LABORATORIO DE PROGRAMACIÓN III (Curso 2011/2012)

PRÁCTICA 3
FECHA LÍMITE: Viernes 25 de mayo a las 11h

Objetivos de la práctica:

1. Inserción de imágenes.
2. Árboles de juego.
3. Algoritmo Minimax con movimientos alternativos.

DESCRIPCIÓN DEL JUEGO

Inicialización. Desarrolla una aplicación Java para jugar a las damas que permita seleccionar el tamaño del tablero (que puede ser de 8x8, 10x10 o 12x12) y la profundidad máxima k del árbol de juego.

El juego consta de un número de piezas (peones) por jugador también variable (a tableros más grandes, más cantidad de fichas) rojas y azules, colocadas en las casillas de un mismo color en las primeras filas de cada lado del tablero (casillas alternas). A lo largo de la partida hay dos tipos de piezas bien diferenciadas: peones y damas. El juego comienza con todos los peones de cada jugador situados sobre el tablero en sus respectivas casillas. Los peones se mueven en diagonal tanto hacia la derecha como hacia la izquierda, siendo los movimientos de los peones siempre en avance de una casilla. Cuando un peón llegue a la primera fila del bando contrario, se corona, convirtiéndose en dama, la cual se distingue del resto de piezas porque mostraremos una figura distinta (en la Figura, la imagen para la dama lleva una K de King). Las damas, al igual que los peones, tienen movimiento diagonal de una casilla, pero con la posibilidad de desplazarse tanto hacia adelante como hacia atrás.

Capturas. El peón captura en diagonal saltando por encima de la ficha contraria que va a ser capturada, cayendo sobre la casilla inmediatamente detrás de ésta (en el sentido de la captura), y siempre que el que captura esté en una casilla adyacente al capturado, y que la casilla inmediatamente detrás esté libre para que acabe el movimiento. La dama puede capturar tanto hacia adelante como hacia atrás.

Tanto con dama como con peón, si tras una captura, la pieza en cuestión estuviera en situación de realizar una nuevamente, tal nueva captura se llevará a cabo de forma encadenada, y así sucesivamente mientras se diera tal circunstancia de poder seguir capturando. Su movimiento y su turno terminan cuando ya no hay más piezas para capturar.

Los movimientos se efectúan pulsando con el ratón una primera vez sobre la ficha que se quiere mover (origen) y a continuación una segunda vez sobre la posición del tablero a la que se quiere mover (destino). Se puede *deseleccionar* la jugada pulsando la segunda vez sobre la misma posición de origen.

Finalización. La finalidad del juego es la captura de la totalidad de las piezas contrarias o el bloqueo de las mismas, de forma que tocándole(s) jugar no le(s) sea posible realizar movimiento.

PARTE 1: DAMAS DOS JUGADORES

Desarrolla una aplicación Java que implemente el juego de damas para dos jugadores tal y como se ha descrito en el apartado anterior. Organiza el código en tres paquetes `aplicacion`, `interfaz` y `movimientos`. En el paquete `aplicacion` implementaremos el método `main` que permite ejecutar la aplicación. A continuación, explicamos el resto de paquetes.

Paquete `interfaz`: Diseño del Interfaz Gráfico.

Define las siguientes clases dentro del paquete `interfaz`:

- La clase abstracta `Tablero` que extiende la clase `Canvas` y define, entre otros, los siguientes métodos:
  - El método `public void paint(Graphics g)` que se encarga de pintar el tablero utilizando los métodos apropiados de la clase `Graphics`.

    - Para pintar las figuras sobre el tablero, `paint` invoca al método `dibujarExtra`. Declara en esta clase el método `abstract void dibujarExtra(Graphics g)` con la idea de que lo implementen las subclases de `Tablero` como se describe a continuación.

- La subclase `TableroFichas` extiende la clase `Tablero` e implementa la interfaz `MouseListener` para poder detectar los movimientos seleccionados sobre el tablero. Como atributo tiene un objeto de tipo `TableroJuego`, que definiremos en el siguiente paquete, y que contiene la información del contenido del tablero que queremos dibujar en el Canvas. Esta clase debe implementar, entre otros, los siguientes métodos:

    - `void dibujarExtra(Graphics g)`: Coloca imágenes para los peones y las damas consultando el estado del tablero de Juego. Además muestra en la parte sur del panel principal el estado con el turno, etc. Este método se invoca desde *paint* para que se actualicen los movimientos de la ficha durante la ejecución.
    - `public void mousePressed(MouseEvent e)`: Lee la posición del Canvas pulsada e identifica a qué fila y columna del tablero corresponde. A continuación invoca al método `procesaMovimiento` que definiremos en el paquete `movimientos` y se encarga de realizar el movimiento, si es posible.

Para acceder a una imagen guardada en disco, por ejemplo, `nombre.gif`, podemos utilizar el siguiente método: `Image imagen= getToolkit().getImage(''nombre.gif'');` que devuelve una imagen que se puede pintar en el Canvas invocando al método: `g.drawImage(imagen,posX,posY,this)`.

Paquete `movimientos`: Programación del Juego.

En este paquete incluiremos la programación del tablero de juego y los movimientos de los jugadores. Define las siguientes clases:

- La clase `TableroJuego` que contiene como atributo la matriz de enteros que representa el tablero. Elegimos distintos valores enteros para indicar cuando una casilla está libre, está ocupada por un peón del jugador 1, una dama del jugador 1, un peón del jugador 2 o una dama del jugador 2.
- La clase `ComprobadorMov` que extiende la clase `TableroJuego` y procesa los movimientos para decidir si es posible llevar a cabo una captura, un desplazamiento, etc y realizarlos. Tiene como atributos las posiciones de origen y destino seleccionadas, el usuario que tiene el turno, el estado que contiene el String que se muestra en la parte sur del Panel (ver Figura), entre otros que se explicarán en clase.

PARTE 2: DAMAS CON ORDENADOR

En la segunda parte, implementaremos el juego de Damas con un único usuario que juega contra el ordenador. Al comenzar el juego elegiremos la estrategia de juego del ordenador:

- *Ofensiva.* Intenta capturar el mayor número posible de fichas del adversario.
- *Defensiva.* Intenta mantener el mayor número posible de fichas a salvo (es decir, que no pueden ser capturadas por el adversario).
- *Mi estrategia.* Con la experiencia adquirida al implementar las estrategias anteriores, diseña tu propia estrategia de juego para el ordenador.

Se implementarán tres métodos `heuristicaOfensiva`, `heuristicaDefensiva`, `miHeuristica` que asignan un valor numérico a un tablero de manera que a mayor valor, mejor es la jugada. Al inicializar la aplicación se eligirá una de las tres estrategias.

El método `mueveOrd` es el encargado de determinar la jugada del ordenador de entre todos los posibles movimientos. La jugada del ordenador se eligirá ensayando la heurística seleccionada sobre los tableros resultantes de realizar todos los posibles movimientos que el ordenador puede efectuar, incluyendo los movimientos normales y las capturas, hasta un nivel de profundidad k en el árbol de juego (recuerda que k es un parámetro de entrada).

Ten en cuenta que desde una misma posición, pueden ser posibles múltiples movimientos y todos se han de ensayar. Para ello, has de disponer de un método `clonarTablero` en la clase `TableroJuego`, que nos permita simular un movimiento concreto y calcular la heurística sin modificar el tablero original. Además, implementa una nueva clase `MovOrdenador` que tiene como atributos la posición origen y destino

a la que se quiere mover el ordenador y el valor de dicho tablero de acuerdo a la heurística seleccionada. La idea es que, al terminar el proceso, el método `mueveOrd` devuelva un objecto de tipo `MovOrdenador` con la mejor jugada posible.

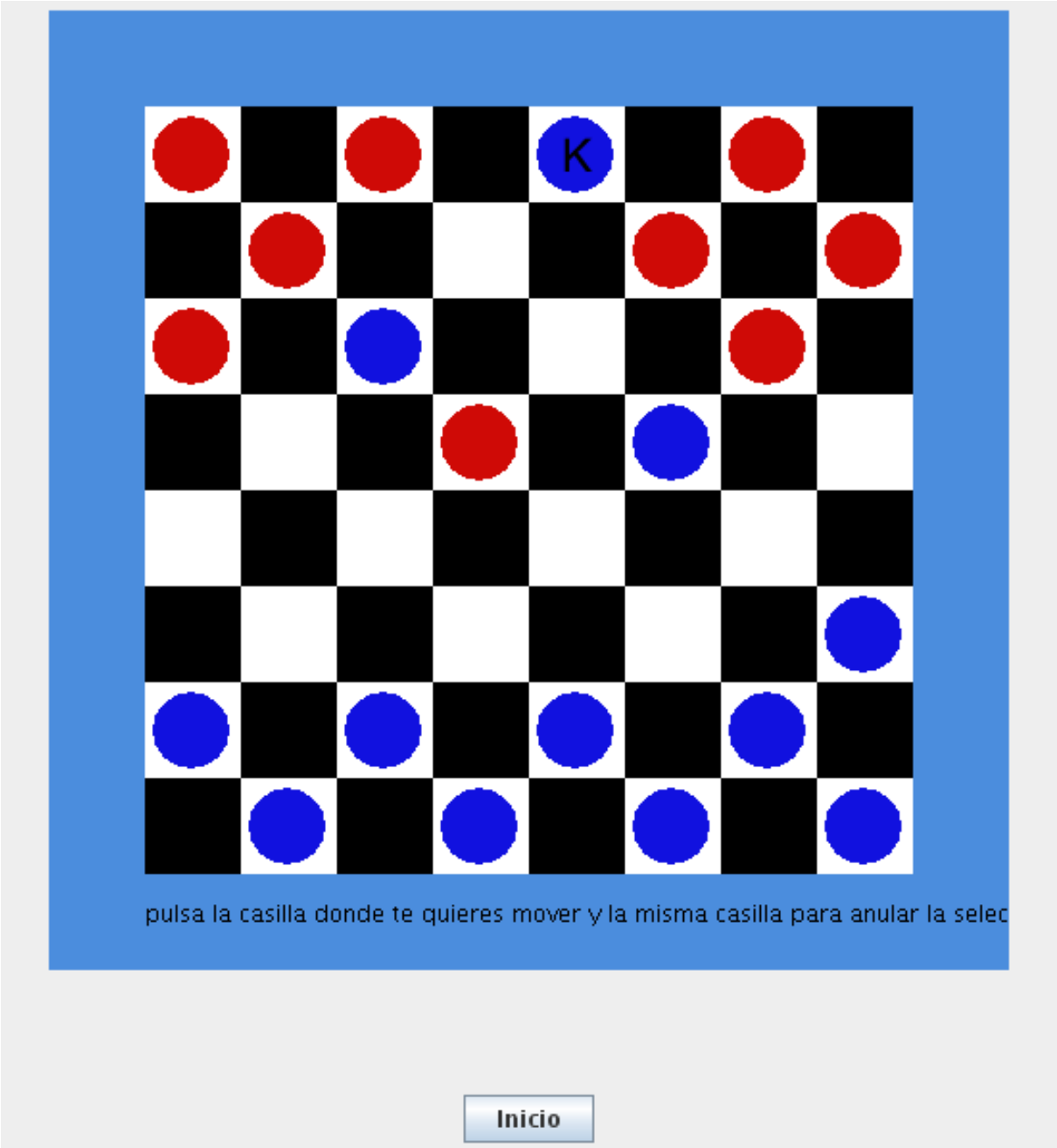

Figura 1: Vista de la aplicación

Los detalles de implementación se ampliarán en clase.