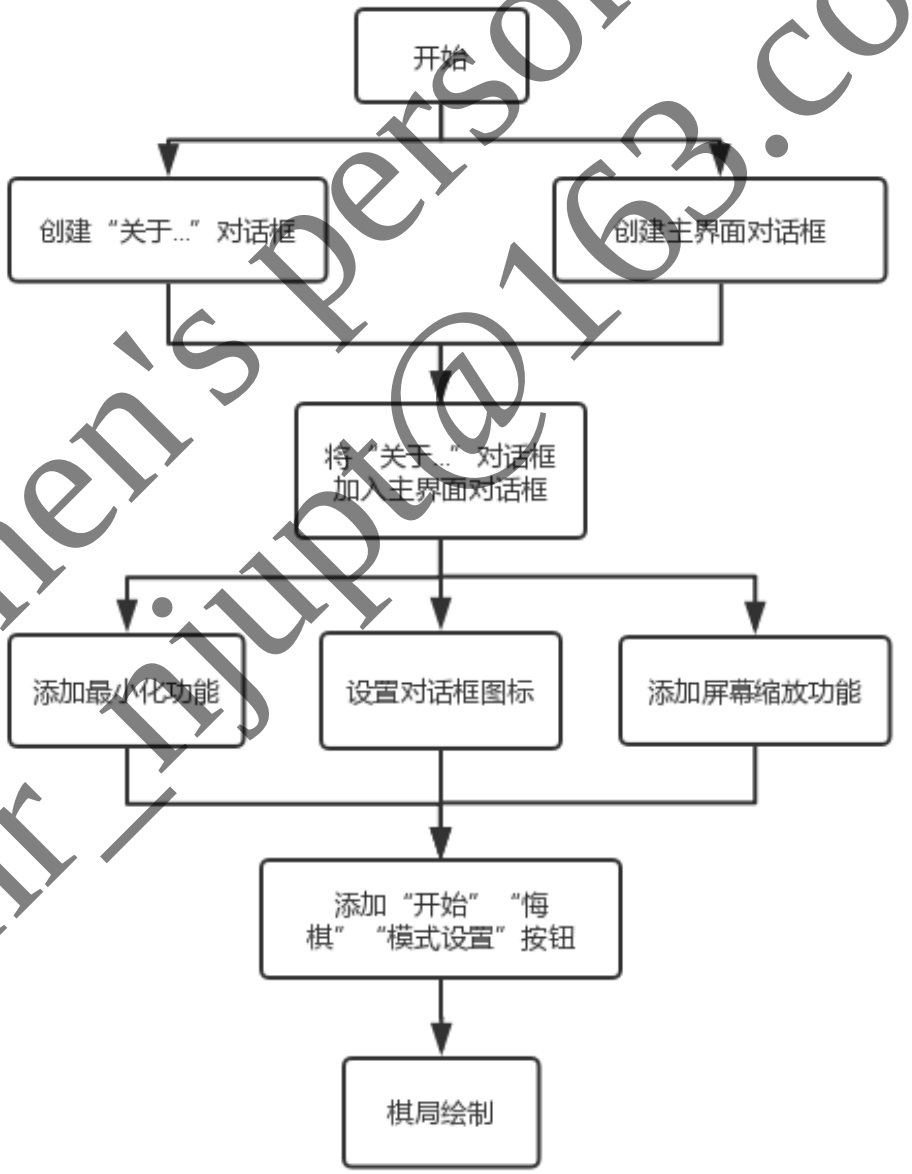
（3）按钮控件设置：包括“开始”、“悔棋”、“模式设置”。

（4）模式功能设计：机器 AI 难度设置，人机对战和人人对战不同接口连接。

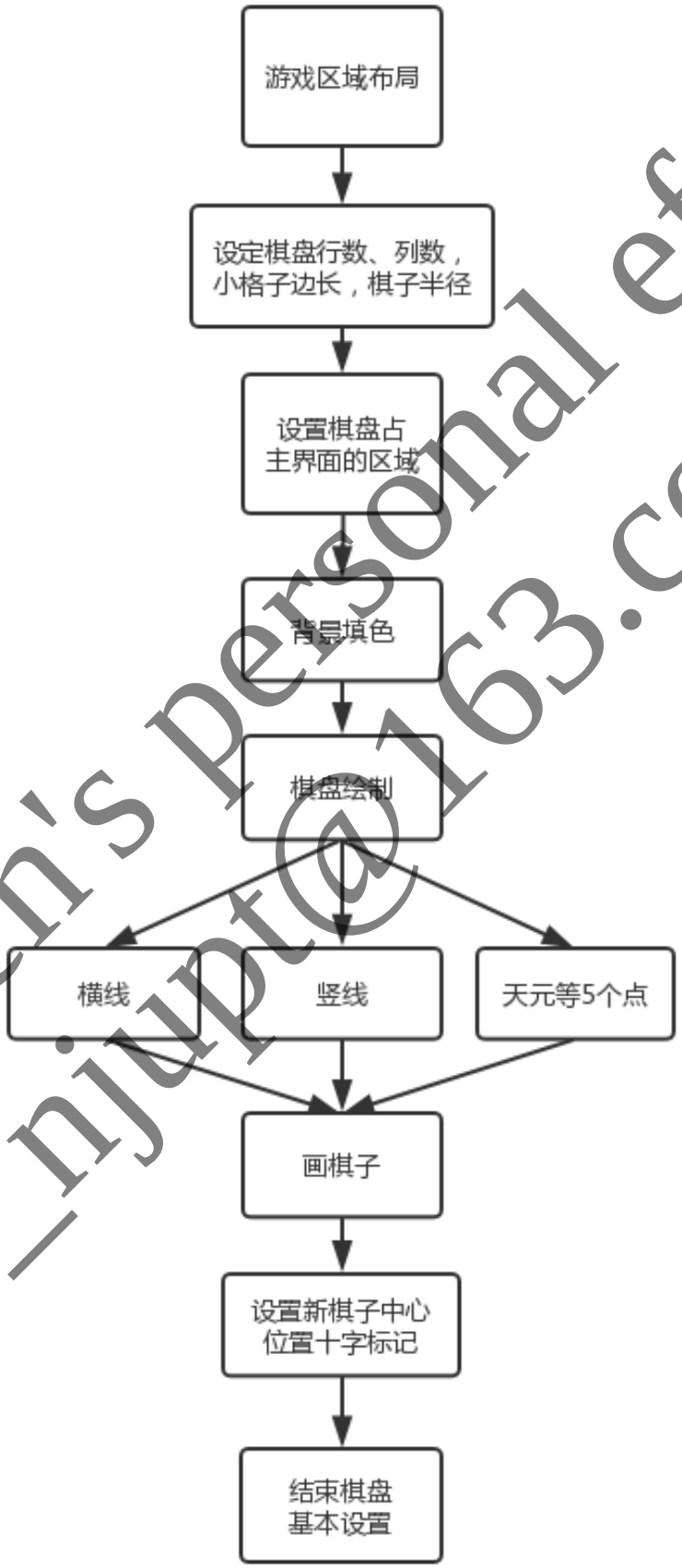
（5）五子棋规则：输赢判断，悔棋设置，黑白棋轮流执子。

各模块的概要设计流程图如下：

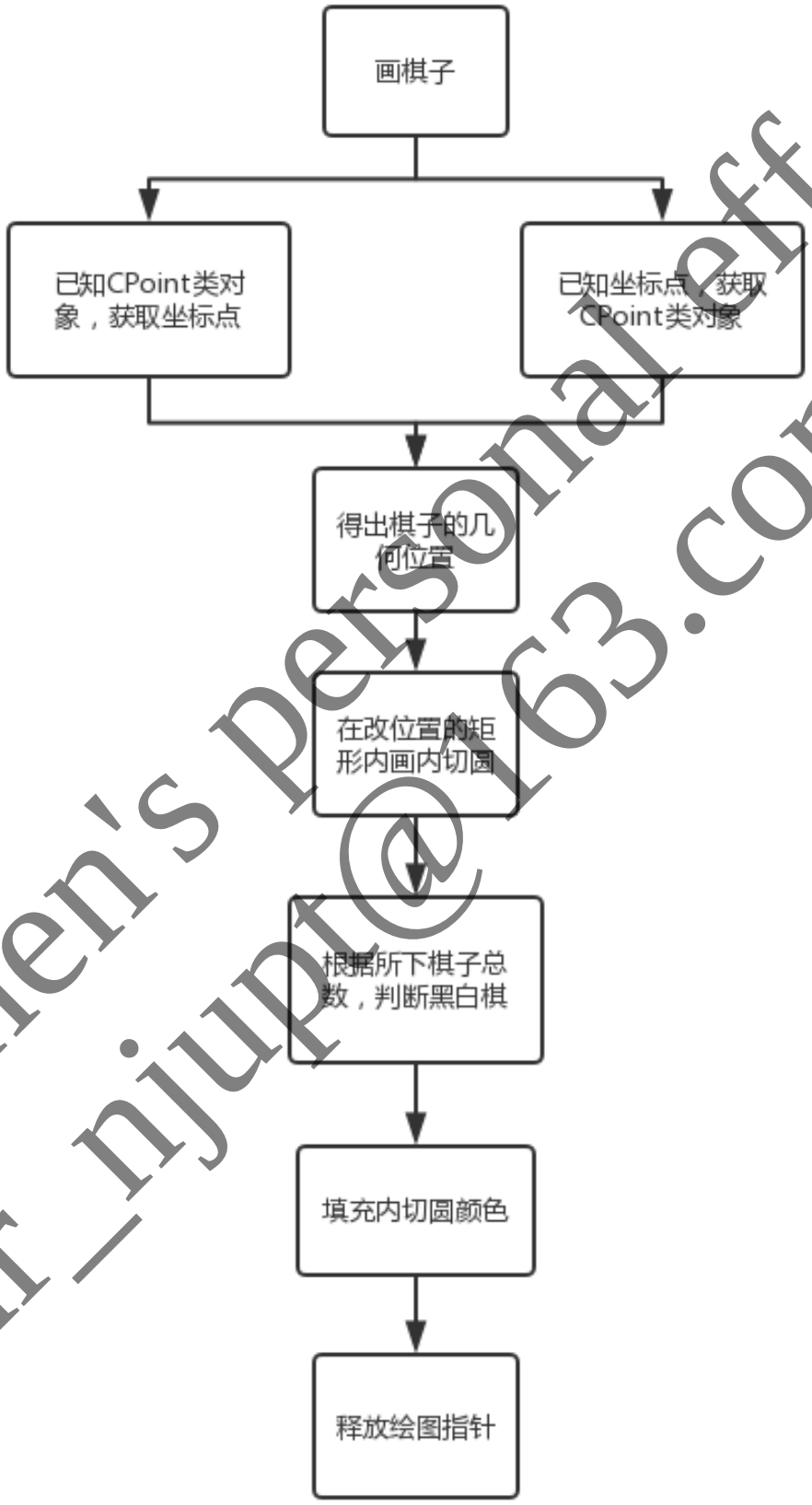
3.1. 程序基础布局设置



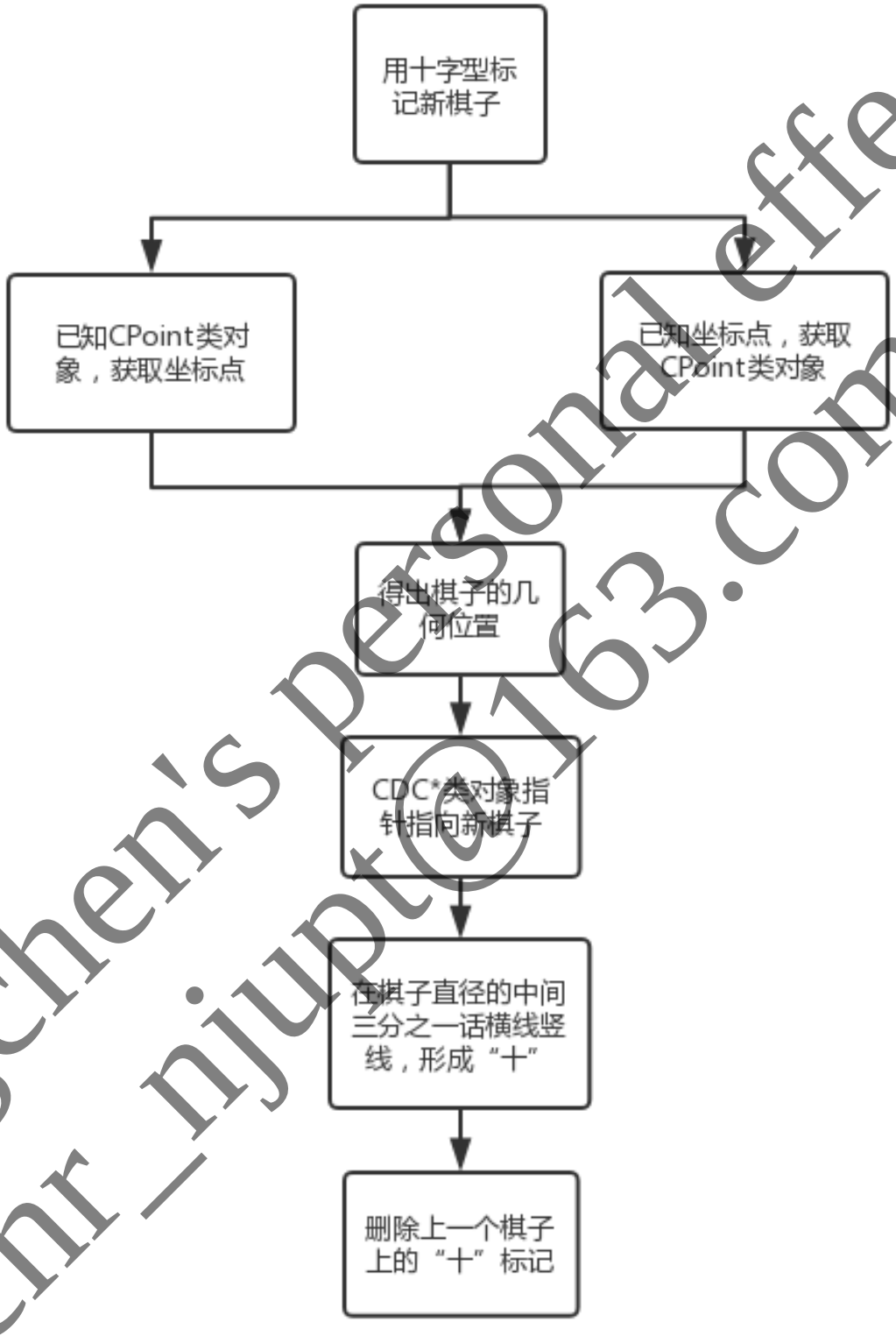
3.2. 棋盘绘制



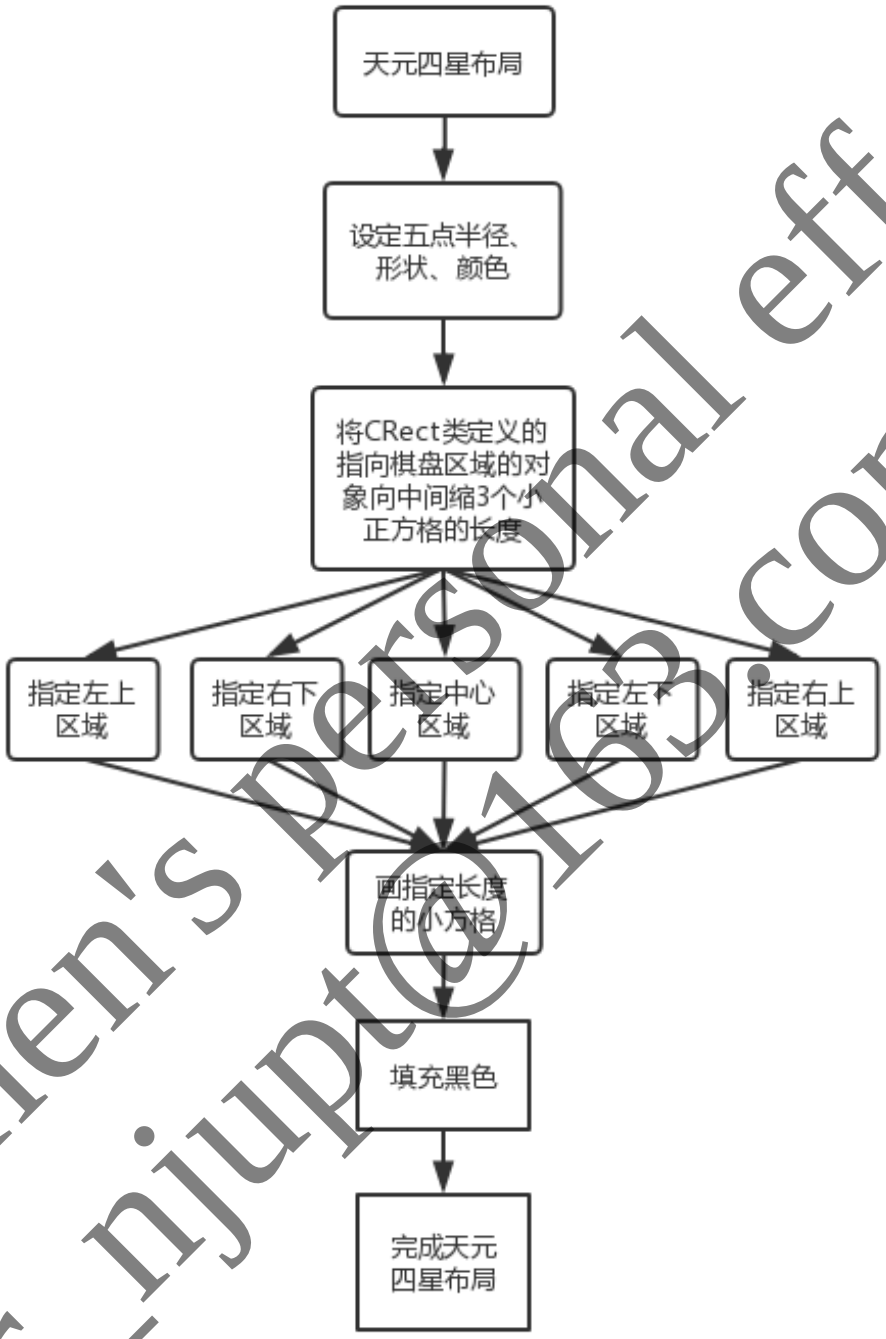
3.3. 黑白棋子绘制



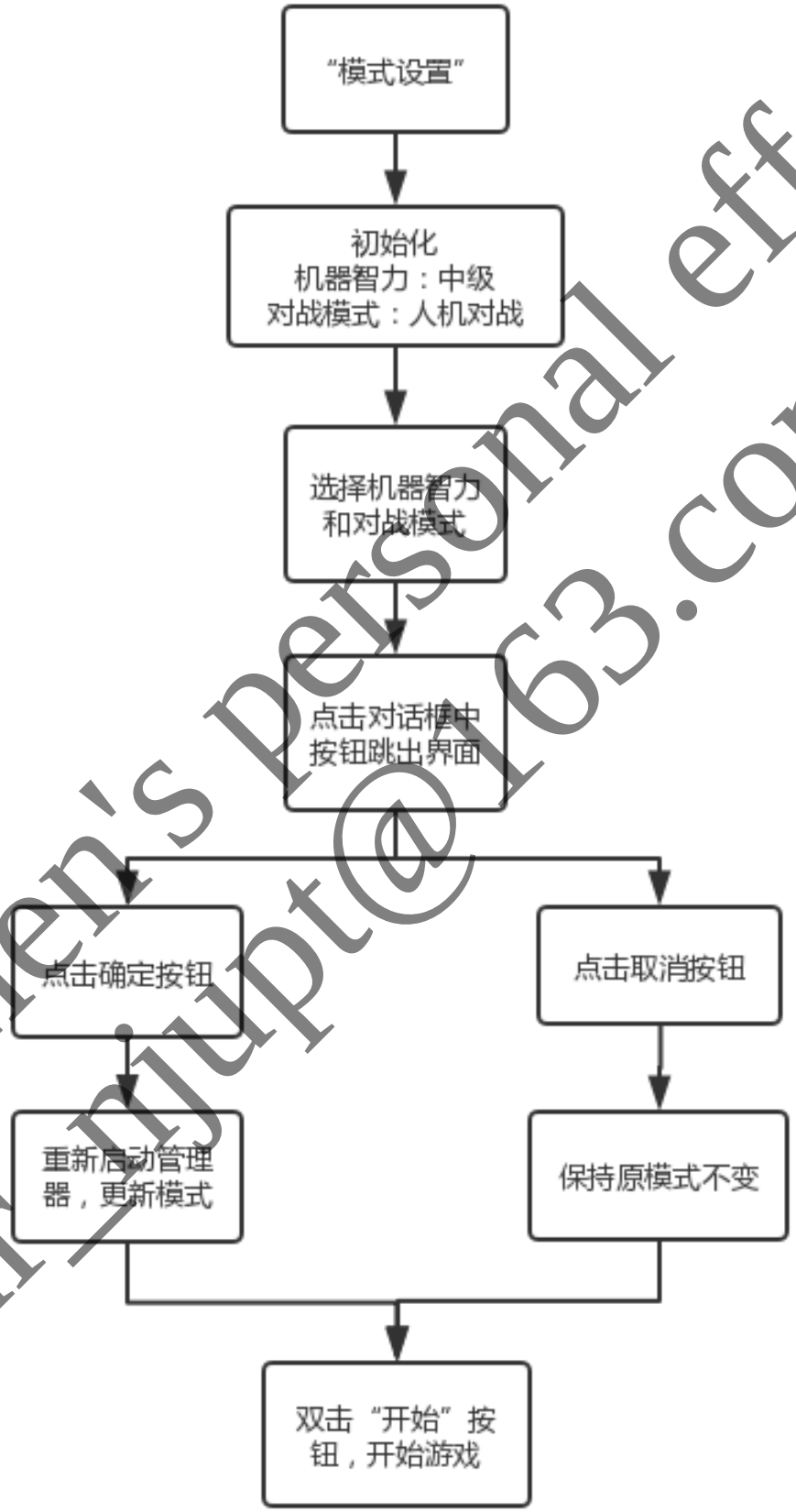
3.4. 新棋子“十”字标记设置



3.5. 棋盘天元四星布局设置



3.6. “模式设置”按钮功能设计



4. 详细设计

4.1. 游戏基本布局实现

首先，创建游戏进行过程中的各种对话框，包括：游戏简介“关于…”对话框，主界面对话框，输赢提示对话框，“模式设置”按钮触发后选择对话框，二次悔棋提醒对话框等。其次，进行对话框直接连接的相关布局。然后，设置对话框放大缩小最小化等基本功能，并实现“开始”“模式设置”“悔棋”按钮的布置，最后为增加游戏体验，将棋盘上的鼠标标记设置为“十”字型。

4.1.1. “关于…”对话框

```
// 用于应用程序“关于”菜单项的 CAboutDlg 对话框
class CAboutDlg : public CDialogEx
{
public:
    CAboutDlg();
// 对话框数据
    enum { IDD = IDD_ABOUTBOX };
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV 支持
// 实现
protected:
    DECLARE_MESSAGE_MAP()
};
CAboutDlg::CAboutDlg() : CDialogEx(CAboutDlg::IDD)
{
}
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);
}
BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
END_MESSAGE_MAP()
```

4.1.2. 主界面对话框

```
// 创建对话框
BEGIN_EASYSIZE_MAP(CGobang_FiveChessDlg)
    EASYSIZE(IDC_BUTTON_GAME_START,  ES_KEEPSIZE, ES_BORDER,
        ES_BORDER,ES_KEEPSIZE,0)
    EASYSIZE(IDC_BUTTON_REGRET,     ES_KEEPSIZE, ES_BORDER,
        ES_BORDER,ES_KEEPSIZE,0)
    EASYSIZE(IDC_BUTTON_MORE,  ES_KEEPSIZE, ES_BORDER,
        ES_BORDER,ES_KEEPSIZE,0)
```

```
END_EASYSIZE_MAP
CGobang_FiveChessDlg::CGobang_FiveChessDlg(CWnd* pParent /*=NULL*/)
    : CDialogEx(CGobang_FiveChessDlg::IDD, pParent)
{
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}
CGobang_FiveChessDlg::~CGobang_FiveChessDlg()
{
}
void CGobang_FiveChessDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);//与对应的变量交换对话框数据
}
BEGIN_MESSAGE_MAP(CGobang_FiveChessDlg, CDialogEx)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_WM_LBUTTONUP()
    ON_WM_ERASEBKGND()
    ON_BN_CLICKED(IDC_BUTTON_GAME_START,
&CGobang_FiveChessDlg::OnBnClickedButtonGameStart)
    ON_BN_CLICKED(IDC_BUTTON_REGRET,
&CGobang_FiveChessDlg::OnBnClickedButtonRegret)
    ON_BN_CLICKED(IDC_BUTTON_MORE,
&CGobang_FiveChessDlg::OnBnClickedButtonMore)
     ON_WM_SIZE()
     ON_WM_MOUSEMOVE()
END_MESSAGE_MAP()
```

4.1.3. 输赢提醒对话框

```
void CGobang_FiveChessDlg::OnLButtonUp(UINT nFlags, CPoint point)
{
    // TODO:  在此添加消息处理程序代码和/或调用默认值
    m_chess.SetPiecePos(point);
    Invalidate();
     switch(m_chess.GetWinFlag())
     {
     case     WHITE_WIN:
          AfxMessageBox(_T("白旗赢"));
          break;
     case      BLACK_WIN:
          AfxMessageBox(_T("黑旗赢"));
          break;
```

```
        case       PEACE:
            AfxMessageBox(_T("平局"));
            break;
        default:
            break;
        }
    CDialogEx::OnLButtonUp(nFlags, point);
}
```

4.1.4. 对话框嵌套及主界面图标设置

```
BOOL CGobang_FiveChessDlg::OnInitDialog()
{
    CDialogEx::OnInitDialog();
    // 将“关于...”菜单项添加到系统菜单中。
    // IDM_ABOUTBOX 必须在系统命令范围内。
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);
    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        BOOL bNameValid;
        CString strAboutMenu;
        bNameValid = strAboutMenu.LoadString(IDS_ABOUTBOX);
        ASSERT(bNameValid);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }
    // 设置此对话框的图标。当应用程序主窗口不是对话框时，框架将自动
    //  执行此操作
    SetIcon(m_hIcon, TRUE);            // 设置大图标
    SetIcon(m_hIcon, FALSE);        // 设置小图标
    // TODO: 在此添加额外的初始化代码
    CRect    rcClient;
    GetClientRect(&rcClient);
    m_chess.Init(rcClient);
    INIT_EASYSIZE;
    return TRUE;  // 除非将焦点设置到控件，否则返回 TRUE
}
```

4.1.5. 屏幕缩放功能及最小化按钮实现

如果向对话框添加最小化按钮，则需要下面的代码来绘制该图标。对于使用文档/视图模型的 MFC 应用程序，这将由框架自动完成。

```
void CGobang_FiveChessDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // 用于绘制的设备上下文
        SendMessage(WM_ICONERASEBKGND,
reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);
        // 使图标在工作区矩形中居中
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;
        // 绘制图标
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialogEx::OnPaint();
         CRect     rcBtnStart, rcBtnRegret, rcBtnMore;
         CRgn      rgnChildWnd,     rgnBtnStart,     rgnBtnRegret,     rgnBtnMore;
        CDC*   pDC     = GetDC();

         // 获取按钮在父窗口的位置
         GetDlgItem(IDC_BUTTON_GAME_START)->GetWindowRect(rcBtnStart);
         GetDlgItem(IDC_BUTTON_REGRET)->GetWindowRect(rcBtnRegret);
         GetDlgItem(IDC_BUTTON_MORE)->GetWindowRect(rcBtnMore);
         ScreenToClient(rcBtnStart);//把屏幕上指定点的屏幕坐标转换成用户坐标
         ScreenToClient(rcBtnRegret);
         ScreenToClient(rcBtnMore);

         // 剪切按钮的位置，创建的剪裁区域大小和控件一致
         rgnBtnStart.CreateRectRgnIndirect(rcBtnStart);
         rgnBtnRegret.CreateRectRgnIndirect(rcBtnRegret);
         rgnBtnMore.CreateRectRgnIndirect(rcBtnMore);
         rgnChildWnd.CreateRectRgn(0, 0, 0, 0);
         rgnChildWnd.CombineRgn(&rgnBtnStart, &rgnBtnRegret, RGN_OR);
         rgnChildWnd.CombineRgn(&rgnChildWnd, &rgnBtnMore, RGN_OR);
         pDC->SelectClipRgn(&rgnChildWnd, RGN_DIFF);
```

```
            // 绘棋局
            m_chess.Draw(pDC);
            // 善后
            pDC->SelectClipRgn(NULL);
            rgnBtnStart.DeleteObject();
            rgnBtnRegret.DeleteObject();
            rgnBtnMore.DeleteObject();
            rgnChildWnd.DeleteObject();
            ReleaseDC(pDC);
        }
}
//当用户拖动最小化窗口时系统调用此函数取得光标显示。
HCURSOR CGobang_FiveChessDlg::OnQueryDragIcon()
{
    return static_cast<HCURSOR>(m_hIcon);
}
```

4.1.6. 游戏按钮触发

```
//“开始游戏”按钮触发
void CGobang_FiveChessDlg::OnBnClickedButtonGameStart()
{
    // TODO:  在此添加控件通知处理程序代码
     m_chess.NewGame();// 新游戏
     Invalidate();
}

//“悔棋”按钮触发
void CGobang_FiveChessDlg::OnBnClickedButtonRegret()
{
    // TODO:  在此添加控件通知处理程序代码
     if(! m_chess.Regret())
     {
         AfxMessageBox(_T("再悔棋就不理你了~"));
     }
     Invalidate();
}

//“模式设置”按钮触发
void CGobang_FiveChessDlg::OnBnClickedButtonMore()
{
     CDialogMore* dlgMore       = new      CDialogMore;

     dlgMore->Create(IDD_DIALOG_MORE);
```

```
    dlgMore->ShowWindow(SW_SHOW);           // 非模态
// dlgMore->DoModal();

    dlgMore->SetChess(&m_chess);
    // TODO: 在此添加控件通知处理程序代码
}
```

4.1.7. 设置鼠标在键盘上形式为“+”字光标

```
void CGobang_FiveChessDlg::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: 在此添加消息处理程序代码和/或调用默认值
    CRect    rcBoard = m_chess.GetRectBoard();// 获取棋盘区域
    if(rcBoard.PtInRect(point))//判断点是否在 CRect 中
    {
        HCURSOR hCursor = AfxGetApp()->LoadStandardCursor(IDC_CROSS);
        //IDC_CROSS 十字光标
      SetCursor(hCursor);
    }
    CDialogEx::OnMouseMove(nFlags, point);
}
```

4.2. 棋盘设置

4.2.1. 棋盘绘制

```
//初始化棋盘
CChessDraw::CChessDraw()
{
    m_pDC = NULL;
    m_rcBK.SetRectEmpty();// 背景区域
    m_rcBoard.SetRectEmpty();// 棋盘区域
    m_crBKBegin      = RGB(255,232,166);// 渐变色起始端
    m_crBKEnd        = RGB(255,220,200);// 结尾端
    m_crBoard        = RGB(0,0,0);// 棋盘线条的颜色

    m_uiPieceRadius= 0;// 棋子半径
    m_uiBoardRows = 0;// 棋盘的行数
    m_uiBoardCols  = 0;// 棋盘的列数
    m_uiBoardWidth= 0; // 每一格棋盘的边长(正方形格子)
}
void    CChessDraw::InitBoard(CRect    rect, UINT    uiRows, UINT    uiCols,          UINT
    uiBoardWidth, UINT uiPieceRadius)
{
```

```
    m_rcBK= rect;
    m_uiBoardRows = uiRows;
    m_uiBoardCols  = uiCols;
    m_uiBoardWidth= uiBoardWidth;
    m_uiPieceRadius= uiPieceRadius;

    // 计算棋盘区域
    m_rcBoard  = m_rcBK;
    m_rcBoard.DeflateRect(CSize(20,20));     // 背景区域的左上点偏移(20,20)再画棋盘
    m_rcBoard.right       = m_rcBoard.left + m_uiBoardWidth * (m_uiBoardCols - 1);
    //右边界位置随左边界和列数和每一格棋盘的边长决定
    m_rcBoard.bottom    = m_rcBoard.top   + m_uiBoardWidth * (m_uiBoardRows - 1);
    //下边界位置随上边界和行数和每一格棋盘的边长决定
}
CRect   CChessDraw::GetRectBoard()// 获取棋盘区域
{
    return   m_rcBoard;
}

// 获取绘图 DC
void     CChessDraw::SetDC(CDC*   pDC)
{
     m_pDC= pDC;
}

// 画背景
void CChessDraw::DrawBackground()
{
     if(! m_pDC)//m_pDC=0,返回；！=0 才画
     {
          return;
     }

     // 将指定区域（棋盘背景区域）填充成渐变颜色
     GradientFillRect(m_pDC,    m_rcBK,    m_crBKBegin,   m_crBKEnd,     TRUE);
}

 // 画棋盘
void CChessDraw::DrawBoard()
{
    if(! m_pDC)//m_pDC=0,返回；！=0 才画
    {
         return;
    }
```

```
    // 画横线   CRect 类型 m_rcBoard 表示棋盘区域
    UINT    ptTop    = m_rcBoard.top;
    for(UINT i   = 0; i < m_uiBoardRows; i ++)//用行数控制，由上至下
    {
        m_pDC->MoveTo(m_rcBoard.left, ptTop);
        m_pDC->LineTo(m_rcBoard.right,     ptTop);
        ptTop    += m_uiBoardWidth;
    }
    // 画竖线
    UINT    ptLeft    = m_rcBoard.left;
    for(UINT i   = 0; i < m_uiBoardCols;   i ++)//用列数控制，由左到右
    {
        m_pDC->MoveTo(ptLeft, m_rcBoard.top);
        m_pDC->LineTo(ptLeft,  m_rcBoard.bottom);
        ptLeft    += m_uiBoardWidth;
    }
    // 五子棋中的天元等 5 个点
    DrawSpecialPoints();
}
```

4.2.2. 黑白棋子绘制

```
// 画棋子
BOOL CChessDraw::DrawPiece(CPoint    pt,   BOOL   bBlack)
//CPoint 类操纵 CPoint 和 POINT 结构的成员函数
{
    //已知 pt，获取棋盘坐标点(puiX,    puiY)
    if(! m_pDC  ||! GetCoordinateWithPoint(pt))
    {
        return    FALSE;
    }
    //CRect rc 定义矩形的左上角和右下角点的成员变量。
    CRect    rc(pt - CSize(m_uiPieceRadius,    m_uiPieceRadius),    pt                    +
CSize(m_uiPieceRadius,  m_uiPieceRadius));
    CRgn    rgn;
    //创建一内切于特定矩形的椭圆区域
    rgn.CreateEllipticRgnIndirect(rc);
    //选择一个区域作为指定设备环境的当前剪切区域,即选取该椭圆区域
    m_pDC->SelectClipRgn(&rgn, RGN_AND);

    if(bBlack)//bBlack=1
    {
        FillGradientEx(m_pDC,   rc,        RGB(140,130,120),   RGB(0,0,0), FALSE);
        // 画黑棋
```

```
    }
    else
    {
        FillGradientEx(m_pDC,  rc,      RGB(255,255,255),  RGB(180,200,180),  FALSE);
        // 画白棋
    }
    m_pDC->SelectClipRgn(NULL);
    return   TRUE;
}
```

4.2.3. 新棋子“十”字标记绘制

```
// 给当前的棋子加上标记
void CChessDraw::DrawPieceCur(CPoint  pt)
{
    if(! m_pDC  || ! GetCoordinateWithPoint(pt))
    {
        return;
    }
    //CPen 类是画笔的对象，用来在 DC 上完成绘制线条的任务。
    CPen    penNew(PS_DASH, 1, RGB(255,0,0));
    CPen*   penOld  = m_pDC->SelectObject(&penNew);
    int  iSize    = m_uiPieceRadius/3;
    // 画横线
    m_pDC->MoveTo(pt.x    -    iSize, pt.y);
    m_pDC->LineTo(pt.x    +    iSize, pt.y);
    // 画竖线
    m_pDC->MoveTo(pt.x,   pt.y -    iSize);
    m_pDC->LineTo(pt.x,    pt.y +   iSize);
    //释放旧棋子上的标记
    m_pDC->SelectObject(penOld);
}
```

4.2.4. 棋盘天元四星布局设置

```
// 五子棋中的天元等 5 个点
void CChessDraw::DrawSpecialPoints()
{
    if(! m_pDC)
    {
        return;
    }
    CRect        rcTmp;
    CRect        rc  = m_rcBoard;// 棋盘区域
```

```
    COLORREFrgb = RGB(0,0,0);
    UINT        uiSpecialRadius = m_uiPieceRadius/2 - 1; // 天元等点的半径(默认为正方形，此值表示边长的一半)
    CSize       szRadius(uiSpecialRadius,    uiSpecialRadius);
    const    int   SPACE  = 3 * m_uiBoardWidth;                    // 偏移

    rc.DeflateRect(CSize(SPACE, SPACE));//通过朝 CRect 的中心移动边以缩小 CRect

    rcTmp   = CRect(rc.TopLeft(),      rc.TopLeft());//指向新区域的左上角
    rcTmp.InflateRect(szRadius);//增大或减小指定矩形的宽和高
    m_pDC->FillSolidRect(rcTmp,          rgb);

    rcTmp   = CRect(rc.BottomRight(),     rc.BottomRight());
    rcTmp.InflateRect(szRadius);
    m_pDC->FillSolidRect(rcTmp,          rgb);//指定填充矩形使用的颜色。

    rcTmp   = CRect(CPoint(rc.right, rc.top),    CPoint(rc.right, rc.top));
    rcTmp.InflateRect(szRadius);
    m_pDC->FillSolidRect(rcTmp,          rgb);

    rcTmp   = CRect(CPoint(rc.left, rc.bottom),      CPoint(rc.left, rc.bottom));
    rcTmp.InflateRect(szRadius);
    m_pDC->FillSolidRect(rcTmp,          rgb);

    rcTmp   = CRect(rc.CenterPoint(), rc.CenterPoint());
    rcTmp.InflateRect(szRadius);
    m_pDC->FillSolidRect(rcTmp,          rgb);
}
```

4.3. 按钮控件设置

```
// CDialogMore  对话框
IMPLEMENT_DYNAMIC(CDialogMore, CDialogEx)//确定运行时对象属于哪一个类
CDialogMore::CDialogMore(CWnd* pParent /*=NULL*/)
    : CDialogEx(CDialogMore::IDD, pParent)
{
     m_pChess     = NULL;
}
CDialogMore::~CDialogMore()
{
}
void CDialogMore::DoDataExchange(CDataExchange* pDX)
{
     CDialogEx::DoDataExchange(pDX);
```

```
    DDX_Control(pDX, IDC_COMBO_AI, m_comboxAI);
    DDX_Control(pDX, IDC_COMBO_VS_MODE, m_comboxVSMode);
}

//宏开始消息映射，为每个消息处理函数加入一个入口，
BEGIN_MESSAGE_MAP(CDialogMore, CDialogEx)
    ON_BN_CLICKED(IDOK, &CDialogMore::OnBnClickedOk)
    //用户单击按钮时,向父对象发送消息
END_MESSAGE_MAP()

// CDialogMore 消息处理程序,,
void CDialogMore::OnBnClickedOk()
{
    // TODO: 在此添加控件通知处理程序代码
    if(m_pChess)
    {
        m_pChess->SetAIDepth(m_comboxAI.GetCurSel());//先选择机器 AI 等级

        switch(m_comboxVSMode.GetCurSel())
        {
        case    0:
            m_pChess->SetVSMode(PERSON_VS_MACHINE);//人机对战
            break;
        case    1:
            m_pChess->SetVSMode(PERSON_VS_PERSON);//人人对战
            break;

        default:
            break;
        }
    }
    CDialogEx::OnOK();//确认键开始
}

BOOL CDialogMore::OnInitDialog()
{
    CDialogEx::OnInitDialog();

    // TODO:  在此添加额外的初始化
    m_comboxAI.SetCurSel(2);                    // 中级
    m_comboxVSMode.SetCurSel(0);    // 人机
    return TRUE;  // return TRUE unless you set the focus to a control
    // 异常: OCX 属性页应返回 FALSE
}
```

```
//// 传入棋类指针
void        CDialogMore::SetChess(CChess    *pChess)
{
      m_pChess        = pChess;
}
```

4.4. 棋盘存储

对于整个棋盘的存储与保存：

若采用静态存储，则需要 15*15*sizeof(UINT) = 900B 的内存空间。在本方案中采用下标做为位置的坐标，也会随之节省 2*900B 的孔家；

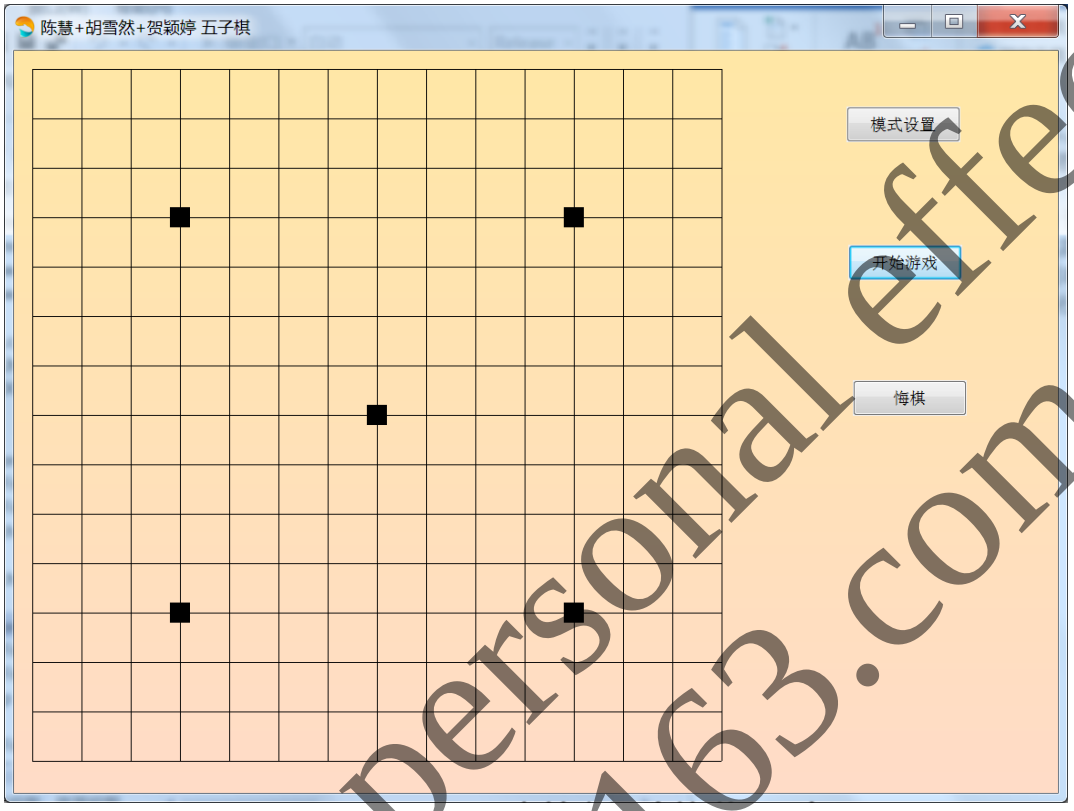
若采用动态存储，大部分情况只需要 3*sizeof(UINT)*25 到 3*sizeof(UINT)*100（一般下 25—100 个棋子便可分出胜负）即 300B—1200B，这样在棋局前期能节省部分内存。

但当判断胜负及执行 AI 时，由于动态存储不具有有序性，所以需采取全部遍历或先排序再判断的方法，由于每次下棋都需要遍历或排序，那么将导致 CPU 增大很多。

由上可知，动态存储 CPU 利用率大，前期才节约了 600B 内存，后期还多耗内存。所以本方案中采用静态存储的方法进行整个棋盘的存储于保存。

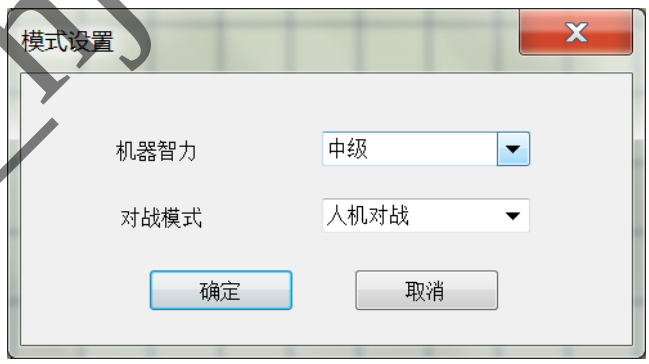
5. 测试数据及其结果分析

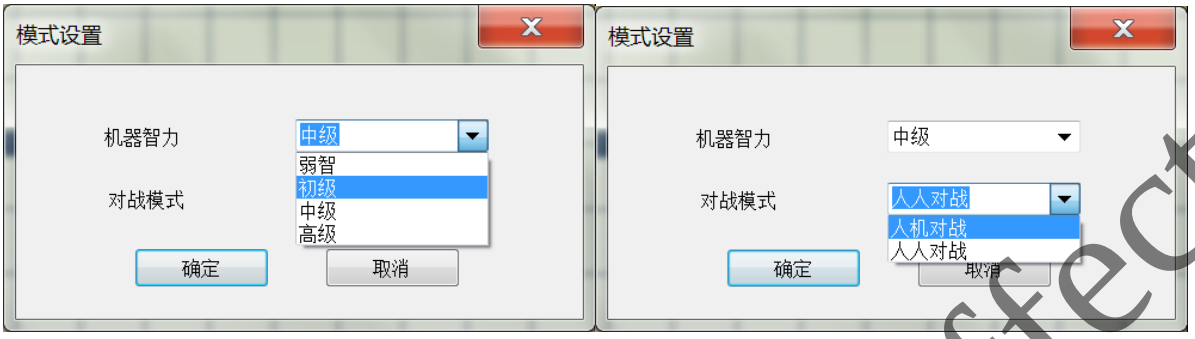
（1）主界面对话框：



在本界面中，棋盘设置距离对话框边界一定距离，按照 15*15 的行数和列数，绘制横线和竖线，完成 225 个小方格的绘制。在棋盘上，将指向棋盘区域的变量向中心缩 3 小格，接着在新区域的左上，右上，左下，右下，中心五个点，绘制小黑正方形，即完成整个棋盘的绘制。同时，在整个对话框的右边区域放置三个触发按钮，以进行后续操作。

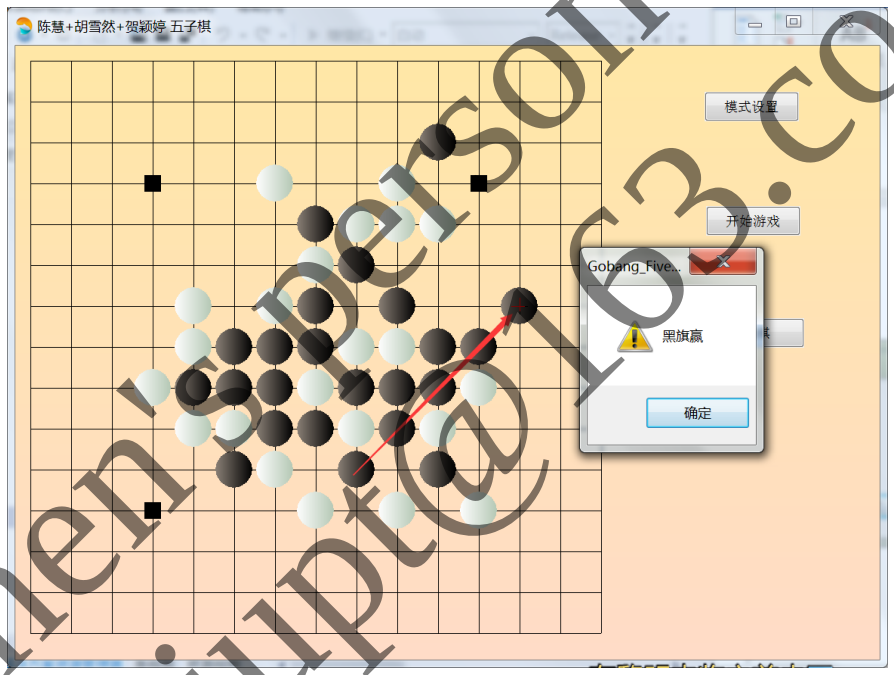
（2）模式设置对话框：





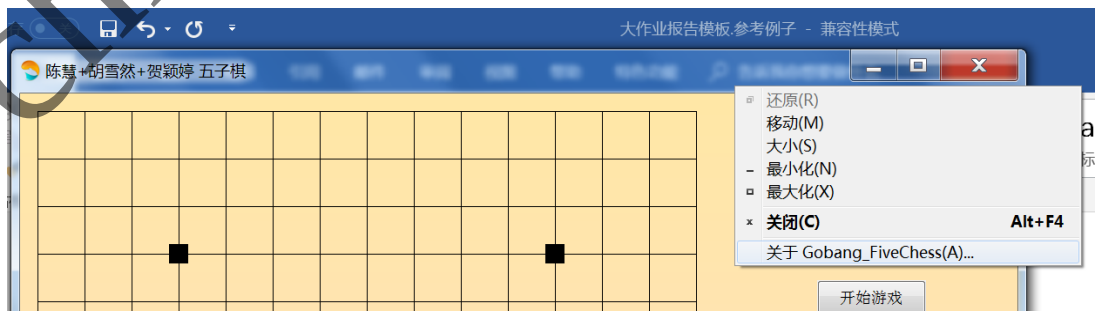
点击“模式设置”按钮后，弹出该对话框，可以选择“机器智力”和“对战模式”，“机器智力”包括“弱智、初级、中级、高级”四个等级，“对战模式”包括“人人对战、人机对战”两种模式，其中任意两个可以两两组合，进行游戏，点击“确定”按钮，客户区即重新设置，开始该种模式的游戏。

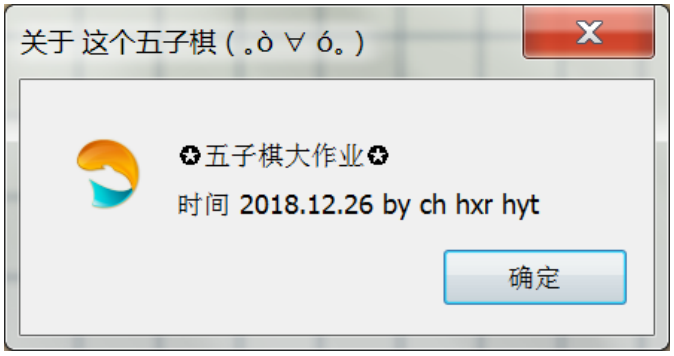
（3）人机大战，获胜，提示“黑棋赢”对话框：



只要系统通过判断输赢的程序，判断某一方胜出，系统立刻自动弹出“黑棋赢”或“白棋赢”对话框。

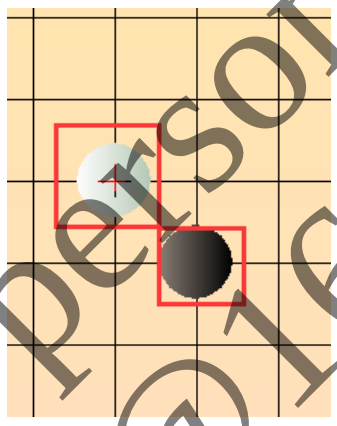
（4）“关于…”对话框





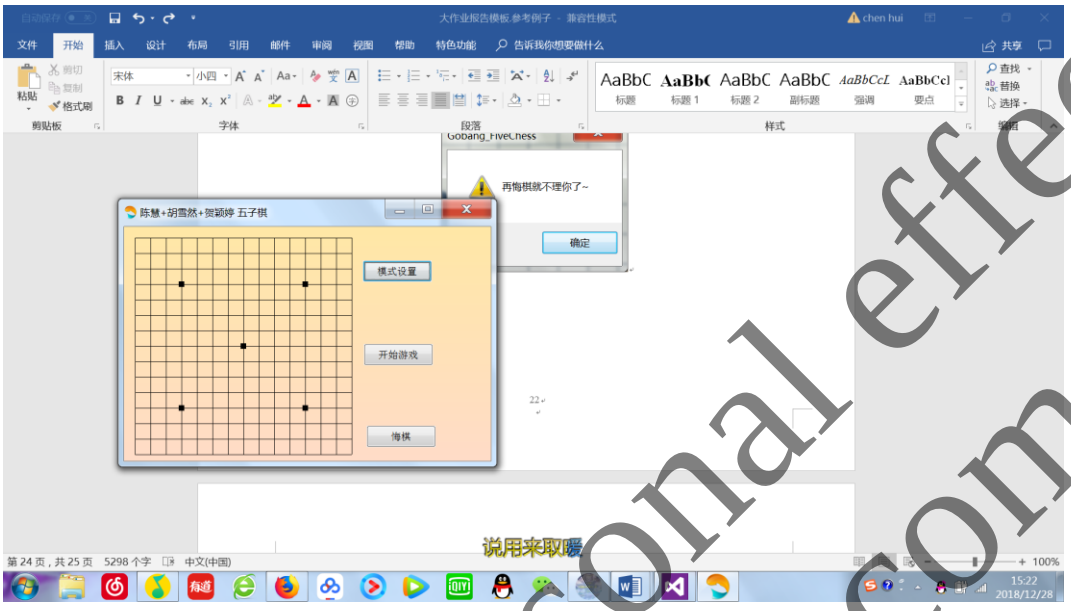
在主界面上边界的空白处，右击，即可发现“关于…”对话框已成功嵌入主界面对话框中，点击，即可弹跳出这个程序的一些简介，包括完成时间和作者。

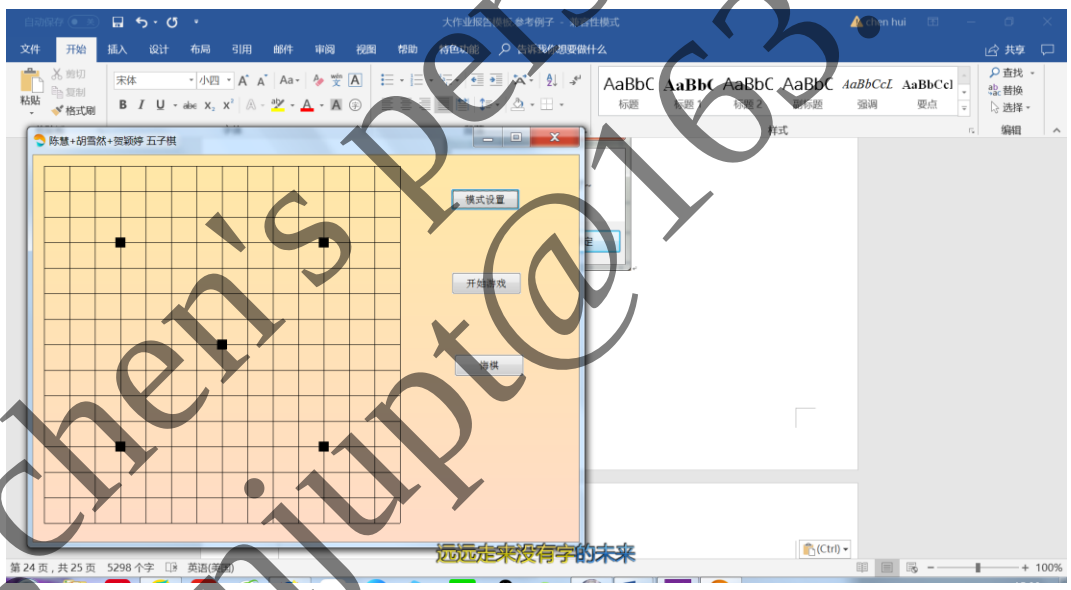
（5）新棋子“十”字型标记，旧棋子取消标记：



（6）允许一次悔棋，第二次悔棋，弹出提示对话框：

（7）屏幕缩放及最小化功能





从上面两图的对比，可以看出主界面可以进行随意缩放，并且界面上的棋盘和按钮不会发生移位和重叠等问题

6. 调试过程中的问题

（1）Windows 对话框分为两类：模态对话框和非模态对话框。模态对话框是当它弹出后，本应用程序其他窗口将不再接受用户输入，只有该对话框响应用户输入，在对它进行相应操作退出后，其他窗口才能继续与用户交互。非模态对话框则是，它弹出后，本程序其他窗口仍能响应用户输入。非模态对话框一般用来显示提示信息等。在本程序中，对于除主界面以外的对话框都需要设置为非模态对话框，才能够进行相应操作，即在弹出对话框时，使用对话框类的 DoModal()函数

```
CDialogMore* dlgMore    = new      CDialogMore;
dlgMore->Create(IDD_DIALOG_MORE);
dlgMore->ShowWindow(SW_SHOW);          // 非模态
dlgMore->DoModal()
dlgMore->SetChess(&m_chess);
```

（2）在对对话框进行缩放和最小化操作时，为了保证主界面上的按钮能够随对话框大小位置而移动，必须要进行定义 CRgn 类对象，对每个按钮操作，首先获取按钮在父窗口的位置 GetDlgItem(IDC_BUTTON_GAME_START)->GetWindowRect(rcBtnStart)，再把屏幕上指定点的屏幕坐标转换成用户坐标 ScreenToClient(rcBtnStart)，接着剪切按钮的位置，创建的剪裁区域大小，保持和控件一致 rgnBtnStart.CreateRectRgnIndirect(rcBtnStart)，最后还要完成善后处理 rgnBtnStart.DeleteObject()，即释放相应对象，才可以成功完成对话框的缩放功能。

7. 大作业报告总结

虽然这学期已经选修了 windows 编程这门课程，对 windows 编程已经有了一些基本的了解。但是凭借自身力量自行编写一个能够在真机上安装运行的程序对于我们而言依旧是一项未知的难题。从书本理论到现实程序调试的跨越并没有一开始想象中的那么一帆风顺。

在大一大二对于 c 语言学习的基础上，我们选择 Visual studio2012 作为我们编程的环境。在查阅资料的基础上，进行五子棋对弈游戏程序的开发。在整个开发过程中，我们遇到了各种问题，但是仔细查看问题原因，会发现只要编程过程中再细心一点就可以解决。对于 windows 编程中的对话框编辑、按钮设计和控件布局有了更深刻的了解。最后，在整个的开发过程中，我们学到了很多之前没有注意过的新知识，这其中最重要的就是 windows 编程库中的众多的类，他们代表不同的功能，又可以映射各种不同的函数，例如：CRgn 封装了一个 Windows 图形设备接口(GDI)区域，这一区域是某一窗口中的一个椭圆或多边形区域；CPen 类是画笔的对象，用来在 DC 上完成绘制线条的任务；宏开始消息映射，可以为每个消息处理函数加入一个入口等等。在之前的书本学习中，只关注了一些基本的操作，并没有想到 windows 可以有这么多的自带库以供我们调用，这对以后的 windows 开发有很大的帮助。

此次期末考核作业的完成，让我体会到了亲身实践的乐趣。书本和实践中相距着难以跨越的鸿沟，而若不通过实践，书本知识则显得索然无味，意义颓然。感谢本次实验周开拓了我的视野，锻炼了我的动手能力，也同时加强了自身查阅资料和与人交流学习的能力。在今后，我也会更加努力，遵循实验周的本意所在，在学习生活中多加尝试，使自己对书本的理解更加深入和透彻。