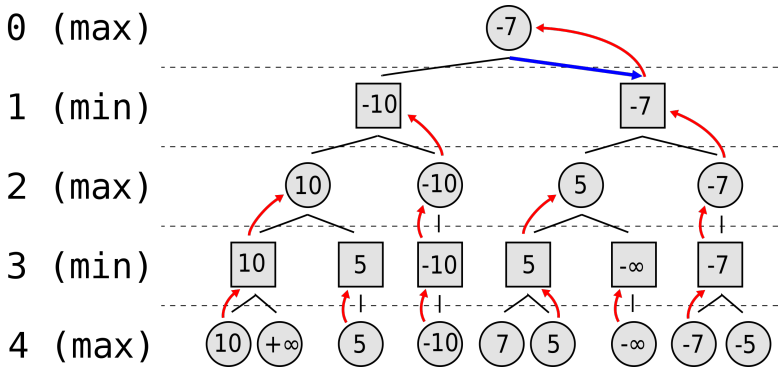
DESCRIPCIÓN DE LAS CLASES Y DE LAS ESTRUCTURAS – Entrega 2

ALGORITMOS

Algoritmo Minimax

Algoritmo que elige el mejor movimiento para ti mismo suponiendo que el contrincante escogerá el movimiento que más daño te hace. Para implementar este algoritmo se genera un árbol de búsqueda, cuyos nodos son los posibles movimientos. Se les debe asignar a los nodos hoja una puntuación según una función de evaluación determinada. A continuación, se calcula el valor de los nodos superiores a partir de los valores de los inferiores. Según si el nivel es MAX o MIN se le asignará a cada nodo del nivel, el valor máximo o mínimo de entre sus hijos. Seguir este proceso hasta la raíz, de modo que devuelva jugada que más nos beneficia

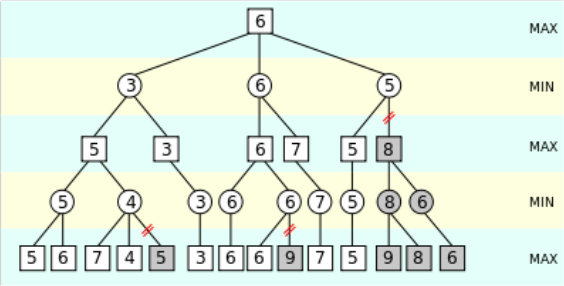


Fuente: *https://es.wikipedia.org*

Algoritmo Poda Alfa-Beta:

Este algoritmo es una mejora del algoritmo Minimax. Esta técnica de búsqueda reduce el número de ramas exploradas del árbol de búsqueda creado por el minimax. Este algoritmo utiliza dos parámetros: alfa y beta. Alfa es el mejor movimiento para max, mientras que Beta es el mejor movimiento para min. Los valores de alfa y beta se actualizan según se recorre el árbol y la poda ocurre

cuando el valor del nodo en el que se está es peor que alfa o beta, según estemos en nivel max o min.



Fuente: *https://es.wikipedia.org*

FUNCIONES DE EVALUACIÓN:

A la hora de dar puntuación al movimiento según el estado actual del tablero, las dos funciones de puntuación tienen en cuenta si la ficha está expuesta, está protegida, si amenaza a otras fichas y si protege a otras fichas.

Función de puntuación "inteligente"

Esta función de puntuación no solamente tiene en cuenta lo expuesto arriba, sino que también tiene en cuenta cuántas y qué fichas se ven implicadas en cada acción. Además, también da más importancia a unas acciones que a otras.

Función de puntuación "tonta"

Esta función funciona igual que la inteligente, pero no tiene en cuenta ni cuántas ni qué fichas están implicadas en el movimiento. Esto resulta en puntuaciones mucho más homogéneas y bajas.

CLASES DOMINIO

Clase Ficha

Clase abstracta que crea un objeto Ficha con los atributos: nombre, posición x, posición y, clau, identificador, color, puntuación base y muerte temporal. Esta clase contiene una función para actualizar la posición de la ficha, otra para comprobar si la posición de la ficha dentro de tablero es correcta y, por último, una función abstracta que retorna una lista de posiciones (x,y) a las que puede ir la ficha.

Clases Peo, Alfil, Torre, Rey, Reina y Cavall

Estas clases son hijas de Ficha. Todas las clases contienen la creadora propia y la función con una implementación propia de cada clase de la función que retorna todos los movimientos posibles. Además, las clases Peo y Rey sobre escriben la función definida en Ficha que comprueba si la posición de la ficha es correcta, ya que estas dos fichas son las únicas que pueden estar en posiciones inválidas.

Clase Taulell

Clase que crea un objeto Taulell con los atributos: fichas blancas, fichas negras, matriz, fen, quien ha empezado, a quien le toca jugar, la posición (x, y) del rey blanco y la posición (x, y) del rey negro. Las fichas blancas y negras se guardan en dos vectores de objetos Ficha, uno para las blancas y otro para las negras.

Los métodos que contiene esta clase son: jaque, mate, mover una ficha, resucitar una ficha, promover un peón, despromover una ficha, todos_movs (que retorna el conjunto de movimientos de todo un color), camino_ok (que utilizamos para saber si es posible realizar o no los movimientos de una ficha), puntua, amenaza, ataca, asegura (estas tres últimas son utilizados por puntua para saber si una pieza está expuesta, est protegida, cuántos puntos amenaza y cuántos asegura) y dos funciones para transformar de fen a matriz y de matriz a fen.

Clase Arbol

Clase que crea el árbol de búsqueda que utilizarán los algoritmos Minimax y Poda Alba-Beta. Esta compuesto por Nodos y la raíz del árbol es la situación actual del tablero.

Clase Nodo

Clase que crea un objeto de tipo Nodo (elemento del árbol), con los atributos: ficha, posición final, hijos, promoción y puntuación. Cada nodo es la representación de un posible movimiento.

Clase IAFacil

Clase que crea el árbol de búsqueda e implementa el algoritmo minimax con la puntuación tonta. Para crear el árbol utiliza la función llenar_arbol y para implementar el algoritmo, la clase contiene tres funciones: best_move, valmin y

valmax. BestMove llama a valmin y entonces comienza una recursión "compartida" entre valmin y valmax que, en cuanto acaba, retorna el mejor movimiento.

Clase IAMedia

Esta clase utiliza también el algoritmo minimax, con la diferencia que ahora a los nodos del árbol se les asigna la puntuación según la puntuación "inteligente". Las funciones que contiene son las mismas que IAFacil.

Clase IADificil

Clase que implementa el algoritmo poda alfa-beta y utiliza la función "inteligente" para puntuar los nodos. Contiene las mismas funciones que las dos clase anteriores, aunque, obviamente, las implementaciones de Valmin y Valmax cambian, ya que ahora utilizan los parámetros alfa y beta.

Clase Player

Crea un objeto de tipo Player con los atributos username, password y código de usuario. Player representa a la persona que ha hecho login en la aplicación.

Clase Problema

Se crea un objeto de tipo Problema que contiene el código del problema, el código del usuario al que pertenece, el numero de jugadas para el mate (mate en n), dos variables que se utilizan para la validación del problema (depth, width), el jugador que empieza moviendo, la dificultad y el fen. La dificultad se establece según width, que depende del número de fichas y del tema. Esta clase contiene las funciones de validación del problema y explore_fen y checkLoser, que realizan una recursividad "compartida" y devuelven si es posible o no realizar un mate en n jugadas. Por último, checkValid compruebasi ExploreFen es true o false.

Clase Ranking

Esta clase crea un objeto Ranking, que contiene el código del ranking (que se corresponde con el código del problema público al que está asignado) y el scoreboard. El scoreboard es el conjunto de códigos de usuarios que están en el ranking y su puntuación correspondiente. Las funciones de esta clase son update_scoreboard que actualiza la puntuación del jugador, order_scoreboard que

implementa un comparador y, por último, check_score que retorna la posición del jugador en el scoreboard o -1 en caso de que no esté.

Clase Joc

Clase controlador de la capa de dominio. Funciona como comunicador entre la capa de dominio y la de presentación. Se encarga de enviar información a la capa de presentación, en formato de tipos básicos (int, String…) para que presentación los pueda interpretar. Esta información se la pide a las diferentes clases de la capa de dominio y a la capa de persistencia. Por eso, Joc es también comunicador entre la capa de persistencia y la de dominio. Esta clase tiene como variable global un objeto de tipo controlador de datos, le pide al controlador de datos toda la información que necesita y este se la envía (también en formatos de tipos básicos). Una vez recibida esta información, a veces la transforma en objetos de tipos de la clases de dominios y otras veces la utiliza en el formato recibido.

CLASES PERSISTENCIA

Los datos se guardan en ficheros. La información que envía esta capa a la capa de dominio es en formato de tipo String.

Controlador Datos

Clase controlador de la capa de persistencia. Esta clase inicializa la base de datos de jugadores, partidas, problemas y rankings. Funciona como comunicador entre la capa de dominio y las clase de la capa de datos. Esta última capa le hace peticiones al controlador de datos y este se encarga de devolverle la información que está almacenada en ficheros.

Jugador Database

Clase que administra y crea la base de datos de Jugadores. Consta de una creadora, de una función de guardar jugador (añade un jugador a la base de datos), actualizar jugador (actualiza la información de un jugador que se encuentra en el fichero) y la función get_jugadores. Esta última función devuelve toda la información que hay en el fichero y es utilizada por la capa de dominio al iniciar la aplicación para “rellenar” el vector de Jugadores que tiene en Joc.

Problema Database

Clase que administra y crea la base de datos de Problemas. Consta de una creadora, de una función de guardar problema y otra llamada get_problemas. Esta última función devuelve toda la información que hay en el fichero y es utilizada por la capa de dominio al iniciar la aplicación para "rellenar" el vector de Jugadores que tiene en Joc. A la función de guardar problema se le pasa toda la información que Joc tiene en su vector de problemas y se reescribe el fichero entero con la nueva información recibida.

Partida Database

Clase que administra y crea la base de datos de Partidas. Consta de una creadora, de una función de guardar partida, una función para eliminar partida y otra llamada get_partidas. Esta última función devuelve la información de todas las partidas de un jugador determinado. La función guardar partida, comprueba si ya hay una partida de ese problema empezada por ese jugador. Si existe, se reescribe la información de la partida pasada, si no, se añade al fichero. Y, por último, la función eliminar partida se llama en el momento en que se acaba una partida guardada.

Ranking Database

Clase que administra y crea la base de datos de Rankings. Consta de una creadora, de una función de guardar ranking y otra llamada get_rankings. Esta última función devuelve la información de todos los rankings. La función guardar Rankings, actualiza la información con la información que se le pasa por parámetro.

CLASES PRESENTACIÓN

En esta capa está implementado una especie de patrón Estado. Cada estado es una pantalla de la aplicación y se cambia de estado en tiempo de ejecución.

Controlador Presentación

Controlador Presentación es la clase que se comunica con la clase de Dominio, más específicamente, con la clase Joc. Le pide información a dominio para pasársela a las diferentes clases de Presentación.

Interface LogIn

Clase que implementa la pantalla de inicio, contiene los botones de login y registrarse. Esta pantalla corresponde al estado 0, el botón de login te lleva al estado 2 (clase LogIn) y el botón de registrarse, al estado 1 (clase Register).

Register

Clase que implementa la pantalla de registro y corresponde al estado 1. Desde esta pantalla se puede llegar solamente al estado 0, o bien a través del botón de atrás, como cuando se valida y acepta la combinación de nombre de usuario y contraseña. Para validar la combinación le envía a dominio los dos Strings (a través del controlador de presentación) y esa capa se encarga de la validación.

LogIn

Esta clase implementa la pantalla de logIn y corresponde al estado 2. Se puede llegar desde este estado al estado 0, mediante el botón de atrás, o al 3, cuando se valida y acepta la combinación de nombre de usuario y contraseña. Para validar la combinación le envía a dominio los dos Strings (a través del controlador de presentación) y esa capa se encarga de la validación.

Account

Esta clase funciona como una especie de menú en donde aparece todo lo que puede hacer el usuario. Esta pantalla es el estado 3 y desde aquí se puede llegar a los estados: 4 (botón jugar partida), 10 (crear problema), 20 (editar problema), 15 (eliminar problema), 11 (editar usuario), 0 (log out) o 14 (compartir problema).

PoP

En esta clase se implementa la pantalla en la que se elige qué tipo de problema se quiere jugar. Esta pantalla se corresponde con el estado 4 y se puede llegar a los estados: 3 (botón atrás), 5 (botón privados o botón públicos) o 18 (botón guardados).

Elige Player

Clase correspondiente al estado 5 e implementa la pantalla de elección de jugador 1 y jugador 2, que pueden ser de tipo Humano, Cpu fácil, Cpu media o Cpu difícil. De este estado se puede llegar solamente a los estados 3, mediante el botón de

atrás, o al 6 o 23, botón de confirmar. Si los dos jugadores elegidos son de tipo máquina, se llega al estado 23, sino al 6.

TemaDif / TemaDifComp / TemaDifElim / TemaDifCreaMod

Clase que implementa la pantalla de elección del tema, dificultad y color de inicio. Son los estados 6, 14, 15 y 20 y permite llegar a los estados 3 (botón atrás) o 7 (botón confirmar).

ChooseP

Esta clase es el estado 7 y la pantalla de elegir problema. Esta pantalla le pide al dominio, a través del controlador de presentación, la información de todos los problemas con las características elegidas en pantallas anteriores. Dominio le envía toda la información necesaria para jugar la partida y ChooseP se encarga de transformarlo a tablero y hacer sets de otras variables. Desde esta pantalla se puede llegar al estado 3 con el botón de atrás o al 8 o 23 con el botón de confirmar. Pasará al estado 23 si se ha elegido IA vs IA, en cualquier otro caso, pasa al estado 8.

Partida

Clase que implementa la pantalla de jugar partida y corresponde al estado 8. En los turnos de jugador tipo Humano, la clase pide a dominio (a través del controlador de presentación) todos los movimientos posibles de la ficha que se ha seleccionado para iluminar así sobre el tablero todas las posibles posiciones finales. En cambio si es turno de la máquina, dominio devuelve solamente el mejor movimiento y presentación se encarga de aplicarlo a la ficha correspondiente. Desde esta pantalla se puede llegar al estado 9, mediante botón de pausa, al estado 3 mediante el botón de guardar y salir o al acabar la partida o al estado 21. Para llegar al estado 3 directamente al acabar es cuando el problema es de tipo privado o bien el jugador 1 es de tipo máquina. El estado 21 corresponde a los rankings y se llega siempre que se juega un problema público y el jugador 1 es humano.

Pause

Esta clase se corresponde con el estado 9 e implementa una pantalla “de espera” que contiene solamente contiene un gif y un botón de continuar. Desde este estado solamente se puede llegar al estado 8 a través del botón de continuar.

CreaMod

Estado 9 e implementación de la pantalla que permite modificar o crear un problema. Desde aquí solamente se puede llegar al estado 3, ya sea a través del botón de atrás o porque se ha validado correctamente el problema modificado o creado. Para validar el problema se envía la información al controlador de presentación y éste a la capa de dominio.

editUser

Clase que implementa la pantalla que permite elegir si modificar usuario o modificar contraseña. Desde este estado (10), se puede llegar al 12 si se quiere modificar usuario, al 13 si se quiere modificar la contraseña o al 3 mediante el botón de atrás.

CambiarUsn

Clase que implementa la pantalla que permite cambiar el nombre de usuario. Desde el estado 12 (correspondiente a esta pantalla), se puede llegar solamente al estado 11 cuando la validación del nuevo nombre de usuario es correcto o mediante el botón de atrás. La validación se realiza al hacer click en el botón validar, que envía la información al controlador presentación y seguidamente este la envía a la capa de dominio.

CambiarPsw

Clase que implementa la pantalla que permite cambiar el nombre de usuario. Desde el estado 13 (correspondiente a esta pantalla), se puede llegar solamente al estado 11 cuando la validación del nuevo nombre de usuario es correcto o mediante el botón de atrás. La validación se realiza al hacer click en el botón validar, que envía la información al controlador presentación y seguidamente este la envía a la capa de dominio.

ChoosePComp

Esta clase es prácticamente idéntica a ChooseP, con la diferencia de que desde este estado (16) se llega solamente al estado 3. Al estado 3 se puede llegar mediante el botón atrás o mediante el botón de compartir. La validación de si se puede compartir o no el problema la hace la capa de datos con la información que le

pasamos desde esta capa. Para saber si se ha podido compartir o no el problema, sale un mensaje indicándolo, según lo que dominio haya dicho.

ChoosePElim

La implementación de esta clase sigue la misma idea que ChoosePComp, con la diferencia que en este caso se elimina el problema en vez de compartirlo. Se le envía la información a dominio para que este lo elimine. Desde este estado (17) solo se puede llegar al 3, igual que antes, mediante el botón de atrás o mediante el botón de eliminar.

ChoosePGuard

Esta clase, correspondiente al estado 18, es prácticamente igual que ChooseP. La única diferencia es que esta pantalla te muestra las partidas empezadas por el jugador.

ChoosePCreaMod

Esta clase funciona como el resto de pantallas de elegir problemas. En esta pantalla solamente se muestran los problemas privados del jugador.

Ranking

Esta clase es meramente informativa y muestra el ranking del problema que se ha jugado, tu puntuación en esta partida y tu mejor puntuación del problema. Desde este estado, el 21, se puede llegar solamente al estado 3 (botón atrás).

ChooseProblemas

Esta clase representa una pantalla en la que se escoge el número de problemas a ejecutar en IA vs IA. Es el estado 22 y puede llegar al estado 3 (botón atrás) y al 23 (confirmar).

SkyNet

Esta clase representa el “marcador” del resultado de que las dos IAs jueguen los k problemas. Esta clase envía toda la información necesaria para que jueguen máquina vs máquina a dominio. Dominio solamente devuelve quien de las dos máquinas ha ganado. Y de esta forma el marcador se va actualizando.