Introducción

Este Laberinto fue desarrollado en Java, mediante el programa (IDE) Eclipse Oxigen. Se comenzó por realizar diferentes aplicaciones sencillas del tipo Cliente/Servidor (Chat, calculadora) a nivel de capacitación para poder lograr la resolución del juego.

Estrategia

Nuestra estrategia se desarrolla de la siguiente manera: Al abrir el servidor, éste predeterminadamente queda a la espera de usuarios que deseen unirse para jugar escuchando el puerto “9020”. Un usuario puede unirse al servidor mediante el formulario de inicio de sesión en donde envía sus credenciales, el puerto y el host desde el cual se conecta. Una vez que el servidor autorice las credenciales, devuelve las coordenadas de entrada. Cuando el cliente recibe las coordenadas de entrada, cierra el formulario de inicio de sesión y se posiciona sobre la coordenada recibida.

Al posicionarse en la entrada o cada vez que el jugador haga un movimiento, el cliente pide al servidor todos los valores de casilleros que se encuentran alrededor y los muestra en el laberinto. Todas las posiciones conocidas son guardadas por el cliente para no volver a pedirlas al servidor.

Por defecto, el Servidor tiene solo un mapa cargado. A nivel paliativo, se encuentran diferentes modelos de laberintos en la ruta src/extras/ Mapa de laberinto.txt los cuales pueden servir para editar el predefinido en Servidor.java. Una oportunidad de mejora para este laberinto sería que el servidor elija un mapa aleatoriamente al autorizar al cliente (Esto no está implementado ya que no era parte de los requisitos del tp)

El cliente es el encargado de verificar en qué posición se encuentra y determinar si es necesario tomar oro, pagar a un guardia o tomar llave. Así mismo, se encuentran predefinidas ciertas reglas para el que jugador no pueda pasar por paredes.

El servidor tiene únicamente dos tareas. Por un lado, cuando se inicia queda a la espera de autorizar credenciales y luego durante el resto del juego devuelve el valor de las coordenadas que le solicitan.

Se encuentra implementando una consola que permite iniciar un servidor sobre un puerto determinado, loguear a un usuario a un servidor determinado, mostrar el formulario de inicio de sesión, etc... Todos estos comandos se encuentran predefinidos y pueden ser consultados con el comando “help” desde la consola.

Protocolo

Para realizar este juego utilizamos predeterminada el puerto 9020, el cual permitirá aceptar una nueva conexión por parte de un usuario y poder unirse al servidor. Como se explicó en la estrategia, el programa cuenta con una consola capaz de iniciar un servidor en cualquier puerto y loguear clientes de forma personalizada utilizando comandos predefinidos. Todas las trasferencias de datos desde y hacia el servidor utilizan los puertos predefinidos por el usuario al iniciar el juego.

Seguridad

En este punto utilizamos una función de encriptación mediante funciones de conversión de los datos que se van a enviar mediante los sockets. Una vez que los mensajes viajan hacia el servidor, se desencriptan para poder entenderlos, se obtienen los datos que el cliente necesita, se encriptan y se devuelven. Finalmente, cuando el cliente recibe los datos, los desencripta mediante otro método que le permite comprender los datos que acaba de recibir de forma útil. En cuanto al acceso al juego, el servidor valida credenciales para aceptar una conexión y enviar las primeras Coordenadas del laberinto, las cuales son imprescindibles para posicionarse en el mapa.

Código fuente:

package modelo;

import java.awt.AWTException;

import java.awt.Color;

import java.awt.Font;

import java.awt.Robot;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.io.DataInputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

import java.net.Socket;

import java.net.UnknownHostException;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.GregorianCalendar;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.SwingUtilities;

public class Cliente {

public static void main(String[] args) {

VentanaLogin ventanaLogin = new VentanaLogin();

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

}

}

class VentanaLogin extends JFrame implements ActionListener {

JButton btnLogin;

JTextField ip;

JTextField puerto;

JTextField user;

JTextField password;

JTextArea respuesta;

public VentanaLogin() {

JLabel titulo = new JLabel("Login");

titulo.setBounds(10, 10, 50, 20);

add(titulo);

JLabel ipLabel = new JLabel("IP:");

ipLabel.setBounds(10, 50, 50, 20);

add(ipLabel);

ip = new JTextField("localhost");

ip.setBounds(70, 50, 100, 20);

add(ip);

JLabel puertoLabel = new JLabel("Puerto: ");

puertoLabel.setBounds(10, 80, 50, 20);

add(puertoLabel);

puerto = new JTextField("9020");

puerto.setBounds(70, 80, 100, 20);

add(puerto);

JLabel userLabel = new JLabel("Usuario: ");

userLabel.setBounds(10, 110, 50, 20);

add(userLabel);

user = new JTextField("admin");

user.setBounds(70, 110, 100, 20);

add(user);

JLabel passLabel = new JLabel("Clave: ");

passLabel.setBounds(10, 140, 50, 20);

add(passLabel);

password = new JTextField("admin");

password.setBounds(70, 140, 100, 20);

add(password);

btnLogin = new JButton();

btnLogin.setText("Ingresar");

btnLogin.setBounds(10, 180, 150, 20); // (x, y, ancho, alto)

btnLogin.addActionListener(this); // para poder programar en el boton

add(btnLogin);

JLabel respuestaLabel = new JLabel("Respuesta del servidor: ");

respuestaLabel.setBounds(10, 290, 150, 20);

add(respuestaLabel);

respuesta = new JTextArea(" Esperando...");

respuesta.setBounds(10, 320, 150, 20);

add(respuesta);

setLayout(null); // para que los controles no esten uno encima del otro

setTitle("Cliente Login");

setSize(400, 400); // tamaño del layout

setResizable(false);

setLocationRelativeTo(null);

setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

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

try {

boolean esperando = true;

Coordenada entradaXY = new Coordenada();

Login login = new Login(ip.getText(), puerto.getText(), user.getText(), password.getText());

Socket cliente = new Socket(ip.getText(), Integer.parseInt(puerto.getText()));

ObjectOutputStream out = new ObjectOutputStream(cliente.getOutputStream());

out.writeObject(login);

while (esperando == true) {

ObjectInputStream entrada = new ObjectInputStream(cliente.getInputStream());

entradaXY = (Coordenada) entrada.readObject();

if (entradaXY != null) {

esperando = false;

}

}

setVisible(false);

VentanaCliente ventanaCliente = new VentanaCliente(entradaXY, ip.getText(), puerto.getText());

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

} catch (Exception ex) {

System.out.println("Error en cliente " + ex.getMessage());

}

}

}

}

class VentanaCliente extends JFrame {

int oro = 0;

boolean llave = false;

public static int key = 346;

int cSize = 30;

int posx = 70;

int posy = 50;

int xEntrada = 0;

int yEntrada = 0;

public static Coordenada posicionActual;

public static KeyListener listener;

JTextArea[][] laberinto = new JTextArea[20][20];

Coordenada[][] mapa = new Coordenada[20][20];

ArrayList<Coordenada> oroYaTomado = new ArrayList<Coordenada>();

JTextArea casillero;

JTextArea txtMensajes;

JLabel oroLabel;

Configuracion config;

public VentanaCliente(Coordenada entrada, String ip, String puerto) {

posicionActual = entrada;

xEntrada = entrada.getX();

yEntrada = entrada.getY();

listener = new MyKeyListener();

addKeyListener(listener);

setFocusable(true);

config = new Configuracion(ip, puerto);

JLabel entradaLabel = new JLabel("Coordenadas de entrada: " + entrada.getX() + "; " + entrada.getY());

entradaLabel.setBounds(10, 10, 200, 20);

add(entradaLabel);

oroLabel = new JLabel("Oro: " + oro);

oroLabel.setBounds(300, 10, 200, 20);

oroLabel.setFont(new Font("Arial", Font.PLAIN, 25));

add(oroLabel);

txtMensajes = new JTextArea();

txtMensajes.setBounds(450, 20, 300, 1000);

// add(txtMensajes);

txtMensajes.setEditable(false);

consola("Inicio de juego.");

dibujarLaberinto(20, false);

initMapa();

setLayout(null);

setTitle("Laberinto");

setSize(800, 800);

setLocationRelativeTo(null);

setVisible(true);

try {

mostrarVecinos();

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

public void initMapa() {

System.out.println("Inicializando mapa");

for (int i = 0; i < 20; i++) {

for (int j = 0; j < 20; j++) {

mapa[i][j] = new Coordenada(i, j, false);

}

}

mapa[xEntrada][yEntrada].setKnown(false);

}

public JTextArea posicionar(JTextArea casillero, Coordenada c) {

casillero.setBounds(laberinto[c.getX()][c.getY()].getX(), laberinto[c.getX()][c.getY()].getY(), 35, 35);

return casillero;

}

public void consola(String mensaje) {

Calendar calendario = new GregorianCalendar();

txtMensajes.append(calendario.get(Calendar.HOUR) + ":" + calendario.get(Calendar.MINUTE) + ":"

+ calendario.get(Calendar.SECOND));

txtMensajes.append(": " + mensaje + "\n");

}

public void mostrarVecinos() throws NumberFormatException, UnknownHostException, IOException {

ArrayList<Coordenada> coordenadas = new ArrayList<Coordenada>();

if (mapa[posicionActual.getX()][posicionActual.getY()].isKnown() == false) {

Coordenada izquierda = new Coordenada(posicionActual.getX() - 1, posicionActual.getY());

Coordenada derecha1 = new Coordenada(posicionActual.getX() + 1, posicionActual.getY());

Coordenada arriba = new Coordenada(posicionActual.getX(), posicionActual.getY() - 1);

Coordenada arriba1 = new Coordenada(arriba.getX() + 1, arriba.getY());

Coordenada abajo = new Coordenada(posicionActual.getX(), posicionActual.getY() + 1);

Coordenada abajo1 = new Coordenada(abajo.getX() + 1, abajo.getY());

coordenadas.add(posicionActual);

coordenadas.add(izquierda);

coordenadas.add(derecha1);

coordenadas.add(arriba);

coordenadas.add(arriba1);

coordenadas.add(abajo);

coordenadas.add(abajo1);

} else {

Coordenada izquierda = new Coordenada(posicionActual.getX() - 1, posicionActual.getY());

Coordenada derecha1 = new Coordenada(posicionActual.getX() + 1, posicionActual.getY());

Coordenada arriba = new Coordenada(posicionActual.getX(), posicionActual.getY() - 1);

Coordenada abajo = new Coordenada(posicionActual.getX(), posicionActual.getY() + 1);

coordenadas.add(izquierda);

coordenadas.add(derecha1);

coordenadas.add(arriba);

coordenadas.add(abajo);

}

for (Coordenada c : coordenadas) {

if (c.getX() >= 0 && c.getX() <= 19 && c.getY() >= 0 && c.getY() <= 19) {

mapa[c.getX()][c.getY()].setLetra(devolverLetra(config.getIp(), config.getPuerto(), c));

if (mapa[c.getX()][c.getY()].isKnown() == false) {

add(posicionar(format(new JTextArea(), mapa[c.getX()][c.getY()].getLetra(), c), c), c);

}

SwingUtilities.updateComponentTreeUI(this);

mapa[c.getX()][c.getY()].setKnown(true);

}

add(posicionar(format(new JTextArea(), " ", posicionActual), posicionActual));

}

}

public static String devolverLetra(String ip, String puerto, Coordenada c)

throws NumberFormatException, UnknownHostException, IOException {

int letraEncript = 0;

String letra = "";

Socket cliente = new Socket(ip, Integer.parseInt(puerto));

ObjectOutputStream pedido = new ObjectOutputStream(cliente.getOutputStream());

c = encriptarCoordenada(c);

pedido.writeObject(c);

DataInputStream recibir = new DataInputStream(cliente.getInputStream());

letraEncript = recibir.readInt();

letra = desencriptarLetra(letraEncript);

cliente.close();

return letra;

}

public static Coordenada encriptarCoordenada(Coordenada c) {

Coordenada encript = new Coordenada();

encript.setX(c.getX() \* key);

encript.setY(c.getY() \* key);

return encript;

}

public static String desencriptarLetra(int encript) {

String decript = "";

int aux = (encript / key);

switch (aux) {

case 1: decript = "P"; break;

case 2: decript = "C"; break;

case 3: decript = "O"; break;

case 4: decript = "G"; break;

case 5: decript = "E"; break;

case 6: decript = "S"; break;

case 7: decript = "K"; break;

}

return decript;

}

public JTextArea format(JTextArea j, String nombre, Coordenada c) {

j.setText(nombre);

j.setFont(new Font("Arial", Font.PLAIN, 25));

j.setEditable(false);

if (j.getText().equalsIgnoreCase("C"))

j.setBackground(Color.GREEN);

if (j.getText().equalsIgnoreCase("P"))

j.setBackground(Color.DARK\_GRAY);

if (j.getText().equalsIgnoreCase("O"))

j.setBackground(Color.YELLOW);

if (j.getText().equalsIgnoreCase("G"))

j.setBackground(Color.MAGENTA);

if (j.getText().equalsIgnoreCase("K"))

j.setBackground(Color.RED);

if (j.getText().equalsIgnoreCase(" "))

j.setBackground(Color.WHITE);

if (posicionActual.equals(c))

j.setBackground(Color.ORANGE);

return j;

}

public void dibujarLaberinto(int size, boolean reiniciar) {

if (!reiniciar) {

int cSize = 30;

JTextArea txtmensajes;

int a = 70, b = 50;

int i = 0, j = 0;

while (i < size) {

txtmensajes = new JTextArea();

txtmensajes.setFont(new Font("Arial", Font.PLAIN, 25));

txtmensajes.setBounds(a, b, cSize, cSize);

txtmensajes.setEditable(false);

laberinto[i][j] = txtmensajes;

while (j < size - 1) {

b += cSize + 5;

j++;

txtmensajes = new JTextArea();

txtmensajes.setFont(new Font("Arial", Font.PLAIN, 25));

txtmensajes.setBounds(a, b, cSize, cSize);

txtmensajes.setEditable(false);

laberinto[i][j] = txtmensajes;

}

b = 50;

j = 0;

a += cSize + 5;

i++;

}

}

}

public boolean checkOro() throws NumberFormatException, UnknownHostException, IOException {

boolean respuesta = false;

boolean tomado = false;

try {

for (Coordenada c : oroYaTomado) {

if (c.equals(posicionActual)) {

consola("Aca no hay mas oro. Ya lo agarraste!");

tomado = true;

}

}

if (tomado == false) {

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX(), posicionActual.getY())).equalsIgnoreCase("O")) {

oro++;

respuesta = true;

laberinto[posicionActual.getX()][posicionActual.getY()].setText("X");

oroYaTomado.add(new Coordenada(posicionActual.getX(), posicionActual.getY()));

}

}

} catch (Exception e) {

System.out.println("\n\n\n\nHA OCURRIDO UN ERROR EN ORO: " + e.getMessage());

e.printStackTrace();

}

return respuesta;

}

public boolean checkGuardia() throws NumberFormatException, UnknownHostException, IOException {

boolean respuesta = false;

try {

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX(), posicionActual.getY())).equalsIgnoreCase("G")) {

if (oro == 0) {

JOptionPane.showMessageDialog(null, "No tiene oro para pagar al guardia. Perdirse!");

posicionActual.setX(xEntrada);

posicionActual.setY(yEntrada);

respuesta = true;

salir();

} else {

oro--;

JOptionPane.showMessageDialog(null, "Guardia! Perdiste 1 oro. Ahora tienes: " + oro + " de oro");

}

}

} catch (Exception e) {

System.out.println("\n\n\n\nHA OCURRIDO UN ERROR EN GUARDIA" + e.getMessage());

e.printStackTrace();

}

return respuesta;

}

public boolean checkSalida() throws NumberFormatException, UnknownHostException, IOException {

boolean respuesta = false;

try {

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX(), posicionActual.getY())).equalsIgnoreCase("S")) {

if (llave == true) {

JOptionPane.showMessageDialog(null, "Ganaste! Llegaste a la salida");

posicionActual.setX(xEntrada);

posicionActual.setY(yEntrada);

respuesta = true;

salir();

} else {

JOptionPane.showMessageDialog(null, "No podes salir, todavia no encontraste la llave!");

respuesta = false;

}

}

} catch (Exception e) {

System.out.println("\n\n\n\nHA OCURRIDO UN ERROR EN GUARDIA" + e.getMessage());

e.printStackTrace();

}

return respuesta;

}

public boolean checkLlave() throws NumberFormatException, UnknownHostException, IOException {

boolean respuesta = false;

try {

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX(), posicionActual.getY())).equalsIgnoreCase("K")) {

llave = true;

JOptionPane.showMessageDialog(null, "Encontraste la llave. Ahora encontra la salida!");

respuesta = true;

}

} catch (Exception e) {

System.out.println("\n\n\n\nHA OCURRIDO UN ERROR EN ORO: " + e.getMessage());

e.printStackTrace();

}

return respuesta;

}

public boolean limites(String direccion) throws NumberFormatException, UnknownHostException, IOException {

boolean resultado = true;

if (direccion.equals("derecha")) { // Derecha

if (posicionActual.getX() == 19) {

resultado = false;

}

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX() + 1, posicionActual.getY())).equalsIgnoreCase("P")) {

resultado = false;

}

}

if (direccion.equals("arriba")) { // Arriba

if (posicionActual.getY() == 0) {

resultado = false;

}

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX(), posicionActual.getY() - 1)).equalsIgnoreCase("P")) {

resultado = false;

}

}

if (direccion.equals("abajo")) { // Abajo

if (posicionActual.getY() == 19) {

resultado = false;

}

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX(), posicionActual.getY() + 1)).equalsIgnoreCase("P")) {

resultado = false;

}

}

if (direccion.equals("izquierda")) { // Izquierda

if (posicionActual.getX() == 0) {

resultado = false;

}

if (devolverLetra(config.getIp(), config.getPuerto(),

new Coordenada(posicionActual.getX() - 1, posicionActual.getY())).equalsIgnoreCase("P")) {

resultado = false;

}

}

return resultado;

}

public void salir() throws NumberFormatException, UnknownHostException, IOException {

devolverLetra(config.getIp(), config.getPuerto(), new Coordenada(-1, -1));

this.setVisible(false);

System.exit(0);

return;

}

public void mover(String direccion) {

try {

if (limites(direccion)) {

if (direccion.equals("izquierda"))

posicionActual.setX(posicionActual.getX() - 1);

if (direccion.equals("arriba"))

posicionActual.setY(posicionActual.getY() - 1);

if (direccion.equals("derecha"))

posicionActual.setX(posicionActual.getX() + 1);

if (direccion.equals("abajo"))

posicionActual.setY(posicionActual.getY() + 1);

if (checkOro()) {

format(new JTextArea(), " ", posicionActual);

}

checkGuardia();

checkSalida();

if (llave == false)

checkLlave();

mostrarVecinos();

}

} catch (NumberFormatException | IOException e1) {

e1.printStackTrace();

}

}

class MyKeyListener implements KeyListener {

@Override

public void keyTyped(KeyEvent e) {

}

@Override

public void keyPressed(KeyEvent e) {

// System.out.println("keyPressed=" + e.getKeyCode());

oroLabel.setText("Oro: " + Integer.toString(oro));

String direccion = "";

switch (e.getKeyCode()) {

case 37: // Izquierda

direccion = "izquierda";

break;

case 38: // Arriba

direccion = "arriba";

break;

case 39: // Derecho

direccion = "derecha";

break;

case 40: // Abajo

direccion = "abajo";

break;

}

mover(direccion);

}

@Override

public void keyReleased(KeyEvent e) {

// TODO Auto-generated method stub

}

}

public void keyReleased(KeyEvent e) {

}

}

class Login implements Serializable {

private String ip;

private String puerto;

private String user;

private String pass;

public Login(String ip, String puerto, String user, String pass) {

this.ip = ip;

this.puerto = puerto;

this.user = user;

this.pass = pass;

}

public Login() {

this.user = "unlogged";

}

public String getIp() {

return ip;

}

public String getPuerto() {

return puerto;

}

public void setUser(String user) {

this.user = user;

}

public String getUser() {

return user;

}

public String getPass() {

return pass;

}

}

class Configuracion {

private String ip;

private String puerto;

public Configuracion(String ip, String puerto) {

this.ip = ip;

this.puerto = puerto;

}

public String getIp() {

return ip;

}

public void setIp(String ip) {

this.ip = ip;

}

public String getPuerto() {

return puerto;

}

public void setPuerto(String puerto) {

this.puerto = puerto;

}

}