## “五子棋”程序技术说明文档

姓名： 陈铭海 学号： 20162180198 班别： 软件5班

**一．设计方案**（功能组成、界面组成、算法、数据结构等等）

1. **五子棋的功能组成**

游戏功能有：选择那方先落子，开始游戏，退出游戏，悔棋功能，电脑端人工智能下棋。详细功能如下：

* 程序开珆；
* 屏幕上显示棋盘。
* 用户选择先走或后走，并开始游戏；
* 由先走者自由落子；
* 后走者自由落子，
* 不论电脑是先者还是后者，都按根据当前棋盘的局势判断应该走哪里最好
* 若步骤5或步骤6棋盘新落子的地方，有在纵向，横向或斜向上有与同这个棋子连成5个连续棋子
* 若有同棋子连成5个连续棋子，则结束本次游戏，根据连成5个连续棋子的颜色判断谁胜谁负。
* 若无则重复回到第三步继续往下执行
* 直至棋盘下满或分出胜负
* 实现了电脑方人工智能。

1. **五子棋的界面组成**

界面主要有“开始游戏”，“悔棋”，“退出”按钮，“计算机先行”复选框，背景图，提示文字。

1. **五子棋的算法、数据结构**

五子棋使用了极大极小值搜索算法，实现简单的人工智能，数据结构主要 使用数组。

**二．开发过程**（如何设计、如何实现、如何检查系统功能）

1. **五子棋的设计**

五子棋不仅能增强思维能力，提高智力，而且富含哲理，有助于修身养性。五子棋既有现代休闲的明显特征“短、平、快”，又有古典哲学的高深学问“阴阳易理”；它既有简单易学的特性，为人民群众所喜闻乐见，又有深奥的技巧和高水平的国际性比赛；它的棋文化源渊流长，具有东方的神秘和西方的直观；既有“场”的概念，亦有“点”的连接。它是中西文化的交流点，是古今哲理的结晶。

因此本人将五子棋设计为人机对战单机版，并加入人工智能技术，使得一个人也能下五子棋，并且具有一定的难度。

人机对战单机版五子棋，首先需要先设计好UI界面，再设计游戏功能：选择那方先落子，开始游戏，退出游戏，悔棋功能。

**设计流程图为：**

先画主体对话框，并添加相应按钮

添加背景图，画棋盘，并显示棋盘

编写判断赢棋方法，判断是否5子相连

编写棋子类，并添加事件，实现单击下棋

实现悔棋，开始游戏，退出功能

实现电脑端人工智能下棋算法，并实现计算机根据算法自动落子

结束

1. **五子棋的实现**

（1）.主类：创建对话框并添加相应组件

public class Five extends JFrame{

private static final long serialVersionUID = 1L;

private JToolBar toolbar;

private JButton startButton,backButton,exitButton;

private ChessBoard boardPanel;

JLabel msg;

JCheckBox computerFirst;

public Five() {

super("五子棋人机对战");

toolbar=new JToolBar();

startButton=new JButton("开始游戏");

backButton=new JButton("悔棋");

exitButton=new JButton("退出");

boardPanel=new ChessBoard(this);

msg=new JLabel("新的游戏未开始!");

computerFirst=new JCheckBox("计算机先下子");

toolbar.add(startButton);

toolbar.add(backButton);

toolbar.add(exitButton);

toolbar.add(computerFirst);

ActionMonitor monitor=new ActionMonitor();

startButton.addActionListener(monitor);

backButton.addActionListener(monitor);

exitButton.addActionListener(monitor);

this.add(toolbar, BorderLayout.NORTH);

this.add(boardPanel,BorderLayout.CENTER);

this.add(msg, BorderLayout.SOUTH);

this.setLocation(350, 20);

this.pack();

this.setResizable(true);

this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

this.setVisible(true);

}

class ActionMonitor implements ActionListener{

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

if(e.getSource()==startButton) {

boardPanel.restartGame();

}

else if(e.getSource()==backButton) {

boardPanel.goback();

}

else if(e.getSource()==exitButton) {

System.exit(0);

}

}

}

public static void main(String[]args) {

new Five();

}

}

1. .添加背景图，画棋盘，并显示棋盘

public class ChessBoard extends JPanel {

private static final long serialVersionUID = 1L;

public static final int Margin=30;

public static final int Span=30;

public static final int Rows=18;

public static final int Cols=18;

private Five five;

Image img;

List<Chess> chessList;

int chessCount;

int computerColor;

int [][] boardStatus;

//保护矩形区域变量

int left;

int top;

int right;

int bottom;

boolean isComputerGo;

boolean isGamming=false;

boolean isBlack=true;

public ChessBoard(Five five) {

img=Toolkit.getDefaultToolkit().getImage("image/background.jpg");

this.addMouseListener(new MouseMonitor());

this.addMouseMotionListener(new MouseMotionMonitor());

this.chessCount=0;

this.chessList=new ArrayList<Chess>();

this.five=five;

this.boardStatus=new int[Cols+1][Rows+1];

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

for(int j=0;j<Rows;j++)

boardStatus[i][j]=0;

}

}

public void paintComponent(Graphics g) {

super.paintComponent(g);

g.drawImage(img, 0, 0, this);

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

g.drawLine(Margin, Margin+i\*Span, Margin+Cols\*Span, Margin+i\*Span);

}

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

g.drawLine(Margin+i\*Span, Margin, Margin+i\*Span, Margin+Rows\*Span);

}

g.fillRect(Margin+3\*Span-2, Margin+3\*Span-2, 5, 5);

g.fillRect(Margin+(Cols/2)\*Span-2, Margin+3\*Span-2, 5, 5);

g.fillRect(Margin+(Cols-3)\*Span-2, Margin+3\*Span-2, 5, 5);

g.fillRect(Margin+3\*Span-2, Margin+(Rows/2)\*Span-2, 5, 5);

g.fillRect(Margin+(Cols/2)\*Span-2, Margin+(Rows/2)\*Span-2, 5, 5);

g.fillRect(Margin+(Cols-3)\*Span-2, Margin+(Rows/2)\*Span-2, 5, 5);

g.fillRect(Margin+3\*Span-2, Margin+(Rows-3)\*Span-2, 5, 5);

g.fillRect(Margin+(Cols/2)\*Span-2, Margin+(Rows-3)\*Span-2, 5, 5);

g.fillRect(Margin+(Cols-3)\*Span-2, Margin+(Rows-3)\*Span-2, 5, 5);

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

chessList.get(i).draw(g);

if(i==chessCount-1) {

int xPos=chessList.get(i).getCol()\*Span+Margin;

int yPos=chessList.get(i).getRow()\*Span+Margin;

g.setColor(Color.RED);

g.drawRect(xPos-Chess.Diameter/2, yPos-Chess.Diameter/2, Chess.Diameter, Chess.Diameter);

}

}

}

public Dimension getPreferredSize() {

return new Dimension(Margin\*2+Span\*Cols,Margin\*2+Span\*Rows);

}

private boolean hasChess(int col,int row) {

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

Chess ch=chessList.get(i);

if(ch!=null && ch.getCol()==col && ch.getRow()==row)

return true;

}

return false;

}

private boolean hasChess(int col,int row,Color color) {

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

Chess ch=chessList.get(i);

if(ch!=null && ch.getCol()==col && ch.getRow()==row && ch.getColor()==color)

return true;

}

return false;

}

private int searchX(int col,int row , Color c,int direct,int i,int continueCount) {

for(int x=col+direct;x\*direct<=i;x+=direct) {

if(hasChess(x,row,c)) continueCount++;

else break;

}

return continueCount;

}

private int searchY(int col,int row , Color c,int direct,int i,int continueCount) {

for(int y=row+direct;y\*direct<=i;y+=direct) {

if(hasChess(col,y,c)) continueCount++;

else break;

}

return continueCount;

}

private boolean isWin(int col,int row) {

int continueCount=1;

Color c=isBlack ? Color.BLACK:Color.WHITE;

//横向搜索

continueCount=searchX(col,row,c,1,Cols,continueCount)+searchX(col,row,c,-1,0,0);

if(continueCount>=5) return true;

else continueCount=1;

//纵向搜索

continueCount=searchY(col,row,c,1,Rows,continueCount)+searchY(col,row,c,-1,0,0);

if(continueCount>=5) return true;

else continueCount=1;

//右上到左下

for(int x=col+1,y=row-1;y>=0 && x<=Cols;x++,y--){

if(hasChess(x,y,c)) continueCount++;

else break;

}

for(int x=col-1,y=row+1;x>=0 && y<=Rows;x--,y++){

if(hasChess(x,y,c)) continueCount++;

else break;

}

if(continueCount>=5) return true;

else continueCount=1;

//左上到右下

for(int x=col-1,y=row-1;x>=0 && y>=0;x--,y--){

if(hasChess(x,y,c)) continueCount++;

else break;

}

for(int x=col+1,y=row+1;x<=Cols && y<=Rows;x++,y++){

if(hasChess(x,y,c)) continueCount++;

else break;

}

if(continueCount>=5) return true;

else return false;

}

public void restartGame() {

chessList.clear();

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

for(int j=0;j<Rows;j++)

boardStatus[i][j]=0;

}

left=7;

top=7;

right=7;

bottom=7;

isBlack=true;

isGamming=true;

isComputerGo=five.computerFirst.isSelected();

computerColor=isComputerGo ? 1:2;

chessCount=0;

if(isComputerGo) {

computerGo();

}

paintComponent(this.getGraphics());

if(isBlack) {

five.msg.setText("请黑棋下子!");

}

else {

five.msg.setText("请白棋下子!");

}

}

private void computerGo() {

Evaluate e=new Evaluate(this);

int pos[]=e.getTheBestPosition();

putChess(pos[0],pos[1],isBlack ? Color.BLACK:Color.WHITE);

}

private void manGo(int col,int row) {

putChess(col,row,isBlack ? Color.BLACK:Color.WHITE);

}

private void putChess(int col, int row, Color color) {

Chess ch=new Chess(ChessBoard.this,col,row,color);

chessList.add(ch);

chessCount++;

boardStatus[col][row]=(color==Color.BLACK) ? 1:2 ;

paintComponent(this.getGraphics());

//更新矩形保护区

if(left>col) left=col;

if(top>row) top=row;

if(right<col) right=col;

if(bottom<row) bottom=row;

if(isWin(col,row)) {

String colorName=isBlack ? "黑棋":"白棋";

String message=String.format("恭喜,%s赢了!", colorName);

JOptionPane.showMessageDialog(ChessBoard.this, message);

isGamming=false;

}

else {

isBlack=!isBlack;

isComputerGo=!isComputerGo;

}

}

public void goback() {

if(isComputerGo || chessCount<2) return;

int i=chessList.get(chessCount-1).getCol();

int j=chessList.get(chessCount-1).getRow();

boardStatus[i][j]=0;

i=chessList.get(chessCount-2).getCol();

j=chessList.get(chessCount-2).getRow();

boardStatus[i][j]=0;

chessList.remove(chessCount-1);

chessList.remove(chessCount-2);

chessCount-=2;

isBlack=!isBlack;

isGamming=true;

paintComponent(this.getGraphics());

if(isBlack) {

five.msg.setText("请黑棋下子!");

}

else {

five.msg.setText("请白棋下子!");

}

}

class MouseMonitor extends MouseAdapter{

public void mousePressed(MouseEvent e) {

if(! isGamming) return ;

if(isComputerGo) return;

int col=(e.getX()-Margin+Span/2)/Span;

int row=(e.getY()-Margin+Span/2)/Span;

if(col<0 || row <0 || col>Cols || row>Rows)

return;

if(hasChess(col,row))

return;

manGo(col,row);

if(!isGamming)

return;

computerGo();

}

}

class MouseMotionMonitor extends MouseMotionAdapter{

public void mouseMoved(MouseEvent e) {

int col=(e.getX()-Margin+Span/2)/Span;

int row=(e.getY()-Margin+Span/2)/Span;

if(col<0 || row <0 || col>Cols || row>Rows ||!isGamming || hasChess(col,row)) {

ChessBoard.this.setCursor(new Cursor(Cursor.DEFAULT\_CURSOR));

}

else {

ChessBoard.this.setCursor(new Cursor(Cursor.HAND\_CURSOR));

}

}

}

}

1. .棋子类

public class Chess {

public static final int Diameter=ChessBoard.Span-2;

private int col;

private int row;

private Color color;

ChessBoard cb;

public Chess(ChessBoard cb, int col, int row, Color color) {

this.col = col;

this.row = row;

this.color=color;

this.cb = cb;

}

public int getCol() {

return col;

}

public int getRow() {

return row;

}

public Color getColor() {

return color;

}

@SuppressWarnings("static-access")

public void draw(Graphics g) {

int xPos=col\*cb.Span+cb.Margin;

int yPos=row\*cb.Span+cb.Margin;

Graphics2D g2d=(Graphics2D) g;

RadialGradientPaint paint=null;

int x=xPos+Diameter/4;

int y=yPos-Diameter/4;

float[] f= {0f,1f};

Color[] c={Color.WHITE,Color.BLACK};

if(color==Color.BLACK) {

paint=new RadialGradientPaint(x,y,Diameter,f,c);

}

else if(color==Color.WHITE) {

paint=new RadialGradientPaint(x,y,Diameter \* 2,f,c);

}

g2d.setPaint(paint);

g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

g2d.setRenderingHint(RenderingHints.KEY\_ALPHA\_INTERPOLATION, RenderingHints.VALUE\_ALPHA\_INTERPOLATION\_DEFAULT);

Ellipse2D e=new Ellipse2D.Float(xPos-Diameter/2, yPos-Diameter/2, Diameter, Diameter);

g2d.fill(e);

}

}

1. .极大极小值搜索算法(由于太多，仅显示估值代码)

int value = 0;

int i, j, k;

int[] line = new int[cb.Cols + 1];

// 对每一行估值

for (j = 0; j <= cb.Rows; j++) {

for (i = 0; i <= cb.Cols; i++) {

line[i] = cb.boardStatus[i][j];

}

value += evaluateLine(line, cb.Cols + 1, 1);

value -= evaluateLine(line, cb.Cols + 1, 2);

}

// 对每一列估值

for (i = 0; i <= cb.Cols; i++) {

for (j = 0; j < cb.Rows; j++) {

line[j] = cb.boardStatus[i][j];

}

value += evaluateLine(line, cb.Rows + 1, 1);

value -= evaluateLine(line, cb.Rows + 1, 2);

}

// 左下到右上斜线估值

for (j = 4; j <= cb.Rows; j++) {

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

line[k] = cb.boardStatus[k][j - k];

}

value += evaluateLine(line, j + 1, 1);

value -= evaluateLine(line, j + 1, 2);

}

for (j = 1; j <= cb.Rows - 4; j++) {

for (k = 0; k <= cb.Cols - j; k++) {

line[k] = cb.boardStatus[k + j][cb.Rows - k];

}

value += evaluateLine(line, cb.Rows + 1 - j, 1);

value -= evaluateLine(line, cb.Rows + 1 - j, 2);

}

// 左上到右下斜线估值

for (j = 0; j <= cb.Rows - 4; j++) {

for (k = 0; k <= cb.Rows - j; k++) {

line[k] = cb.boardStatus[k][k + j];

}

value += evaluateLine(line, cb.Rows + 1 - j, 1);

value -= evaluateLine(line, cb.Rows + 1 - j, 2);

}

for (i=1;i<=cb.Cols-4;i++) {

for (k = 0; k <= cb.Rows - i; k++) {

line[k] = cb.boardStatus[k+i][k];

}

value += evaluateLine(line, cb.Rows + 1 - i, 1);

value -= evaluateLine(line, cb.Rows + 1 - i, 2);

}

if (cb.computerColor == 1) {

return value;

} else {

return -value;

}

1. **五子棋的测试**

主要是通过黑盒测试，经过测试，发现如果极大极小值搜索算法的深度过大，电脑端下棋响应时间会很久。测试如下：

**三．技术讨论**（存在问题、改进方向）

1. **存在问题**

在开发的过程中，存在一些问题：没有保存棋谱的功能，无法提供棋谱欣赏功能，也没有联网功能。人工智能的算法的搜索深度过小，可能不太“智能”，但深度太大又会很费时间，交互性就会变差。

1. **改进方向**

可以加上数据库的技术，将数据持久化，也可以使用文件来序列化对象。可是使用Socket编程来加上联网功能。对应人工智能的极大极小值搜索算法，可以加上剪枝算法来减少搜索时间从而实现更加“智能”。