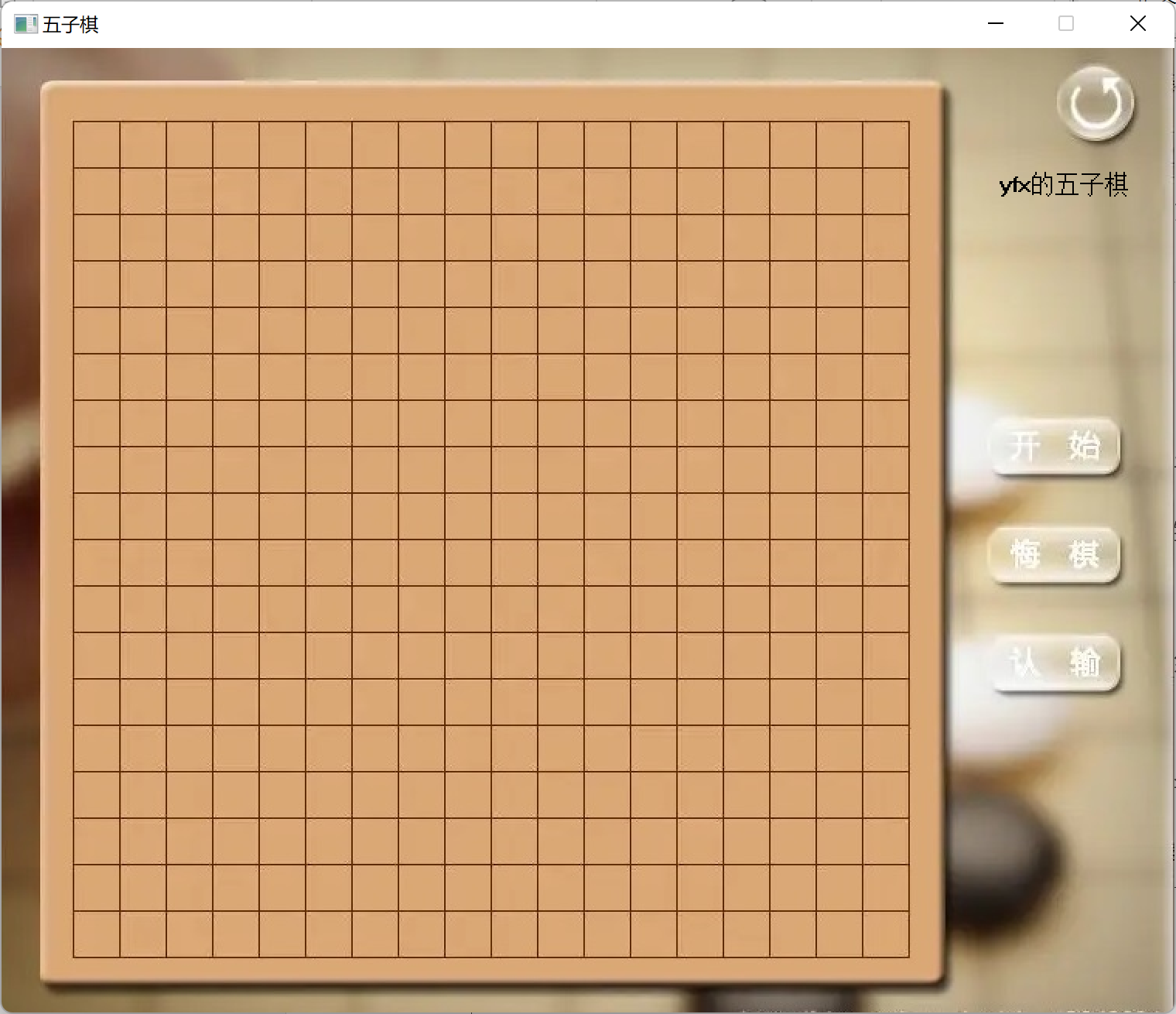
山东大学 计算机科学与技术 学院

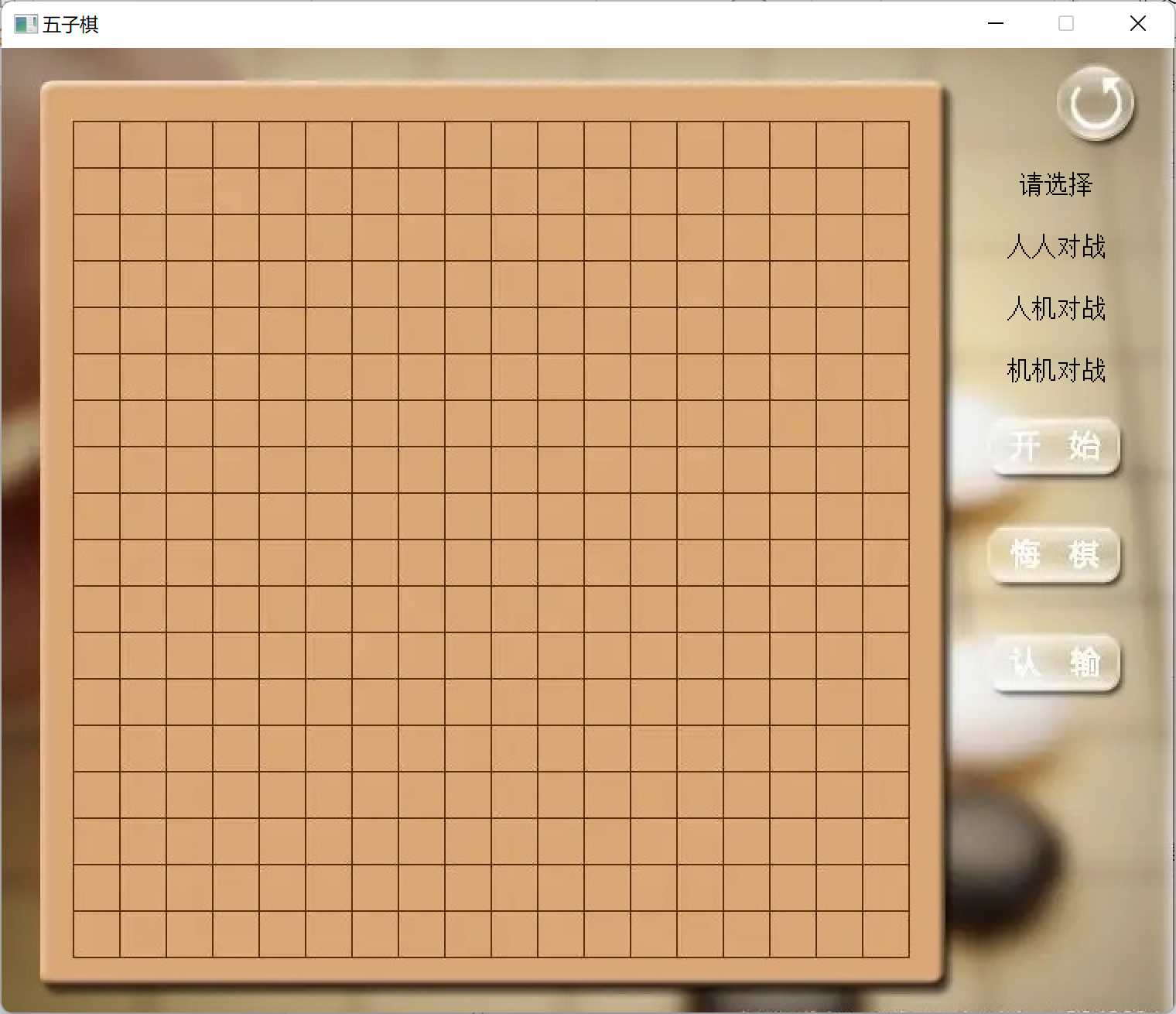
计算导论与程序设计 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202222130195 | 姓名： 阎发祥 | | 班级： 3班 |
| 实验题目：五子棋 | | | |
| 实验学时：64 | | 实验日期： 2022.11.28——2022.12.22 | |
| 实验目的：  1.实现一个五子棋游戏，训练对C语言综合知识以及相关算法的设计与实现能力  2.在实际应用中了解人工智能算法的实现  3.了解相关软件开发的知识，了解easyx的初步应用 | | | |
| 硬件环境：  （所使用的机器的型号、CPU型号等硬件信息）  联想thinkbook14+  AMD 6800H | | | |
| 软件环境：  （所使用的操作系统，编程环境IDE的名称，版本等软件信息）  Windows 11  Microsoft Visual Studio 2022 | | | |
| 实验内容与实验步骤：  （1）实验内容：  用c语言实现五子棋游戏，能够拥有简单游戏界面提示、下棋功能、人机对战功能、胜负判断功能、机机对战演示功能、对局复盘功能，通过鼠标进行游戏控制。  （2）实验步骤：  1.添加素材 :游戏开始出现初始界面，加载图片，设置字体大小和位置  2.添加棋盘：使用easyx库导入棋盘素材  3.添加棋子 ：先设置具体颜色和大小，再使用solidcircle函数直接绘制实心棋子  4.判断输赢 : 对落子位置进行判断，判断该位置在横、竖、左斜右斜4个方向上是否有5颗连续的同类棋子，使用数组进行判断，如果五子连珠，则根据下棋步骤判断谁胜，游戏结束。  5.判断和棋:遍历棋盘的每一个位置，如果每个位置都有棋子，则本局和棋  6.完善界面：制作游戏结束的弹窗，已经结束后能够实现重复对局的功能  7.认输判断：如果棋手单击认输键，则本局自动结束  8.保存游戏： 使用相应的文件操作，把每一步的横纵坐标存储在文件里，以便记录对局数据  9.复盘功能： 打开存储的文件，按照顺序读取文件中的横纵坐标，再完成落子操作，使得重新落子与上一局的位置完全一致  10.计算评分：先计算棋手如果在这个位置落子，会有多大的价值，然后再计算自己如果在这个位置落子，有多大价值。计算如果黑棋或者白棋在这个位置落子，那么在这个位置的某个方向上， 一共有连续几个黑子或者连续几个白子，根据连续数量的不同以及间隔点的不同来相应赋分。遍历棋盘后选择评分价值最大的点落子。  11.人机对战：具体操作与人人对战类似，先用鼠标控制人落子，然后开始计算评分，选择评分最高的点给机器落子。  12.机机对战：计算评分一次，机器落子一次，然后切换颜色，再计算评分落子，重复上述过程，直到能分出胜负或者和棋。 | | | |
| 结论分析与体会：  通过五子棋项目的实践，我对C语言系列操作有了更加深刻的理解和认识，锻炼了知识综合应用的能力，激发了我对于学习计算机的兴趣，编写机器的算法也为我开启了人工智能领域的大门。在实操过程中发现了许多不足之处，比如界面设置不美观，电脑棋手算力不足等情况，这也为以后知识储备丰富后进一步改进预留了空间。总体而言，五子棋代码的编写需要投入大量的精力，我深刻的体会到了付出就会有收获，这一定是一次难忘的经历。 | | | |

附录：程序代码、调试过程及测试用例（运行结果截图）

（1）初始界面

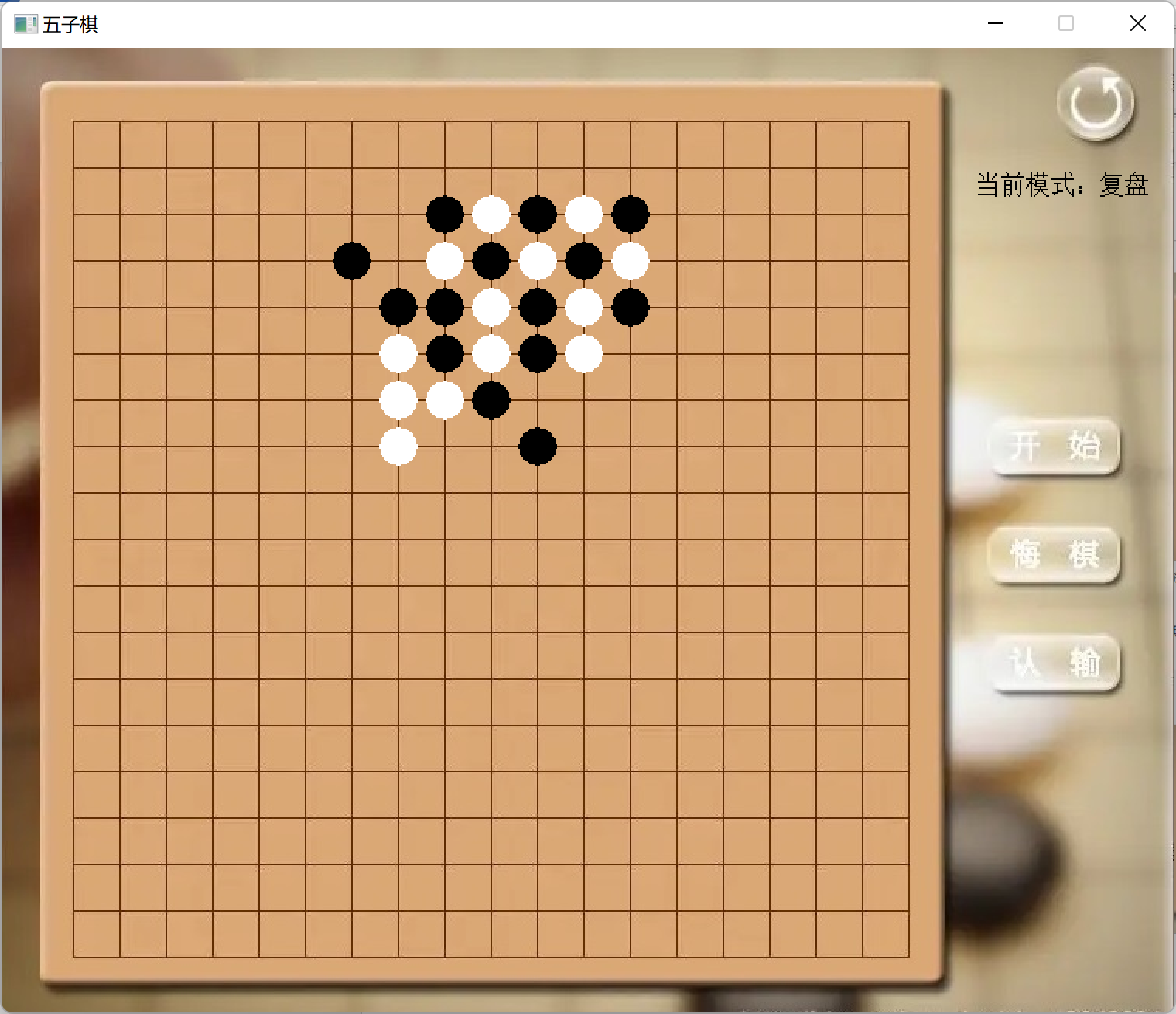
（2）菜单页



（3）胜负判断



（4）复盘游戏



（5）完整代码

#include <graphics.h> //图形库，在easyx下载

#include <windows.h>

#include<math.h>

#include<stdio.h>

int flag = 0;//下棋次数

int board[21][21] = { 0 };//0表示没有棋子

int sta[21][21] = { 0 };//存储数据

int score[21][21] = { 0 };//人机对战评分

int judge(int a, int b); //判断输赢

int isfull(int a, int b);//判断棋盘是否已满

void save\_chess(int sta[21][21], int a, int b); //保存对局数据

void replay\_chess(int flag); //复盘

void playpve();//人机对战

void playchess();//人人对战

void playeve();//机机对战

int a = 0, b = 0,x2 = 0,y2 = 0;

void init() //创建棋盘的窗口

{

initgraph(758, 623); //图片大小

loadimage(0, "素材/棋盘.jpg"); //加载图片,写地址

setbkmode(TRANSPARENT);//设置字体背景透明，默认不透明

settextcolor(BLACK);

outtextxy(645, 80, "yfx的五子棋");

}

void playpve()//人机对战

{

int x = 0, y = 0, a = 0, b = 0;

void calculate();

FILE\* fp;

HWND hwnd = GetHWnd();//用于提前显示窗口

loadimage(0, "素材/棋盘.jpg");

outtextxy(630, 80, "当前模式：PVE");

outtextxy(640, 120, "玩家1：黑棋");

outtextxy(640, 160, "电脑：白棋");

while (1)

{

MOUSEMSG msg = GetMouseMsg(); //获取鼠标信息并保存

for (int i = 1; i < 20; i++) //校正棋子位置

{

for (int j = 1; j < 20; j++)

{

if (abs(msg.x - 16 - i \* 30) < 12 && abs(msg.y - 17 - j \* 30) < 12)

{

x = i \* 30 + 16;

y = j \* 30 + 17;

a = i;

b = j;

}

}

}

if (msg.uMsg == WM\_LBUTTONDOWN) //左键按下

{

//判断是否认输

if (abs(msg.x - 680) < 30 && abs(msg.y - 395) < 15)

{

if (flag % 2 == 0)

{

MessageBox(hwnd, "玩家1认输，电脑胜利", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

}

else if (board[a][b] != 0)

{

MessageBox(hwnd, "该位置已有棋子，请重试！", "提示", MB\_OK);

continue;//退出循环重新进入

}

else if (flag % 2 == 0)

{

setfillcolor(BLACK); //黑白交替

solidcircle(x, y, 12);//在当前位置获取棋子

board[a][b] = 1; //1表示黑棋占用

sta[a][b] = flag + 1;//表示第n步下的棋子数据

flag++;

}

if (judge(a, b))

{

if (flag % 2 == 0)

{

MessageBox(hwnd, "电脑胜利", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

if (flag % 2 == 1)

{

MessageBox(hwnd, "玩家1胜利", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

}

if (isfull(a, b) == 1) //判断棋盘是否已满

{

MessageBox(hwnd, "本局和棋", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

//电脑下棋

setfillcolor(WHITE);

Sleep(500);

calculate();

a = x2, b = y2;

solidcircle(a \* 30 + 16, b \* 30 + 17, 12);//在当前位置获取棋子

board[a][b] = 2; //2表示白棋占用

sta[a][b] = flag + 1;

flag++;

}

if (judge(a, b))

{

if (flag % 2 == 0)

{

MessageBox(hwnd, "电脑胜利", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

if (flag % 2 == 1)

{

MessageBox(hwnd, "玩家1胜利", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

}

if (isfull(a, b) == 1) //判断棋盘是否已满

{

MessageBox(hwnd, "本局和棋", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

}

//游戏结束后清空棋盘，初始化数据

x = 0, y = 0, a = 0, b = 0;

for (int i = 1; i < 20; i++)

{

for (int j = 1; j < 20; j++)

{

board[i][j] = 0;

sta[i][j] = 0;

score[i][j] = 0;

}

}

clearcircle(200, 300, 700);//清空棋子

loadimage(0, "素材/棋盘.jpg");

setbkmode(TRANSPARENT);//设置字体背景透明，默认不透明

settextcolor(BLACK);

outtextxy(650, 200, "复盘游戏");

while (1)

{

MOUSEMSG m = GetMouseMsg(); //获取鼠标信息并保存

if (m.uMsg == WM\_LBUTTONDOWN)

{

if (abs(m.x - 680) < 40 && abs(m.y - 200) < 15)

{

replay\_chess(flag);//开始复盘

fp = fopen("Save\_chess.txt", "w");//删除数据

fclose(fp);

for (int i = 1; i < 20; i++)//初始化数据

{

for (int j = 1; j < 20; j++)

{

sta[i][j] = 0;

}

}

flag = 0;

break;

}

else

break;

}

}

flag = 0;

}

void calculate()

{

int personnum = 0;

int botnum = 0;

int emptynum = 0;

int m = 0;

for (int i = 1; i < 20; i++)

{

for (int j = 1; j < 20; j++)

{

score[i][j] = 0;

}

}

for (int row = 1; row < 20; row++)

{

for (int col = 1; col < 20; col++)

{

for (int x = -1; x <= 1; x++)

{ //Y的范围是-1,0,1

for (int y = -1; y <= 1; y++)

{ //X的范围是 -1,0,1

if (y == 0 && x == 0)

{

continue;

}

personnum = 0;

botnum = 0;

emptynum = 0;

//对黑棋评分(正向)

for (int i = 1; i <= 4; i++)

{

int curRow = row + i \* x;

int curCol = col + i \* y;

if (curRow >= 1 && curRow < 20 &&

curCol >= 1 && curCol < 20 && board[curRow][curCol] == 1)//该方向第t个棋子的坐标

{

personnum++; //遇到玩家的黑棋

}

else if (curRow >= 1 && curRow < 20 &&

curCol >= 1 && curCol < 20 && board[curRow][curCol] == 0)

{

emptynum++; //遇到空白停止搜索

break;

}

else

break; //出边界或者有白棋

}

//对黑棋评分（反向）

for (int i = 1; i <= 4; i++)

{

int curRow = row - i \* x;

int curCol = col - i \* y;

if (curRow >= 1 && curRow < 20&&

curCol >= 1 && curCol < 20&& board[curRow][curCol] == 1)//该方向第t个棋子的坐标

{

personnum++; //遇到玩家的黑棋

}

else if (curRow >= 1 && curRow < 20 &&

curCol >= 1 && curCol < 20 && board[curRow][curCol] == 0)

{

emptynum++; //遇到空白停止搜索

break;

}

else

break; //出边界或者有白棋

}

//开始打分

if (personnum == 1) //杀二

score[row][col] += 10;

else if (personnum == 2)

{ //杀三

if (emptynum == 1) //死三

score[row][col] += 30;

else if (emptynum == 2)//活三

score[row][col] += 40;

}

else if (personnum == 3)//杀四

{

if (emptynum == 1)//死四

score[row][col] += 60;

else if (emptynum == 2)//活四

score[row][col] += 200;

}

else if (personnum == 4)//杀五

score[row][col] += 20000;

//清空一次

emptynum = 0;

//对白棋评分(正向)

for (int i = 1; i <= 4; i++)

{

int curRow = row + i \* x;

int curCol = col + i \* y;

if (curRow >= 1 && curRow < 20 &&

curCol >= 1 && curCol < 20 && board[curRow][curCol] == 2)//该方向第t个棋子的坐标

{

botnum++; //遇到白棋

}

else if (curRow >= 1 && curRow < 20 &&

curCol >= 1 && curCol < 20 && board[curRow][curCol] == 0)

{

emptynum++; //遇到空白停止搜索

break;

}

else

break; //出边界

}

//对白棋评分（反向）

for (int i = 1; i <= 4; i++)

{

int curRow = row - i \* x;

int curCol = col - i \* y;

if (curRow >= 1 && curRow < 20 &&

curCol >= 1 && curCol < 20 && board[curRow][curCol] == 2)//该方向第t个棋子的坐标

{

botnum++; //遇到白棋

}

else if (curRow >= 1 && curRow < 20 &&

curCol >= 1 && curCol < 20 && board[curRow][curCol] == 0)

{

emptynum++; //遇到空白停止搜索

break;

}

else

break; //出边界

}

//开始打分

if (botnum == 0) //普通下棋

score[row][col] += 5;

else if (botnum == 1)//活二

score[row][col] += 10;

else if (botnum == 2)

{

if (emptynum == 1) //死三

score[row][col] += 25;

else if (emptynum == 2)//活三

score[row][col] += 50;

}

else if (botnum == 3)

{

if (emptynum == 1)//死四

score[row][col] += 55;

else if (emptynum == 2)//活四

score[row][col] += 1000;

}

else if (botnum == 4)

score[row][col] += 30000;//活五

if (board[row][col] != 1 && board[row][col]!=2)

{

if (m < score[row][col])

{

m = score[row][col];

x2 = row;

y2 = col;

}

}

else

break;

}

}

}

}

}

void playeve()

{

int x = 0, y = 0, a = 0, b = 0;

FILE\* fp;

HWND hwnd = GetHWnd();//用于提前显示窗口

loadimage(0, "素材/棋盘.jpg");

outtextxy(630, 80, "当前模式：EVE");

outtextxy(640, 120, "电脑1：黑棋");

outtextxy(640, 160, "电脑2：白棋");

while (1)

{

if (flag % 2 == 0)

{

setfillcolor(BLACK);

Sleep(600);

calculate();

a = x2, b = y2;

solidcircle(a \* 30 + 16, b \* 30 + 17, 12);//在当前位置获取棋子

board[a][b] = 1;

sta[a][b] = flag + 1;

flag++;

}

//判断输赢

if (judge(a, b))

{

if (flag % 2 == 0)

{

MessageBox(hwnd, "电脑2胜利", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

else if (flag % 2 == 1)

{

MessageBox(hwnd, "电脑1胜利", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

}

if (isfull(a, b) == 1) //判断棋盘是否已满

{

MessageBox(hwnd, "本局和棋", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

if (flag % 2 == 1)

{

setfillcolor(WHITE);

Sleep(600);

calculate();

a = x2, b = y2;

solidcircle(a \* 30 + 16, b \* 30 + 17, 12);//在当前位置获取棋子

board[a][b] = 2; //白棋占用

sta[a][b] = flag + 1;

flag++;

}

//判断输赢

if (judge(a, b))

{

if (flag % 2 == 0)

{

MessageBox(hwnd, "电脑2胜利", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

else if (flag % 2 == 1)

{

MessageBox(hwnd, "电脑1胜利", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

}

if (isfull(a, b) == 1) //判断棋盘是否已满

{

MessageBox(hwnd, "本局和棋", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

}

//游戏结束后清空棋盘，初始化数据

x = 0, y = 0, a = 0, b = 0;

for (int i = 1; i < 20; i++)

{

for (int j = 1; j < 20; j++)

{

board[i][j] = 0;

sta[i][j] = 0;

score[i][j] = 0;

}

}

clearcircle(200, 300, 700);//清空棋子

loadimage(0, "素材/棋盘.jpg");

setbkmode(TRANSPARENT);//设置字体背景透明，默认不透明

settextcolor(BLACK);

outtextxy(650, 200, "复盘游戏");

while (1)

{

MOUSEMSG m = GetMouseMsg(); //获取鼠标信息并保存

if (m.uMsg == WM\_LBUTTONDOWN)

{

if (abs(m.x - 680) < 40 && abs(m.y - 200) < 15)

{

replay\_chess(flag);//开始复盘

fp = fopen("Save\_chess.txt", "w");//删除数据

fclose(fp);

for (int i = 1; i < 20; i++)//初始化数据

{

for (int j = 1; j < 20; j++)

{

sta[i][j] = 0;

}

}

flag = 0;

break;

}

else

break;

}

}

flag = 0;

}

void playchess() //创建棋子开始人人对战

{

int x = 0, y = 0, a = 0, b = 0;

FILE\* fp;

HWND hwnd = GetHWnd();//用于提前显示窗口

loadimage(0, "素材/棋盘.jpg");

outtextxy(630, 80, "当前模式：PVP");

outtextxy(640, 120, "玩家1：黑棋");

outtextxy(640, 160, "玩家2：白棋");

while (1)

{

MOUSEMSG msg = GetMouseMsg(); //获取鼠标信息并保存

for (int i = 1; i < 20; i++) //校正棋子位置

{

for (int j = 1; j < 20; j++)

{

if (abs(msg.x - 16 - i \* 30) < 12 && abs(msg.y - 17 - j \* 30) < 12)

{

x = i \* 30 + 16;

y = j \* 30 + 17;

a = i;

b = j;

}

}

}

if (msg.uMsg == WM\_LBUTTONDOWN) //左键按下

{

//判断是否认输

if (abs(msg.x - 680) < 30 && abs(msg.y - 395) < 15)

{

if (flag % 2 == 0)

{

MessageBox(hwnd, "玩家1认输，玩家2胜利", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

if (flag % 2 == 1)

{

MessageBox(hwnd, "玩家2认输，玩家1胜利", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

}

else if (board[a][b] != 0)

{

MessageBox(hwnd, "该位置已有棋子，请重试！", "提示", MB\_OK);

continue;//退出循环重新进入

}

else if (flag % 2 == 0)

{

setfillcolor(BLACK); //黑白交替

solidcircle(x, y, 12);//在当前位置获取棋子

board[a][b] = 1; //1表示黑棋占用

sta[a][b] = flag + 1;//表示第n步下的棋子数据

}

else if (flag % 2 == 1)

{

setfillcolor(WHITE);

solidcircle(x, y, 12);

board[a][b] = 2; //2表示白棋占用

sta[a][b] = flag + 1;

}

flag++;

}

//判断输赢

if (judge(a, b))

{

if (flag % 2 == 0)

{

MessageBox(hwnd, "玩家2胜利", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

else if (flag % 2 == 1)

{

MessageBox(hwnd, "玩家1胜利", "游戏结束", MB\_OK);

save\_chess(sta, a, b);

break;

}

}

if (isfull(a, b) == 1) //判断棋盘是否已满

{

MessageBox(hwnd, "本局和棋", "游戏结束", MB\_OK);

break;

}

}

//游戏结束后清空棋盘，初始化数据

x = 0, y = 0, a = 0, b = 0;

for (int i = 1; i < 20; i++)

{

for (int j = 1; j < 20; j++)

{

board[i][j] = 0;

sta[i][j] = 0;

}

}

clearcircle(200, 300, 700);//清空棋子

loadimage(0, "素材/棋盘.jpg");

setbkmode(TRANSPARENT);//设置字体背景透明，默认不透明

settextcolor(BLACK);

outtextxy(650, 200, "复盘游戏");

while (1)

{

MOUSEMSG m = GetMouseMsg(); //获取鼠标信息并保存

if (m.uMsg == WM\_LBUTTONDOWN)

{

if (abs(m.x - 680) < 40 && abs(m.y - 200) < 15)

{

replay\_chess(flag);//开始复盘

fp = fopen("Save\_chess.txt", "w");//删除数据

fclose(fp);

for (int i = 1; i < 20; i++)//初始化数据

{

for (int j = 1; j < 20; j++)

{

sta[i][j] = 0;

}

}

flag = 0;

break;

}

else

break;

}

}

flag = 0;

}

//判断输赢

int judge(int a, int b)

{

int i, j;

int t = 2 - flag % 2; //1表示黑棋，2表示白棋

for (i = a - 4, j = b; i <= a; i++) //横判断

{

if (i >= 0 && i < 16 && t == board[i][j] && t == board[i + 1][j] && t == board[i + 2][j] && t == board[i + 3][j] && t == board[i + 4][j])

{

return 1;

}

}

for (i = a, j = b - 4; j <= b; j++) //竖判断

{

if (i >= 0 && i < 16 && t == board[i][j] && t == board[i][j + 1] && t == board[i][j + 2] && t == board[i][j + 3] && t == board[i][j + 4])

{

return 1;

}

}

for (i = a - 4, j = b - 4;i <= a ,j <= b; i++,j++) //斜判断

{

if (i >= 0 && i < 16 && t == board[i][j] && t == board[i + 1][j + 1] && t == board[i + 2][j + 2] && t == board[i + 3][j + 3] && t == board[i + 4][j + 4])

{

return 1;

}

}

for (i = a - 4, j = b + 4; i <= a , j >= 1; i++, j--) //斜判断

{

if (i >= 0 && i < 16 && t == board[i][j] && t == board[i + 1][j - 1] && t == board[i + 2][j - 2] && t == board[i + 3][j - 3] && t == board[i + 4][j - 4])

{

return 1;

}

}

}

int isfull(int a, int b) //判断棋盘是否已满

{

int i, j;

int ret = 1;

for (i = 1; i < 20; i++)

{

for (j = 1; j < 20; j++)

{

if (board[i][j] == 0) //遍历数组，当有一个位置为空，则棋盘不满

{

ret = 0;

break;

}

}

}

return ret;

}

void save\_chess(int sta[21][21], int a, int b) //保存对局数据

{

FILE\* fp;

fp = fopen("Save\_chess.txt", "w");

if ((fp = fopen("Save\_chess.txt", "w")) == NULL)

printf("存储失败");

else

{

for (int i = 1; i <= flag; i++)

{

for (int j = 1; j < 20; j++)

{

for (int k = 1; k < 20; k++)

{

if (sta[j][k] == i)

{

fprintf(fp, "%d %d ", j, k);

}

}

}

}

fclose(fp);

}

}

void replay\_chess(int flag)//复盘游戏

{

FILE\* fp;

loadimage(0, "素材/棋盘.jpg");

setbkmode(TRANSPARENT);//设置字体背景透明，默认不透明

settextcolor(BLACK);

outtextxy(630, 80, "当前模式：复盘");

fp = fopen("Save\_chess.txt", "r");

if (fp== NULL)

{

printf("复盘失败");

exit(1);

}

else

{

for (int a = 0; a < flag; a++)

{

int i = 0, j = 0;

fscanf(fp, "%d %d", &i, &j);

if (a % 2 == 0)

{

setfillcolor(BLACK);

solidcircle(i \* 30 + 16, j \* 30 + 17, 12);

}

else if (a % 2 == 1)

{

setfillcolor(WHITE);

solidcircle(i \* 30 + 16, j \* 30 + 17, 12);

}

Sleep(600);

}

fclose(fp);

}

}

int main()

{

init(); //初始化棋盘

FILE\* fp;

while (1)

{

flag = 0;

MOUSEMSG msg = GetMouseMsg(); //获取鼠标信息并保存

if (msg.uMsg == WM\_LBUTTONDOWN)

{

if (abs(msg.x - 680) < 40 && abs(msg.y - 260) < 15)//点开始

{

fp = fopen("Save\_chess.txt", "w");//删除数据

fclose(fp);

loadimage(0, "素材/棋盘.jpg");

outtextxy(658, 80, "请选择");

outtextxy(650, 120, "人人对战");

outtextxy(650, 160, "人机对战");

outtextxy(650, 200, "机机对战");

}

if (abs(msg.x - 680) < 40 && abs(msg.y - 120) < 15)

{

playchess();//创建棋子并开始人人对战

}

if (abs(msg.x - 680) < 40 && abs(msg.y - 160) < 15)

{

playpve();//人机对战

}

if (abs(msg.x - 680) < 40 && abs(msg.y - 200) < 15)

{

playeve();//机机对战

}

}

}

system("pause");

return 0;

}