Wuziqi.h:

//包含的头文件库

#include <graphics.h>

#include <conio.h>

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

//宏定义与全局变量声明

#define HIGH 800//UI界面的高度

#define WIDTH 1400//UI界面的宽度

#define ROW 15//棋盘行数

#define COL 15//棋盘列数

#define DEPTH 4

#define THEBLACK 1

#define THEWHITE -1

extern int board[ROW][COL] ;//棋盘数组 0代表空棋盘 1代表黑子 -1代表白子

extern int vision[ROW][COL];

extern int step;//记录步数

typedef struct {

int sp;

int\* data;

} Stack;

extern Stack\* stack;

//函数声明

int InitUI(void);//开始界面的UI

int GameUI(void);//游戏主界面的UI

void Game(int t);//选择游戏模式

void InitBoard();//初始化棋盘

int ButtonClick1();//进入游戏主界面

int ButtonClick2();//再来一局或者退出游戏

void count\_step(int x, int y, int color);//记录对战步数

void AI(int color);//AI落子

int Move(void);//玩家对下

int PlayerMove(int color);//人机对战时的人类玩家

void set(int row, int col);//落子功能的集成模块

void Judge(int row, int col);//判断胜利与平局（平局判定）

int Kill(int depth, int player, int color);//算杀

int check(int row, int col);//检查部分棋型

int thedefend(int row, int col, int color);//防御

int fobidden(int row, int col);//禁手

int Chess\_type(int row, int col, int direction);//棋型判断 辅助估值

int Opposite\_shore(int row, int col, int direction, int color);//辅助判断变种棋型

int SituationValuation(int row, int col, int color);//将棋型判断与估值分割成两个模块

int Settled\_Situation(int color);//全局估值

int min\_max\_dfs(int depth, int player, int color, int a, int b);//dfs

void Neighbor\_Location(int row, int col, int t);//位置减枝

int Compete\_Chess\_Repentance();//悔棋

Stack\* creatRepentanceStack(int size);//栈相关函数

void stackpush(Stack\* pStack, int data);

int stackpop(Stack\* pStack);

AI.cpp

#include"wuziqi.h"

//AI功能集成函数

void AI(int color) {

int score = -200, Max = -1e9;//初始化分数跟最大值

int x = 7, y = 7;//初始化落子位置

int flag1 = 0, flag2 = 0, flag3 = 0, flag4 = 0, flag5 = 0;

int attack = 0, defend = 0,kill = 0;

if (step == 0 && color== THEBLACK) {//先手第一步落子天元

board[x][y] = THEBLACK;

Neighbor\_Location(x, y, 1);

setfillcolor(RGB(00, 00, 00));

solidcircle((y + 1) \* 50, (x + 1) \* 50, 25);

//记录步数

count\_step(x, y, THEBLACK);

}

else {

for (int i = 0; i < ROW; i++) {

for (int j = 0; j < COL; j++) {

if (board[i][j] == 0 && vision[i][j] != 0 && flag1 == 0

&& ( (color==THEBLACK && fobidden(i, j) == 0 ) || color == THEWHITE) ) {

defend = thedefend(i, j, -color);

board[i][j] = color;

Neighbor\_Location(i, j, 1);

attack = check(i, j);

kill = Kill(2, 1, color);

if (attack == 1) {

x = i;

y = j;

flag1 = 1;

}//先看我方成五

if (flag1 == 0 && flag2 == 0 && defend == 1) {

x = i;

y = j;

flag2 = 1;

}//再看敌方下一步成五 若有 则阻断

if (flag1 == 0 && flag2 == 0 && flag3 == 0) {

if (attack == 7) {

x = i;

y = j;

flag3 = 1;

}//我方活四

}

if (flag1 == 0 && flag2 == 0 && flag3 == 0 && flag4 == 0) {

if (attack == 5 || attack == 6 || (attack == 4 && kill == 1) || defend == 2) {

x = i;

y = j;

flag4 = 1;

}//然后找我方含冲四算杀的点和敌方下一步成活四的点

}

if (flag1 == 0 && flag2 == 0 && flag3 == 0 && flag4 == 0 && flag5==0) {

if ((kill == 1 && attack == 2) || attack == 3) {

x = i;

y = j;

flag5 = 1;

}//然后找我方活三算杀的点

else {

score = min\_max\_dfs(2, 1, color, Max, 1e9);

if (score > Max) {

Max = score;

x = i;

y = j;

}

}//以上都无则执行深搜 寻找4层之内最优点

}

board[i][j] = 0;

Neighbor\_Location(i, j, -1);

}

}

}

set(x, y);

}

}

Chess\_type.cpp

#include"wuziqi.h"

//棋型判断函数

int Chess\_type(int row,int col,int direction) {//只实现棋型判断的功能

int num = 1;//用于记录连续的棋子数

int empty = 0;//用于记录空格数

int x, y;//记录坐标

int remote[2] = { 0 };//实现空格棋型判断

if (direction == 1) {

// 方向1 行分级

x = row; y = col;

while (true) {//往上

if (x == 0) {

break;

}

if (board[x - 1][y] == board[x][y]) {

num++;

x--;

}

else if (board[x - 1][y] == 0) {

empty++;

remote[0] = Opposite\_shore(x - 2, y, 1, board[row][col]);

break;

}

else {

break;

}

}

x = row; y = col;

while (true) {//往下

if (x == 14) {

break;

}

if (board[x + 1][y] == board[x][y]){

num++;

x++;

}

else if (board[x + 1][y] == 0){

empty++;

remote[1] = Opposite\_shore(x + 2, y, 2, board[row][col]);

break;

}

else{

break;

}

}

}

else if (direction == 2) {

//列分级

x = row; y = col;

while (true) {//往左

if (y == 0) {

break;

}

if (board[x][y - 1] == board[x][y]) {

num++;

y--;

}

else if (board[x][y - 1] == 0) {

empty++;

remote[0] = Opposite\_shore(x, y - 2, 3, board[row][col]);

break;

}

else{

break;

}

}

x = row; y = col;

while (true) {//往右

if (y == 14) {

break;

}

if (board[x][y + 1] == board[x][y]) {

num++;

y++;

}

else if (board[x][y + 1] == 0) {

empty++;

remote[1] = Opposite\_shore(x, y + 2, 4, board[row][col]);

break;

}

else{

break;

}

}

}

else if (direction == 3) {

//左斜线分级

x = row; y = col;

while (true) {//左上

if (x == 0 || y == 0) {

break;

}

if (board[x - 1][y - 1] == board[x][y]) {

num++;

x--;

y--;

}

else if (board[x - 1][y - 1] == 0) {

empty++;

remote[0] = Opposite\_shore(x - 2, y - 2, 5, board[row][col]);

break;

}

else{

break;

}

}

x = row; y = col;

while (true) {//右下

if (x == 14 || y == 14) {

break;

}

if (board[x + 1][y + 1] == board[x][y]) {

num++;

x++;

y++;

}

else if (board[x + 1][y + 1] == 0) {

empty++;

remote[1] = Opposite\_shore(x + 2, y + 2, 6, board[row][col]);

break;

}

else{

break;

}

}

}

else if (direction == 4) {

//右斜线分级

x = row; y = col;

while (true) {//右上

if (x == 0 || y == 14) {

break;

}

if (board[x - 1][y + 1] == board[x][y]) {

num++;

x--;

y++;

}

else if (board[x - 1][y + 1] == 0) {

empty++;

remote[0] = Opposite\_shore(x - 2, y + 2, 7, board[row][col]);

break;

}

else {

break;

}

}

x = row; y = col;

while (true) {//左下

if (x == 14 || y == 0) {

break;

}

if (board[x + 1][y - 1] == board[x][y]) {

num++;

x++;

y--;

}

else if (board[x + 1][y - 1] == 0) {

empty++;

remote[1] = Opposite\_shore(x + 2, y - 2, 8, board[row][col]);

break;

}

else {

break;

}

}

}

//棋型 //remote数组里每个位置上 1表示同色棋子 2表示空格 0表示异色棋子

if (empty == 0 && num == 1) {//死一

return 1;

}

if (num == 1 && empty == 1 && (remote[0] == 222 || remote[1] == 222)) {

return 2;

}//眠一

if (num == 1 && empty == 2 && (remote[0] / 10 == 22 || remote[1] / 10 == 22)) {

return 3;

}//活一

if (empty == 0 && num == 2) {

return 4;

}//死二

if (empty == 1 && num == 2 && (remote[0] == 0 || remote[1] == 0)) {

return 4;

}//一型死二变种 ★ ■ ■ □ ★

if (empty == 1 && num == 2 && (remote[0] == 20 || remote[1] == 20)) {

return 4;

}//二型死二变种 ★ ■ ■ □ □ ★

if (empty == 1 && num == 2 && (remote[0] / 10 == 22 || remote[1] / 10 == 22)) {

return 5;

}//眠二 ★ ■ ■ □ □ □ 左图左右子

if (empty == 1 && num == 1 && (remote[0] == 122 || remote[1] == 122)) {

return 5;

}//一型眠二变种 ★ ■ □ ■ □ □ 左图左子

if (empty == 2 && num == 1 && ((remote[0] / 10 == 10 && remote[1] / 100 == 2) || (remote[1] / 10 == 10 && remote[0] / 100 == 2))) {

return 5;

}//一型眠二变种 ★ ■ □ ■ □ □ 左图右子

if (empty == 1 && num == 1 && ((remote[0] / 100 == 0 && remote[1] / 10 == 212) || (remote[0] / 10 == 10 && remote[1] / 10 == 22))) {

return 5;

}//二型眠二变种 ★ ■ □ □ ■ □ 左图左子

if (empty == 2 && num == 1 && (remote[0] == 210 || remote[1] == 210)) {

return 5;

}//二型眠二变种 ★ ■ □ □ ■ □ 左图右子

if (empty != 0 && num == 1 && (remote[0] == 221 || remote[1] == 221)) {

return 5;

}//三型眠二变种 ■ □ □ □ ■ 左图左右子皆可

if (empty == 2 && num == 2 && (remote[0] / 100 == 2 && remote[1] / 100 == 2)) {

return 6;

}//活二 □ □ ■ ■ □ □ 左右子

if (empty == 2 && num == 1 && (remote[0] / 10 == 12 && remote[1] / 10 == 12)) {

return 6;

}//一型活二变种 □ ■ □ ■ □ 左右子

if (empty == 2 && num == 1 && (remote[0] == 212 || remote[1] == 212)) {

return 6;

}//二型活二变种 □ ■ □ □ ■ □ 左右子

if (empty == 0 && num == 3 ) {

return 7;

}//死三

if (empty == 1 && num == 3 && (remote[0] == 0 || remote[1] == 0)) {

return 7;

}//一型死三变种

if (empty == 1 && num == 3 && (remote[0] / 100 == 2 || remote[1] / 100 == 2)) {

return 8;

}//眠三 ★ ■ ■ ■ □ □ 左图左右子

if (empty == 1 && num == 2 && (remote[0] / 10 == 12 || remote[1] / 10== 12)) {

return 8;

}//一型眠三变种 ★ ■ ■ □ ■ □ 左图左子域

if (empty == 2 && num == 1 && (remote[0] == 110 || remote[1] == 110)) {

return 8;

}//一型眠三变种 ★ ■ ■ □ ■ □ 左图右子

if (empty == 1 && num == 1 && (remote[0] == 110 || remote[1] == 110)) {

return 8;

}//二型眠三变种 ★ ■ □ ■ ■ □ 左图左子

if (empty == 2 && num == 2 && (remote[0] / 10 == 10 || remote[1] / 10== 10)) {

return 8;

}//二型眠三变种 ★ ■ □ ■ ■ □ 左图右子域

if (empty != 0 && num == 2 && (remote[0] / 10 == 21 || remote[1] / 10== 21)) {

return 8;

}//三型眠三变种 ■ ■ □ □ ■ 左图左子域

if (empty != 0 && num == 2 && (remote[0] == 211 || remote[1] == 211)) {

return 8;

}//三型眠三变种 ■ ■ □ □ ■ 左图右子

if (empty != 0 && num == 1 && (remote[0] == 121 || remote[1] == 121)) {

return 8;

}//四型眠三变种 ■ □ ■ □ ■ 左图左右子

if (empty == 2 && num == 1 && (remote[0] / 100 == 1 || remote[1] / 100== 1)) {

return 8;

}//四型眠三变种 ■ □ ■ □ ■ 左图中子

if (empty == 2 && num == 3 && (remote[0] / 100 == 0 || remote[1] / 100 == 0)) {

return 8;

}//五型眠三变种 ★ □ ■ ■ ■ □ 左图中子

if (empty == 1 && num == 3 && (remote[0] / 10 == 20 || remote[1] / 10== 20)) {

return 8;

}//六型眠三变种 ★ ■ ■ ■ □ □ ★左图中子

if (empty == 2 && num == 3) {

return 9;

}//活三

if (empty == 2 && num == 2 && (remote[0]/10 == 12 || remote[1]/10 == 12)) {

return 9;

}//一型嵌四（活三）双子处 □ ■ ■ □ ■ □

if (empty == 2 && num == 1 && (remote[0] == 112 || remote[1] == 112)) {

return 9;

}//一型嵌四（活三）单子处

if (empty == 0 && num == 4) {

return 10;

}//死四

if (empty == 1 &&num == 4 ) {

return 11;

}//冲四

if (empty != 0 && num == 3 && (remote[0] / 100 == 1 || remote[1] / 100 == 1)) {

return 11;

}//一型冲四变种 ■ ■ ■ □ ■ 三子处

if (empty != 0 && num == 1 && (remote[0] == 111 || remote[1] == 111)) {

return 11;

}//一型冲四变种 ■ ■ ■ □ ■ 一子处

if (empty != 0 && num == 2 && (remote[0]/ 10 == 11 || remote[1] / 10 == 11)) {

return 11;

}//二型冲四变种 ■ ■ □ ■ ■ 左右二子处

if (empty == 2 && num == 4) {

return 12;

}//活四

if (num == 5) {

return 13;

}//成五

//其他必胜棋型

if (empty == 2 && num == 2 && remote[0] / 10 == 11 && remote[1] / 10 == 11) {

return 13;//■ ■ □ ■ ■ □ ■ ■ 型

}

if (empty == 2 && num == 3 && remote[0] / 100 == 1 && remote[1] / 100 == 1) {

return 13;//■ □ ■ ■ ■ □ ■ 型

}

if (num > 5) {

return 14;//长连

}

return 1;

}

Opposite\_shore.cpp

#include"wuziqi.h"

//变种棋型判断辅助函数

//为了实现棋型的判断 比如中间空一个的冲四 活三之类的棋型

//所以在一个方向上找到空位置之后不是停止搜索而是进一步往前找

//往这个方向搜索2步

int Opposite\_shore(int row,int col,int direction,int color) {

int go = 0;//空格对面的步数

int num = 0;//记录空格对面棋型

const int Maxgo=3;

//用1表示同色棋子 2表示空 0表示异色棋子或者是棋盘壁

switch (direction) {

case 1://往上

while (row >= 0 && go < Maxgo) {

if (board[row][col] == color) {

num = num \* 10 + 1;

row--;

}

else if (board[row][col] == 0) {

num = num \* 10 + 2;

row--;

}

else {

break;

}

go++;

}

break;

case 2://往下

while (row <=14 && go < Maxgo) {

if (board[row][col] == color) {

num = num \* 10 + 1;

row++;

}

else if (board[row][col] == 0) {

num = num \* 10 + 2;

row++;

}

else {

break;

}

go++;

}

break;

case 3://往左

while (col >= 0 && go < Maxgo) {

if (board[row][col] == color) {

num = num \* 10 + 1;

col--;

}

else if (board[row][col] == 0) {

num = num \* 10 + 2;

col--;

}

else {

break;

}

go++;

}

break;

case 4://往右

while (col <=14 && go < Maxgo) {

if (board[row][col] == color) {

num = num \* 10 + 1;

col++;

}

else if (board[row][col] == 0) {

num = num \* 10 + 2;

col++;

}

else {

break;

}

go++;

}

break;

case 5://左上

while (row >= 0 && col >= 0 && go < Maxgo) {

if (board[row][col] == color) {

num = num \* 10 + 1;

row--;

col--;

}

else if (board[row][col] == 0) {

num = num \* 10 + 2;

row--;

col--;

}

else {

break;

}

go++;

}

break;

case 6://右下

while (row <= 14 && col <= 14 && go < Maxgo) {

if (board[row][col] == color) {

num = num \* 10 + 1;

row++;

col++;

}

else if (board[row][col] == 0) {

num = num \* 10 + 2;

row++;

col++;

}

else {

break;

}

go++;

}

break;

case 7://右上

while (row >= 0 && col <= 14 && go < Maxgo) {

if (board[row][col] == color) {

num = num \* 10 + 1;

row--;

col++;

}

else if (board[row][col] == 0) {

num = num \* 10 + 2;

row--;

col++;

}

else {

break;

}

go++;

}

break;

case 8://左下

while (row <= 14 && col >= 0 && go < Maxgo) {

if (board[row][col] == color) {

num = num \* 10 + 1;

row++;

col--;

}

else if (board[row][col] == 0) {

num = num \* 10 + 2;

row++;

col--;

}

else {

break;

}

go++;

}

break;

}

if (num < 100 && num > 0) {

num = num \* 10;

}

return num;

}

Game.cpp

#include"wuziqi.h"

//初始化游戏主界面

void Game(int t) {

InitBoard();

GameUI();

while (1) {

if (t == 1) {

PlayerMove(1);

AI(THEWHITE);

}

else if (t == 2) {

Move();//玩家对战

}

else if (t == 3) {

AI(THEBLACK);

PlayerMove(-1);

}

else {//电脑对战

AI(THEBLACK);

AI(THEWHITE);

}

}

getchar();

}

InitBoard.cpp

#include"wuziqi.h"

//初始化棋盘

void InitBoard() {

int i, j;

for (i = 0; i < ROW; i++) {

for (j = 0; j < ROW; j++) {

board[i][j] = 0;//0代表空棋盘 1代表黑子 -1代表白子

vision[i][j] = 0;

}

}

}

main.cpp

#include"wuziqi.h"

//定义棋盘为全局变量

int board[ROW][COL] = { 0 };

int vision[ROW][COL] = { 0 };

int step = 0;

int main() {

InitUI();

return 0;

}

stack.cpp

#include"wuziqi.h"

//栈的创建 Push Pop

int stackpop(Stack\* pStack) {

return pStack->data[--pStack->sp];

}

void stackpush(Stack\* pStack, int data) {

pStack->data[pStack->sp++] = data;

}

Stack\* creatRepentanceStack(int size) {

Stack\* pStack = (Stack\*)malloc(sizeof(Stack));//声明并初始化；

pStack->data = (int\*)malloc(sizeof(int) \* size);

pStack->sp = 0;

return pStack;

}

Stack\* stack = creatRepentanceStack(510);

Chess\_Repentance.cpp

#include"wuziqi.h"

//悔棋

int Compete\_Chess\_Repentance() {

int x, y;

if (step == 0) {

return 0;

}

setfillcolor(RGB(255, 153, 00));

setcolor(BLACK);

y = stackpop(stack);

x = stackpop(stack);

board[x][y] = 0;

Neighbor\_Location(x, y, -1);

solidcircle((y + 1) \* 50, (x + 1) \* 50, 25);

line((y + 1) \* 50 - 25, (x + 1) \* 50, (y + 1) \* 50 + 25, (x + 1) \* 50);

line((y + 1) \* 50, (x + 1) \* 50 - 25, (y + 1) \* 50, (x + 1) \* 50 + 25);

step--;

return 0;

}

check.cpp

#include"wuziqi.h"

//棋型判断函数

int check(int row, int col) {

int rank = 0,flag = 0;

int ko = 0, ko1 = 0, ko2 = 0, ko3 = 0;

for (int i = 0; i < 4; i++) {

rank = Chess\_type(row, col, i + 1);

if (rank >= 13) {

ko = 1;

break;

}

else if (rank == 9) {//活三

ko1++;

}

else if (rank == 11) {//冲四

ko2++;

}

else if (rank == 12) {//活四

ko3 = 1;

}

}

if (ko1 == 1 && ko2 == 0) {

flag = 2;//单活三

}

else if (ko1 >= 2 && ko2 == 0) {

flag = 3;//双活三

}

else if (ko1 == 0 && ko2 == 1) {

flag = 4;//单冲四

}

else if (ko1 == 0 && ko2 >= 2) {

flag = 5;//双冲四

}

else if (ko1 >= 1 && ko2 >= 1) {

flag = 6;//一活三，一冲四

}

if (ko3 == 1) {

flag = 7;//活四

}

if (ko == 1) {

flag = 1;//成五返回1

}

return flag;

}

defend.cpp

#include"wuziqi.h"

//防守函数

int thedefend(int row, int col, int color) {

int rank = 0, flag = 0;

int ko = 0, ko1 = 0;;

board[row][col] = color;

for (int i = 0; i < 4; i++) {

rank = Chess\_type(row, col, i + 1);

if (rank >= 13) {//成五

ko = 1;

break;

}

else if (rank == 12) {//活四

ko1 = 1;

}

}

if (ko == 1) {

flag = 1;

}

else if (ko1 == 1) {

flag = 2;

}

board[row][col] = 0;

return flag;//在对手下一步成五返回1 在对手下一步活四返回2 下一步无成五活四返回0

}

forbidden.cpp

#include"wuziqi.h"

//禁手函数

int fobidden(int row, int col) {

int rank = 0,flag = 0;

int ko1 = 0,ko2 = 0,ko3 = 0;

board[row][col] = THEBLACK;

for (int i = 0; i < 4; i++) {

rank =Chess\_type(row, col, i + 1);

if (rank == 14) {

flag = 1;

}

else if (rank == 9) {

ko1++;//用于记录活三数量

}

else if (rank == 11) {

ko2++;//用于记录冲四数量

}

else if (rank == 12) {

ko3++;//用于记录活四数量

}

}

if (ko1 == 2 || ko2 == 2 || ko3 == 2) {

flag = 1;

}

board[row][col] = 0;

return flag;//触发禁手返回1 否则返回0

}

Judge.cpp

#include"wuziqi.h"

//输赢判断函数

void Judge(int row, int col) {//判断胜利与平局（平局判定）

int flag = 0;

flag = check(row, col);

settextstyle(40, 20, L"楷体");

settextcolor(BLACK);

if (step == 255) {

flag = 1;

outtextxy(950, 250, L"双方平局");

}

if (flag == 1 && board[row][col] == THEBLACK) {

outtextxy(950, 250, L"黑子获胜");

}

else if (flag == 1 && board[row][col] == THEWHITE) {

outtextxy(950, 250, L"白子获胜");

}

if (flag == 1) {

step = 0;

for (int i = 0; i < ROW; i++) {

for (int j = 0; j < COL; j++) {

vision[i][j] = 0;

}

}

ButtonClick2();

return;

}

return;

}

kill.cpp

#include"wuziqi.h"

//算杀函数

int Kill(int depth ,int player,int color) {

//player=1表示接下来是对手下棋来进入下一层 player=-1表示自己下棋来进入下一层

int key = 0; //返回值

int v = 0;

if (depth == DEPTH + 7) {

return 0;

}//算杀失败返回0

for (int i = 0; i < ROW; i++) {

for (int j = 0; j < COL; j++) {

if (board[i][j] == 0 && vision[i][j] != 0) {

if (player == -1 && key == 0) {

board[i][j] = color;

Neighbor\_Location(i, j, 1);

v = check(i, j);

if (v == 2 || v==4 ) {

key = Kill(depth + 1, 1, color);

}//算杀半成功

else if (v == 1 || v == 3 || v == 5 || v == 6) {

key = 1;

}//算杀成功;

board[i][j] = 0;

Neighbor\_Location(i, j, -1);

}

else if(player == 1 && thedefend(i, j, color) != 0){

board[i][j] = -color;

Neighbor\_Location(i, j, 1);

key = Kill(depth + 1, -1, color);

board[i][j] = 0;

Neighbor\_Location(i, j, -1);

}

}

}

}

return key;

}

set.cpp

#include"wuziqi.h"

//落子函数

void set(int row ,int col) {

int color = -1, t = 0;

if (step % 2 == 1) {

color = 255;

t = -1;

}

else if (step % 2 == 0) {

color = 0;

t = 1;

}

board[row][col] = t;

Neighbor\_Location(row, col, 1);

setfillcolor(RGB(color, color, color));

solidcircle((col + 1) \* 50, (row + 1) \* 50, 25);

count\_step(row, col, t);//记录步数

Judge(row, col);//判断胜负

}

Move.cpp

#include"wuziqi.h"

//双人对战移动

int Move(void){

int row, col;

TCHAR stepch[10];

while (true){

ExMessage msg = { 0 };

if (peekmessage(&msg, EM\_MOUSE)){//检测鼠标信息

switch (msg.message){//筛选鼠标信息类型

case WM\_LBUTTONDOWN://左键信息 落黑子

if ( board[(msg.y + 25) / 50 - 1][(msg.x + 25) / 50 - 1] == 0 && ((msg.y + 25) / 50 - 1 >= 0 && (msg.y + 25) / 50 - 1 <= 14) && ((msg.x + 25) / 50 - 1 >= 0 && (msg.x + 25) / 50 - 1 <= 14)) {

//注意x y与row col的关系 y对应row x对应col

row = (msg.y + 25) / 50 - 1;

col = (msg.x + 25) / 50 - 1;

if (step % 2==0 && fobidden(row, col) == 1) {

break;

}

else {

set(row, col);

}

}

else if (msg.x >= 950 && msg.x <= 1050 && msg.y >= 300 && msg.y <= 380) {

Compete\_Chess\_Repentance();

}

break;

default:

break;

}

}

}

return 0;

}

PlayerMove.cpp

#include"wuziqi.h"

//人机对战玩家移动

int PlayerMove(int color)

{

int f1 = 1;

int row, col;

TCHAR stepch[10];

//玩家移动

while (f1)

{

ExMessage msg = { 0 };

if (peekmessage(&msg, EM\_MOUSE)){//检测鼠标信息

switch (msg.message){//筛选鼠标信息类型

case WM\_LBUTTONDOWN://左键信息

if (board[(msg.y + 25) / 50 - 1][(msg.x + 25) / 50 - 1] == 0 && ((msg.y + 25) / 50 - 1 >= 0 && (msg.y + 25) / 50 - 1 <= 14) && ((msg.x + 25) / 50 - 1 >= 0 && (msg.x + 25) / 50 - 1 <= 14)) {

f1 = 0;

row = (msg.y + 25) / 50 - 1;//注意x y与row col的关系 y对应row x对应col

col = (msg.x + 25) / 50 - 1;

if (color == THEBLACK && fobidden(row, col) == 1) {

f1 = 1;

}

else{

set(row, col);

}

}

else if (msg.x >= 950 && msg.x <= 1050 && msg.y >= 300 && msg.y <= 380) {

Compete\_Chess\_Repentance();

Compete\_Chess\_Repentance();

}

break;

default:

break;

}

}

}

return 0;

}

ButtonClick.cpp

#include"wuziqi.h"

//进入游戏主界面

int ButtonClick1() {

while (true) {

ExMessage msg = { 0 };

if (peekmessage(&msg, EM\_MOUSE));//检测鼠标信息

{

switch (msg.message)//筛选鼠标信息类型

{

case WM\_LBUTTONDOWN://左键信息

closegraph();

if (msg.x >= 250 && msg.x <= 650 && msg.y >= 350 && msg.y <= 450) {

Game(1);

}

if (msg.x >= 720 && msg.x <= 1120 && msg.y >= 350 && msg.y <= 450) {

Game(2);

}

if (msg.x >= 250 && msg.x <= 650 && msg.y >= 500 && msg.y <= 600) {

Game(3);

}

if (msg.x >= 720 && msg.x <= 1120 && msg.y >= 500 && msg.y <= 600) {

Game(4);

}

break;

default:

break;

}

}

}

return 0;

}

//再来一局或者退出游戏

int ButtonClick2() {

settextstyle(40, 20, L"楷体");

setbkmode(TRANSPARENT);

setfillcolor(RGB(255, 255, 255));

setcolor(BLACK);

fillrectangle(950, 300, 1130, 380);

outtextxy(960, 320, L"人机黑子");

fillrectangle(950, 400, 1130, 480);

outtextxy(960, 420, L"人机白子");

fillrectangle(950, 500, 1130, 580);

outtextxy(960, 520, L"双人对战");

fillrectangle(950, 600, 1130, 680);

outtextxy(955, 620, L"机器对下");

fillrectangle(950, 700, 1155, 780);

outtextxy(955, 720, L"返回主界面");

while(true){

ExMessage msg = { 0 };

if (peekmessage(&msg, EM\_MOUSE));//检测鼠标信息

{

switch (msg.message)//筛选鼠标信息类型

{

case WM\_LBUTTONDOWN://左键信息

closegraph();

if (msg.x >= 950 && msg.x <= 1130 && msg.y >= 300 && msg.y <= 380) {

Game(1);

}

else if (msg.x >= 950 && msg.x <= 1130 && msg.y >= 400 && msg.y <= 480) {

Game(3);

}

else if (msg.x >= 950 && msg.x <= 1130 && msg.y >= 500 && msg.y <= 580) {

Game(2);

}

else if (msg.x >= 950 && msg.x <= 1130 && msg.y >= 600 && msg.y <= 680) {

Game(4);

}

else if (msg.x >= 950 && msg.x <= 1155 && msg.y >= 700 && msg.y <= 780) {

InitUI();

}

break;

default:

break;

}

}

}

return 0;

}

countstep.cpp

#include"wuziqi.h"

//记录并绘制步数

void count\_step(int x, int y,int color) {

TCHAR stepch[10];

step++;

stackpush(stack, x);

stackpush(stack, y);

if (step%2 == 1) {

settextcolor(RGB(255, 255, 255));

}

else{

settextcolor(RGB(0, 0, 0));

}

\_stprintf\_s(stepch, \_T("%d"), step);

if (step >= 100){

LOGFONT f;

gettextstyle(&f);

f.lfHeight = 13;

f.lfQuality = ANTIALIASED\_QUALITY;

settextstyle(&f);

setbkmode(TRANSPARENT);

outtextxy((y + 1) \* 50 - 30, (x + 1) \* 50 - 6, stepch);

}

else if (step >= 10){

LOGFONT f;

gettextstyle(&f);

f.lfHeight = 20;

f.lfQuality = ANTIALIASED\_QUALITY;

settextstyle(&f);

setbkmode(TRANSPARENT);

outtextxy((y + 1) \* 50 - 20, (x + 1) \* 50 - 11, stepch);

}

else{

outtextxy((y + 1) \* 50 - 9, (x + 1) \* 50 - 20, stepch);

}

}

UI.cpp

#include"wuziqi.h"

//对战界面UI

int GameUI(void)

{

initgraph(WIDTH, HIGH, EX\_DBLCLKS);

setfillcolor(RGB(255, 82, 178));

solidrectangle(0, 0, WIDTH, HIGH);

setfillcolor(RGB(255, 153, 00));

solidrectangle(25, 25, 775, 775);//棋盘的图形区块

setfillcolor(RGB(255, 255, 255));

fillrectangle(950, 300, 1050, 380);//悔棋的图形区块

setcolor(BLACK);

for (int i = 50; i <= 775; i += 50){

line(50, i, 750, i);//画15根横线

}

for (int i = 50; i <= 775; i += 50){

line(i, 50, i, 750);//画15根竖线

}

settextstyle(40, 20, L"楷体");

setbkmode(TRANSPARENT);

outtextxy(950, 50, L"五子棋5.0");

outtextxy(960, 320, L"悔棋");

return 0;

}

//初始界面UI

int InitUI(void) {

initgraph(WIDTH, HIGH, EX\_DBLCLKS);

setfillcolor(RGB(33, 82, 178));

solidrectangle(0, 0, WIDTH, HIGH);

setfillcolor(RGB(255, 255, 255));

fillrectangle(250, 350, 650, 450);

fillrectangle(720, 350, 1120, 450);

fillrectangle(250, 500, 650, 600);

fillrectangle(720, 500, 1120, 600);

settextstyle(80, 40, L"楷体");

setbkmode(TRANSPARENT);

outtextxy(250, 200, L"欢迎来到人工智能五子棋");

settextstyle(100, 50, L"楷体");

setbkmode(TRANSPARENT);

setcolor(BLACK);

outtextxy(255, 350, L"人机黑子");

outtextxy(725, 350, L"双人对战");

outtextxy(255, 500, L"人机白子");

outtextxy(725, 500, L"机器对下");

ButtonClick1();

getchar();

return 0;

}

Location\_pruning.cpp

#include"wuziqi.h"

//减枝

void Neighbor\_Location(int row, int col,int t) {//t=1是插入；t=-1是删去

for (int i = row - 2; i <= row + 2; i++) {

for (int j = col - 2; j <= col + 2; j++) {

if (i >= 0 && i <= 14 && j >= 0 && j <= 14 &&(i != row || j != col)) {

if (t == -1) {

vision[i][j]--;

}

else {

vision[i][j]++;

}

}

}

}

}

min\_max\_dfs.cpp

#include"wuziqi.h"

//depth表示层数 color表示AI的颜色 a是下界 b是上界

int min\_max\_dfs(int depth, int player,int color, int a, int b){

//player=1表示接下来是对手下棋来进入下一层 player=-1表示自己下棋来进入下一层

int key; //返回值

int v;

if (depth == DEPTH + 1){

return Settled\_Situation(color);

} //到到最深层时 即叶子节点时 估值

if (player==1) key = b;

else key = a;

for (int i = 0; i < ROW; i++){

for (int j = 0; j < COL; j++){

if (board[i][j] == 0 && vision[i][j] != 0){

if (player == -1){

board[i][j] = color;

Neighbor\_Location(i, j, 1);

v = min\_max\_dfs(depth + 1, 1,color, a, b);

if (v > a){

a = v;

if (a >= b){

board[i][j] = 0;

Neighbor\_Location(i, j, -1);

return 1e9;

}

key = v; //返回更新的下界值 亦是所有子节点里最大的下界值

}

board[i][j] = 0;

Neighbor\_Location(i, j, -1);

}

else

{

board[i][j] = -color;

Neighbor\_Location(i, j, 1);

v = min\_max\_dfs(depth + 1, -1,color, a, b);

if (v < b){

b = v;

if (a >= b){//减枝 向第上一层即Max层返回一个无用的极大负数

board[i][j] = 0;

Neighbor\_Location(i, j, -1);

return -1e9;

}

key = v; //返回更新的上界值 亦是所有子节点里最小的上界值

}

board[i][j] = 0;

Neighbor\_Location(i, j, -1);

}

}

}

}

return key;

}

Settled\_Situation.cpp

#include"wuziqi.h"

//子节点估值

int Settled\_Situation(int color){//color表示是对黑子还是白子估值

int total = 0,pointscore = 0;

for (int i = 0; i < ROW; i++) {

for (int j = 0; j < COL; j++) {

if (board[i][j] != 0) {

pointscore = SituationValuation(i, j, color);

total = total + pointscore;

pointscore = 0;

}

}

}

return total;

}

SituationValuation.cpp

#include"wuziqi.h"

//局面估值

int SituationValuation(int row, int col, int color) {//注意 进入这个函数的条件的是board[col][row]上已经落子

int Valuation = 0;//用于估值

int Val\_White[13] = { 0,2,5,2,5,20,5,15,200,20,200,1000000 , 9999999};

int Val\_BLACK[13] = { 0,2,5,2,5,20,5,15,200,20,200,1000000 , 9999999};

int flag = 0;

if (board[row][col] == color) {

flag = 1;

}

else {

flag = -1;

}

if (color == THEWHITE) {

for (int i = 0; i < 4; i++) {

Valuation += flag \* Val\_White[Chess\_type(row, col, i + 1) - 1];

}

}

else if (color== THEBLACK) {

for (int i = 0; i < 4; i++) {

Valuation += flag \* Val\_BLACK[Chess\_type(row, col, i + 1) - 1];

}

}

return Valuation;

}