Gomoku

Ver 1.0

09016401
顾婷瑄
Botzone id: Gutingxuan

1. Old versions review

This is my very first version of Gomoku, and I am excited to introduce it to you.

2. Literatures I read

[1] 王志水. 基于搜索算法的人工智能在五子棋博弈中的应用研究[D].山东：中国石油大学, 2006

[2] Blog: 五子棋基本棋型@我是老邱

五子棋估值算法@maxuewei2

3. Current idea of design

i. Essence

A function based on the evaluation of each available position.

ii. Origin of the idea

Myself

iii. Expectation

The function can comprehensively take every potential board type into account and then make the best move.

iv. Implementation

① A function that collect all the positions that has neighbors in the distance of 2 so that it can save searching time.

② Two evaluation functions which will separately evaluate the attacking potential and the defense potential of each available position.

③ A function that can evaluate the chess from eight directions.

④ A function which will choose the best position to make the next move.

v. Advantages of the design

It is easy to implement and it is convenient to edit the current known board type so that the computer can give out a more accurate score.

vi. Weakness

The computer can only evaluate the current board and give the score. It can not evaluate the potential moves in future turns and give out a better solution.

vii. Improvement directions

I will implement a search algorithm which can foresee future moves and thus give out a better solution.

4. Evaluation of the performance

It is still pretty weak now since even I can beat it. However, after being posted on the Botzone, sometimes it can actually beat other Gomoku AI players. But since this is just the very first version, I am not ready to put it in the ranking list because the result might hurt feelings so there will not be any rank or score now. I am still glad it can successfully detective the winning move of the player and then prevent in advance and the accident victories really give me a thrill.

5. Code structure and detailed implementations

i. The code structure

```
#pragma once
#include <iostream>
#include <stack>
#include <iomanip>
using namespace std;

class Player
{
public:
    int choice;             //玩家决定为黑子还是白子，0为黑子，1为白子
    int x, y;               //玩家当前落子位置
};

class ChessBoard
{
public:
    int choice;             //电脑执子
    int board[15][15];      //棋盘
    int score1[15][15];     //每个空位防守得分
    int score2[15][15];     //每个空位攻击得分
    int move1[225][2];      //可移动的空位
    int lastx, lasty;

    int Winning();
    int EvaluateA(int x, int y, int choice);    //攻击估值函数
    int EvaluateD(int x, int y, int choice);    //防守估值函数
    int evaluate();                             //静态估值函数
    int getLine(int x, int y, int i, int j);    //不同方向不同距离
    int gen();                                  //只选择有邻居的空位
    void nextPosition();                        //静态估值计算下一个落子位置
```

ii. The score table

Originally, the table goes:

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0
0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 0
0, 1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 1, 0
0, 1, 2, 3, 4, 4, 4, 4, 4, 4, 4, 3, 2, 1, 0
0, 1, 2, 3, 4, 5, 5, 5, 5, 5, 4, 3, 2, 1, 0
0, 1, 2, 3, 4, 5, 6, 6, 6, 5, 4, 3, 2, 1, 0
0, 1, 2, 3, 4, 5, 6, 7, 6, 5, 4, 3, 2, 1, 0
0, 1, 2, 3, 4, 5, 6, 6, 6, 5, 4, 3, 2, 1, 0
0, 1, 2, 3, 4, 5, 5, 5, 5, 5, 4, 3, 2, 1, 0
0, 1, 2, 3, 4, 4, 4, 4, 4, 4, 4, 3, 2, 1, 0
0, 1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 1, 0
0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 0
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Different position is given different initial score so that if the player goes black and makes the first move, then the computer will choose the position that near the center to make the move instead of making the move randomly from the first row.

```
/*
*: 当前空位置
0：其他空位置
1：当前player
2：对手
*/
```

Different board type is given different scores:

		Attack	Defense
活四	01111*	300000	300000
冲四A	21111*	250000	200000
冲四B	1*111	250000	200000
冲四C	11*11	250000	200000
活三A-1	0111*0	60000	50000
活三B-1	01*110	60000	50000
活三B-2	*10110	40000	30000
活三B-3	01011*	40000	30000
眠三A-1	2111*0	3000	2000
眠三A-2	21110*	1500	500
眠三B-1	211*10	3000	2000
眠三B-2	21101*	1800	800
眠三C-1	21*110	3000	2000
眠三C-2	21011*	1800	800

眠三D-1	211*01	1600	600
眠三D-2	2110*1	1600	600
眠三E	21*101	1550	550
活二A-1	0*0110	1650	650
活二A-2	0*110	4000	3000
活二B-1	0*1010	1650	650
活二B-2	01*10	4000	3000
活二C-1	01*010	1650	650
活二C-2	*10010	1650	650
眠二A-1	211*00	1150	150
眠二A-2	2110*0	1150	150
眠二A-3	21100*	1150	150
眠二B-1	21*100	1250	250
眠二B-2	2101*0	1250	250
眠二B-3	21010*	1250	250
眠二C-1	21*010	1200	200
眠二C-2	210*10	1200	200
眠二C-3	21001*	1200	200
眠二D-1	21*001	1100	100
眠二D-2	210*01	1100	100
Numoftwo>=2		3000	2000

Different board type scores can help computer choose the best position to make the move. Attack score is the one that computer can gain if it places the stone on a position and the defense score is the one that player can gain if he places the stone on a position. Therefore, the computer will have to choose whether to make a move to attack or to defense in case the player will make a move to gain a higher score. And I choose to make the computer a more competitive one so the attack score will be a little bit higher than defense score in the same board type case. And after playing a few games, I found that the number of alive-two board types sometimes means a lot so I give extra points to the position that meets at least two alive-two board types.

iii. How to evaluate the score of each position

Based on the score table and the search from eight directions that whether this available position is in any valuable board type.

```
int ChessBoard::EvaluateD(int x, int y, int choice)
{
    int numoftwo = 0;
    for (int i = 1; i <= 8; i++)
    {
        //活四  01111*
        if (getLine(x, y, i, 1) == choice && getLine(x, y, i, 2) == choice && getLine(x, y, i, 3) == choice
            && getLine(x, y, i, 4) == choice && getLine(x, y, i, 5) == -1)
            score1[x][y] += 300000;

        //冲四A  21111*
        if (getLine(x, y, i, 1) == choice && getLine(x, y, i, 2) == choice && getLine(x, y, i, 3) == choice
            && getLine(x, y, i, 4) == choice && getLine(x, y, i, 5) == 1 - choice)
            score1[x][y] += 200000;

        //冲四B  1*111
        if (getLine(x, y, i, -1) == choice && getLine(x, y, i, 1) == choice && getLine(x, y, i, 2) == choice
            && getLine(x, y, i, 3) == choice)
            score1[x][y] += 200000;

        //冲四C  11*11
        if (getLine(x, y, i, -2) == choice && getLine(x, y, i, -1) == choice && getLine(x, y, i, 1) == choice
            && getLine(x, y, i, 2) == choice)
            score1[x][y] += 200000;

        //活三A-1 0111*0
        if (getLine(x, y, i, 1) == -1 && getLine(x, y, i, -1) == choice && getLine(x, y, i, -2) == choice
            && getLine(x, y, i, -3) == choice && getLine(x, y, i, -4) == -1)
            score1[x][y] += 50000;

        //活三B-1  01*110
        if (getLine(x, y, i, -1) == choice && getLine(x, y, i, -2) == -1 && getLine(x, y, i, 1) == choice
            && getLine(x, y, i, 2) == choice && getLine(x, y, i, 3) == -1)
            score1[x][y] += 50000;

        //活三B-2  *10110
        if (getLine(x, y, i, 1) == choice && getLine(x, y, i, 2) == -1 && getLine(x, y, i, 3) == choice
            && getLine(x, y, i, 4) == choice && getLine(x, y, i, 5) == -1)
            score1[x][y] += 30000;

        //活三B-3  01011*
        if (getLine(x, y, i, -1) == choice && getLine(x, y, i, -2) == choice && getLine(x, y, i, -3) == -1
            && getLine(x, y, i, -4) == choice && getLine(x, y, i, -5) == -1)
            score1[x][y] += 30000;

        //眠三A-1 2111*0
        if (getLine(x, y, i, 1) == -1 && getLine(x, y, i, -1) == choice && getLine(x, y, i, -2) == choice
            && getLine(x, y, i, -3) == choice && getLine(x, y, i, -4) == 1 - choice)
            score1[x][y] += 2000;

        //眠三A-2 21110*
        if (getLine(x, y, i, -1) == -1 && getLine(x, y, i, -2) == choice && getLine(x, y, i, -3) == choice
            && getLine(x, y, i, -4) == choice && getLine(x, y, i, -5) == 1 - choice)
            score1[x][y] += 500;

        //眠三B-1 211*10
        if (getLine(x, y, i, 1) == choice && getLine(x, y, i, 2) == -1 && getLine(x, y, i, -1) == choice
            && getLine(x, y, i, -2) == choice && getLine(x, y, i, -3) == 1 - choice)
            score1[x][y] += 2000;

        //眠三B-2 21101*
        if (getLine(x, y, i, -1) == 1 - choice && getLine(x, y, i, -2) == -1 && getLine(x, y, i, -3) == choice
            && getLine(x, y, i, -4) == choice && getLine(x, y, i, -5) == 1 - choice)
            score1[x][y] += 800;
```

```
//眠三C-1  21*110
if (getLine(x, y, i, 1) == -1 && getLine(x, y, i, 2) == choice && getLine(x, y, i, 3) == choice
    && getLine(x, y, i, 4) == choice)
    score1[x][y] += 2000;

//眠三C-2  21011*
if (getLine(x, y, i, -1) == choice && getLine(x, y, i, -2) == choice && getLine(x, y, i, -3) == -1
    && getLine(x, y, i, -4) == choice && getLine(x, y, i, -5) == 1 - choice)
    score1[x][y] += 800;

//眠三D-1  211*01
if (getLine(x, y, i, -3) == 1 - choice && getLine(x, y, i, -1) == choice && getLine(x, y, i, -2) == choice
    && getLine(x, y, i, 1) == -1 && getLine(x, y, i, 2) == choice)
    score1[x][y] += 600;

//眠三D-2  2110*1
if (getLine(x, y, i, 1) == choice && getLine(x, y, i, -1) == -1 && getLine(x, y, i, -2) == choice
    && getLine(x, y, i, -3) == choice && getLine(x, y, i, -4) == 1 - choice)
    score1[x][y] += 600;

//眠三E    21*101
if (getLine(x, y, i, -2) == 1 - choice && getLine(x, y, i, -1) == choice && getLine(x, y, i, 1) == choice
    && getLine(x, y, i, 2) == -1 && getLine(x, y, i, 3) == choice)
    score1[x][y] += 550;

//活二A-1  0*0110
if (getLine(x, y, i, -1) == -1 && getLine(x, y, i, 1) == -1 && getLine(x, y, i, 2) == choice
    && getLine(x, y, i, 3) == choice && getLine(x, y, i, 4) == -1)
{
    score2[x][y] += 650;
    numoftwo++;
}

//活二A-2  0*110
if (getLine(x, y, i, -1) == -1 && getLine(x, y, i, 1) == choice && getLine(x, y, i, 2) == choice
    && getLine(x, y, i, 3) == -1)
{
    score2[x][y] += 3000;
    numoftwo++;
}

//活二B-1  0*1010
if (getLine(x, y, i, -1) == -1 && getLine(x, y, i, 1) == choice && getLine(x, y, i, 2) == -1
    && getLine(x, y, i, 3) == choice && getLine(x, y, i, 4) == -1)
{
    score2[x][y] += 650;
    numoftwo++;
}

//活二B-2  01*10
if (getLine(x, y, i, -1) == choice && getLine(x, y, i, -2) == -1 && getLine(x, y, i, 1) == choice
    && getLine(x, y, i, 2) == -1)
{
    score1[x][y] += 3000;
    numoftwo++;
}

//活二C-1  01*010
if (getLine(x, y, i, -2) == -1 && getLine(x, y, i, -1) == choice && getLine(x, y, i, 1) == -1
    && getLine(x, y, i, 2) == choice && getLine(x, y, i, 3) == -1)
{
    score1[x][y] += 650;
    numoftwo++;
}
```

```
    //活二C-2  *10010
    if (getLine(x, y, i, 1) == choice && getLine(x, y, i, 2) == -1 && getLine(x, y, i, 3) == -1
    ¦   && getLine(x, y, i, 4) == choice && getLine(x, y, i, 5) == -1)
    {
    ¦   score1[x][y] += 650;
    ¦   numoftwo++;
    }

    //眠二A-1  211*00
    if (getLine(x, y, i, -1) == choice && getLine(x, y, i, -2) == choice && getLine(x, y, i, -3) == 1 - choice
        && getLine(x, y, i, 1) == -1 && getLine(x, y, i, 2) == -1)
        score1[x][y] += 150;

    //眠二A-2  2110*0
    if (getLine(x, y, i, 1) == -1 && getLine(x, y, i, -1) == -1 && getLine(x, y, i, -2) == choice
        && getLine(x, y, i, -3) == choice && getLine(x, y, i, -4) == 1 - choice)
        score1[x][y] += 150;

    //眠二A-3  21100*
    if (getLine(x, y, i, -1) == -1 && getLine(x, y, i, -2) == -1 && getLine(x, y, i, -3) == choice
        && getLine(x, y, i, -4) == choice && getLine(x, y, i, -5) == 1 - choice)
        score1[x][y] += 150;

    //眠二B-1  21*100
    if (getLine(x, y, i, -1) == choice && getLine(x, y, i, -2) == 1 - choice && getLine(x, y, i, 1) == choice
        && getLine(x, y, i, 2) == -1 && getLine(x, y, i, 3) == -1)
        score1[x][y] += 250;

    //眠二B-2  2101*0
    if (getLine(x, y, i, 1) == -1 && getLine(x, y, i, -1) == choice && getLine(x, y, i, -2) == -1
        && getLine(x, y, i, -3) == choice && getLine(x, y, i, -4) == 1 - choice)
        score1[x][y] += 250;

    //眠二B-3  21010*
    if (getLine(x, y, i, -1) == -1 && getLine(x, y, i, -2) == choice && getLine(x, y, i, -3) == -1
        && getLine(x, y, i, -4) == choice && getLine(x, y, i, -5) == 1 - choice)
        score1[x][y] += 250;

    //眠二C-1  21*010
    if (getLine(x, y, i, -1) == choice && getLine(x, y, i, -2) == 1 - choice && getLine(x, y, i, 1) == -1
        && getLine(x, y, i, 2) == choice && getLine(x, y, i, 3) == -1)
        score1[x][y] += 200;

    //眠二C-2  210*10
    if (getLine(x, y, i, 1) == choice && getLine(x, y, i, 2) == -1 && getLine(x, y, i, -1) == -1
        && getLine(x, y, i, -2) == choice && getLine(x, y, i, -3) == 1 - choice)
        score1[x][y] += 200;

    //眠二C-3  21001*
    if (getLine(x, y, i, 1) == -1 && getLine(x, y, i, -1) == -1 && getLine(x, y, i, -2) == choice
        && getLine(x, y, i, -3) == choice && getLine(x, y, i, -4) == 1 - choice)
        score1[x][y] += 200;

    //眠二D-1  21*001
    if (getLine(x, y, i, -2) == 1 - choice && getLine(x, y, i, -1) == choice && getLine(x, y, i, 1) == -1
        && getLine(x, y, i, 2) == -1 && getLine(x, y, i, 3) == choice)
        score1[x][y] += 100;

    //眠二D-2  210*01
    if (getLine(x, y, i, -3) == 1 - choice && getLine(x, y, i, -1) == -1 && getLine(x, y, i, -2) == choice
        && getLine(x, y, i, 1) == -1 && getLine(x, y, i, 2) == choice)
        score1[x][y] += 100;
}
if (numoftwo >= 2) score1[x][y] += 2000;
return score1[x][y];
```

iv. How to evaluate 8 directions

As we all know, there are four kinds of rows in a Gomoku game so if we need to value a position, we need to value it in eight different directions. The getLine() function can help us get the coordinate in eight directions in a easy way and we can choose the distance we want.

// x , y means the current position

// i is the direction

// j is the relative distance

```
int ChessBoard::getLine(int x, int y, int i, int j)
{
    int current;
    //i:方向 j:相对位置
    switch (i)
    {
    case 1://左
        x = x - j;
        break;
    case 2://右
        x = x + j;
        break;
    case 3://上
        y = y + j;
        break;
    case 4://下
        y = y - j;
        break;
```

```
    case 5://右上
        x = x + j;
        y = y + j;
        break;
    case 6://右下
        x = x + j;
        y = y - j;
        break;
    case 7://左上
        x = x - j;
        y = y + j;
        break;
    case 8://左下
        x = x - j;
        y = y - j;
        break;
    }
    if (x < 0 || y < 0 || x > 14 || y > 14) return -2;
    current = board[x][y];
    return current;
}
```

v. The winning function

If the evaluation process detective a five-in-a-row situation, the score of that position will be -500. And if the winning function detective a score of -500, then the game will come to an end. This function will not be used when the program is put on the Botzone.

```
int ChessBoard::Winning()
{
    for (int i = 0; i < 15; i++)
    {
        for (int j = 0; j < 15; j++)
        {
            if (board[i][j] == -1)
            {
                if (EvaluateA(i, j, choice) == -500)
                {
                    return 0;
                }
                else if (EvaluateA(i, j, 1 - choice) == -500)
                {
                    return 1;
                }
            }
        }
    }
    return -1;
}
```

vi. The function that collect all the available position that has neighbors in the distance of 2

Based on experience, it is usually a waste of time to evaluate the positions that have no neighbors nearby. Therefore, I decided to evaluate only those positions that has neighbors in the distance of two in order to save the time of searching. Basically it just uses the getLine() function to search whether there is a black or white stone in the distance of two.

However, when applying this function on the Botzone, there seems to be a problem in the later period of the game, so I decided to use it in the future search algorithm instead of the current version.

```
int ChessBoard::gen()
{
    int k = 0;
    for (int i = 0; i < 225; i++)
    {
        move[i][0] = -1;
        move[i][1] = -1;
    }
    for (int x = 0; x < 15; x++)
    {
        for (int y = 0; y < 15; y++)
        {
            if (board[x][y] == -1)
            {
                for (int i = 1; i <= 8; i++)
                {
                    if ( getLine(x, y, i, 1) == 0 || getLine(x, y, i, 1) == 1
                        || getLine(x, y, i, 2) == 0 || getLine(x, y, i, 2) == 1)
                    {
                        move[k][0] = x;
                        move[k][1] = y;
                        k++;
                        break;
                    }
                }
            }
        }
    }
    return k;
}
```

vii. The function that makes the next move

This is the main function in this version of design that makes the next move. It evaluates the attack potential and the defense potential of each position and picks out the position which has the max attack score and the position that has the max defense potential. Compare these two scores and choose the bigger one to make the move. If two positions have the same attack score, choose the one that has a bigger defense score in it to make the move.

```
int k = 0;
for (int i = 0; i < 15; i++)
{
    for (int j = 0; j < 15; j++)
    {

        if (board[i][j] == -1)
        {
            move1[k][0] = i;
            move1[k][1] = j;
            k++;
        }
    }
}
int maxA = 0, maxD = 0; //最大进攻值和防守值
int x1 = 0, y1 = 0; ;   //进攻点
int x2 = 0, y2 = 0;     //防守点
for (int i = 0; i<k; i++)
{
    if (EvaluateA(move1[i][0], move1[i][1], choice) > maxA)
    {
        maxA = EvaluateA(move1[i][0], move1[i][1], choice);
        x1 = move1[i][0];
        y1 = move1[i][1];
    }
    else if (EvaluateA(move1[i][0], move1[i][1], choice) == maxA
        && EvaluateD(move1[i][0], move1[i][1], 1 - choice) > EvaluateD(x1, y1, 1 - choice))

    {
        x1 = move1[i][0];
        y1 = move1[i][1];
    }
```

```
    if (EvaluateD(move1[i][0], move1[i][1], 1 - choice) > maxD)
    {
        maxD = EvaluateD(move1[i][0], move1[i][1], 1 - choice);
        x2 = move1[i][0];
        y2 = move1[i][1];
    }
}
if (maxA >= maxD)
{
    board[x1][y1] = choice;
}
else if (maxD > maxA)
{
    board[x2][y2] = choice;
}
```