## DrawGobang.java

package gobang;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

public class DrawGobang {

public static void main(String[] args) {

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.add(new DrawChessBoard());

frame.setSize(800, 800);

frame.setVisible(true);

}

}

class DrawChessBoard extends JPanel implements MouseListener,ActionListener

{

static final int sx = 80;//游戏区域方块的起始横坐标

static final int sy = 40;//游戏区域方块的起始纵坐标

static final int w = 40;//每个小方格的边长

static final int rw = 560;//游戏区域方块的边长

static final int Rows = 15;//行列数

int chessmanWidth = 30; //棋子直径

static int[][] chessmanArray = new int[Rows][Rows];//棋盘状态，0无子，1白子，2黑子

static final int computer = 1;

static final int player = 0;

int user = -1;//该谁下，1电脑，0玩家

static int color = 2;//该落什么颜色的棋子，1白，2黑

JButton computerBtn=new JButton("电脑先手");

JButton playerBtn=new JButton("玩家先手");

JButton comformBtn = new JButton();

public DrawChessBoard()

{

addMouseListener(this);

add(computerBtn);

add(playerBtn);

add(comformBtn);

computerBtn.addActionListener(this);

playerBtn.addActionListener(this);

comformBtn.addActionListener(this);

comformBtn.setVisible(false);

}

/\*\*画棋盘\*\*/

void DrawBoard(Graphics g)

{

g.setColor(new Color(230, 189, 128));

g.fillRect(sx, sy, rw, rw);

g.setColor(Color.RED);

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

g.drawLine(sx, sy + i \* w, sx + rw, sy + i \* w);

g.drawLine(sx + i \* w, sy, sx + i \* w, sy + rw);

}

}

/\*\*画棋子\*\*/

void DrawChessMan(Graphics g)

{

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

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

if (chessmanArray[j][i]==1) {

g.setColor(Color.WHITE);

g.fillOval(sx + w \* i - chessmanWidth/2, sy + w \* j - chessmanWidth/2, chessmanWidth, chessmanWidth);

}

else if (chessmanArray[j][i]==2) {

g.setColor(Color.BLACK);

g.fillOval(sx + w \* i - chessmanWidth/2, sy + w \* j - chessmanWidth/2, chessmanWidth, chessmanWidth);

}

}

}

}

@Override

public void paint(Graphics g) {

// TODO Auto-generated method stub

super.paint(g);

DrawBoard(g);

DrawChessMan(g);

}

@Override

public void mouseClicked(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

if (user!=player) {

System.out.println("user:"+user);

return;

}

int x = e.getX();

int y = e.getY();

int px = (x - (sx - chessmanWidth / 2)) / w;

int py = (y - (sy - chessmanWidth / 2)) / w;

if (px >= 0 && px < Rows && py >= 0 && py < Rows

&& (x - (sx - chessmanWidth / 2)) % w <= chessmanWidth

&& (y - (sy - chessmanWidth / 2)) % w <= chessmanWidth

&& chessmanArray[py][px]==0)

{

chessmanArray[py][px] = color;

System.out.println("color1:"+color+" y:"+py + " x:"+px);

if (Evaluate.isEnd(new Point(px, py), color)) {

comformBtn.setVisible(true);

comformBtn.setText("玩家获胜");

user = -1;

}

else {

color = 3 - color;

user = 1 - user;

Point p = Evaluate.computerGo();

chessmanArray[p.y][p.x] = color;

System.out.println("color2:"+color+" y:"+p.y + " x:"+p.x);

if (Evaluate.isEnd(p, color)) {

comformBtn.setVisible(true);

comformBtn.setText("电脑获胜");

user = -1;

}

else {

color = 3 - color;

user = 1 - user;

}

}

repaint();

}

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

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

color = 2;

computerBtn.setVisible(false);

playerBtn.setVisible(false);

chessmanArray = new int[Rows][Rows];

chessmanArray[7][7] = color;

user = player;

color = 1;

repaint();

}

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

user = player;

color = 2;

computerBtn.setVisible(false);

playerBtn.setVisible(false);

chessmanArray = new int[Rows][Rows];

repaint();

}

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

computerBtn.setVisible(true);

playerBtn.setVisible(true);

comformBtn.setVisible(false);

}

}

}

### Evaluate.java

package gobang;

class Dir

{

int x;

int y;

public Dir(int x,int y) {

// TODO Auto-generated constructor stub

this.x = x;

this.y = y;

}

}

class Point

{

int x;

int y;

public Point(int x,int y) {

// TODO Auto-generated constructor stub

this.x = x;

this.y = y;

}

public Point newPoint(int len, Dir dir)

{

return new Point(x + dir.x \* len, y + dir.y \* len);

}

}

public class Evaluate {

static final Dir d1 = new Dir(0, 1); //横

static final Dir d2 = new Dir(1, 0); //竖

static final Dir d3 = new Dir(1, -1); //撇

static final Dir d4 = new Dir(1, 1); //捺

public static boolean isInBoard(Point p) {

return p.x >= 0 && p.x < DrawChessBoard.Rows

&& p.y >= 0 && p.y < DrawChessBoard.Rows;

}

public static boolean isEnd(Point p,int color)

{

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

Dir d = null;

int count = 0;

switch(i){

case 1:

d = d1;

break;

case 2:

d = d2;

break;

case 3:

d = d3;

break;

case 4:

d = d4;

break;

}

for(int j = -4; j <= 4; j ++){

Point p1 = p.newPoint(j,d);

if(isInBoard(p1) && DrawChessBoard.chessmanArray[p1.y][p1.x] == color){

count ++;

}

else{

count = 0;

}

if(count == 5){

return true;

}

}

}

return false;

}

public static Point computerGo() {

int computer = DrawChessBoard.color;

int player = 3 - DrawChessBoard.color;

Point attackPoint = null, defendPoint = null;

int attScore = 0, b1 = 0, defScore = 0, b2 = 0;

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

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

if(DrawChessBoard.chessmanArray[j][i] != 0){

continue;

}

Point cur = new Point(i, j);

//寻找最佳进攻点

int m1 = getScore(cur, computer);

int m2 = getScore(cur, player);

if(m1 > attScore || (m1 == attScore && m2 > b1)){

attackPoint = cur;

attScore = m1;

b1 = m2;

}

//寻找最佳防守点

int n1 = getScore(cur, player);

int n2 = getScore(cur, computer);

if(n1 > defScore || (n1 == defScore && n2 > b2)){

defendPoint = cur;

defScore = n1;

b2 = n2;

}

}

}

if(attScore >= defScore){

return attackPoint;

}

else{

return defendPoint;

}

}

/\*判断棋型，评估分数\*/

public static int getScore(Point p,int color)

{

int win5 = 0, alive4 = 0, die4 = 0, ddie4 = 0, alive3 = 0,

dalive3 = 0, die3 = 0, alive2 = 0, dalive2 = 0, die2 = 0, nothing = 0;

int opp = 3 - color;

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

Dir d = null;

switch(j){

case 1:

d = d1;

break;

case 2:

d = d2;

break;

case 3:

d = d3;

break;

case 4:

d = d4;

break;

}

int count = 1;

Point le, ri, p1;

int[] left = new int[5], right = new int[5];

//计算连子数量

p1 = p.newPoint(-1, d);

le = p;

while(isInBoard(p1) && DrawChessBoard.chessmanArray[p1.y][p1.x] == color){

le = p1;

p1 = p1.newPoint(-1,d);

count++;

}

p1 = p.newPoint(1,d);

ri = p;

while(isInBoard(p1) && DrawChessBoard.chessmanArray[p1.y][p1.x] == color){

ri = p1;

p1 = p1.newPoint(1,d);

count++;

}

//计算连子左右棋子情况

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

p1 = le.newPoint(-i,d);

if(isInBoard(p1)){

left[i] = DrawChessBoard.chessmanArray[p1.y][p1.x];

}

else{

left[i] = opp;

}

p1 = ri.newPoint(i,d);

if(isInBoard(p1)){

right[i] = DrawChessBoard.chessmanArray[p1.y][p1.x];

}

else{

right[i] = opp;

}

}

//具体棋型判断

if(count == 5){ //连五

win5++;

}

else if(count == 4){

if(left[1] == 0 && right[1] == 0){ //活四

alive4++;

}

else if(left[1] == 0 || right[1] == 0){ //死四

die4++;

}

else{ //nothing

nothing++;

}

}

else if(count == 3){

if((left[1] == 0 && left[2] == color) || (right[1] == 0 && right[2] == color)){//冲四

ddie4 ++;

}

else if(left[1] == 0 && right[1] == 0 && (left[2] == 0 || right[2] == 0)){//活三

alive3 ++;

}

else if((left[1] == 0 && left[2] == 0) || (right[1] == 0 && right[2] == 0)){//死三

die3 ++;

}

else if(left[1] == 0 && right[1] == 0){//死三

die3 ++;

}

else{//nothing

nothing ++;

}

}

else if(count == 2){

if((left[1] == 0 && left[2] == color && left[3] == color) &&

(right[1] == 0 && right[2] == color && right[3] == color)){//冲四

ddie4 ++;

}

else if(left[1] == 0 && right[1] == 0 &&

((left[2] == color && left[3] == 0) || (right[2] == color && right[3] == 0))){//跳活3

dalive3 ++;

}

else if((left[1] == 0 && left[3] == 0 && left[2] == color) ||

(right[1] == 0 && right[3] == 0 && right[2] == color)){//die3

die3 ++;

}

else if((left[1] == 0 && right[1] == 0) &&

(left[2] == color || right[2] == color)){//die3

die3 ++;

}

else if((left[1] == 0 && left[2] == 0 && left[3] == color) ||

(right[1] == 0 && right[2] == 0 && right[3] == color)){//die3

die3 ++;

}

else if((left[1] == 0 && right[1] == 0 && right[2] == 0 && right[3] == 0) ||

(left[1] == 0 && left[2] == 0 && right[1] == 0 && right[2] == 0) ||

(left[1] == 0 && left[2] == 0 && left[3] == 0 && right[1] == 0)){//alive2

alive2 ++;

}

else if((left[1] == 0 && left[2] == 0 && left[3] == 0) ||

(right[1] == 0 && right[2] == 0 && right[3] == 0)){//die2

die2 ++;

}

else{//nothing

nothing ++;

}

}

else if(count == 1){

if((left[1] == 0 && left[2] == color && left[3] == color && left[4] == color) ||

(right[1] == 0 && right[2] == color && right[3] == color && right[4] == color)){//ddie4

ddie4 ++;

}

else if((left[1] == 0 && right[1] == 0) && ((left[2] == color && left[3] == color && left[4] == 0) ||

(right[2] == color && right[3] == color && right[4] == 0))){//dalive3

dalive3 ++;

}

else if((left[1] == 0 && right[1] == 0) &&

((left[2] == color && left[3] == color) || (right[2] == color && right[3] == color))){//die3

die3 ++;

}

else if((left[1] == 0 && left[4] == 0 && left[2] == color && left[3] == color) ||

(right[1] == 0 && right[4] == 0 && right[2] == color && right[3] == color)){//die3

die3 ++;

}

else if((left[1] == 0 && left[2] == 0 && left[3] == color && left[4] == color) ||

(right[1] == 0 && right[2] == 0 && right[3] == color && right[4] == color)){//die3

die3 ++;

}

else if((left[1] == 0 && left[3] == 0 && left[2] == color && left[4] == color) ||

(right[1] == 0 && right[3] == 0 && right[2] == color && right[4] == color)){//die3

die3 ++;

}

else if((left[1] == 0 && right[1] == 0 && right[3] == 0 && right[2] == color) && (left[2] == 0 || right[4] == 0)){//dalive2

dalive2 ++;

}

else if((right[1] == 0 && left[1] == 0 && left[3] == 0 && left[2] == color) &&

(right[2] == 0 || left[4] == 0)){//dalive2

dalive2 ++;

}

else if((left[1] == 0 && right[1] == 0 && right[2] == 0 && right[4] == 0 && right[3] == color) ||

(right[1] == 0 && left[1] == 0 && left[2] == 0 && left[4] == 0 && left[3] == color)){//dalive2

dalive2 ++;

}

else if((left[1] == 0 && left[3] == 0 && left[4] == 0 && left[2] == color) ||

(right[1] == 0 && right[3] == 0 && right[4] == 0 && right[2] == color)){//die2

die2 ++;

}

else if((left[1] == 0 && right[1] == 0 && right[2] == 0 && left[2] == color) ||

(right[1] == 0 && left[1] == 0 && left[2] == 0 && right[2] == color)){//die2

die2 ++;

}

else if((left[1] == 0 && left[2] == 0 && left[4] == 0 && left[3] == color) ||

(right[1] == 0 && right[2] == 0 && right[4] == 0 && right[3] == color)){//die2

die2 ++;

}

else if((left[1] == 0 && left[2] == 0 && right[1] == 0 && left[3] == color) ||

(right[1] == 0 && right[2] == 0 && left[1] == 0 && right[3] == color)){//die2

die2 ++;

}

else if((left[1] == 0 && left[2] == 0 && left[3] == 0 && left[4] == color) ||

(right[1] == 0 && right[2] == 0 && right[3] == 0 && right[4] == color)){//die2

die2 ++;

}

else{//nothing

nothing ++;

}

}

}

//计算分值

if (win5 >= 1)

return 14;//赢5

if (alive4 >= 1 || die4 >= 2 || (die4 >= 1 && alive3 >= 1))

return 13;//活4 双死4 死4活3

if (alive3 >= 2)

return 12;//双活3

if (die3 >= 1 && alive3 >= 1)

return 11;//死3高级活3

if (die4 >= 1)

return 10;//高级死4

if (ddie4 >= 1)

return 9;//低级死4

if (alive3 >= 1)

return 8;//单活3

if (dalive3 >= 1)

return 7;//跳活3

if (alive2 >= 2)

return 6;//双活2

if (alive2 >= 1)

return 5;//活2

if (dalive2 >= 1)

return 4;//低级活2

if (die3 >= 1)

return 3;//死3

if (die2 >= 1)

return 2;//死2

return 1;//没有威胁

}

}