Computer and Information Science; Vol. 14, No. 4; 2021
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Six-in-a-Row Artificial Intelligence Design and Its Implementation Based on Java

Liujue Zhang[1]

[1] FuZhou University, Maynooth University, FuZhou, FuJian, 350000 China

Correspondence: Liujue Zhang, FuZhou University, Maynooth University, FuZhou, FuJian, 350000 China.

Received: August 2, 2021 Accepted: October 7, 2021 Online Published: October 12, 2021

doi:10.5539/cis.v14n4p47 URL: https://doi.org/10.5539/cis.v14n4p47

Abstract

We can use java language to design the game of chess on eclipse platform and design a new rule for it : “Six in a Row”, this game is more complex than Five-in-a-Row,and we can add a function of Man-Machine Battle in the program.This paper which based on the practical programming experience introduce the algorithm of the above game. The basic mentality of the algorithm is creating a scoring table and all points of the chessboard are scored by using ergodic method.Finally, the machine will placing the chess in the point which have the highest score.This algorithm is clearly operable, and the machine has a high rate of winning when play with human.

Keywords: Six-in-a-Row, artificial intelligence, design and implementation, Java

Foreword

“In 1952, Alan Mathison Turing once said: Playing chess is a very abstract activity, and it’s a field of intelligence that machine can compete with human.”(Li Hao,2020)“Chess game”is a competitive two-player board game, classified as zero-sum and perfect information (meaning one player win is the other's loss and both players see everything that goes on in the game). “Chess game” is one of the most popular games worldwide, played in clubs, tournaments, by correspondence, on the Internet or informally, and we choose to design a six-in-a-row chess game.

“Computer game,also known as machine game,is,the product of the combination of game theory and computer technology,and is an important research direction in the field of artificial intelligence.”(Liu Rui,2012)AI has been a key direction of people’s research in the past few years, and it has a relatively important position in people’s production and life. In the game industry, the human - computer battle is a mode which is used widely. Game manufacturers develop human-computer battle in their production.It can reduce the waiting time of the user due to the insufficient number of human players,and increase the fun and confrontation of the game.

The AI algorithm in the game has a great influence on the fun and playable of the game.A good game AI should have a certain level of gameplay.The programmer should design the algorithm according to the game rules,difficulty,and collected battle data,

1. Introduction of the Chess Game

The chess is a game played by two people. The rule of the game which we introduced is very similar to Gobang. Players should choose to play the black or white piece. The first player moves his piece on the board, the other player then moves, and the two players take turns on the board. Once there are six pieces which have same color that has formed a line in the horizontal, vertical, or diagonal directions, then the player who charges above color piece wins.

The rule of the chess game with rule”six in a row” is simple and fair.The first player has no advantage,its complexity is higher than Gobang,and the game process is more interesting.

2. Introduction of Algorithm

2.1 Overall Introduction

The idea of the algorithm is that the machine scores points on the board at any time. The score is based on the score table.The score table is based on the data of everyone's battle and the six-piece chess design.It takes both offensive and defensive situations into account.The offense has a higher priority.

2.2 Specific Ideas

2.2.1 Set Up Chessboard

A chessboard with a size of 19*19 was drawn using the AWT component in java, and the function buttons were drawn at the same time. We use the filling method to draw the chess pieces.

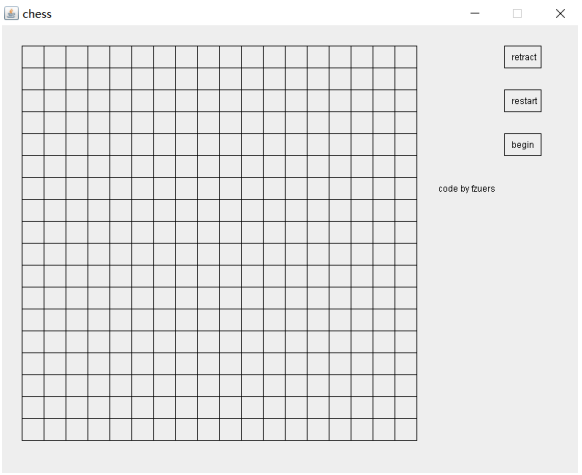

Figure 1. Interface

2.2.2 Judge Method

For judging rules, take horizontal for example,we first set a variable,named “trans”to temporarily store how many chess pieces in same color are in a row.

From the rule we know that, after a chess piece is placed, we should judge from the left 5 squares and right 5 squares of the current position. But the prerequisite is that the chess piece within the bounds.So we firstly write a method to judge whether the chess piece is out of bounds.If so, the cycle ends and the next cycle begins. If there is an adjacent chess piece of the same color, the variable "trans" will increase by 1.If "trans" equals to 6, it means that one side has won. Then it will return the color of the winner.If “trans”is less than 6,(it means that there is one or more chess piece of the other color adjacent to it,which is invalid) the "trans" will reset to zero.For vertical,it is similar to the transverse direction, but it is checked in the Y direction.The diagonal are the same——once we know the correspondence of the abscissa and ordinate.We can easily find the exact expression.

The codes of Judge method:

```
private int checkVictory(int x,int y,int var) {
    int trans = 0;
    for(int i=x-5;i<x+6;i++) {
        if(i<0||i>=this.x) {
            continue;
        }
        if(core[i][y]==var) {
            trans++;
        }
        else {
            trans=0;
        }
        if(trans==6) {
            return(var);
```

```
            }
        }
        int longitudinal=0;
        for(int i=y-5;i<y+6;i++) {
            if(i<0||i>=this.y) {
                continue;
            }
            if(core[x][i]==var) {
                longitudinal++;
            }
            else {
                longitudinal=0;
            }
            if(longitudinal==6) {
                return(var);
            }
        }
        int leftUPToDown=0;
        for(int i=x-5,j=y+5;i<x+6&&j>y-6;i++,j--) {
            if(i<0||i>=this.x||j<0||j>=this.y) {
                continue;
            }
            if(core[i][j]==var) {
                leftUPToDown++;
            }else {
                leftUPToDown=0;
            }
            if(leftUPToDown==6) return(var);
        }
        int leftDownToUP=0;
        for(int i=x+5,j=y+5;i>x-6&&j>y-6;i--,j--) {
            if(i<0||i>=this.x||j<0||j>=this.y) {
                continue;
            }
            if(core[i][j]==var) {
                leftDownToUP++;
            }else {
                leftDownToUP=0;
            }
            if(leftDownToUP==6) return(var);
        }
        return(0);
}
```

2.2.3 Create Array

The keyboard size will be 19*19 which is larger than the normal Five-in-a-row chess game. Create a matrix chessboard[19][19], and we use the row of matrix to represent the abscissa, column represents ordinate, and the element of the matrix represents the piece,2 represents black piece,1 represent white piece,0 represent blank.For example ,there is a black piece on the point which signed in the picture,its ordinate is 6,abscissa is 5,then chessboard[6][5]=2.

The codes of Creating array:

```
int[][] core;
public void original(int x,int y) {
          for(int i=0;i<x;i++) {
          for(int j=0;j<y;j++) {
         core[i][j]=0;
     }
     }
     }
```

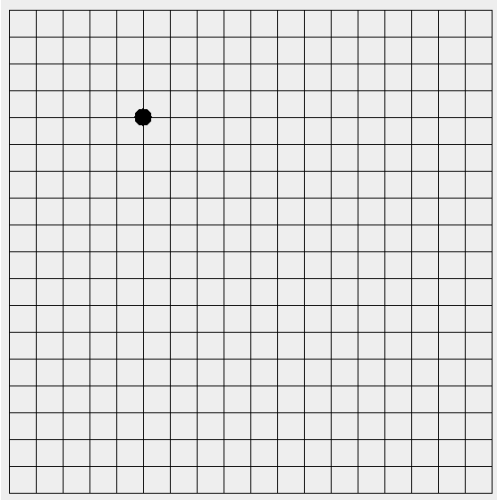

Figure 2. Sample picture

2.2.4 Divide Chessboard into Groups

Divide the chessboard into groups which contain six points in three different directions: transverse,Portrait and Oblique.

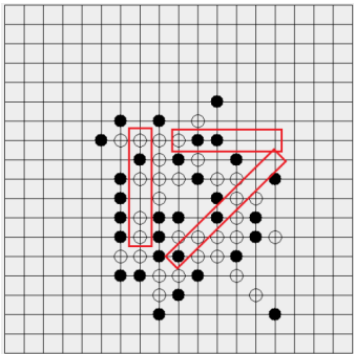

Figure 3. Dividing groups

2.2.5 Traversing the Chessboard

In a human-computer battle, after the player plays chess, the computer judges several six-tuples in the horizontal, vertical and oblique directions, counts the number of black and white chess in each six-tuple, and calculates it as six yuan according to the scoring table. Points are assigned to each point in the group.

The codes of Traversing:

```
public int searchLocation(){
        int[][] chessboard=core.getCore();
        int maxScore=-1000;
        int goalX=-1;
        int goalY=-1;
        for(int i=0;i<19;i++){
            for(int j=0;j<19;j++){
                score[i][j]=0;
            }
        }
        int humanNum=0;
        int machineNum=0;
        int tupleScoreTmp=0;
        for(int i=0;i<19;i++){
            for(int j=0;j<14;j++){
                int k=j;
                while(k<j+6){
                    if(chessboard[i][k]==1) {
                        machineNum++;
                    }
                    if(chessboard[i][k]==2) {
                        humanNum++;
                    }
                    k++;
                }
                tupleScoreTmp=tupleScore(humanNum, machineNum);
                for(k=j;k<j+6;k++){
                    score[i][k]+=tupleScoreTmp;
                }
                humanNum=0;
                machineNum=0;
                tupleScoreTmp=0;
            }
        }
        for(int i=0;i<19;i++){
            for(int j=0;j<14;j++){
                int k=j;
                while(k<j+6){
                    if(chessboard[k][i]==1) {
```

```
                    machineNum++;
                }
                if(chessboard[k][i]==2) {
                    humanNum++;
                }
                k++;
            }
            tupleScoreTmp = tupleScore(humanNum, machineNum);
            for(k=j;k<j+6;k++){
                score[k][i]+=tupleScoreTmp;
            }
            humanNum=0;
            machineNum=0;
            tupleScoreTmp=0;
        }
    }
    for(int i=18;i>=5;i--){
        for(int k=i,j=0;j<19&&k>=0;j++,k--){
            int m=k;
            int n=j;
            while(m>k-6&&k-6>=-1){
                if(chessboard[m][n]==1) {
                    machineNum++;
                }
                if(chessboard[m][n]==2) {
                    humanNum++;
                }
                m--;
                n++;
            }
            if(m==k-6){
                tupleScoreTmp=tupleScore(humanNum,machineNum);
                for(m=k,n=j;m>k-6;m--,n++){
                    score[m][n]+=tupleScoreTmp;
                }
            }
            humanNum=0;
            machineNum=0;
            tupleScoreTmp=0;
}
    }
    for(int i=1;i<19;i++){
        for(int k=i,j=18;j>=0&&k<19;j--,k++){
```

```
            int m=k;
            int n=j;
            while(m<k+6&&k+6<=19){
                if(chessboard[n][m]==1) {
                    machineNum++;
                }
                if(chessboard[n][m]==2) {
                    humanNum++;
                }
                m++;
                n--;
            }
            if(m==k+6){
                tupleScoreTmp=tupleScore(humanNum, machineNum);
                for(m=k,n=j;m<k+6;m++,n--){
                    score[n][m]+=tupleScoreTmp;
                }
            }
            humanNum=0;
            machineNum=0;
            tupleScoreTmp=0;
        }
    }
    for(int i=0;i<14;i++){
        for(int k=i,j=0;j<19&&k<13;j++,k++){
            int m=k;
            int n=j;
            while(m<k+6&&k+6<=19){
                if(chessboard[m][n]==1) {
                    machineNum++;
                }
                if(chessboard[m][n]==2) {
                    humanNum++;
                }
                m++;
                n++;
            }
            if(m==k+6){
                tupleScoreTmp = tupleScore(humanNum, machineNum);
                for(m=k,n=j;m<k+6;m++,n++){
                    score[m][n]+=tupleScoreTmp;
                }
            }
```

```
                humanNum=0;
                machineNum=0;
                tupleScoreTmp=0;
            }
        }
        for(int i=1;i<14;i++){
            for(int k=i,j=0;j<19&&k<19;j++,k++){
                int m=k;
                int n=j;
                while(m<k+6&&k+6<=19){
                    if(chessboard[n][m]==1) {
                        machineNum++;
                    }
                    if(chessboard[n][m]==2) {
                        humanNum++;
                    }
                    m++;
                    n++;
                }
                if(m==k+6){
                    tupleScoreTmp=tupleScore(humanNum,machineNum);
                    for(m=k,n=j;m<k+6;m++,n++){
                        score[n][m]+=tupleScoreTmp;
                    }
                }
                humanNum=0;
                machineNum=0;
                tupleScoreTmp=0;
            }
        }
```

2.2.6 Machine Make a Move

After traversing the chessboard, the program uses a round-robin algorithm to compare the value of each point and place a place on the point with the largest value.

The codes of moving:

```
for(int i=0;i<19;i++){
            for(int j=0;j<19;j++){
                if(chessboard[i][j]==0&&score[i][j]>maxScore){
                    goalX=i;
                    goalY=j;
                    maxScore=score[i][j];
                }
```

```
            }
}
```

2.3 Set Up the Score Sheet

The algorithm sets the machine's chess style to tend to offensive.

Score sheet:

```
public int tupleScore(int humanNum, int machineNum){
        if(humanNum>0&&machineNum>0) {
            return 0;
        }
        if(humanNum==0&&machineNum==0){
            return 7;
        }
        if(machineNum==1){
            return 35;
        }
        if(machineNum==2){
            return 800;
        }
        if(machineNum==3){
            return 15000;
        }
        if(machineNum==4){
            return 800000;
        }
        if(machineNum==5) {
            return 200000000;
        }
        if(humanNum==1){
            return 15;
        }
        if(humanNum==2){
            return 400;
        }
        if(humanNum==3){
            return 1800;
        }
        if(humanNum==4){
            return 100000;
        }
        if(humanNum==5) {
            return 10000000;
        }
```

```
            return -1;
        }
    }
```

3. Conclusion

Basically achieve the required functions of the game,the program can show the chessboard to the player, and can make computer play the game with player,it achieves the final desired effect.

In the future, we will optimize the interface design of the program. Perhaps we can set the chessboard to an antique wood color, thicken the lines, and make the function buttons into physical shapes. We need to improve our algorithm, streamline it, and improve the human-machine algorithm to increase the intelligence of the machine. At the same time, we may be able to increase networking capabilities so that two players can play chess happily even if they are thousands of miles apart.

References

Li, H. (2020). Research and Implementation of Human Computer Game Algorithm Optimization of Gobang (Unpublished master's thesis). *Dalian Maritime University, DaLian, China.*

Liu, R. (2012). Five-in-a-Row Artificial Intelligence Design and Its Implementation (Unpublished master's thesis). *South China University of Technology, GuangZhou, China.*