# **C语言五子棋AI实验报告**

苏徐涛

1. **实验目标**

用C语言设计出一个简单的五子棋游戏。人机对战，黑棋白棋可选，可以投降，无禁手限制、时间限制，暂无悔棋功能。

游戏规则：与五子棋的规则一样，只是没有禁手限制。

1. **构建开发环境**

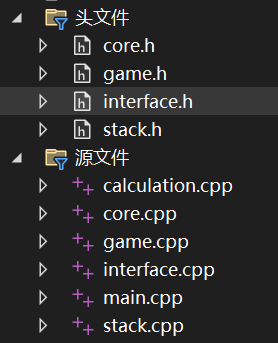
集成开发环境：visual studio 2022

程序设计语言：C语言

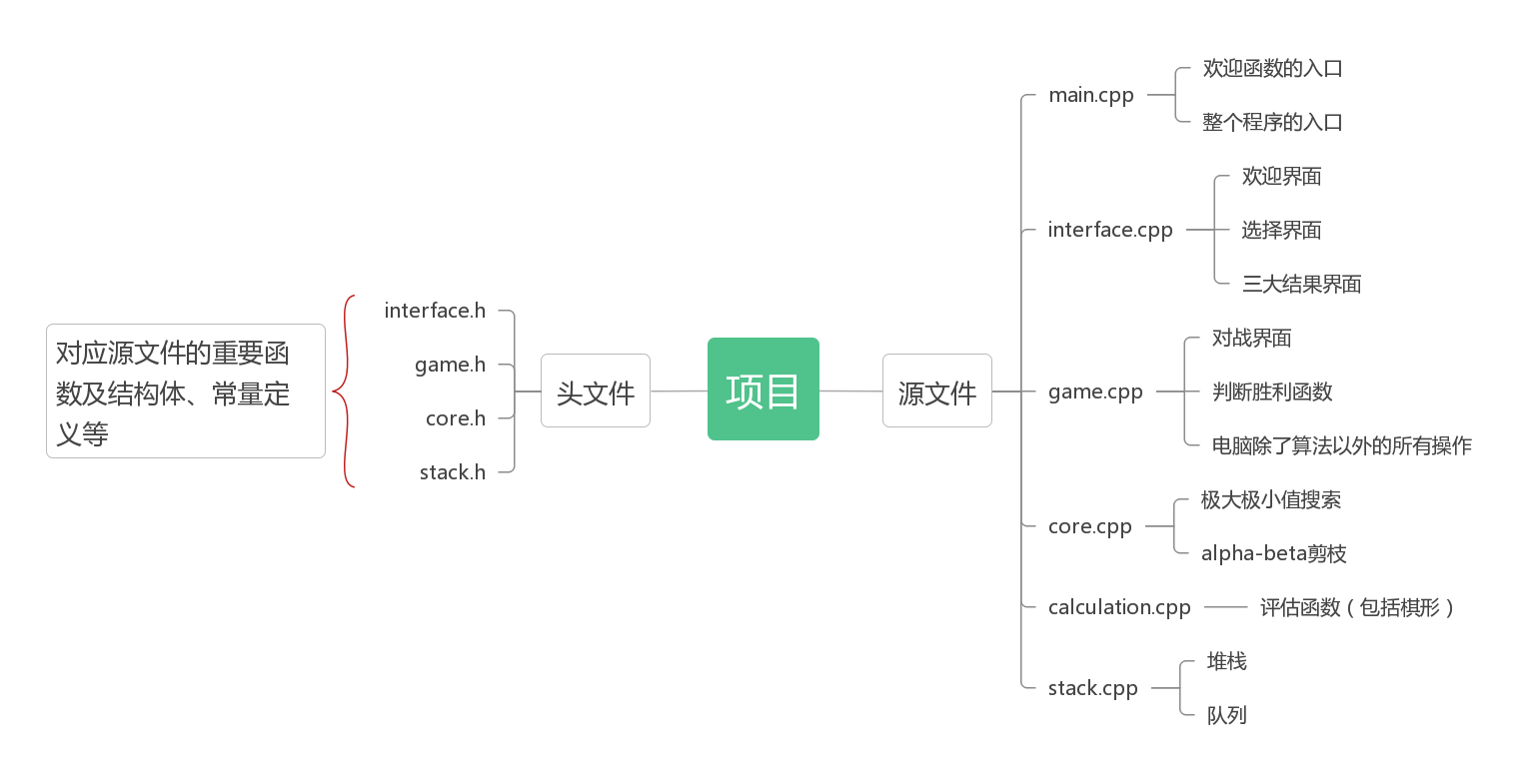
图形库插件：EasyX图形库

运行平台：Windows 11

1. **文件分工**

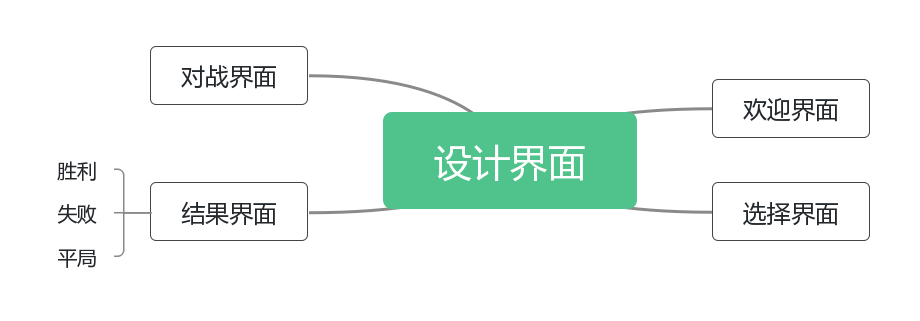


我一共写了这些文件，下面是示意图：



1. **界面设计**

首先设计各种界面：



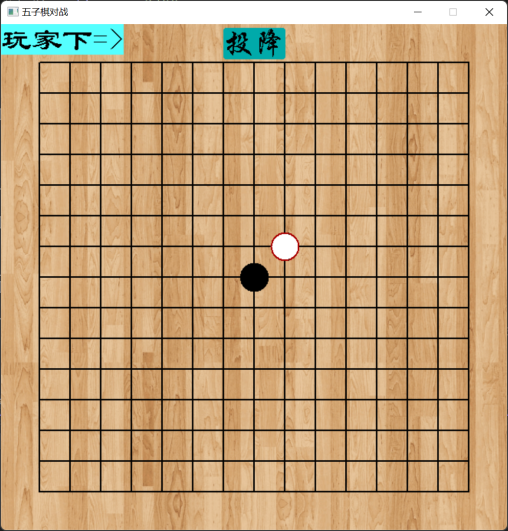
**1 欢迎界面**



**2 选择界面**



**3 对战界面**



**4 结果界面**



附上头文件代码：

void welcome();

void choice();

void victory();

void failure();

void draw();

1. **核心算法**

算法部分作为整个项目的精华，我在这上面花了最多的时间（大部分时间都花在了Debug上），接下来一一论述：

**1 数学原理**

五子棋作为完全信息动态零和博弈，需要博弈论和决策树相关的知识。对于下面的几个步骤，都必须先理解其中的数学原理。

**2 评估函数**

对于五子棋，计算机需要解决的只有一个问题，那就是五子棋下一步需要下在什么地方。对此，我们首先需要先完成一个评估函数，这个函数能够对当前的局面做出一个比较准确的评分。评估函数的优劣就在于它能多大程度上完整刻画局面。

首先，需要获取整个局面的完整信息。因为棋盘是固定的，我的选择是使用15×15的二维数组，用1代表黑棋，用2代表白棋，用0代表空位。

其次，选择恰当的评估方法，我查阅了网上很多资料，认为六元组是其中一种比较容易的实现方法。下面给出一部分关键代码：

int sixpart[6] = { 0,0,0,0,0,0 };

//横向搜索

for (int a = 0; a < 15; a++)

{

for (int b = 0; b < 10; b++)

{

int d = b;

for (int c = 0; c < 6; c++)

{

sixpart[c] = helparr[a][d++];

}

grade += judge\_sixBlack(sixpart);

grade -= judge\_sixWhite(sixpart);

}

}

//纵向搜索······

//斜向搜索······

//次斜向搜索······

（上面给出的代码我调整了很久，可能有不完善的地方）

最后，就是对五子棋规则的诠释。因为这部分代码仅仅表示大家都知道的五子棋棋形（连五、活四、活三······），没有任何算法难度，只是需要注意棋形的评分及不漏掉任何棋形，这里就不贴出来了。

**3 极大极小值搜索**

这里开始有一定的算法难度，遵循深度优先原则。这里给出一部分代码：

for (int a = 0; a < 15; a++)

{

for (int b = 0; b < 15; b++)

{

if (copyarr[a][b] != 0)//有棋就跳过

continue;

if (JudgeEmpty(copyarr, a, b))//距离太远也要跳过

continue;

copyarr[a][b] = 2;

SearchFirst(depth - 1, copyarr, CutLeaves);

copyarr[a][b] = 0;

MINgrade = min(MINgrade, pop(pStackFirst));

}

}

push(pStackFirst, MINgrade);

return pStackFirst;

这是min层，去掉了剪枝代码。这个算法越往下算棋力越高，但相对而言计算的时间也越长，算到四层时时间就已经相当长了（指到达人无法忍受的地步），因此还需要一些优化算法。

**4 alpha-beta剪枝**

alpha-beta剪枝是一种安全的剪枝策略，理解原理后实现起来也不难，只有短短几行代码，下面给出加了剪枝的代码（与上面的代码是同一个地方，仅仅加上了剪枝代码）：

for (int a = 0; a < 15; a++)

{

for (int b = 0; b < 15; b++)

{

if (copyarr[a][b] != 0)//有棋就跳过

continue;

if (JudgeEmpty(copyarr, a, b))//距离太远也要跳过

continue;

copyarr[a][b] = 2;

SearchFirst(depth - 1, copyarr, CutLeaves);

copyarr[a][b] = 0;

MINgrade = min(MINgrade, pop(pStackFirst));

//剪枝

CutLeaves->beta = MINgrade;

if (MINgrade < CutLeaves->alpha)

goto exit;

}

}

exit:

push(pStackFirst, MINgrade);

return pStackFirst;

剪枝后能够明显感到速度的提升，目前我的算法只有四层，剪枝前最长能到1分多钟，剪枝后基本能控制在10秒以内。如果层数再加深的话就需要配合进一步的优化算法了。

1. **其他功能**

为了增加游戏的趣味性，需要给游戏增加音乐，这里使用mciSendString函数播放音乐，在程序前加上宏定义“#pragma comment(lib,”winmm.lib”)”就可以任意加上音乐了。在这里我使用了多首音乐随机挑选：

void playmusic()

{

srand((unsigned)time(NULL));

switch (rand()%4+1)

{

case 1:

mciSendString(L"open D:\\CloudMusic\\询问1.mp3 alias 询问1", NULL, 0, NULL);

mciSendString(L"play 询问1 repeat", NULL, 0, NULL);

break;

case 2:

mciSendString(L"open D:\\CloudMusic\\询问2.mp3 alias 询问2", NULL, 0, NULL);

mciSendString(L"play 询问2 repeat", NULL, 0, NULL);

break;

case 3:

mciSendString(L"open D:\\CloudMusic\\询问3.mp3 alias 询问3", NULL, 0, NULL);

mciSendString(L"play 询问3 repeat", NULL, 0, NULL);

break;

case 4:

mciSendString(L"open D:\\CloudMusic\\询问4.mp3 alias 询问4", NULL, 0, NULL);

mciSendString(L"play 询问4 repeat", NULL, 0, NULL);

break;

default:

printf("没有找到此音乐！\a\n");

exit(0);

}

}

对于五子棋的棋盘与棋子，我犹豫了很久，经反复测试还是放弃使用精致的图片，只能粗糙地直接写程序画上去。

然后，在实际对战时我发现了一个问题，那就是在后期棋子越来越多的时候，无法轻易辨别出电脑或者自己下在了什么位置，依次我也写了三个函数，归纳成函数指针数组，方便调用。其主要功能就是在电脑或自己下的最新一步棋上画一个红色圆圈，调试了很久，终于达到了令人满意的效果。下面给出效果截图（电脑是黑棋）和关键代码：

void printred(int x, int y)

{

setlinecolor(RED);

circle(x \* 40 + 50, y \* 40 + 50, 18);

}

void clearredWhite(int x, int y)

{

setlinecolor(WHITE);

circle(x \* 40 + 50, y \* 40 + 50, 18);

}

void clearredBlack(int x, int y)

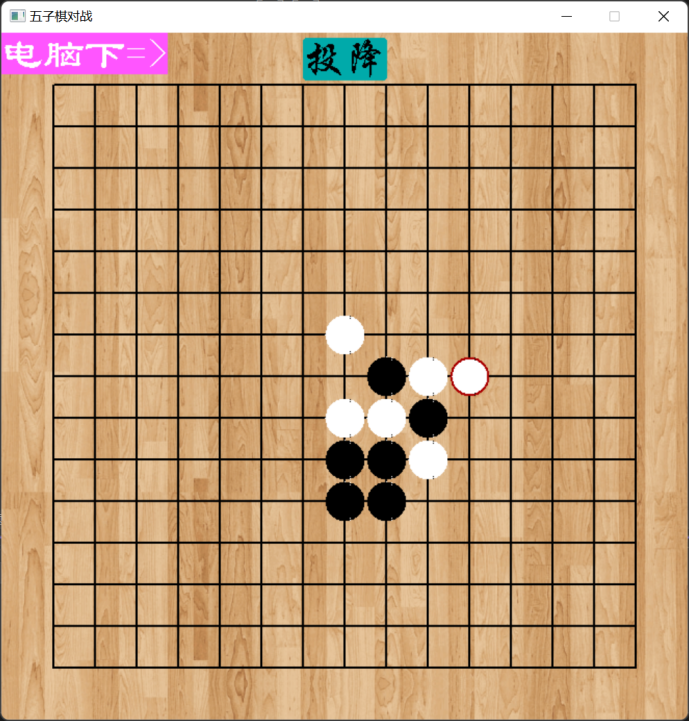
{

setlinecolor(BLACK);

circle(x \* 40 + 50, y \* 40 + 50, 18);

}

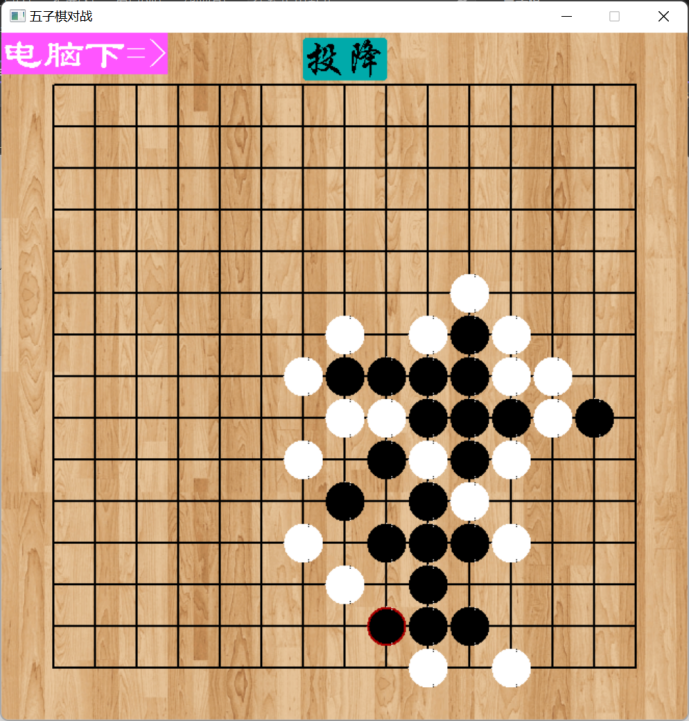
void (\*rp[3])(int, int) = { printred,clearredBlack,clearredWhite };



另外，五子棋还有一个投降功能按键，当你觉得此局必输无疑时，你可以直接按下结束游戏。

1. **结果评价**

正常对战的游戏截图：（这张图我是黑棋，我本以为连成两个活三后稳操胜券，却没看到白棋已经连成了一个四，然后我就输了）



程序运行流畅，exe文件大小有71KB。

下面来评价一下AI的智能水平：

首先，从算法的优劣来评价。我所使用的极大极小值搜索和αβ剪枝，只有四层棋力的水平，略强于普通玩家。

然后，从玩家体验上来说，我经过实际对战，电脑对我差不多六四开，尽管如此，我也发现了电脑的“思路”有一些很严重的问题：如无视自己的四而先堵对方的四等。

最后，讨论需要改进的地方。一是需要提升棋力，二是需要加入更多的优化算法，三是解决电脑不够“聪明”的问题。这些仍然需要不断改进。

**参考文献**

1. 五子棋AI算法

<https://blog.csdn.net/qq_44732921/article/details/102648408>

1. 五子棋断奶班基础知识（三）五子棋基本棋型及其特点

<http://game.onegreen.net/wzq/HTML/142336.html>

1. 五子棋AI教程

<https://github.com/lihongxun945/myblog/labels/%E4%BA%94%E5%AD%90%E6%A3%8BAI%E6%95%99%E7%A8%8B%E7%AC%AC%E4%BA%8C%E7%89%88>

[4] Alpha-Beta剪枝算法原理

<https://blog.csdn.net/qq_24178985/article/details/115858780>