Tarea programada 2: Othello

Descripción

Un cliente busca un desarrollador capaz de programar el juego Othello:

Reversi u Othello es un juego basado en turnos para dos personas. Se van colocando fichas en un tablero de tamaño 8 x 8 y el objetivo es: al llenar las 64 posiciones del tablero tener la mayor cantidad de fichas del color del jugador.

La movilidad media de un jugador a lo largo de la partida es de 8 movimientos. Como en total se pueden hacer 60 movimientos, el número máximo de posibles partidas es de aproximadamente 10^{54}. Por otra parte, el número máximo de posiciones posibles se calcula aproximadamente en 10^{30}.

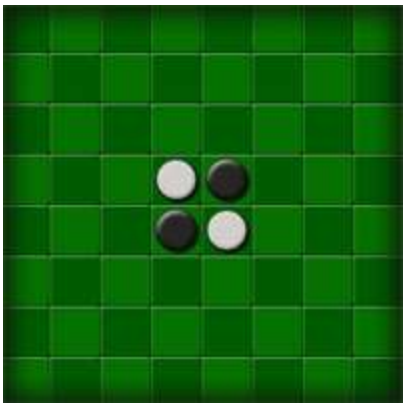
Origen e historia

Las primeras referencias que se tienen de juegos similares se remontan a finales del siglo XIX, y se trataba de juegos a los que se aplicaba nombres diferentes del actual y en los que cambiaba la forma o tamaño del tablero.

En 1870 aparece un juego similar que se juega sobre un tablero en forma de cruz. Posteriormente aparece ya un juego que se juega en tablero cuadrado de 8x8.

Reglas básicas

Cada ficha de Reversi posee dos colores (blanco y negro). Los jugadores deberán escoger un color para las fichas. El juego inicia con 4 fichas, dos de cada equipo colocadas en diagonal en el centro del tablero como muestra la siguiente imagen:



El primer movimiento lo realizará quien juegue con las fichas negras e irá alternando con el otro jugador por turnos a menos que algún jugador no tenga opciones válidas para colocar una ficha.

Posibles movimientos

Los movimientos consisten en incorporar fichas al tablero a razón de una por turno, nunca en desplazar fichas de las que ya estuvieran sobre el tablero.

Las incorporaciones deberán hacerse en orden a las siguientes normas:

- Sólo podrá incorporarse una ficha al flanquear a una o varias fichas contrarias.

- Por flanquear se entiende el hecho de colocar la nueva ficha en un extremo de una hilera de fichas del color del contrario (una o más fichas) en cuyo extremo opuesto hay una ficha del color de la que se incorpora, sin que existan casillas libres entre ninguna de ellas. Esta hilera puede ser indistintamente vertical, horizontal o diagonal. De este modo, las fichas del contrincante quedan encerradas entre una que ya estaba en el tablero y la nueva ficha.

- Cada vez que un jugador incorpora una ficha, y por lo tanto encierra a otras del contrario, debe dar la vuelta a las fichas encerradas convirtiéndolas así en propias.

- Si en una sola incorporación se provocase esta situación de flanqueo en más de una línea, se voltearán todas las fichas contrarias que estuvieran implicadas en cada uno de los flanqueos.

- Si no fuera posible para un jugador encerrar a ninguna ficha, deberá pasar en su turno, volviendo el mismo a su oponente.

Final del juego

La partida finaliza cuando todas las casillas del tablero son ocupadas o cuando ninguno de los 2 jugadores tiene un movimiento posible. Sin importar el caso, el vencedor es el jugador con más fichas sobre el tablero. Cabe resaltar que existe la posibilidad de un empate.

Referencia de juego

Puede encontrar un ejemplo de juego en el siguiente enlace: http://www.ireversi.com/

Lineamientos para la tarea

Los requerimientos solicitados por el cliente son la posibilidad de tener dos variantes de juego: un jugador humano contra un jugador virtual y un jugador humano contra otro jugador humano (hot-seat). También se quieren cuatro variantes para los tamaños del tablero: 8x8, 10x10, 10x14 y 14x10 que podrá elegir el usuario para jugar.

Se quiere tener también un sistema de records en donde se desplieguen los 10 mejores jugadores (deberá desplegar máximo 10). El sistema de records deberá ser persistente, es decir deberá guardar los records en un medio de almacenamiento no volátil para que se puedan ver los records entre corridas del programa. La forma en la que se desea tener el sistema es fragmentada por tablero, dentro si el juego fue de un jugador contra la computadora o jugador contra jugador ordenado por la cantidad de puntos (fichas) obtenidas.

Podrá elegir cómo desea realizar la interfaz. Puede usar salida en consola, JOptionPane o una mezcla de ambos.

El programa deberá ser a prueba de fallos, en el caso que un jugador no elija una opción adecuada, el programa pedirá nuevamente una opción al usuario.

Opcional extra (10%)

Puede optar por un 10% extra en la tarea al realizar una interfaz gráfica utilizando BufferedImage (como referencia puede utilizar el objeto ObraDeArte explicado en clase).

Entregables

Se deberá realizar un análisis y diseño del problema. Se espera que en un documento se especifique el problema, los pasos para llevar a cabo la solución, un diseño de clases (en formato UML) y una explicación de los algoritmos complejos utilizando pseudocódigo. Un plan de pruebas y los resultados de las pruebas.

Adicionalmente agregar una sección de problemas encontrados a la hora de resolver la tarea y un porcentaje de distribución del 100% del trabajo de cada uno de los integrantes. Por ejemplo si el grupo es de dos personas y cada estudiante trabajo la misma cantidad, el porcentaje será 50% cada uno.

El código fuente deberá contar con documentación interna y deberá generar la documentación utilizando Doxygen.

Deberá programar el juego para que sea completamente funcional y utilizar todas las técnicas de programación que considere necesarias y entregar el código fuente.

Adicionalmente, agregar el nombre de su repositorio que utilizó el Github. Se espera que utilice el sistema de control de versiones para agregar frecuentemente los cambios que está haciendo mediante el uso de commits y push al servidor de Github.

Se evaluará el uso de la herramienta de control de versiones y un avance.

Desglose

(23%) Documentación que incluya: análisis, diseño, pseudocódigo de los algoritmos complejos, diagramas UML, descripción de las pruebas, documentación interna del código fuente usando Doxygen y los demás puntos explicados anteriormente para la elaboración de la documentación.

(77%) Codificación de la solución (incluye uso del sistema de control de versiones, solución propuesta, funcionalidad, buen diseño, apego a los estándares de código, detección de errores, efectividad y eficiencia de las estructuras de datos y los algoritmos).

Notas adicionales

- Se recomienda utilizar www.draw.io para la generación del UML por facilidad de uso.
- Se debe crear una cuenta en Github utilizando el correo dado en clase.
- Se recomienda descargar Sourcetree o Git para llevar a cabo los commits.
- Se recomienda usar la estructura con branches.
- Recuerde que no se debe commitear a master código inestable.