**OTHELLO GUI**

package gui;

import kernel.Kernel;

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

import javax.swing.\*;

@SuppressWarnings("serial")

public class GUI extends JApplet implements ActionListener{

public static final int SIZE = 8;

public static Container cont;

public static JPanel menu, pieces, info, alert;

public static JButton newGame, loadGame, credits, handOver, giveUp, save, backMenu;

public static JButton[][] button = new JButton[SIZE][SIZE];

public static Icon b = new ImageIcon(Kernel.path+ "black.png"), br = new ImageIcon(Kernel.path+ "blackOver.png");

public static Icon w = new ImageIcon(Kernel.path+ "white.png"), wr = new ImageIcon(Kernel.path+ "whiteOver.png");

public static Icon g = new ImageIcon(Kernel.path+ "blank.png");

public static Icon t, tr;

public static JLabel time, pnts, alrt;

public static JList plays;

public static DefaultListModel playsList;

public static int whtPnts=2, blckPnts=2, x, y, moves;

public static long whtTm, blckTm, actTm, rgstrTm, tmTPlWht, tmTPlBlck, tmRstntWht, tmRstntBlck;

public static boolean noLmts[]=new boolean[2], canPlay, sameSong;

public static Thread tmApp;

public static AudioClip sng[] = new AudioClip[4];

public void init(){

sameSong = true;

sng[0] = getAudioClip(getDocumentBase(), Kernel.path+ "inMenu.wav");

sng[1] = getAudioClip(getDocumentBase(), Kernel.path+ "inGameTime.wav");

sng[2] = getAudioClip(getDocumentBase(), Kernel.path+ "inGameNoTime.wav");

sng[3] = getAudioClip(getDocumentBase(), Kernel.path+ "gameOver.wav");

//Cambiar el tamaño de la ventana del programa

setSize(560, 450);

iniciarMenu();

}

//Inicia la interface del menú principal del juego

public void iniciarMenu(){

cambiarCancion(0);

//System.out.println(System.getProperty("user.dir"));

cont = getContentPane();

cont.removeAll();

cont.setLayout(new FlowLayout(FlowLayout.CENTER));

menu = new JPanel();

menu.setLayout(new GridLayout(7,1));

menu.add(new JLabel("Othello MegaGus"));

menu.add(new JLabel(" "));

newGame = new JButton("Juego Nuevo");

newGame.addActionListener(this);

menu.add(newGame);

menu.add(new JLabel(" "));

loadGame = new JButton("Cargar Juego");

loadGame.addActionListener(this);

menu.add(loadGame);

menu.add(new JLabel(" "));

credits = new JButton("Creditos");

credits.addActionListener(this);

menu.add(credits);

cont.add(menu);

}

//Inicia un nuevo juego con el tiempo límite provisto

public void iniciarJuego(long timeWhite, long timeBlack){

noLmts[0] = timeWhite==-1 ? true : false;

noLmts[1] = timeBlack==-1 ? true : false;

if(!sameSong){

cambiarCancion(noLmts[0] ? 2 : 1);

}else{

cambiarCancion(2);

}

whtTm = timeWhite;

blckTm = timeBlack;

rgstrTm = System.currentTimeMillis();

t = w;

tr = wr;

whtPnts = blckPnts = 2;

canPlay = true;

moves = 1;

cont = getContentPane();

cont.removeAll();

cont.setLayout(new FlowLayout(FlowLayout.LEFT));

//Botones de acción adicional

handOver = new JButton("Ceder Turno");

handOver.addActionListener(this);

cont.add(handOver);

giveUp = new JButton("Rendirse");

giveUp.addActionListener(this);

cont.add(giveUp);

save = new JButton("Guardar");

save.addActionListener(this);

cont.add(save);

backMenu = new JButton("Volver al Menú");

backMenu.addActionListener(this);

cont.add(backMenu);

//Tabla para el tablero de juego

pieces = new JPanel();

pieces.setLayout(new GridLayout(SIZE, SIZE));

pieces.setPreferredSize(new Dimension(400,400));

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

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

if((i==SIZE/2-1 && j==SIZE/2-1) || (i==SIZE/2 && j==SIZE/2)){

button[i][j] = new JButton(b);

button[i][j].setRolloverIcon(br);

}else if((i==SIZE/2-1 && j==SIZE/2) || (i==SIZE/2 && j==SIZE/2-1)){

button[i][j] = new JButton(w);

button[i][j].setRolloverIcon(wr);

}else{

button[i][j] = new JButton(g);

button[i][j].setRolloverIcon(w);

}

button[i][j].setBackground(new Color(0,125,15));

button[i][j].addActionListener(this);

pieces.add(button[i][j]);

}

}

cont.add(pieces);

//Lugar para guardar los datos adicionales

info = new JPanel();

info.setLayout(new GridLayout(4,1));

pnts = new JLabel("<html>Puntos:<br>- Blancas:  " +whtPnts+ "<br>- Negras:\t" +blckPnts+ "</html>");

pnts.setToolTipText("Puntos actuales de cada uno de los jugadores!");

info.add(pnts);

String tms = "<html>Tiempo Restante:";

tms += "<br>- Blancas:\t";

tms += noLmts[0] ? "N/A" : (tmTPlWht/60000)+ " mins";

tms += "<br>- Negras:\t";

tms += noLmts[1] ? "N/A" : (tmTPlBlck/60000)+ " mins";

tms += "</html>";

time = new JLabel(tms);

time.setToolTipText("Tiempo restante disponible para cada jugador!");

info.add(time);

alrt = new JLabel("<html><br>Juegan Blancas<br>...</html>");

alrt.setPreferredSize(new Dimension(1,1));

info.add(alrt);

playsList = new DefaultListModel();

playsList.addElement("Equipo [posX, posY]");

plays = new JList(playsList);

plays.setVisibleRowCount(4);

JScrollPane jsp = new JScrollPane(plays);

jsp.setPreferredSize(new Dimension(140, 100));

info.add(jsp);

cont.add(info);

tmApp = new Thread(new RunTime(), "Tiempo");

tmApp.start();

info.updateUI();

}

//Muestra los créditos

public void mostrarCreditos(){

String msg = "<html><a style='font-family:courier new;'>";

msg += "|---------------Créditos xD---------------|<br>";

msg += "|. No hay ................................|<br>";

msg += "|.. Pero los habría ......................|<br>";

msg += "|... Si sobrara tiempo ...................|<br>";

msg += "|.... Pero cómo no sobra .................|<br>";

msg += "|..... Mejor lo gastamos bien ............|<br>";

msg += "|...... xD xD xD .........................|<br>";

msg += "|....... Disfrute del Othello MegaGus ....|<br>";

msg += "|-----------------Fin xD------------------|<br>";

msg += "</a></html>";

alertarGrande(msg);

}

//Pone play a la canción especificada y para las otras xD

public static void cambiarCancion(int id){

for(int i=0; i<sng.length; i++){

if(i != id){

sng[i].stop();

}else{

if(id != 3){

sng[i].loop();

}else{

sng[i].play();

}

}

}

}

//Cambia el turno

public static void cambiarTurno(){

plays.updateUI();

if(moves != 0){

playsList.addElement((moves<10 ? "0" : "")+moves+ ". " +(GUI.t.equals(GUI.b)?"Negras ":"Blancas")+ " [" +x+ ", " +y+ "]");

}

moves++;

if(t.equals(b)){

if(!sameSong){

cambiarCancion(noLmts[0] ? 2 : 1);

}

t = w;

tr = wr;

alrt.setText("<html><br>Juegan Blancas<br>Alertas...</html>");

}else if(t.equals(w)){

if(!sameSong){

cambiarCancion(noLmts[1] ? 2 : 1);

}

t = b;

tr = br;

alrt.setText("<html><br>Juegan Negras<br>Alertas...</html>");

}

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

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

if(button[i][j].getIcon().equals(g)){

button[i][j].setRolloverIcon(t);

}

}

}

alrt.updateUI();

}

//Actualiza la cantidad de fichas de cada jugador

public static void cambiarPuntos(int x, Icon who){

if(who.equals(w)){//El enemigo era el blanco

whtPnts -= x;

blckPnts += x+1;

}else if(who.equals(b)){//El enemigo era el negro

whtPnts += x+1;

blckPnts -= x;

}

pnts.setText("<html>Puntos:<br>- Blancas:\t" +whtPnts+ "<br>- Negras:\t" +blckPnts+ "<html>");

pnts.updateUI();

if(whtPnts+blckPnts==64 || whtPnts==0 || blckPnts==0){

acabaJuego(' ');

}

}

//Cambia el tiempo después de cada vez que se activa un botón

public static void cambiarTiempo(Icon turno){

actTm = System.currentTimeMillis();

if(turno.equals(b) && !noLmts[1]){

blckTm += (actTm - rgstrTm);

tmRstntBlck = tmTPlBlck - blckTm;

if(tmRstntBlck<=0){

acabaJuego('t');

return;

}

}else if(turno.equals(w) && !noLmts[0]){

whtTm += (actTm - rgstrTm);

tmRstntWht = tmTPlWht - whtTm;

if(tmRstntWht<=0){

acabaJuego('t');

return;

}

}else if(turno.equals(g)){

tmRstntWht = tmTPlWht - whtTm;

tmRstntBlck = tmTPlBlck - blckTm;

}

rgstrTm = actTm;

int minWht = (int)(((tmRstntWht-(tmRstntWht%(60000)))/60000)%60);

int secWht = (int)(((tmRstntWht-(tmRstntWht%1000))/1000)%60);

int milWht = (int)(tmRstntWht%1000);

int minBlck = (int)(((tmRstntBlck-(tmRstntBlck%(60000)))/60000)%60);

int secBlck = (int)(((tmRstntBlck-(tmRstntBlck%1000))/1000)%60);

int milBlck = (int)(tmRstntBlck%1000);

String tms = "<html>Tiempo Restante:";

tms += "<br>- Blancas:\t";

tms += noLmts[0] ? "N/A" : minWht+ " : " +(secWht < 10 ? "0" : "") + secWht+ " . " +(milWht<10 ? "00" : (milWht<100 ? "0" : ""))+ milWht;

tms += "<br>- Negras:\t";

tms += noLmts[1] ? "N/A" : minBlck+ " : " +(secBlck < 10 ? "0" : "") + secBlck+ " . " +(milBlck<10 ? "00" : (milBlck<100 ? "0" : ""))+ milBlck;

tms += "</html>";

time.setText(tms);

time.repaint();

}

//Juego Finalizado

public static void acabaJuego(char id){

cambiarCancion(3);

String msg = "El Juego está Over jajaja\n";

boolean whtWns = id=='w'? false : true;

boolean blckWns = id=='b'? false : true;

if(whtWns && blckWns){

if(whtPnts > blckPnts){

blckWns = false;

}else if(blckPnts > whtPnts){

whtWns = false;

}

}else{

msg += "El juego fue terminado por que las " +(whtWns ? "negras" : "blancas");

msg += " se dieron porvencidas!";

}

if(id == 't'){

msg += "El juego se acabó porque las ";

if(tmRstntWht < 0){

msg += "blancas";

whtWns = false;

blckWns = true;

}else{

msg += "negras";

whtWns = true;

blckWns = false;

}

msg += " se quedaron sin tiempo";

}

msg += "\n";

if(whtWns && blckWns){//Ganan ambas

msg += "Fue un empate, ambos colores con " +whtPnts+ " fichas";

}else if(blckWns){

msg += "Ganaron las negras! " +blckPnts+ " a " +whtPnts;

}else{

msg += "Ganaron las blancas! " +whtPnts+ " a " +blckPnts;

}

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

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

if(button[i][j].getIcon().equals(g)){

button[i][j].setRolloverIcon(null);

}

}

}

canPlay = false;

handOver.setEnabled(false);

giveUp.setEnabled(false);

save.setEnabled(false);

alertar("¡¡¡Juego Finalizado!!!");

alertarGrande(msg);

//Con ésta línea se manda al menú principal luego de mostrar el aviso de gameover

//Pero es mejor dejar que el usuario vea el tablero y que luego decida volver al menú

//backMenu.doClick();

}

//Alerta de eventos en el juego

public static void alertar(String txt){

String turn = t.equals(w)?"Blancas":"Negras";

if(txt == ""){txt += "...";}

alrt.setText("<html><br>Juegan " +turn+ "<br>" +txt+ "</html>");

alrt.updateUI();

}

//Alerta de eventos GRANDES en el juego

public static void alertarGrande(String txt){

JOptionPane.showMessageDialog(null, txt);

}

//Pendiente de los clicks en los botones

public void actionPerformed(ActionEvent evnt){

if(evnt.getSource() == newGame){//Comenzar nuevo juego

long timeLongW = -1;

long timeLongB = -1;

boolean seRepite = true;

//Pedir tiempo para las blancas

while(seRepite){

String timeW = JOptionPane.showInputDialog("Minutos máximos de juego para las blancas\n[Vacío = Sin límite]");

if(timeW != null){

if(timeW.length() == 0){

timeLongW = -1;

seRepite = false;

}else{

try{

timeLongW = Long.parseLong(timeW);

tmTPlWht = tmRstntWht = (long)timeLongW\*60000;

if(timeLongW > 0){

seRepite = false;

}else{

throw new Exception();

}

}catch(Exception e){

alertarGrande("Usted no indicó un número entero mayor que 0");

}

}

}

}

seRepite = true;

//Pedir tiempo para las negras

while(seRepite){

String timeB = JOptionPane.showInputDialog("Minutos máximos de juego para las negras\n[Vacío = Sin límite]");

if(timeB != null){

if(timeB.length() == 0){

timeLongB = -1;

seRepite = false;

}else{

try{

timeLongB = Long.parseLong(timeB);

tmTPlBlck = tmRstntBlck = (long)timeLongB\*60000;

if(timeLongB > 0){

seRepite = false;

}else{

throw new Exception();

}

}catch(Exception e){

alertarGrande("Usted no indicó un número entero mayor que 0");

}

}

}

}

iniciarJuego(timeLongW, timeLongB);

}else if(evnt.getSource() == loadGame){//Carga el último juego guardado

iniciarJuego(-1,-1);

Kernel.loadGame();

}else if(evnt.getSource() == credits){//Muestra los créditos

mostrarCreditos();

}else if(evnt.getSource() == handOver){//Cede turno el jugador actual

cambiarTurno();

}else if(evnt.getSource() == giveUp){//Alguien se rindió

acabaJuego(t.equals(b) ? 'b' : 'w');

}else if(evnt.getSource() == save){//Salvar juego

Kernel.saveGame();

iniciarMenu();

cont.repaint();

menu.updateUI();

}else if(evnt.getSource() == backMenu){//Vuelve al menu principal

iniciarMenu();

cont.repaint();

menu.updateUI();

}else{//Se clickeó uno de los botones del juego

//System.out.println("\n----------\n\nwhtTm = " +whtTm+ "\nblckTm = " +blckTm+ "\nactTm = " +actTm+ "\nrgstrTm = " +rgstrTm+ "\ntmTPlWht = " +tmTPlWht+ "\ntmTPlBlck = " +tmTPlBlck+ "\ntmRstntWht = " +tmRstntWht+ "\ntmRstntBlck = " +tmRstntBlck);

if(canPlay){//Si se puede jugar

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

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

if(evnt.getSource() == button[i][j]){

x = j+1;

y = i+1;

Kernel.play(i, j);

}

}

}

}

}

}

public class RunTime implements Runnable{

public void run(){

while(canPlay){

int id = (t.equals(b) ? 1 : 2);

if(id == 1){

cambiarTiempo(b);

}else{

cambiarTiempo(w);

}

}

}

}

}

**OTHELLO KERNEL**

package kernel;

import gui.GUI;

import java.io.\*;

import javax.swing.\*;

public class Kernel{

public static final String path = "..\\";

public static final String file = path+ "blankOver.png";

//Intenta jugar en el cuadro de la coordenada x,y

public static void play(int x, int y){

Icon base = GUI.t, baser = GUI.tr, enemy = GUI.b, actual = GUI.button[x][y].getIcon();

if(base.equals(GUI.b)){enemy = GUI.w;}

int howMany = 0;

boolean play = false, other = false;

if(actual.equals(GUI.g)){

//Chequeo vertical hacia arriba

for(int i=x-1; i>=0; i--){

Icon act = GUI.button[i][y].getIcon();

if(act!=null && act.equals(base)){

if(other){

play = true;

for(int k=i+1; k<=x; k++){

GUI.button[k][y].setIcon(base);

GUI.button[k][y].setRolloverIcon(baser);

howMany++;

}

howMany--;

break;

}

break;

}else if(act.equals(enemy)){

other = true;

}else{

break;

}

}

//Chequeo vertical hacia abajo

other = false;

for(int i=x+1; i<GUI.SIZE; i++){

Icon act = GUI.button[i][y].getIcon();

if(act.equals(base)){

if(other){

play = true;

for(int k=i-1; k>=x; k--){

GUI.button[k][y].setIcon(base);

GUI.button[k][y].setRolloverIcon(baser);

howMany++;

}

howMany--;

break;

}

break;

}else if(act.equals(enemy)){

other = true;

}else{

break;

}

}

//Chequeo horizontal hacia la izquierda

other = false;

for(int i=y-1; i>=0; i--){

Icon act = GUI.button[x][i].getIcon();

if(act.equals(base)){

if(other){

play = true;

for(int k=i+1; k<=y; k++){

GUI.button[x][k].setIcon(base);

GUI.button[x][k].setRolloverIcon(baser);

howMany++;

}

howMany--;

break;

}

break;

}else if(act.equals(enemy)){

other = true;

}else{

break;

}

}

//Chequeo horizontal hacia la derecha

other = false;

for(int i=y+1; i<GUI.SIZE; i++){

Icon act = GUI.button[x][i].getIcon();

if(act.equals(base)){

if(other){

play = true;

for(int k=i-1; k>=y; k--){

GUI.button[x][k].setIcon(base);

GUI.button[x][k].setRolloverIcon(baser);

howMany++;

}

howMany--;

break;

}

break;

}else if(act.equals(enemy)){

other = true;

}else{

break;

}

}

//Chequeo diagonal: arriba izquierda

other = false;

for(int i=x-1, j=y-1; i>=0 && j>=0; i--, j--){

Icon act = GUI.button[i][j].getIcon();

if(act.equals(base)){

if(other){

play = true;

for(int k=i+1, l=j+1; k<=x && l<=y; k++, l++){

GUI.button[k][l].setIcon(base);

GUI.button[k][l].setRolloverIcon(baser);

howMany++;

}

howMany--;

break;

}

break;

}else if(act.equals(enemy)){

other = true;

}else{

break;

}

}

//Chequeo diagonal: arriba derecha

other = false;

for(int i=x-1, j=y+1; i>=0 && j<GUI.SIZE; i--, j++){

Icon act = GUI.button[i][j].getIcon();

if(act.equals(base)){

if(other){

play = true;

for(int k=i+1, l=j-1; k<=x && l>=y; k++, l--){

GUI.button[k][l].setIcon(base);

GUI.button[k][l].setRolloverIcon(baser);

howMany++;

}

howMany--;

break;

}

break;

}else if(act.equals(enemy)){

other = true;

}else{

break;

}

}

//Chequeo diagonal: abajo derecha

other = false;

for(int i=x+1, j=y+1; i<GUI.SIZE && j<GUI.SIZE; i++, j++){

Icon act = GUI.button[i][j].getIcon();

if(act.equals(base)){

if(other){

play = true;

for(int k=i-1, l=j-1; k>=x && l>=y; k--, l--){

GUI.button[k][l].setIcon(base);

GUI.button[k][l].setRolloverIcon(baser);

howMany++;

}

howMany--;

break;

}

break;

}else if(act.equals(enemy)){

other = true;

}else{

break;

}

}

//Chequeo diagonal: abajo izquierda

other = false;

for(int i=x+1, j=y-1; i<GUI.SIZE && j>=0; i++, j--){

Icon act = GUI.button[i][j].getIcon();

if(act.equals(base)){

if(other){

play = true;

for(int k=i-1, l=j+1; k>=x && l<=y; k--, l++){

GUI.button[k][l].setIcon(base);

GUI.button[k][l].setRolloverIcon(baser);

howMany++;

}

howMany--;

break;

}

break;

}else if(act.equals(enemy)){

other = true;

}else{

break;

}

}

}

//Se pudo jugar

if(play){

GUI.cambiarTurno();

GUI.cambiarPuntos(howMany, enemy);

GUI.cambiarTiempo(base);

if(GUI.canPlay){

GUI.alertar("");

}

}else{

GUI.alertar("Jugada no posible!");

}

}

//Guardar el juego

public static void saveGame(){

try{

BufferedWriter game = new BufferedWriter(new FileWriter(file));

game.write(GUI.whtTm+ "\t" +GUI.blckTm+ "\t" +GUI.tmTPlWht+ "\t" +GUI.tmTPlBlck+ "\t");

game.newLine();

game.write(GUI.whtPnts+ "\t" +GUI.blckPnts+ "\t");

if(GUI.t.equals(GUI.b)){

game.write("1\t");

}else{

game.write("2\t");

}

game.newLine();

for(int i=0; i<GUI.SIZE; i++){

for(int j=0; j<GUI.SIZE; j++){

Icon act = GUI.button[i][j].getIcon();

if(act.equals(GUI.b)){//Negro

game.write("1\t");

}else if(act.equals(GUI.w)){//Blanco

game.write("2\t");

}else{//Verde

game.write("0\t");

}

}

game.newLine();

}

game.close();

GUI.canPlay = false;

GUI.alertarGrande("Juego guardado correctamente!");

}catch(Exception e){

String sge = "Error al guardar el juego!";

sge += "\nPuede deberse a que:";

sge += "\n- El archivo está en uso.";

sge += "\n- Usted está jugando desde un navegador web.";

GUI.alertarGrande(sge);

}

}

//Cargar el juego

public static void loadGame(){

try{

BufferedReader game = new BufferedReader(new InputStreamReader(new DataInputStream(new FileInputStream(file))));

String gameData = "";

String gameLine = "";

int lines = 0;

int numbers = 0;

int[] tms = new int[4], pnts = new int[2];

int turn;

Icon[][] bg = new Icon[GUI.SIZE][GUI.SIZE];

while((gameLine = game.readLine()) != null){

gameData += gameLine;

lines++;

}

if(lines != GUI.SIZE + 2){//No se leyeron las n+2 líneas

throw new Exception();

}

tms[0] = Integer.parseInt(gameData.substring(0, gameData.indexOf("\t")));

gameData = gameData.substring(gameData.indexOf("\t") + 1);

numbers++;

tms[1] = Integer.parseInt(gameData.substring(0, gameData.indexOf("\t")));

gameData = gameData.substring(gameData.indexOf("\t") + 1);

numbers++;

tms[2] = Integer.parseInt(gameData.substring(0, gameData.indexOf("\t")));

gameData = gameData.substring(gameData.indexOf("\t") + 1);

numbers++;

tms[3] = Integer.parseInt(gameData.substring(0, gameData.indexOf("\t")));

gameData = gameData.substring(gameData.indexOf("\t") + 1);

numbers++;

pnts[0] = Integer.parseInt(gameData.substring(0, gameData.indexOf("\t")));

gameData = gameData.substring(gameData.indexOf("\t") + 1);

numbers++;

pnts[1] = Integer.parseInt(gameData.substring(0, gameData.indexOf("\t")));

gameData = gameData.substring(gameData.indexOf("\t") + 1);

numbers++;

turn = Integer.parseInt(gameData.substring(0, gameData.indexOf("\t")));

gameData = gameData.substring(gameData.indexOf("\t") + 1);

numbers++;

while(gameData.length() > 0){

for(int i=0; i<GUI.SIZE; i++){

for(int j=0; j<GUI.SIZE; j++){

int id = Integer.parseInt(gameData.substring(0, gameData.indexOf("\t")));

gameData = gameData.substring(gameData.indexOf("\t") + 1);

numbers++;

bg[i][j] = (id==2 ? GUI.w : (id==1 ? GUI.b : GUI.g));

}

}

}

if(numbers != GUI.SIZE\*GUI.SIZE + 7){//No se leyeron los n\*n+7 datos

throw new Exception();

}

GUI.rgstrTm = System.currentTimeMillis();

GUI.whtTm = tms[0];

GUI.blckTm = tms[1];

GUI.tmTPlWht = tms[2];

GUI.tmTPlBlck = tms[3];

GUI.noLmts[0] = tms[0]==-1 ? true : false;

GUI.noLmts[1] = tms[1]==-1 ? true : false;

GUI.cambiarTiempo(GUI.g);

GUI.whtPnts = pnts[0];

GUI.blckPnts = pnts[1];

GUI.cambiarPuntos(0, GUI.g);

GUI.t = turn==1 ? GUI.w : GUI.b;

GUI.moves = 0;

GUI.cambiarTurno();

for(int i=0; i<GUI.SIZE; i++){

for(int j=0; j<GUI.SIZE; j++){

GUI.button[i][j].setIcon(bg[i][j]);

if(bg[i][j].equals(GUI.b)){

GUI.button[i][j].setRolloverIcon(GUI.br);

}else if(bg[i][j].equals(GUI.w)){

GUI.button[i][j].setRolloverIcon(GUI.wr);

}

}

}

game.close();

}catch(Exception e){

String lge = "Error al cargar el juego!";

lge += "\nPuede deberse a que:";

lge += "\n- El archivo no existe.";

lge += "\n- El archivo está en uso.";

lge += "\n- El archivo fue modificado fuera del juego.";

lge += "\n- Usted está jugando desde un navegador web.";

GUI.alertarGrande(lge);

GUI.newGame.doClick();

}

}

}

**QUICK LOGIC QUICK LOGIC.c**

#include <stdio.h>

#include <stdlib.h>

#include "logica.h"

#include "archivos.h"

#include "interfazGrafica.h"

int main()

{

struct Juego juego;

struct Datos datos[10];

int f=0, c=0, n=0, m=0, s=0;

system("color 8a");

n=cargarDatos(datos);

sIn(1);

pantallaEntrada(&juego);

printf ("\t\t\t\tQUICK LOGIC\n\n\n\n");

imprimirLab(&juego);

//sOut();

system("pause");

while (s==0)

{

//pantallaFinal(&juego, 1);

//imprimirLab(&juego);

s = menu1(&juego, n, datos);

if (s==1)

{

return;

}

/\*lab(1, &juego);

printf ("\t\t\t\tQUICK LOGIC\n\n\n\n");

imprimirLab(&juego);

system("pause");

printf ("laberinto1();");

m = mover(&juego);

pantallaFinal(&juego, m);

printf ("\t\t\t\tQUICK LOGIC\n\n\n\n");

imprimirLab(&juego);

system("pause");

menu1(&juego, n, datos);

system("PAUSE");\*/

}

return 0;

}

**QUICK LOGIC archivos.c**

#include <stdio.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include "logica.h"

int pantallaEntrada(struct Juego\* juego)

{

int f, c;

FILE\* pantalla = fopen("grafico/pantallaEntrada.txt", "r");

if (pantalla!=0)

{

for (f=0;f<20;f++)

{

fread(juego->laberinto[f], 1, 81, pantalla);

}

for (f=0;f<20;f++)

{

for (c=0;c<80;c++)

{

if (juego->laberinto[f][c]!=32)

{

juego->laberinto[f][c]=219;

}

}

}

}

else

{

printf ("no se puede leer la pantalla inicial");

}

fclose(pantalla);

}

int lab(int n, struct Juego\* juego)

{

int f, c;

FILE \*archivo;

if (n==1)

{

archivo = fopen("grafico/laberinto1.txt", "r");

if (archivo!=0)

{

for (f=0;f<20;f++)

{

fread(juego->laberinto[f], 1, 81, archivo);

}

for (f=0;f<20;f++)

{

for (c=0;c<80;c++)

{

if (juego->laberinto[f][c]!=32)

{

juego->laberinto[f][c]=219;

}

}

}

fclose (archivo);

}

else

{

printf ("error abriendo el laberinto 1.");

}

}/\*hasta aquí va el código del laberinto 1\*/

else if (n==2)

{

archivo = fopen("grafico/laberinto2.txt", "r");

if (archivo!=0)

{

for (f=0;f<20;f++)

{

fread(juego->laberinto[f], 1, 81, archivo);

}

for (f=0;f<20;f++)

{

for (c=0;c<80;c++)

{

if (juego->laberinto[f][c]!=32)

{

juego->laberinto[f][c]=219;

}

}

}

fclose (archivo);

}

else

{

printf ("error abriendo el laberinto 2.");

}

}/\*hasta aquí va el laberinto 2\*/

if (n==3)

{

archivo = fopen("grafico/laberinto3.txt", "r");

if (archivo!=0)

{

for (f=0;f<20;f++)

{

fread(juego->laberinto[f], 1, 81, archivo);

}

for (f=0;f<20;f++)

{

for (c=0;c<80;c++)

{

if (juego->laberinto[f][c]!=32)

{

juego->laberinto[f][c]=219;

}

}

}

fclose (archivo);

}

else

{

printf ("error abriendo el laberinto 3.");

}

}/\*hasta aquí va el código del laberinto 3\*/

if (n==4)

{

archivo = fopen("grafico/laberinto4.txt", "r");

if (archivo!=0)

{

for (f=0;f<20;f++)

{

fread(juego->laberinto[f], 1, 81, archivo);

}

for (f=0;f<20;f++)

{

for (c=0;c<80;c++)

{

if (juego->laberinto[f][c]!=32)

{

juego->laberinto[f][c]=219;

}

}

}

fclose (archivo);

}

else

{

printf ("error abriendo el laberinto 4.");

}

}/\*hasta aquí va el código del laberinto 4\*/

if (n==5)

{

archivo = fopen("grafico/laberinto5.txt", "r");

if (archivo!=0)

{

for (f=0;f<20;f++)

{

fread(juego->laberinto[f], 1, 81, archivo);

}

for (f=0;f<20;f++)

{

for (c=0;c<80;c++)

{

if (juego->laberinto[f][c]!=32)

{

juego->laberinto[f][c]=219;

}

}

}

fclose (archivo);

}

else

{

printf ("error abriendo el laberinto 5.");

}

}/\*hasta aquí va el código del laberinto 5\*/

}

int cargarDatos(struct Datos\* datos)

{

FILE\* archivo;

archivo = fopen("manejoDeArchivos/datosJugador.txt", "r");

if (archivo!=0)

{

int f=0, c=0;

while( fscanf (archivo, "%s\t%d\t%d\t%x\n", &datos[f].perfil, &datos[f].puntaje, & datos[f].robot, & datos[f].color) == 4)

{

f++;

}

fclose (archivo);

return f;

}

else

{

return -1;

}

}

int pantallaFinal(struct Juego\* juego, int m)

{

int f, c;

if (m==1)

{

FILE\* archivo = fopen ("grafico/pantallaGanadora.txt", "r");

if (archivo != 0)

{

for (f=0;f<20;f++)

{

fread(juego->laberinto[f], 1, 81, archivo);

}

for (f=0;f<20;f++)

{

for (c=0;c<80;c++)

{

if (juego->laberinto[f][c]!=32)

{

juego->laberinto[f][c]=219;

}

}

}

}

else

{

//system ("cls");

printf ("error abriendo la pantalla ganadora\n");

}

}

else if (m==2)

{

FILE\* archivo = fopen ("grafico/pantallaEscape.txt", "r");

if (archivo != 0)

{

for (f=0;f<20;f++)

{

fread(juego->laberinto[f], 1, 81, archivo);

}

for (f=0;f<20;f++)

{

for (c=0;c<80;c++)

{

if (juego->laberinto[f][c]!=32)

{

juego->laberinto[f][c]=219;

}

}

}

}

else

{

system ("cls");

printf ("error abriendo la pantalla escape");

}

}

}

**QUICK LOGIC archivos.h**

//#include "logica.h"

int pantallaEntrada(struct Juego\* juego);

int lab(int n, struct Juego\* juego);

int cargarDatos(struct Datos\* datos);

int pantallaFinal(struct Juego\* juego, int m);

**QUICK LOGIC interfazGrafica.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

#include "logica.h"

#include "interfazGrafica.h"

#include "archivos.h"

void imprimirLab(struct Juego\* juego)

{

/\*char color[9];

sprintf (color, "color %x", juego->datos.color);

system (color);\*/

int f, c;

//system ("cls");

//printf ("\t\t\t\tQUICK LOGIC\n\n\n\n");

for (f=0;f<20;f++)

{

for (c=0;c<80;c++)

{

if(juego->laberinto[f][c] == juego->datos.robot)

{

printf ("%c", juego->datos.robot);

}

else if (juego->laberinto[f][c]!=32)

{

printf ("%c", 219);

}

else

{

printf ("%c", 32);

}

}

}

//printf ("%1600s", juego->laberinto);

//system("pause");

//menu1(&juego,n,datos) ;

}

int menu1(struct Juego\* juego, int n, struct Datos\* datos)

{

/\*char color[9];

sprintf (color, "color %x", juego->datos.color);

system (color);\*/

int seleccion=0;

while (seleccion==0 || (seleccion != '1' && seleccion != '2' && seleccion != '3' && seleccion != '4' && seleccion != '5'))

{

system ("cls");

printf ("\t\t\t\tQUICK LOGIC\n\n\n\n");

printf ("1. crear perfil.\n");

printf ("2. cargar perfil.\n");

printf ("3. opciones.\n");

printf ("4. mayores puntajes.\n");

printf ("5. salir.\n\n");

seleccion = getch();

}

if(seleccion == '1')

{

menuCrearPerfil(juego, n, datos);

return 0;

}

else if (seleccion == '2')

{

menuCargarPerfil(juego, n, datos);

return 0;

}

else if (seleccion=='3')

{

printf ("\n\t\tlo lamentamos, esta opcion aun no esta completa.\n");

system ("pause");

//menuOpciones(&juego, n, datos);

return 0;

}

else if (seleccion=='4')

{

mayoresPuntajes(juego, n, datos);

return 0;

}

else if (seleccion=='5')

{

int r=0;

system ("cls");

printf ("Esta seguro que desea salir del juego?\n1. si\n2. no");

r=getch();

if (r=='1')

{

system("exit");

}

else if (r=='2')

{

menu1(juego, n, datos);

}

return 1;

}

return 0;

}

void menuCrearPerfil(struct Juego\* juego, int n, struct Datos\* datos)

{

int a;

FILE\* archivo=0;

system("cls");

printf ("\t\t\t\tCrear perfil.\n\t\t\t\t-------------\n");

printf ("Por favor entre su nombre de usuario:\n");

scanf ("%s", juego->datos.perfil);

system ("cls");

printf ("Ahora entre el color de su robot junto con el color de sus paredes:\n\n\n");

system ("color/?");

scanf ("%x", &juego->datos.color);

system ("cls");

printf ("Ahora entre el codigo para el robot que desee usar en sus laberintos:\n\n");

for (a=1;a<256;a++)

{

printf ("%d. %c\t", a, a);

}

printf ("Si desea usar un robot que no este mostrado en la tabla puede ingresar este codigo si lo desea:\n");

scanf ("%d", &juego->datos.robot);

juego->datos.puntaje=0;

archivo= fopen ("manejoDeArchivos/datosJugador.txt", "a");

if (archivo!=0)

{

fprintf (archivo, "\n%s\t%d\t%d\t%hx", juego->datos.perfil, juego->datos.puntaje, juego->datos.robot, juego->datos.color);

fclose(archivo);

}

else

{

printf ("error abriendo y guardando datos del nuevo perfil.\n");

}

menuJuego(juego, n, datos);

}

void menuCargarPerfil(struct Juego\* juego, int n, struct Datos\* datos)

{

char r, s=0;

while (s==0)

{

system("cls");

printf ("\t\t\t\tCargar perfil.\n\t\t\t\t--------------\n\n");

for (r=0;r<n-1;r++)

{

printf (("%d. %s\n"), (r+1)%10, datos[r].perfil);

}

s = getch();

if (s=='1')

{

strcpy(juego->datos.perfil, datos[0].perfil);

juego->datos.color=datos[0].color;

juego->datos.puntaje=datos[0].puntaje;

juego->datos.robot=datos[0].robot;

}

else if (s=='2')

{

strcpy(juego->datos.perfil, datos[1].perfil);

juego->datos.color=datos[1].color;

juego->datos.puntaje=datos[1].puntaje;

juego->datos.robot=datos[1].robot;

}

else if (s=='3')

{

strcpy(juego->datos.perfil, datos[2].perfil);

juego->datos.color=datos[2].color;

juego->datos.puntaje=datos[2].puntaje;

juego->datos.robot=datos[2].robot;

}

else if (s=='4')

{

strcpy(juego->datos.perfil, datos[3].perfil);

juego->datos.color=datos[3].color;

juego->datos.puntaje=datos[3].puntaje;

juego->datos.robot=datos[3].robot;

}

else if (s=='5')

{

strcpy(juego->datos.perfil, datos[4].perfil);

juego->datos.color=datos[4].color;

juego->datos.puntaje=datos[4].puntaje;

juego->datos.robot=datos[4].robot;

}

else if (s=='6')

{

strcpy(juego->datos.perfil, datos[5].perfil);

juego->datos.color=datos[5].color;

juego->datos.puntaje=datos[5].puntaje;

juego->datos.robot=datos[5].robot;

}

else if (s=='7')

{

strcpy(juego->datos.perfil, datos[6].perfil);

juego->datos.color=datos[6].color;

juego->datos.puntaje=datos[6].puntaje;

juego->datos.robot=datos[6].robot;

}

else if (s=='8')

{

strcpy(juego->datos.perfil, datos[7].perfil);

juego->datos.color=datos[7].color;

juego->datos.puntaje=datos[7].puntaje;

juego->datos.robot=datos[7].robot;

}

else if (s=='9')

{

strcpy(juego->datos.perfil, datos[8].perfil);

juego->datos.color=datos[8].color;

juego->datos.puntaje=datos[8].puntaje;

juego->datos.robot=datos[8].robot;

}

else if (s=='0')

{

strcpy(juego->datos.perfil, datos[9].perfil);

juego->datos.color=datos[9].color;

juego->datos.puntaje=datos[9].puntaje;

juego->datos.robot=datos[9].robot;

}

menuJuego(juego, n, datos);

return;

}

}

int menuOpciones(struct Juego\* juego, int n, struct Datos\* datos)

{

return 1;

}

void mayoresPuntajes(struct Juego\* juego, int n, struct Datos\* datos)

{

system ("cls");

printf ("\t\t\t\tQUICK LOGIC\n\n\n\n");

int a;

for (a=0;a<10;a++)

{

printf ("%d. Jugador: %s\t\tPuntaje: %d\n", a+1, datos[a].perfil, datos[a].puntaje);

}

system ("pause");

menu1(juego, n, datos);

}

/\*void mayoresPuntajes(struct Juego\* juego, int n, struct Datos\* datos)

{

int f=0;

for (f=0;f<10;f++)

{

printf ("%d. %20s\t%d", f+1, datos[f].perfil, datos[f].puntaje);

}

menu1(&juego, n, datos);

}\*/

void menuJuego(struct Juego\* juego, int n, struct Datos\* datos)

{

int l=1, m=1;

system ("cls");

printf ("\t\t\t\tQUICK LOGIC\n\n\n\n");

printf ("entre el numero del laberinto que desea resolver:\n");

scanf ("%d", &l);

lab(l, juego);

system("cls");

printf ("\t\t\t\tQUICK LOGIC\n\n\n\n");

imprimirLab(juego);

system("pause");

m = mover(juego);

pantallaFinal(juego, m);

printf ("\t\t\t\tQUICK LOGIC\n\n\n\n");

imprimirLab(juego);

system("pause");

return;

}

**QUICK LOGIC interfazGrafica.h**

int menu1(struct Juego\* juego, int n, struct Datos\* datos);

void menuCrearPerfil(struct Juego\* juego, int n, struct Datos\* datos);

void menuCargarPerfil(struct Juego\* juego, int n, struct Datos\* datos);

int menuOpciones(struct Juego\* juego, int n, struct Datos\* datos);

void mayoresPuntajes(struct Juego\* juego, int n, struct Datos\* datos);

void imprimirLab(struct Juego\* juego);

void mayoresPuntajes(struct Juego\* juego, int n, struct Datos\* datos);

void menuJuego(struct Juego\* juego, int n, struct Datos\* datos);

**QUICK LOGIC lógica.c**

#include <stdio.h>

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

#include <conio.h>

#include <windows.h>

//#include <C:/Dev-Cpp/include/c++/3.4.2/backward/iostream.h>

#include <mmsystem.h>

//#pragma comment(lib,"winmm.lib")

#include "logica.h"

#include "interfazGrafica.h"

#define C 80

#define F 20

int mover(struct Juego\* juego)

{

//system ("cls");

//printf("\n\n\n\n");

//char datos [][];

int x=0, y=1, key, x1, y1, t1, t2, t3, puntaje=0, pasos=0;

/\*char color[9];

sprintf (color, "color %x", juego->datos.color);

system (color);\*/

t1=GetTickCount();

key = getch();

x1=0;

y1=1;

while(key != 0xd && juego->laberinto[18][79] != juego->datos.robot)

{

if(key == 0xe0)

{

key = getch();

if(key != 0xd || juego->laberinto[18][80] != juego->datos.robot)

{

if(key == 0x4b && x > 0)

{

if (juego->laberinto[y][x-1]==32)

{

x1=x-1;

y1=y;

pasos++;

juego->laberinto[y1][x1]=juego->datos.robot;

juego->laberinto[y][x]=32;

}

}

if(key == 0x4d && x < C-1)

{

if (juego->laberinto[y][x+1]==32)

{

x1=x+1;

y1=y;

pasos++;

juego->laberinto[y1][x1]=juego->datos.robot;

juego->laberinto[y][x]=32;

}

}

if(key == 0x48 && y > 0)

{

if (juego->laberinto[y-1][x]==32)

{

x1=x;

y1=y-1;

pasos++;

juego->laberinto[y1][x1]=juego->datos.robot;

juego->laberinto[y][x]=32;

}

}

if(key == 0x50 && y < F-1)

{

if (juego->laberinto[y+1][x]==32)

{

x1=x;

y1=y+1;

pasos++;

juego->laberinto[y1][x1]=juego->datos.robot;

juego->laberinto[y][x]=32;

}

}

}

if (key == 0xd)

{

return 2;

}

if (juego->laberinto[18][80] == juego->datos.robot)

{

return 1;

}

system("cls");

//textattr("%x", juego->datos.color);

x=x1;

y=y1;

t2=GetTickCount();

t3=t2-t1;

puntaje = ((1000000-100\*pasos)/t3)\*100;

/\*for(i = 0; i < F; i++)

{

for(j = 0; j < C; j++)

{

laberinto[i][j] = 32;

}

} \*/

//laberinto[y][x] = "%c", 115;

juego->laberinto[F-1][C-1] = 0;//laberinto[F-1][C-1] = 0;

printf (" QUICK LOGIC \n");

printf ("--------------------------------------------------------------------------------");

printf ("Jugador: %s. Mayor puntaje: %d\n", juego->datos.perfil, juego->datos.puntaje);

printf ("Tiempo: %d Puntaje: %d\n", t3, puntaje);

//printf("%1600s", juego->laberinto);//printf("%1600s", laberinto);

imprimirLab(juego);

}

key = getch();

}

if(juego->datos.puntaje<puntaje)

{

juego->datos.puntaje=puntaje;

}

if (key == 0xd)

{

return 2;

}

if (juego->laberinto[18][80] == juego->datos.robot)

{

return 1;

}

}

void aletorio(struct Juego\* juego)

{

char a=219;

int x,y,f,c;

for (f=0;f<20;f++)

{

juego->laberinto[f][0]=a;

juego->laberinto[f][79]=a;

}

for (c=0;c<80;c++)

{

juego->laberinto[0][c]=a;

juego->laberinto[19][c]=a;

}

x=rand();

x=(x%19)+1;

for (f=1;f<79;f++)

{

juego->laberinto[x][f]=a;

}

y=rand();

y=(y%19)+1;

for (c=1;c<79;c++)

{

juego->laberinto[y][c]=a;

}

for (f=0;f<20;f++)

{

for (c=0;c<80;c++)

{

printf("%c", juego->laberinto[y][c]);

}

}

}

void sIn(int n)

{

if (n==1)

{

PlaySound("manejoDeArchivos/jamesBond.wav", 0, SND\_LOOP | SND\_ASYNC);

}

else if (n==2)

{

//PlaySound();

}

else if (n==3)

{

//PlaySound();

}

return;

}

void sOut()

{

PlaySound(0, 0, SND\_ASYNC);

return;

}

**QUICK LOGIC logica.h**

struct Datos

{

char perfil[20];

int puntaje;

int robot;

int color;

};

struct Juego

{

char laberinto[20][80];

struct Datos datos;

};

int mover(struct Juego\* juego);

void aletorio(struct Juego\* juego);

//void sIn(int n);

//void sOut();