高级语言程序设计大作业（三）

大作业报告

恐暝棋

学　号：　2353900

姓　名：　汪嘉晨

学　院：　国豪书院

班　级：　工科二班

二零二四年六月十八日

一、设计思路与功能概述

(一)设计思路

孔明棋，又名~~恐暝棋~~（Peg Solitaire），是一种智力游戏，通常由一个棋盘和若干颗棋子组成。最经典的 Solitaire 棋盘盘面如下图所示，一个 33 孔的的盘面上摆放了 32 颗棋子（如图 1）。

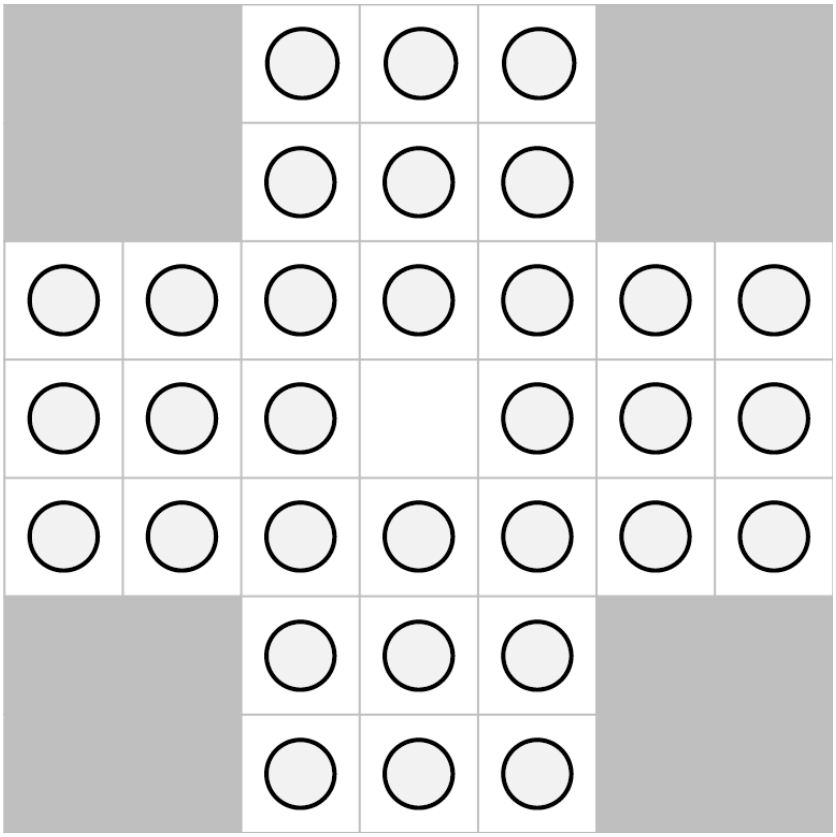

图 1 照抄原题图片

孔明棋小游戏整体的设计思路如下

0.头文件：一些有的没的+easy.x 要用的+栈要用的，并定义保存移动信息的结构体与保存移动历史的栈（如图 2）；

```
#include <iostream>
#include <graphics.h>
#include <easyx.h>
#include <stack>
using namespace std;

// 定义保存移动信息的结构体
struct Move {
    int x1, y1; // 移动前的坐标
    int x2, y2; // 移动后的坐标
    int xOver, yOver; // 被跳过的棋子的坐标
};
stack<Move> moveHistory; // 创建保存移动历史的栈
```

图 2 主要设计思路——写在前面的一些有的没的

1.棋盘&计算类：主要负责棋盘数组定义，计算剩余棋子，计算可移动位置，选中某棋子，判断胜利、失败等（如图 3）；

①数组定义：初始化孔明棋棋盘——0：空 1：棋子 2：选中的棋子 3：可移动的位置 4：非游戏区域与一个二维数组；

②计算剩余棋子：即遍历整个数组查看 1 与 2 棋子的数目；

③计算可移动位置：如果选中了某个棋子，那么隔一个棋子且不出棋盘的棋子为可以移动的位置；

④选中棋子：即将 1 号转化成 2 号，并套用“③”，得到可移动的位置；

⑤判断胜利：只剩下一个棋子；

⑥判断失败：如果遇到走不了&&还剩下多于一枚棋子的情况，则判定为失败。这里多加了一个 canMove 函数判断能都上下左右移动某个棋子，如果有就可以继续走。

⑦悔棋：首先定义了一个 Move 结构体来保存每次移动的详细信息。在 moveChess 函数中，每次移动时都会创建一个 Move 对象并将其压入 moveHistory 栈中。undoMove 函数则用于实现悔棋功能，它从栈中弹出最后一次移动的信息，并据此恢复棋盘状态。

对于多次悔棋也是支持的：每次玩家移动棋子时，移动的信息都会被压入 moveHistory 这个栈中。当执行悔棋操作时，它会从栈中弹出最后一次移动的信息并恢复。只要栈中还有元素，就可以继续执行悔棋操作。

```
15    //棋盘&计算类：
16    //初始化孔明棋棋盘——0：空 1：棋子 2：选中的棋子 3：可移动的位置 4：非游戏区域
17    int board[7][7] = { ... }
25    //计算剩余棋子
26    int countChessPieces() { ... }
41    //计算可移动位置
42    void countMove(int x, int y) { ... }
64    //移动棋子并将移动信息压入栈中
65    void moveChess(int x1, int y1, int x2, int y2) { ... }
76    //选中棋子并计算可移动位置
77    void selectChess(int x, int y) { ... }
82    // 悔棋
83    void undoMove() { ... }
95
96    //判断胜利
97    bool judgeWin() { ... }
108   //判断失败（检查能否继续移动）
109   bool canMove(int x, int y) { ... }
133   bool judgeLose() { ... }
```

图 3 主要设计思路——棋盘&计算类

2.绘制类：绘制棋盘、棋子、控制按钮、标题、提示剩余棋子、主要界面等（如图 4）；

①绘制棋盘：根据定义的页面大小与棋盘数组，通过调整颜色，调整线条粗细，可以制作精美的棋盘（它橙黄相间一目了然，~~粉色打底也不要认为我两次元~~）；

②绘制棋子，按照数组，数值为 0 的代表空，就不画；数值为 1 的是普通棋子，画个红圆；数值为 2 的是选中棋子，画个浅一点的红圆；数值为 3 的是

选中棋子可以走的地方，画个小点的绿圆代表可以通行；

③显示剩余棋子位于左上角，通过显示剩余棋子函数返回的数值进行显示；

④孔明棋标题位于右下角，标题，采用某种花里胡哨的字体“汉仪尚巍手书 W”书写“孔明棋”三个大字，亲测这个字体霸气！

⑤棋盘右侧又设置了五个控制按钮，从上到下为退出、重玩、悔棋、规则与更多随机，5 个按钮绘制了橙边的圆角矩形，醒目美观；

⑥为了方便，有设置了“显示主要界面函数”，方便统一调用。

```
153    //绘制类:
154    //绘制棋盘
155    void drawChessBoard() { ... }
184    //绘制棋子
185    void drawChessPieces() { ... }
210    //绘制控制按钮
211    void drawButtons() { ... }
235    //绘制显示剩余棋子
236    void drawCount() { ... }
253    //绘制“孔明棋”标题
254    void drawTitle()
255    {
256        LOGFONT f;
257        gettextstyle(&f);                       // 获取当前字体设置
258        _tcscpy_s(f.lfFaceName, _T("汉仪尚巍手书W")); // 设置字体
259        f.lfHeight = 160;                        // 设置字体大小为48
260        f.lfQuality = ANTIALIASED_QUALITY;      // 设置抗锯齿效果
261        settextstyle(&f);                       // 应用字体设置
262        settextcolor(BLACK);                    // 设置文字颜色为黑色
263        setbkmode(TRANSPARENT);                 // 设置文字背景为透明
264        outtextxy(600, 560, _T("孔明棋"));
265    }
266    //显示主要界面
267    void drawGameInterface() { ... }
```

图 4 主要设计思路——绘制类

3. 界面类：由于程序较为简洁，只包括显示规则的界面（如图 5），玩家点选提示后点击“返回游戏”（这里又设置了个按钮（圆角矩形））。

```
// 规则界面:
void rule()
{
    initgraph(850, 450, 1);
    setbkcolor(RGB(255, 210, 220)); //淡粉色
    cleardevice();
    // 设置文字样式
    LOGFONT f;
    gettextstyle(&f);                       // 获取当前字体设置
    _tcscpy_s(f.lfFaceName, _T("隶书"));    // 设置字体
    f.lfHeight = 32;                        // 设置字体大小为36
    f.lfQuality = ANTIALIASED_QUALITY;      // 设置抗锯齿效果
    settextstyle(&f);                       // 应用字体设置
    settextcolor(BLACK);                    // 设置文字颜色为黑色
    setbkmode(TRANSPARENT);                 // 设置文字背景为透明
    // 绘制规则界面
    outtextxy(50, 40, _T("孔明棋规则："));
    outtextxy(50, 80, _T("1. 棋子为红色圆形，选中的位置为浅红色圆形，可移"));
    outtextxy(50, 120, _T("    动的位置为绿色圆形。"));
    outtextxy(50, 160, _T("2. 棋子只能横向或纵向移动，且只能跳过一个棋子。"));
    outtextxy(50, 200, _T("3. 棋子移动后，被跳过的棋子会消失。"));
    outtextxy(50, 240, _T("4. 棋子移动后，若无法继续移动，则游戏失败。"));
    outtextxy(50, 280, _T("5. 如果只剩下一枚棋子，则游戏胜利。"));
    outtextxy(50, 320, _T("6. 通过悔棋按钮可以实现悔棋。"));
    outtextxy(50, 360, _T("7. 关闭窗口请点击“返回游戏”>>"));

    // 绘制一个“返回游戏”按钮
    setfillcolor(YELLOW);
    setlinecolor(RGB(255, 165, 0));
    fillroundrect(550, 360, 700, 400, 10, 10);
    _tcscpy_s(f.lfFaceName, _T("等线"));
    settextcolor(RED);                    // 设置文字颜色为黑色
    setbkmode(TRANSPARENT);               // 设置文字背景为透明
    settextstyle(&f);                     // 应用字体设置
    outtextxy(560, 365, _T("返回游戏"));
    // 消息循环，等待用户操作
    bool running = true;
    while (running)
    {
        ExMessage m;        // 获取鼠标消息
        if (peekmessage(&m, EM_MOUSE))
        {
            // 检查是否点击了“返回游戏”按钮的区域
            if (m.message == WM_LBUTTONDOWN && m.x >= 550 && m.x <= 700 && m.y >= 360 && m.y <= 400)
            {
                running = false; // 结束循环
            }
        }
    }
    // 关闭规则窗口后重新绘制游戏界面
    initgraph(1000, 760, 0);
    setbkcolor(RGB(255, 210, 220)); //淡粉色
    drawGameInterface();
}
```

图 5 主要设计思路——界面类

4.处理鼠标操作（如图 6）。使用了 EasyX 图形库中的 GetMouseMsg 函数来获取鼠标事件，具体函数见下图。

```cpp
// 处理鼠标点击事件
pair<int, int> handleMouseClick()
{
    MOUSEMSG m = GetMouseMsg();
    switch (m.uMsg)
    {
    case WM_LBUTTONDOWN: // 如果是左键按下
        return { m.x, m.y }; // 返回鼠标点击的屏幕坐标
    }
    return { -1, -1 }; // 如果没有点击，返回无效坐标
}
```

图 6 主要设计思路——处理鼠标操作

5. 主函数，主要分为初始化图形窗口，显示页面，鼠标操作进行相应运行的循环与关闭窗口（如图 7）。

```cpp
int main()
{
    // 初始化图形窗口，窗口大小为1000*760，背景白色
    initgraph(1000, 760, 1);
    setbkcolor(RGB(255, 210, 220)); //淡粉色
    //绘制棋盘，棋子和控制按钮
    drawGameInterface();
    int moregame = 0;
    //处理鼠标点击并执行操作
    while (true)
    {
        pair<int, int> clickPos = handleMouseClick();
        if (clickPos.first != -1) // 检查是否有有效的点击
        {
            //退出按钮
            if (clickPos.first >= 800 && clickPos.first <= 900 && clickPos.second >= 50 && clickPos.second <= 100)
            {
                break; // 退出游戏
            }
            //重玩按钮
            else if (clickPos.first >= 800 && clickPos.first <= 900 && clickPos.second >= 150 && clickPos.second <= 200){ ... }
            //悔棋按钮
            else if (clickPos.first >= 800 && clickPos.first <= 900 && clickPos.second >= 250 && clickPos.second <= 300){ ... }
            //规则按钮
            else if (clickPos.first >= 800 && clickPos.first <= 900 && clickPos.second >= 350 && clickPos.second <= 400){ ... }
            //随机残局按钮
            else if (clickPos.first >= 800 && clickPos.first <= 900 && clickPos.second >= 450 && clickPos.second <= 500){ ... }
            //点击棋盘
            MOUSEMSG m = GetMouseMsg();
            int x = (m.y - 120) / 80; // 计算行坐标
            int y = (m.x - 120) / 80; // 计算列坐标
            if (x >= 0 && x < 7 && y >= 0 && y < 7){ ... }
            // 重新绘制棋盘和棋子
            drawGameInterface();
            //判断成功失败
            if (judgeWin())
            {
                MessageBox(GetHWnd(), _T("恭喜你，游戏胜利！"), _T("游戏结束"), MB_OK);
                break;
            }
            if (judgeLose())
            {
                MessageBox(GetHWnd(), _T("很遗憾，游戏失败！"), _T("游戏结束"), MB_OK);
                break;
            }
        }
    }
    closegraph();// 关闭图形窗口
    return 0;
}
```

图 7 主要设计思路——主函数

(二)功能概述

孔明棋小游戏整体的功能如下：

0.主界面，包括游戏棋盘，退出，悔棋，重玩，规则与随机残局（照着某音某站一些资料编制，~~数量有限并没有那么随机~~），~~（easyX 画的）~~简约而~~（程序实在是）~~不简单（如图 8）；

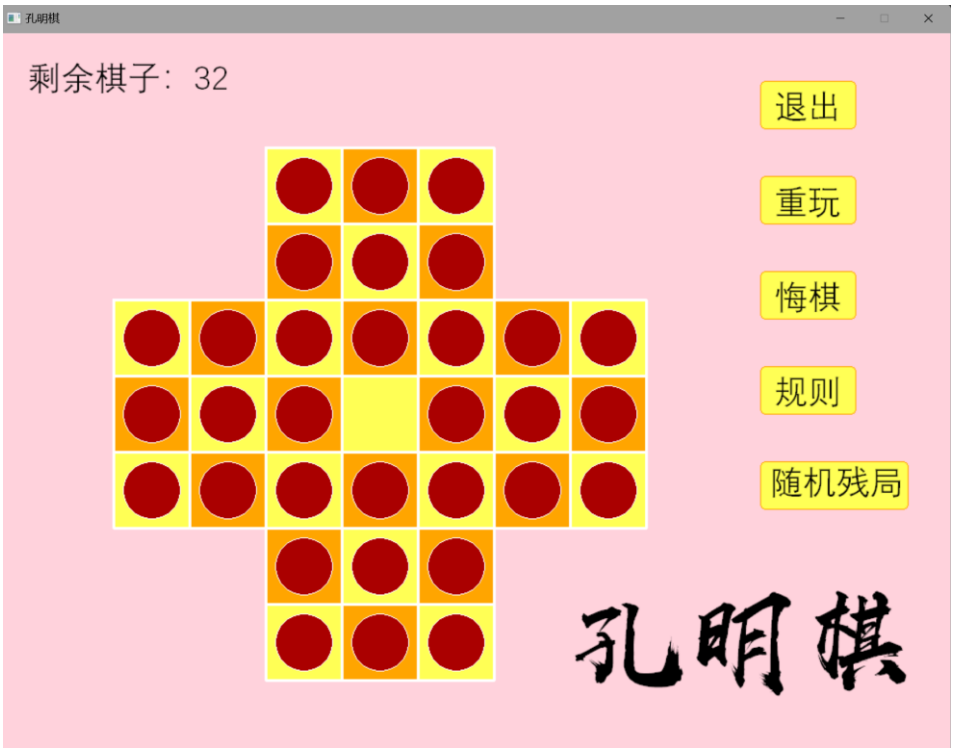

图 8 主界面

1.退出，用于游戏退出，直接关闭并 return 0；

2.悔棋、重玩，自如其名，棋盘也会进行相应的更新；

3.规则，点选规则以后会弹出规则界面，点击“返回游戏”会回到主界面（如图 9）；

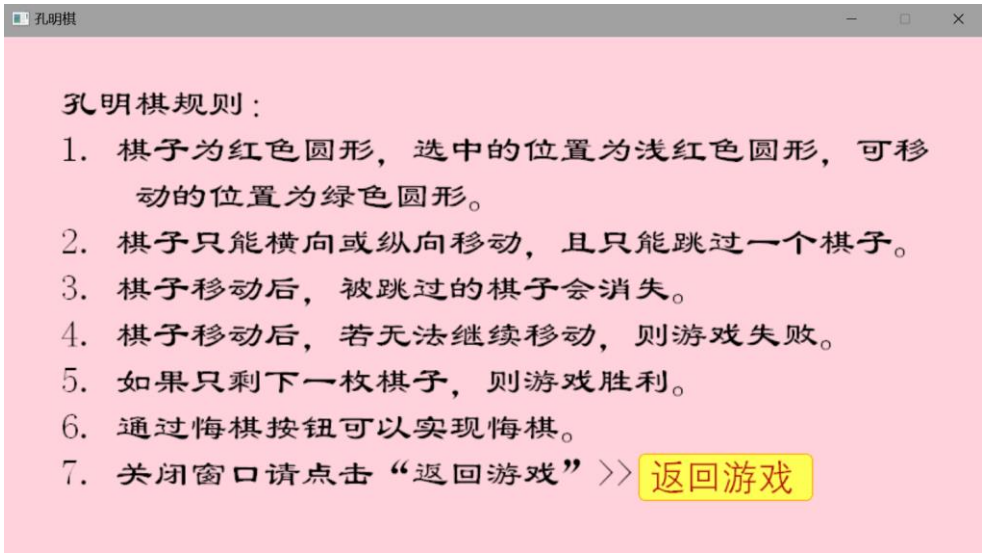

图 9 规则界面

4. 残局：点击随机残局，会显示一些残局关卡（见图 10）；

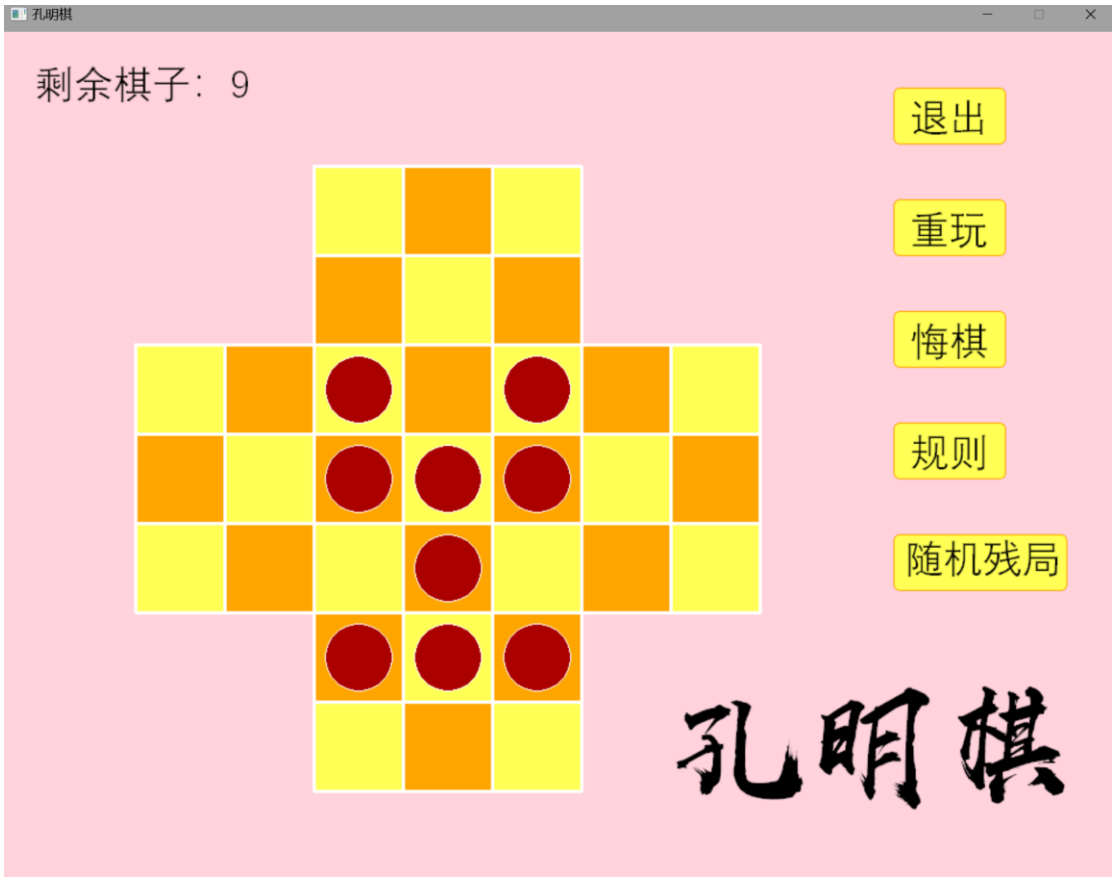

图 10 残局

5. 判断成功失败，如图组 11-13，不同类别的失败都能进行显示！

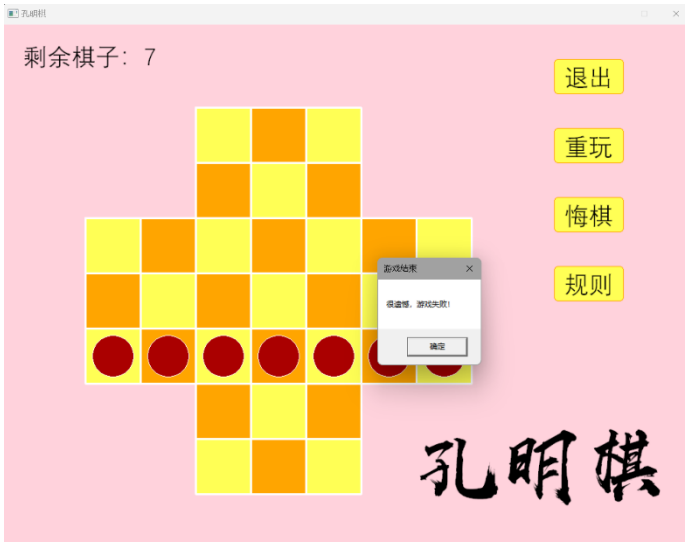

图 11 一行卡住的判定为失败

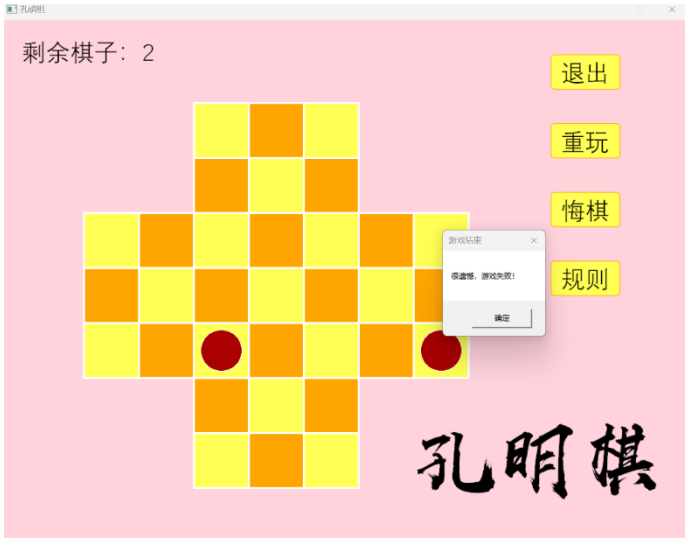

图 12 两棋不相邻判定为失败

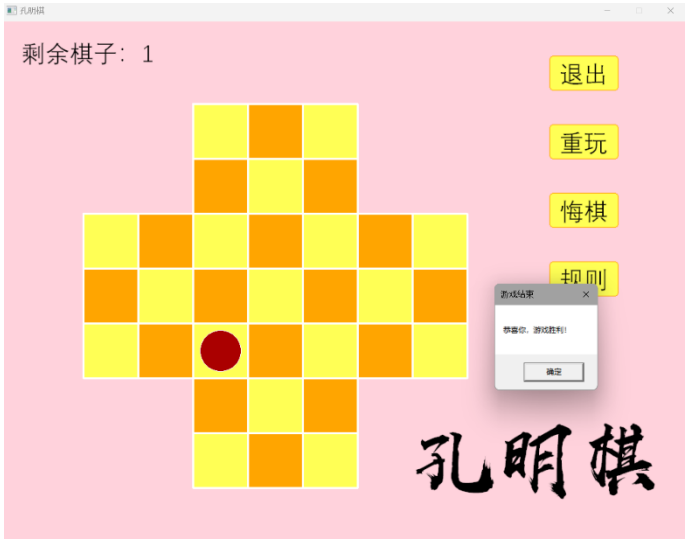

图 13 只剩一个棋子判定成功

二、问题与解决方法

Q1：快没时间写孔明棋了，怎么办？

A1：焦虑不会解决问题，抓紧上手！“种树最好的时间是十年前，其次是现在！”于是一个通宵再加上两个晚上，把孔明棋基本的要求再加上 easyx 搞完，只要不赶 ddl 什么时间都不晚！

Q2：界面怎么去绘制？上手好困难！

A2：（绘制界面的时长远多于编写棋盘与计算函数的时长!!!）运用 easyx 工具，浅读了教程并刷了 b 站上的一堆速成视频，终于明白如何去绘制，于是我就画了草图，大概什么地方放什么？具体的行宽列宽是多少？颜色是什么？为什么输入 ORANGE 会报错？（最后用 RGB 格式代替了）棋子如何和二维数组建立联系？这些问题思考以后绘制界面的问题就迎刃而解。

Q3:卡顿问题：一下点数次会引发闪屏现象

A3：经过网上的搜索，引入了：

```
BeginBatchDraw();
EndBatchDraw();
```

来减少卡顿，发现效果立竿见影。

Q4:如何解决这烦人的悔棋？

A4：首先定义了一个 Move 结构体来保存每次移动的详细信息。在 moveChess 函数中，每次移动时都会创建一个 Move 对象并将其压入 moveHistory 栈中。undoMove 函数则用于实现悔棋功能，它从栈中弹出最后一次移动的信息，并据此恢复棋盘状态。当然还是感谢伟大的**蔻派乐特博士**和**文心先生**的大力帮助！

Q5：我加了文字后为什么会有白色底纹难看死了！

A5：底纹换成透明颜色即可 `setbkmode(TRANSPARENT);`

Q6:怎么去掉烦人的控制台啊？？？？？

A6：似乎无解呜呜呜┬┬﹏┬┬，编写程序的时候经常忘记关掉，导致再打开程序的时候会报错，但是为什么在我写报告准备截报错的图这个问题又解决了 hhh

三、心得体会

0.本次大作业如果对于只是完成可操作的程序感觉较为简单，但是在这个基础上加上 easyx 再加上自己的风格再加上残局再加上 AI 指导确实不分是力不从心（如果没有懒惰的心理||如果没有考试周兴许能够完成 AI 并优化的更好）。

1.由于拖到了距离 ddl 倒数第二周，再加上期末周复习（预习）的压力，我确实进度加快了不少，一天再加上一个晚上进行页面的制作&&一天时间进行棋盘函数等等的调试&&半天时间加了点诸如残局的有的没的，在花上半天写个打报告，确实**有效率**！正如顶针所说：

2.本次大作业，我学会了使用 easyx 绘画界面，处理鼠标信息，对人机交互理解更加深刻，看着大家的成果我也尽力让自己的变得更好，互相学习互相进步互相给压力。

3.做完了本次大作业，有幸和**程序本人**合影，接下来就是海报和视频啦(*ˆ▽ˆ*)被高程折磨得四个月到此结束，至于九月的高程进阶，下学期的事下学期再说吧(*ˆ▽ˆ*)

4.哦对了，国豪最大地下组织，由张博士牵头发起的嫜姮书院成立（bushi），感谢他们对我高程学习的帮助。（不知道为什么打了马赛克）

5.最后还是想在此对陈老师和她的各位助教们道声感谢！感谢群内的各路大神！没有你们就不会有现在的我！

四、源代码

~~感觉自我良好的我就黄底啦！~~

```
#include <iostream>
#include <graphics.h>
#include <easyx.h>
#include <stack>
using namespace std;

// 定义保存移动信息的结构体
struct Move {
	int x1, y1; // 移动前的坐标
	int x2, y2; // 移动后的坐标
	int xOver, yOver; // 被跳过的棋子的坐标
};
stack<Move> moveHistory; // 创建保存移动历史的栈


//棋盘类:
//初始化孔明棋棋盘——0：空 1：棋子 2：选中的棋子 3：可移动的位置 4：非游戏区域
int board[7][7] = {
	{4,4,1,1,1,4,4},
	{4,4,1,1,1,4,4},
	{1,1,1,1,1,1,1},
	{1,1,1,0,1,1,1},
	{1,1,1,1,1,1,1},
	{4,4,1,1,1,4,4},
	{4,4,1,1,1,4,4} };


//计算类:
//计算剩余棋子
int countChessPieces()
{
	int count = 0;
	for (int i = 0; i < 7; i++)
	{
		for (int j = 0; j < 7; j++)
		{
			if (board[i][j] == 1 || board[i][j] == 2)
			{
				count++;
			}
		}
	}
	return count;
}
```

```
//计算可移动位置
void countMove(int x, int y)
{
    if (board[x][y] == 2)
    {
        if (x - 2 >= 0 && board[x - 1][y] == 1 && board[x - 2][y] == 0)
        {
            board[x - 2][y] = 3;
        }
        if (x + 2 < 7 && board[x + 1][y] == 1 && board[x + 2][y] == 0)
        {
            board[x + 2][y] = 3;
        }
        if (y - 2 >= 0 && board[x][y - 1] == 1 && board[x][y - 2] == 0)
        {
            board[x][y - 2] = 3;
        }
        if (y + 2 < 7 && board[x][y + 1] == 1 && board[x][y + 2] == 0)
        {
            board[x][y + 2] = 3;
        }
    }
}
//移动棋子并将移动信息压入栈中
void moveChess(int x1, int y1, int x2, int y2)
{
    if (board[x1][y1] == 2 && board[x2][y2] == 3)
    {
        Move move = { x1, y1, x2, y2, (x1 + x2) / 2, (y1 + y2) / 2 };
        moveHistory.push(move); //  将移动信息压入栈中
        board[x1][y1] = 0;
        board[x2][y2] = 1;
        board[(x1 + x2) / 2][(y1 + y2) / 2] = 0;
    }
}
//选中棋子并计算可移动位置
void selectChess(int x, int y)
{
    board[x][y] = 2;
    countMove(x, y);
}
//  悔棋
void undoMove()
{
```

```
    if (!moveHistory.empty())
    {
        Move lastMove = moveHistory.top(); // 获取最后一次移动信息
        moveHistory.pop(); // 从栈中移除
        // 恢复棋盘状态
        board[lastMove.x1][lastMove.y1] = 1; // 恢复移动前的棋子
        board[lastMove.x2][lastMove.y2] = 0; // 清除移动后的位置
        board[lastMove.xOver][lastMove.yOver] = 1; // 恢复被跳过的棋子
    }
}

//判断胜利
bool judgeWin()
{
    if (countChessPieces() == 1)
    {
        return true;
    }
    else
    {
        return false;
    }
}
//判断失败（检查能否继续移动）
bool canMove(int x, int y)
{
    // 检查向左移动
    if (x - 2 >= 0 && (board[x - 1][y] == 1 || board[x - 1][y] == 2) && (board[x - 2][y] == 0 ||
board[x - 2][y] == 3))
    {
        return true;
    }
    // 检查向右移动
    if (x + 2 < 7 && (board[x + 1][y] == 1 || board[x + 1][y] == 2) && (board[x + 2][y] == 0 ||
board[x + 2][y] == 3))
    {
        return true;
    }
    // 检查向上移动
    if (y - 2 >= 0 && (board[x][y - 1] == 1 || board[x][y - 1] == 2) && (board[x][y - 2] == 0 ||
board[x][y - 2] == 3))
    {
        return true;
    }
```

```
    // 检查向下移动
    if (y + 2 < 7 && (board[x][y + 1] == 1 || board[x][y + 1] == 2) && (board[x][y + 2] == 0 ||
board[x][y + 2] == 3))
    {
        return true;
    }
    return false;
}
bool judgeLose()
{
    for (int i = 0; i < 7; i++)
    {
        for (int j = 0; j < 7; j++)
        {
            // 对于每个1号、2号棋子，检查是否有合法的移动
            if (board[i][j] == 1 || board[i][j] == 2)
            {
                if (canMove(i, j))
                {
                    return false; // 如果找到至少一个棋子有合法移动，游戏未失败
                }
            }
        }
    }
    return true; // 如果没有找到任何合法移动，游戏失败
}


//绘制类:
//绘制棋盘
void drawChessBoard()
{
    //绘制棋盘线条
    //棋盘7行7列，每个格子边长为80，但是四个角各2行2列格子不绘制
    for (int i = 0; i < 7; i++)
    {
        for (int j = 0; j < 7; j++)
        {
            if (board[i][j] == 4)
            {
                continue;
            }
            else
            {
```

```
                    if ((i + j) % 2 == 0)
                        setfillcolor(YELLOW); //  偶数格黄色
                    else
                        setfillcolor(RGB(255, 165, 0)); //  奇数格橙色
                    // 绘制格子
                    setlinestyle(PS_SOLID, 3);//粗一点
                    setlinecolor(WHITE);
                    fillrectangle(120 + j * 80, 120 + i * 80, 200 + j * 80, 200 + i * 80);
                    // 恢复线条样式为默认
                    setlinestyle(PS_SOLID, 1);
                }
            }
        }

}
//绘制棋子
void drawChessPieces()
{
    for (int i = 0; i < 7; i++)
    {
        for (int j = 0; j < 7; j++)
        {
            setlinestyle(PS_SOLID, 1);
            if (board[i][j] == 1)
            {
                setfillcolor(RED);
                fillcircle(160 + j * 80, 160 + i * 80, 30);
            }
            else if (board[i][j] == 2)
            {
                setfillcolor(RGB(255, 102, 102));//浅红色
                fillcircle(160 + j * 80, 160 + i * 80, 30);
            }
            else if (board[i][j] == 3)
            {
                setfillcolor(GREEN);
                fillcircle(160 + j * 80, 160 + i * 80, 15);
            }
        }
    }
}
//绘制控制按钮
void drawButtons()
{
```

```
    // 设置填充颜色为黄色，线条颜色为橙色
    setfillcolor(YELLOW);
    setlinecolor(RGB(255, 165, 0));
    // 设置文字样式
    LOGFONT f;
    gettextstyle(&f);                       // 获取当前字体设置
    _tcscpy_s(f.lfFaceName, _T("等线"));    // 设置字体为等线
    f.lfHeight = 36;                        // 增大字体大小
    setbkmode(TRANSPARENT); // 设置文字背景为透明
    settextcolor(BLACK);      // 设置文字颜色为黑色
    f.lfQuality = ANTIALIASED_QUALITY;      // 设置抗锯齿效果
    settextstyle(&f);                       // 应用字体设置
    // 绘制圆角矩形按钮
    fillroundrect(800, 50, 900, 100, 10, 10);
    outtextxy(815, 60, _T("退出"));
    fillroundrect(800, 150, 900, 200, 10, 10);
    outtextxy(815, 160, _T("重玩"));
    fillroundrect(800, 250, 900, 300, 10, 10);
    outtextxy(815, 260, _T("悔棋"));
    fillroundrect(800, 350, 900, 400, 10, 10);
    outtextxy(815, 360, _T("规则"));
    fillroundrect(800, 450, 955, 500, 10, 10);
    outtextxy(810, 455, _T("随机残局"));
}
//绘制显示剩余棋子
void drawCount()
{
    LOGFONT f;
    gettextstyle(&f);                       // 获取当前字体设置
    _tcscpy_s(f.lfFaceName, _T("等线"));    // 设置字体为等线
    f.lfHeight = 36;                        // 设置字体大小为36
    f.lfQuality = ANTIALIASED_QUALITY;      // 设置抗锯齿效果
    settextstyle(&f);                       // 应用字体设置
    settextcolor(BLACK);                    // 设置文字颜色为黑色
    setbkmode(TRANSPARENT);                 // 设置文字背景为透明
    // 获取剩余棋子数量
    int count = countChessPieces();
    TCHAR str[100];
    _stprintf_s(str, _T("剩余棋子：%d"), count); // 格式化字符串，包含剩余棋子数量
    // 绘制文字
    outtextxy(30, 30, str); // 输出包含剩余棋子数量的字符串
}
//绘制“孔明棋”标题
void drawTitle()
```

```
{
    LOGFONT f;
    gettextstyle(&f);                       // 获取当前字体设置
    _tcscpy_s(f.lfFaceName, _T("汉仪尚巍手书W")); // 设置字体
    f.lfHeight = 160;                       // 设置字体大小为48
    f.lfQuality = ANTIALIASED_QUALITY;      // 设置抗锯齿效果
    settextstyle(&f);                       // 应用字体设置
    settextcolor(BLACK);                    // 设置文字颜色为黑色
    setbkmode(TRANSPARENT);                 // 设置文字背景为透明
    outtextxy(600, 560, _T("孔明棋"));
}
//显示主要界面
void drawGameInterface()
{
    BeginBatchDraw();
    EndBatchDraw();
    cleardevice();
    drawChessBoard();
    drawChessPieces();
    drawButtons();
    drawTitle();
    drawCount();
}

int main()
{
    // 初始化图形窗口，窗口大小为1000*760，背景白色
    initgraph(1000, 760, 1);
    setbkcolor(RGB(255, 210, 220)); //淡粉色
    //绘制棋盘，棋子和控制按钮
    drawGameInterface();
    int moregame = 0;
    //处理鼠标点击并执行操作
    while (true)
    {
        pair<int, int> clickPos = handleMouseClick();
        if (clickPos.first != -1) // 检查是否有有效的点击
        {
            //退出按钮
            if (clickPos.first >= 800 && clickPos.first <= 900 && clickPos.second >= 50 &&
clickPos.second <= 100)
            {
                break; // 退出游戏
            }
```

```
            //重玩按钮
            else if (clickPos.first >= 800 && clickPos.first <= 900 && clickPos.second >= 150
&& clickPos.second <= 200)
            {

                int refreshboard[7][7] = {
                    {4,4,1,1,1,4,4},
                    {4,4,1,1,1,4,4},
                    {1,1,1,1,1,1,1},
                    {1,1,1,0,1,1,1},
                    {1,1,1,1,1,1,1},
                    {4,4,1,1,1,4,4},
                    {4,4,1,1,1,4,4} };
                //重新初始化棋盘
                for (int i = 0; i < 7; i++)
                {
                    for (int j = 0; j < 7; j++)
                    {
                        board[i][j] = refreshboard[i][j];
                    }
                }
                // 重新绘制棋盘和棋子
                cleardevice();
                drawChessBoard();
                drawChessPieces();
                drawButtons();
                drawTitle();
                drawCount();
            }
            //悔棋按钮
            else if (clickPos.first >= 800 && clickPos.first <= 900 && clickPos.second >= 250
&& clickPos.second <= 300)
            {
                undoMove();
            }
            //规则按钮
            else if (clickPos.first >= 800 && clickPos.first <= 900 && clickPos.second >= 350
&& clickPos.second <= 400)
            {
                rule();
            }
            //随机残局按钮
            else if (clickPos.first >= 800 && clickPos.first <= 900 && clickPos.second >= 450
&& clickPos.second <= 500)
```

```
{
    if (moregame % 5 == 0)
    {
        int refreshboard[7][7] = {
          {4,4,0,0,0,4,4},
          {4,4,0,0,0,4,4},
          {0,0,0,0,0,0,0},
          {0,0,0,0,0,0,0},
          {0,1,1,0,1,1,0},
          {4,4,1,1,1,4,4},
          {4,4,0,1,0,4,4} };
        //重新初始化棋盘
        for (int i = 0; i < 7; i++)
        {
            for (int j = 0; j < 7; j++)
            {
                board[i][j] = refreshboard[i][j];
            }
        }
    }
    else if (moregame % 5 == 1)
    {
        int refreshboard[7][7] = {
            {4,4,1,1,1,4,4},
            {4,4,0,1,1,4,4},
            {0,0,0,0,0,0,1},
            {0,0,1,1,1,1,1},
            {0,1,1,0,0,1,1},
            {4,4,1,0,1,4,4},
            {4,4,0,1,1,4,4} };
        //重新初始化棋盘
        for (int i = 0; i < 7; i++)
        {
            for (int j = 0; j < 7; j++)
            {
                board[i][j] = refreshboard[i][j];
            }
        }
     }
    else if (moregame % 5 == 2)
    {
        int refreshboard[7][7] = {
            {4,4,0,0,0,4,4},
            {4,4,0,0,0,4,4},
```

```
            {0,0,1,0,1,0,0},
            {0,0,1,1,1,0,0},
            {0,0,0,1,0,0,0},
            {4,4,1,1,1,4,4},
            {4,4,0,0,0,4,4} };
        //重新初始化棋盘
        for (int i = 0; i < 7; i++)
        {
            for (int j = 0; j < 7; j++)
            {
                board[i][j] = refreshboard[i][j];
            }
        }
    }
    else if (moregame % 5 == 3)
    {
        int refreshboard[7][7] = {
            {4,4,0,0,0,4,4},
            {4,4,0,0,0,4,4},
            {0,0,0,1,1,1,0},
            {0,0,0,1,1,1,0},
            {0,0,0,1,0,0,0},
            {4,4,0,1,0,4,4},
            {4,4,0,1,0,4,4} };
        //重新初始化棋盘
        for (int i = 0; i < 7; i++)
        {
            for (int j = 0; j < 7; j++)
            {
                board[i][j] = refreshboard[i][j];
            }
        }
    }
    else if (moregame % 5 == 4)
    {
        int refreshboard[7][7] = {
            {4,4,0,0,0,4,4},
            {4,4,0,0,0,4,4},
            {0,0,0,0,1,0,0},
            {0,0,1,1,1,1,0},
            {0,0,0,0,1,0,0},
            {4,4,0,0,0,4,4},
            {4,4,0,0,0,4,4} };
        //重新初始化棋盘
```

```
                for (int i = 0; i < 7; i++)
                {
                    for (int j = 0; j < 7; j++)
                    {
                        board[i][j] = refreshboard[i][j];
                    }
                }
            }
            moregame++;
            // 重新绘制棋盘和棋子
            cleardevice();
            drawChessBoard();
            drawChessPieces();
            drawButtons();
            drawTitle();
            drawCount();
        }
        //点击棋盘
        MOUSEMSG m = GetMouseMsg();
        int x = (m.y - 120) / 80; // 计算行坐标
        int y = (m.x - 120) / 80; // 计算列坐标
        if (x >= 0 && x < 7 && y >= 0 && y < 7) // 检查坐标是否在棋盘范围内
        {
            //选中1号棋子，变成2号棋子，其他棋子变回1号棋子，并且计算可移动位置变成3号棋子
            if (board[x][y] == 1)
            {
                //把其他的2号棋子变成1号棋子，3号棋子变成0号棋子
                for (int i = 0; i < 7; i++)
                {
                    for (int j = 0; j < 7; j++)
                    {
                        if (board[i][j] == 2)
                        {
                            board[i][j] = 1;
                        }
                        else if (board[i][j] == 3)
                        {
                            board[i][j] = 0;
                        }
                    }
                }
                selectChess(x, y);//改变选中的1号棋子为2号棋子，同时计算可移动位置
            }
```

```
                    //点选3号棋子，移动棋子，中间的棋子消失，同时去掉其他的3号棋子
                    else if (board[x][y] == 3)
                    {
                        for (int i = 0; i < 7; i++)
                        {
                            for (int j = 0; j < 7; j++)
                            {
                                if (board[i][j] == 2)
                                {
                                    moveChess(i, j, x, y);
                                }
                            }
                        }
                        //去掉其他的3号棋子
                        for (int i = 0; i < 7; i++)
                        {
                            for (int j = 0; j < 7; j++)
                            {
                                if (board[i][j] == 3)
                                {
                                    board[i][j] = 0;
                                }
                            }
                        }
                    }
                }
                // 重新绘制棋盘和棋子
                drawGameInterface();
                //判断成功失败
                if (judgeWin())
                {
                    MessageBox(GetHWnd(), _T("恭喜你，游戏胜利！"), _T("游戏结束"), MB_OK);
                    break;
                }
                if (judgeLose())
                {
                    MessageBox(GetHWnd(), _T("很遗憾，游戏失败！"), _T("游戏结束"), MB_OK);
                    break;
                }
            }
        }
        closegraph();// 关闭图形窗口
        return 0;
}
```