实验报告

1. 由于无禁手，玩家默认执黑棋，AI默认执黑棋，通过点击鼠标即可落子且无时间限制，横，竖，斜任意方向连成五子即算胜利.
2. 主要工具为EasyX和Visual Studio 2022.引用的库有

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<mmsystem.h>

#include"tools.h"

#include<math.h>

#include<minwindef.h>

#include <graphics.h>

#include<string.h>

#include<time.h>

#include<stdlib.h>

#include <stdio.h>

#include <Windows.h>

1. 关于配置开发环境（Visual Studio 2022同时是编辑，编译，运行环境）

首先下载Visual Studio 2022，接着安装C/C++组件，然后创建项目与各类文件，经过试错无误后，开始正式编写代码.

1. 模块分为2个头文件（ChessData.h；tools.h），3个源文件（ChessData.cpp；tools.cpp；main.cpp）

其中，ChessData.h负责棋盘及棋子的各类数据，ChessData.cpp与main.cpp均需引用ChessData.h.而tools.h中有负责载入PNG图并使背景透明化的函数声明，tools.cpp与main.cpp均需引用tools.h.整体来说，程序存在一定耦合度且实现了封装与接口的使用.

1. 关键实现代码有

首先是对于棋盘的初始化及更新

void initChessData(ChessData\* data){

if (!data) {

return; }

memset(data->chessmap, 0, sizeof(data->chessmap));

memset(data->chesspoint, 0, sizeof(data->chesspoint));

data->sideflag = true;

}

void newChessmap(ChessData\*data, int row, int col) {

if (!data) {

return;

}

if (data->sideflag) {

data->chessmap[row][col] = 1;

}

else {

data->chessmap[row][col] = -1;

}

data->sideflag = !data->sideflag;

}

其次是关于判断胜负的代码

bool Win(ChessData\* game, int row, int col) {

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

if (col - i >= 0 && col - i + 4 < boardsize && game->chessmap[row][col - i] == game->chessmap[row][col - i + 1] && game->chessmap[row][col - i] == game->chessmap[row][col - i + 2] && game->chessmap[row][col - i] == game->chessmap[row][col - i + 3] && game->chessmap[row][col - i] == game->chessmap[row][col - i + 4])

return true;

}

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

if (row - i >= 0 && row - i + 4 < boardsize && game->chessmap[row - i][col] == game->chessmap[row - i + 1][col] && game->chessmap[row - i][col] == game->chessmap[row - i + 2][col] && game->chessmap[row - i][col] == game->chessmap[row - i + 3][col] && game->chessmap[row - i][col] == game->chessmap[row - i + 4][col])

return true;

}

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

if (row + i - 4 >= 0 && row + i < boardsize && col - i >= 0 && col - i + 4 < boardsize && game->chessmap[row + i][col - i] == game->chessmap[row + i - 1][col - i + 1] && game->chessmap[row + i][col - i] == game->chessmap[row + i - 2][col - i + 2] && game->chessmap[row + i][col - i] == game->chessmap[row + i - 3][col - i + 3] && game->chessmap[row + i][col - i] == game->chessmap[row + i - 4][col - i + 4])

return true;

}

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

if (row - i >= 0 && row - i + 4 < boardsize && col - i > 0 && col - i + 4 < boardsize && game->chessmap[row - i][col - i] == game->chessmap[row - i + 1][col - i + 1] && game->chessmap[row - i][col - i] == game->chessmap[row - i + 2][col - i + 2] && game->chessmap[row - i][col - i] == game->chessmap[row - i + 3][col - i + 3] && game->chessmap[row - i][col - i] == game->chessmap[row - i + 4][col - i + 4])

return true;

}

return false;

}

接着是关于玩家落子的代码

bool click(MOUSEMSG msg) {

int x = msg.x;

int y = msg.y;

int col = (x - distance\_x) / size;

int row = (y - distance\_y) / size;

int lefttopX = distance\_x + size \* col;

int lefttopY = distance\_y + size \* row;

int len, len1, len2, len3, len4;

int choosePos = false;

len1 = sqrt((x - lefttopX) \* (x - lefttopX) + (y - lefttopY) \* (y - lefttopY));//左上

len2 = sqrt((x - lefttopX - size) \* (x - lefttopX - size) + (y - lefttopY) \* (y - lefttopY));//右上

len3 = sqrt((x - lefttopX) \* (x - lefttopX) + (y - lefttopY - size) \* (y - lefttopY - size));//左下

len4 = sqrt((x - lefttopX - size) \* (x - lefttopX - size) + (y - lefttopY - size) \* (y - lefttopY - size));//右下

len = minimum(len1, len2, len3, len4);

if (len == len1&& len < threshold) {

clickcol = col;

clickrow = row;

if (game.chessmap[clickrow][clickcol] == 0) {

choosePos = true;

}

}

if (len == len2 &&len < threshold) {

clickcol = col+1;

clickrow = row;

if (game.chessmap[clickrow][clickcol] == 0) {

choosePos = true;

}

}

if (len == len3 && len < threshold) {

clickcol = col;

clickrow = row+1;

if (game.chessmap[clickrow][clickcol] == 0) {

choosePos = true;

}

}

if (len == len4 && len < threshold) {

clickcol = col + 1;

clickrow = row + 1;

if (game.chessmap[clickrow][clickcol] == 0) {

choosePos = true;

}

}

return choosePos;

}

void Hand(int row, int col, Kind kind) {

int x = distance\_x + col \* size - 0.5 \* size;

int y = distance\_y + row \* size - 0.5 \* size;

if (kind == White) {

putimagePNG(x, y, &WhiteImg);

}

else { putimagePNG( x, y, &BlackImg);

}

}

通过这两段段代码，实现了对鼠标点击的获取及玩家落子的正常进行，且通过模糊边界，给予了点击一定的容错，点歪了也能进行一定的修正，能保证棋子不会出现下歪的情况.

当然，更重要的是对AI落子的编写：

void value(ChessData\* data) {

if (!data) {

return;

}

int playernum = 0;

int ainum = 0;

int emptynum = 0;

memset(data->chesspoint, 0, sizeof(data->chesspoint));

for (int row = 0;row < boardsize;row++)

for (int col = 0;col < boardsize;col++) {

if (row >= 0 && col >= 0 && data->chessmap[row][col] == 0) {

int pos[4][2] = { {1,0},{1,1},{0,1},{-1,1} };

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

int x = pos[k][0];

int y = pos[k][1];

playernum = 0;

ainum = 0;

emptynum = 0;

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

if (row + i \* y >= 0 && row + i \* y < boardsize && col + i \* x >= 0 && col + i \* x < boardsize && data->chessmap[row + i \* y][col + i \* x] == 1) {

playernum++;

}

else if (row + i \* y >= 0 && row + i \* y < boardsize && col + i \* x >= 0 && col + i \* x < boardsize && data->chessmap[row + i \* y][col + i \* x] == 0) {

emptynum++;

break;

}

else break;

}

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

if (row - i \* y >= 0 && row - i \* y < boardsize && col - i \* x >= 0 && col - i \* x < boardsize && data->chessmap[row - i \* y][col - i \* x] == 1) {

playernum++;

}

else if (row - i \* y >= 0 && row - i \* y < boardsize && col - i \* x >= 0 && col - i \* x < boardsize && data->chessmap[row - i \* y][col - i \* x] == 0) {

emptynum++;

break;

}

else break;

}

if (playernum == 1) {

data->chesspoint[row][col] += 10;

}

else if (playernum == 2) {

if (emptynum == 1) {

data->chesspoint[row][col] += 30;

}

else if (emptynum == 2) {

data->chesspoint[row][col] += 40;

}

}

else if (playernum == 3) {

if (emptynum == 1) {

data->chesspoint[row][col] += 60;

}

else if (emptynum == 2) {

data->chesspoint[row][col] += 200;

}

}

else if (playernum == 4) {

data->chesspoint[row][col] += 20000;

}

emptynum = 0;

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

if (row + i \* y >= 0 && row + i \* y < boardsize && col + i \* x >= 0 && col + i \* x < boardsize && data->chessmap[row + i \* y][col + i \* x] == -1) {

ainum++;

}

else if (row + i \* y >= 0 && row + i \* y < boardsize && col + i \* x >= 0 && col + i \* x < boardsize && data->chessmap[row + i \* y][col + i \* x] == 0) {

emptynum++;

break;

}

else break;

}

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

if (row - i \* y >= 0 && row - i \* y < boardsize && col - i \* x >= 0 && col - i \* x < boardsize && data->chessmap[row - i \* y][col - i \* x] == -1) {

ainum++;

}

else if (row - i \* y >= 0 && row - i \* y < boardsize && col - i \* x >= 0 && col - i \* x < boardsize && data->chessmap[row - i \* y][col - i \* x] == 0) {

emptynum++;

break;

}

else break;

}

if (ainum == 0) {

data->chesspoint[row][col] += 5;

}

else if (ainum == 1) {

data->chesspoint[row][col] += 10;

}

else if (ainum == 2) {

if (emptynum == 1) {

data->chesspoint[row][col] += 25;

}

else if (emptynum == 2) {

data->chesspoint[row][col] += 50;

}

}

else if (ainum == 3) {

if (emptynum == 1) {

data->chesspoint[row][col] += 55;

}

else if (emptynum == 2) {

data->chesspoint[row][col] += 300;

}

}

else if (ainum >= 4) {

data->chesspoint[row][col] += 30000;

}

}

}

}

}

Cpoint move(ChessData\* data) {

value(data);

int maxscore = 0;

Cpoint maxpoints[boardsize \* boardsize] = { 0 };

int k = 0;

for (int row = 0;row < boardsize;row++)

for (int col = 0;col < boardsize;col++) {

if (data->chessmap[row][col] == 0) {

if (data->chesspoint[row][col] > maxscore) {

memset(maxpoints, 0, sizeof(maxpoints));

k = 0;

maxscore = data->chesspoint[row][col];

maxpoints[k].row = row;

maxpoints[k].col = col;

}

else if (data->chesspoint[row][col] == maxscore) {

k++;

maxpoints[k].row = row;

maxpoints[k].col = col;

}

}

}

if (k > 0) {

srand((unsigned)time(0));

int renju = rand() % k;

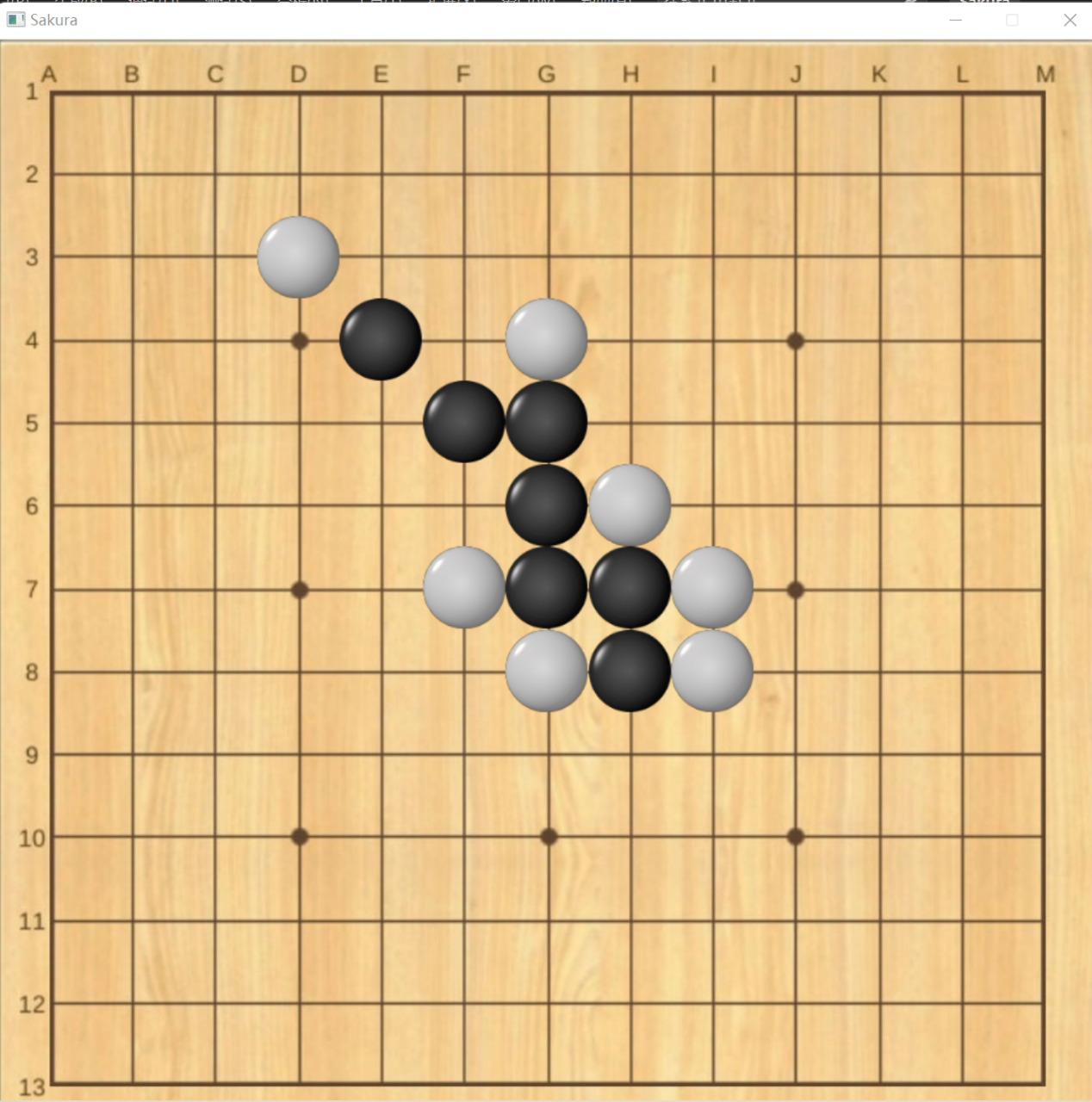
return maxpoints[renju];

}

else {return maxpoints[k];}

}

首先是第一段代码，为每个棋子的落点进行了加权与赋分，这为AI后续下棋打下基础，并且预防了越界的问题.接着是第二段代码，在上一段代码计算出来的分值的基础上实现了落子.

6.运行界面如图所示

所做的五子棋最大的特点之一就是通过tools.h与tools.cpp贝叶斯定理，实现了棋子图像背景的透明化，使得棋子无边界，无阴影，十分精致美观。同时程序运行速度快，帧率稳定且毫无卡顿，几乎可以做到在玩家落子后瞬间实现落子.

7.程序包括所有图片总共466 MB .

8.AI的棋力大概有三段水平，该结论来自于我做实验，让一些朋友与AI对弈，其中一名三段水平的朋友说一般爱好者是难下赢的，他需要慢下，如果快下也可能会输掉.

9.参考文献有

# **1.《使用 EASYX 载入PNG图并透明背景》，下为链接：**

<https://blog.csdn.net/weixin_45779485/article/details/102757315>

# 《**C语言五子棋、三子棋人机对战篇的详细介绍**》，下为链接：

<https://blog.csdn.net/qq_64318258/article/details/126602578>

实验报告如上，感谢俸老师过目.