75.41 - Algoritmos y Programación II

Cátedra Ing. Patricia Calvo - 1er cuatrimestre 2021

Trabajo Práctico 2: CUATRO EN LÍNEA V1.0

Objetivo

Generar una pieza de software que simule el funcionamiento del juego 4 en línea en su versión multijugador.

Enunciado

Conecta 4 (también conocido como 4 en Línea en algunas versiones) es un juego de mesa para dos jugadores distribuido por Hasbro, en el que se introducen fichas en un tablero vertical con el objetivo de alinear cuatro consecutivas de un mismo color. Fue creado en 1974 por Ned Strongin y Howard Wexler para Milton Bradley Company.

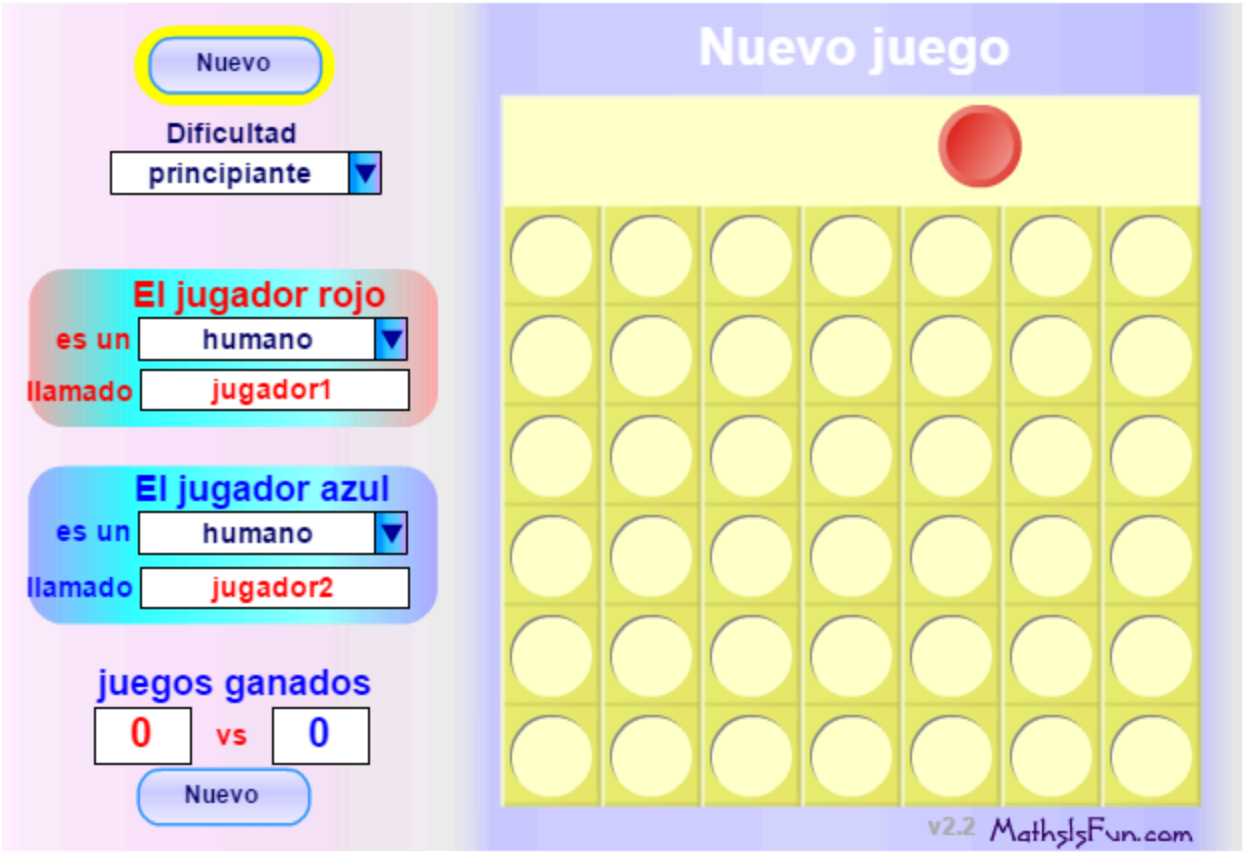
Reglas del juego

El objetivo de Conecta 4 es alinear N (parámetro) fichas sobre un tablero formado por L (parámetro) filas, M (parámetro) columnas y Z (parámetro) profundidad. Cada jugador dispone de X (parámetro) fichas de un color (por lo general, rojas o amarillas). Por turnos, los jugadores deben introducir una ficha en la columna que prefieran (siempre que no esté completa) y ésta caerá a la posición más baja. Gana la partida el primero que consiga alinear N fichas consecutivas de un mismo color en horizontal, vertical o diagonal. Si todas las columnas están llenas pero nadie ha hecho una fila válida, hay empate.

Conecta 4 es un juego de estrategia abstracta donde los contrincantes disponen de información perfecta. Por norma general, el primer jugador tiene más posibilidades de ganar si introduce la primera ficha en la columna central. Si lo hace en las contiguas se puede forzar un empate, mientras que si la mete en las más alejadas del centro su rival puede vencerle con mayor facilidad.

Existe una versión de mesa y otra de viaje. Conecta 4 es una variante de Captain's Mistress, un juego de mecánica similar pero con bolas de madera en vez de fichas de plástico.

A modo de ejemplo sugerimos esta página:



En el modo Humano vs Humano.

http://www.disfrutalasmatematicas.com/juegos/4-en-linea.html

Interfaz de usuario

Toda la interfaz de usuario debe estar basada en texto. El estado del tablero tiene que mostrarse usando caracteres dispuestos en filas y columnas. Por ejemplo, usando ‘ ’ para representar un casillero vacío, ‘X’ para representar un casillero ocupado por un jugador y ‘O’ para el otro, y así sucesivamente. Además, después de cada turno, el programa debe exportar el tablero en un archivo de bmp (utilizando la libreria) con el estado del tablero.

No es necesario que se limpie la pantalla, simplemente escribir el estado del tablero luego de cada jugada.

Adicionales

1) Al iniciar el turno, cada jugador saca una carta. Una carta tiene una función y puede ser canjeada antes de ejecutar la jugada. Hay 2 tipos de cartas fijos y 2 tipos de cartas que el grupo tiene que definir. Cada jugador puede tener hasta 3 cartas.
    a) Carta Bloquear turno: le hace perder un turno al jugador siguiente o el primer jugador del siguiente turno si es el último.
    b) Carta Juega Doble: le permite al jugador jugar dos fichas en vez de una.
2) El jugador antes de ejecutar su jugada puede elegir utilizar una carta.

Normas de entrega

Trabajo práctico individual: 5 personas. Cada grupo debe elegir un nombre.

Reglas generales: respetar el Apéndice A.

Se deberá subir un archivo comprimido al campus, en un link que se habilitará para esta entrega. Este archivo deberá tener un nombre formado de la siguiente manera:

Nombre del grupo-TP2.zip

Deberá contener los archivos fuentes (no los binarios), el informe del trabajo realizado, las respuestas al cuestionario, el manual del usuario y el manual del programador (Todo en el mismo PDF).

La fecha de entrega vence el día lunes 24/06/21 a las 23.59hs.

Se evaluará: funcionalidad, eficiencia, algoritmos utilizados, buenas prácticas de programación, modularización, documentación, gestión de memoria y estructuras de datos.

Apéndice A

1) Usar las siguientes convenciones para nombrar identificadores.
    a) Clases y structs: Los nombres de clases y structs siempre deben comenzar con la primera letra en mayúscula en cada palabra, deben ser simples y descriptivos. Se concatenan todas las palabras. Ejemplo: Coche, Vehiculo, CentralTelefonica.
    b) Métodos y funciones: Deben comenzar con letra minúscula, y si está compuesta por 2 o más palabras, la primera letra de la segunda palabra debe comenzar con mayúscula. De preferencia que sean verbos. Ejemplo: arrancarCoche(), sumar().
    c) Variables y objetos: las variables siguen la misma convención que los métodos. Por Ejemplo: alumno, padronElectoral.
    d) Constantes: Las variables constantes o finales, las cuales no cambian su valor durante todo el programa se deben escribir en mayúsculas, concatenadas por "_". Ejemplo: ANCHO, VACIO, COLOR_BASE.
2) El lenguaje utilizado es C++, esto quiere decir que se debe utilizar siempre C++ y no C, por lo tanto una forma de darse cuenta de esto es no incluir nada que tenga .h, por ejemplo #include <iostream> .
3) Los enumerados se nombran como una clase, al igual que sus valores.
4) No usar sentencias 'using namespace' en los .h, solo en los .cpp. Por ejemplo, para referenciar el tipo string en el .h se pone std::string.
5) No usar 'and' y 'or', utilizar los operadores '&&' y '||' respectivamente.
6) Compilar en forma ANSI. Debe estar desarrollado en linux con eclipse y g++. Utilizamos el estándar C++98.
7) Chequear memoria antes de entregar. No tener accesos fuera de rango ni memoria colgada.
8) En el TP 2 no se pueden utilizar grandes bloques de memoria contiguos.
9) Si el trabajo práctico requiere archivos para procesar, entregar los archivos de prueba en la entrega del TP. Utilizar siempre rutas relativas y no absolutas.

10) Entregar el informe explicando el TP realizado, manual de usuario y manual del programador.
11) Comentar el código. Todos los tipos, métodos y funciones deberían tener sus comentarios en el .h que los declara.
12) Modularizar el código. No entregar 1 o 2 archivos, separar cada clase o struct con sus funcionalidades en un .h y .cpp
13) No inicializar valores dentro del struct o .h.
14) Si cualquier estructura de control tiene 1 línea, utilizar {} siempre, por ejemplo:

```
for(int i = 0; i < 10; i++) {
      std::cout << i;
}
```