Memoria proyecto

-

Juego del molino

Yamandú Sotelo Souto
Diego Gálvez Ruiz

Índice

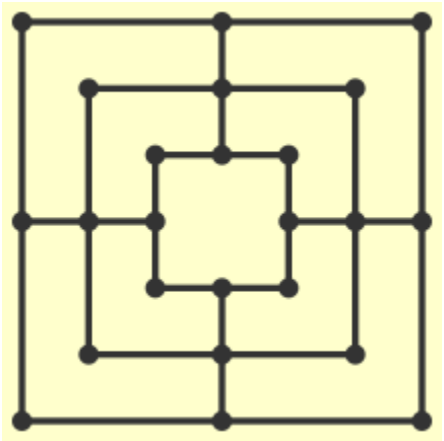
1. Resumen

En el siguiente proyecto se ha programado el juego de mesa del molino para el SO Windows usando el lenguaje de programación Prolog mediante la interfaz SWI. El juego cuenta con dos modalidades de dificultad, la cual será escogida al principio de la partida por el usuario y acto seguido jugará contra la máquina programada para intentar ganar al jugador. Una vez terminada la partida, se le preguntará al usuario si desea seguir jugando o salir de la aplicación.

Para comenzar la aplicación, basta con abrirlo en el entorno SWI he introducir el predicado molinero.

2. Descripción del juego

Cada jugador dispone de nueve piezas, u "hombres", que se mueven en el tablero entre veinticuatro intersecciones. El objetivo del juego es dejar al oponente con menos de tres piezas o sin movimiento posible.



El tablero al principio del juego, antes de que ninguna pieza sea colocada.

El juego sigue las siguientes reglas:

- El juego comienza con un tablero vacío.
- Los jugadores se turnan para colocar sus piezas en las intersecciones vacías.

- Una vez que todas las dieciocho piezas se han colocado, los jugadores se turnan moviendo.
- Para moverse, el jugador desliza una de sus piezas a lo largo de una línea en el tablero a una intersección vacía adyacente. Si no puede hacerlo, ha perdido el juego.
- Si un jugador es capaz de formar una fila de tres piezas a lo largo de una de las líneas del tablero (tanto en la fase de colocación, como en la de movimiento), tiene un "molino" y puede eliminar una de las piezas de su oponente en el tablero.
- No se puede quitar una pieza que forme parte de un molino, a no ser que todas las piezas restantes formen parte de un molino.
- Las piezas quitadas no podrán ser colocadas de nuevo.
- Cualquier jugador que es reducido a dos piezas no es capaz de eliminar más piezas del oponente y, por tanto, pierde la partida.

3. Desarrollo del proyecto

El juego se define mediante un bucle principal del juego en el cual se bifurca al movimiento que hace el jugador sobre el tablero y el movimiento que hace la máquina. Para representar el tablero se han usado 3 listas de longitud 8, cada una representa las posiciones de los 3 cuadrados que forman el tablero. Además de tener en estas listas la información del tipo de ficha que hay en cada posición guardamos en la base de conocimiento 2 listas con las posiciones en las que ha puesto cada jugador. También es necesario saber si estamos en la fase de posicionar fichas o moverlas, ya que no podemos comprobar si el juego se ha bloqueado hasta que se hayan puesto todas las fichas, de modo que se aserta en la base de conocimiento la fase en la que estamos.

Elegimos esta representación para el tablero dada la simplicidad para pintarlo por pantalla y para comprobar rápidamente mediante las listas las posiciones del rival del juego. Gracias a esto la maquina intentará quitar antes las fichas peligrosas del rival, cuando haga un molino, mediante el predicado “buscaPeligrosas” y lo hace en un coste lineal según el número de elementos de la lista.

Para implementar la “inteligencia” del rival, se ha empleado el algoritmo minimax con poda alfa-beta y la representación del tablero antes mencionada como nodos del árbol. En la heurística se ha tenido en cuenta tanto el número de fichas que hay en el tablero, como las que están formando molinos y el número de fichas por cuadrado del tablero.

Al terminar la partida no solo se informa de cómo ha acabado la partida sino que se propone seguir jugando al usuario.

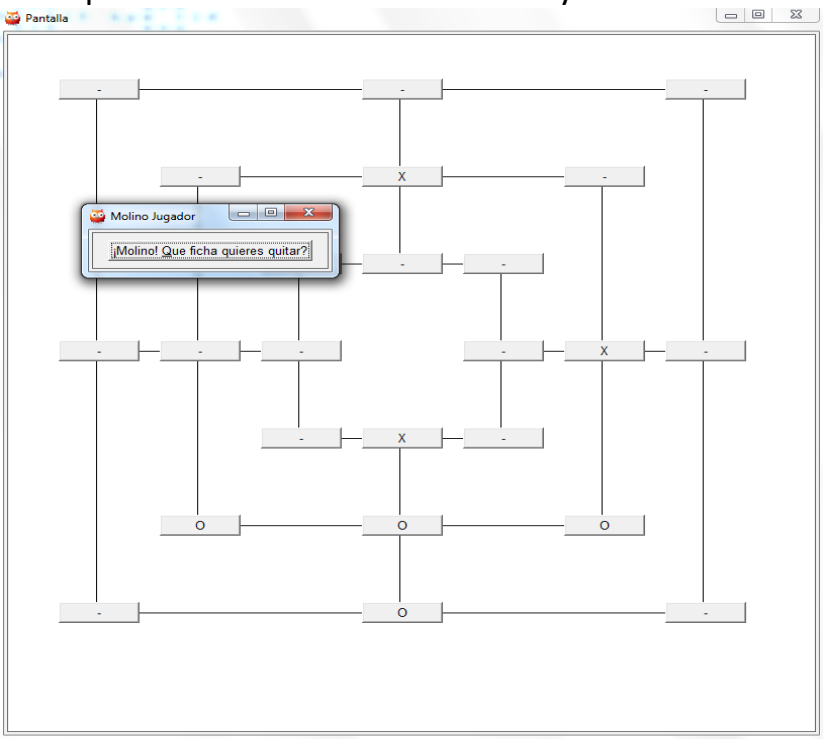
3.1 Desarrollo de la interfaz con XPCE

Una vez terminada la funcionalidad del juego, se buscaron maneras de hacerlo más atractivo y fácil de usar para el usuario.

Con este objetivo se exploraron métodos para integrar interfaces gráficas con Prolog, encontrando en la librería XPCE para gráficos una solución aceptable.

La integración con esta librería (y sus limitaciones), así como las formas de interactuar con el juego (eventos de botones en lugar de escritura en consola), obligó a modificar el código en gran medida para mantener el mismo flujo de programa. Una consecuencia de esto es que en este modo con interfaz si se quiere volver a jugar contra la máquina hay que cerrar y volver a abrir.

El resultado es una interfaz bastante más legible y fácil de usar, en la cual las fichas del usuario se representan mediante un círculo y las fichas del oponente máquina mediante un aspa.



4. Conclusiones

Durante el desarrollo del proyecto la mayor dificultad encontrada fue la de adaptar un algoritmo de poda alfa beta de un código de otra aplicación al código. Esto suponía comprender con claridad el funcionamiento del código y aun así al unirlo seguía habiendo problemas que no detectamos.

A pesar de los problemas encontrados, la programación mediante un lenguaje declarativo de un juego de estas características aporta una inmediata ventaja al usar algoritmos recursivos en árboles, ya que el propio lenguaje incluye la recursión necesaria.

5. Bibliografía

Código poda alfa-beta: https://code.google.com/p/tictactoe-prolog/

XPCE: http://www.dccia.ua.es/logica/prolog/docs/ProgGUI.pdf

SWI-Prolog: http://www.swi-prolog.org/