PROGRAMACIÓN DE AJEDREZ HASKELL

Temática: Juegos Visuales Avanzados

Autores:

- **Antonio José Díaz Gonzalez (antdiagon1) antoniojosedigo@gmail.com**
- **Samuel Buzón Gil (sambuzgil) samuelbg5c@gmail.com**

Resumen

El proyecto consiste en la realización del mítico juego del Ajedrez, (piezas, tablero movimientos, modos de juego, …), implementado en Haskell.

1º Introducción:

Con el trabajo se pretendía modelar todo aquello relacionado y necesario para poder jugar una partida de ajedrez.
En una partida de ajedrez deberemos de tener un tablero donde se situarán las fichas, con un orden específico dicho tablero deberá para jugar una partida de ajedrez estándar deberá de ser de 8x8 y contendrá las siguientes fichas:

- Peones: contendrá 16 peones (8 Blancos, 8 Negros) colocados a lo largo de las filas 2 y 7. Disponen de un movimiento de 1 solo salto hacia adelante, no podrán moverse a la siguiente fila si existe alguna ficha enemiga o amiga en esta y podrán eliminar fichas contrarias en diagonal a 1 salto de longitud. Estos deberán de tener la capacidad de poder convertirse en Reinas si llegan a la última fila posible del bando contrario.

- Torres: contendrá 4 torres (2 Blancas, 2 Negras). Estas podrán realizar movimientos de forma horizontal o vertical, hacia adelante o atrás de la longitud que se desee, siempre que no exista una ficha aliada en su camino o una ficha enemiga la cual podrá eliminar.

- Alfiles: contendrá 4 alfiles (2 Blancos, 2 Negros). Este podrá realizar movimientos de forma diagonal hacia adelante o atrás siempre que no exista una ficha aliada o enemiga la cual podrá eliminar.

Caballos: contendrá 4 caballos (2 Blancos, 2 Negros). Este podrá realizar movimientos de forma que se desplaza dos casillas horizontalmente y una casilla verticalmente o dos casillas en posición vertical y una horizontal, siempre que este hueco este vacío o se encuentre una ficha rival la cual eliminará.

- Reinas: contendrá 1 reina (1 Blancos, 1 Negros). Esta ficha resulta de la unión de movimientos de Torres y Alfiles (diagonales, verticales y horizontales), y podrá realizarlos siempre que no exista ninguna ficha aliada en su camino o enemiga la cual eliminará.

- Reyes: contendrá 1 rey (1 Blancos, 1 Negros). Esta ficha podrá realizar movimientos verticales, horizontales y diagonales de longitud máxima 1, podrá eliminar fichas enemigas si existen en su rango y si esta ficha, el rey es eliminada el jugador contrario ganará la partida.

2º Estructura del código

El código está dividido en 3 archivos/módulos, los cuáles se dividen las funciones de la siguiente forma:

-Tablero.hs -> este archivo se encarga de crear el tablero de juego, así como de dar información como las fichas disponibles de un jugador en un momento, o si se ha terminado la partida.

-Movimientos.hs-> este archivo tiene el código referente a los movimientos de las fichas, tanto comprobar si un movimiento es válido, como cambiar un pero por una reina al llegar al final del tablero, como el propio movimiento de las piezas.

-Juegos.hs-> aquí nos encontramos con el código para poder jugar tanto contra otro jugador como contra la IA, son las funciones IO y las necesarias para obtener ciertos datos como las posiciones de la ficha que queremos mover, también nos encontramos el código de generación de movimiento de nuestra IA (es una IA “tonta”, realiza un movimiento aleatorio a una ficha aleatoria de las que dispone, siempre será un movimiento válido que cumpla con las normas del ajedrez).

Nuestro código está basado en Matrices del tipo Matrix, pues nos será de utilidad para obtener valores y cambiarlos con facilidad.Además usamos listas por comprensión es

gran parte de las funciones, como pueden ser “vacio n” para crear n listas que tengan 8 huecos (en nuestro caso los huecos del tablero vienen representados por “ * “) o “recorridoPieza” para obtener el recorrido que va a realizar una pieza y así comprobar si es válido o no.

Un ejemplo de recursividad lo encontramos en la función “filaString”, en la cual convertimos una lista a un string para poder imprimirlo por pantalla.

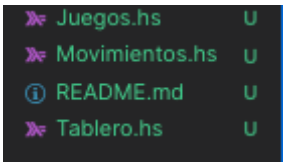
Como ejemplo de guardas podemos ver funciones como “recorridoTor” o “recorridoRei” las cuales nos dan el recorrido que queremos que realice una torre o reina. Además podemos encontrar ejemplo de usos de patrones en funciones como “finalizado” para comprobar en caso de cada jugador si ha ganado, o “peonReina” para ver si despues del movimiento de cada jugador, un peón suyo ha llegado al final de tablero para convertirse en reina.

Como funciones de orden superior podemos encontrar el uso de filter en la función “peonReina” o foldr en la función “fichasJugador” y “fichasLinea”.

3º Cómo compilar y ejemplos de uso.

A al hora de compilar y ejecutar el proyecto se realizará la siguiente secuencia de comandos:

1. Se descargará y se mantendrán todos los archivos extraídos del zip en mismo directorio.



Ejemplo de estructuración correcta.

2. Será necesario tener instalado Haskell y tener instalados los plugins de Haskell Syntax Highlighting y Haskell Runner de Visual Studio, además de módulos comentados en la siguiente sección.

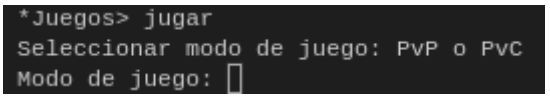
3. Una vez tengamos el entorno configurado realizaremos “ :l Juegos.hs ” en la terminal de Visual Studio (Tras esto tendremos compilado el programa y podremos ejecutar el código.

```
*Juegos> :l Juegos.hs
[1 of 3] Compiling Movimientos      ( Movimientos.hs, interpreted )
[2 of 3] Compiling Tablero          ( Tablero.hs, interpreted )
[3 of 3] Compiling Juegos           ( Juegos.hs, interpreted )
Ok, three modules loaded.
```

Ejemplo de compilación exitosa.

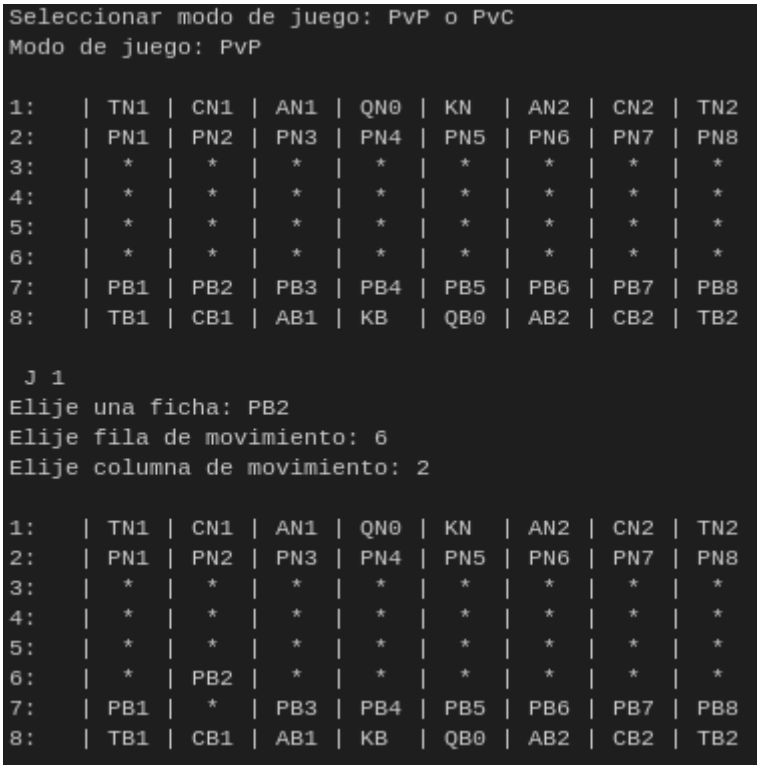
4. Para comenzar una partida deberemos escribir por la terminal “jugar” y comenzará la partida.

5. Tras esto deberemos elegir diferentes parámetros como el *modo de juego* Persona vs Persona ó Persona vs Computador. (PvC realizará movimientos de forma aleatorio)



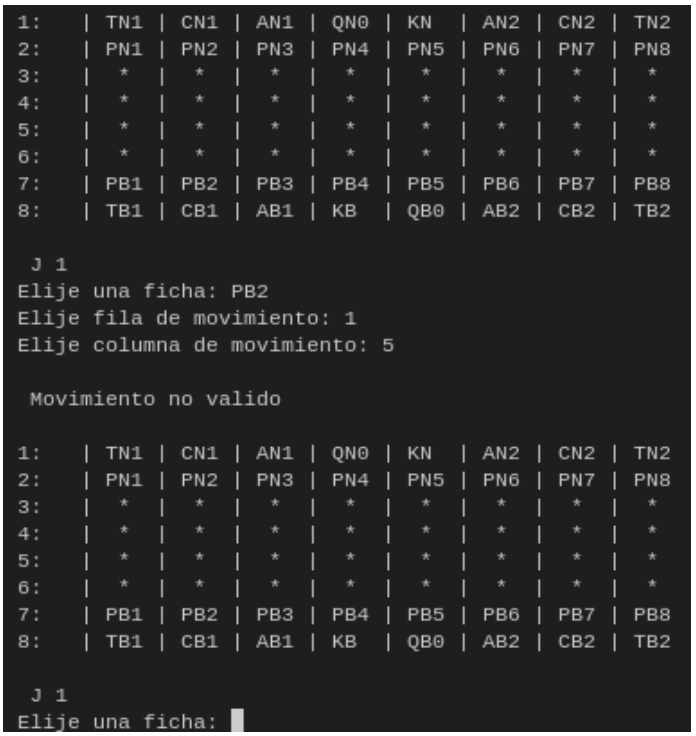
Ejemplo de cómo elegir modo de juego

6. Una vez elegido el modo de juego empezamos a mover las fichas del tablero para ello escribiremos el nombre que está indicado en la terminal en la figura del tablero.



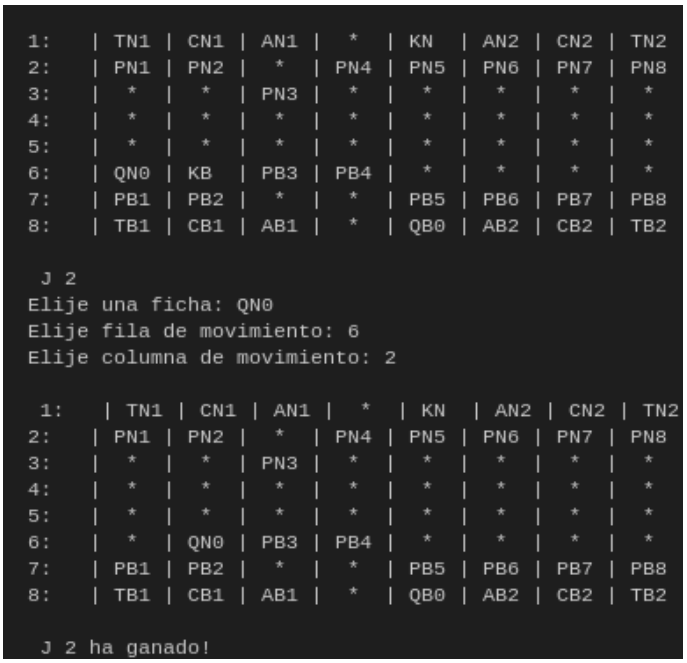
Ejemplo de cómo realizar un movimiento.

7. Si el usuario realiza movimientos incorrectos para una fila se avisará por pantalla.



Ejemplo de movimiento no válido.

8. Por último el juego acabará si alguno de los jugadores logra acabar con el rey enemigo.



Ejemplo de partida finalizada.

Librerías externas usadas

Como librerías externas la única que hemos usado la cual es necesaria su instalación es la de Matrix.

Para ello ejecutaremos el siguiente comando: cabal install matrix

Además de esta también hemos creado y usado otros módulos como son:

```
import Tablero
import Movimientos

import Data.Maybe
import Data.List
import Data.Matrix
import Data.Char
import Data.String
import System.Random
import Data.Time.Clock
import Data.Time.LocalTime
import Data.Array
```