Universidad Mariano Gálvez

Emilio Jose Reyes Aragón 9959-23-3303

Ing. Esduardo Del Aguila

Programación I

Documentación del juego Busca Minas

30/03/2025

Tablero.cpp

```
#include "Tablero.h"
#include <sstream>
#include <iostream>

using namespace std;

// Constructor por defecto de la clase Tablero
Tablero::Tablero()
{
}

// Constructor que inicializa el tablero con dimensiones y modo desarrollador
Tablero::Tablero(int alturaTablero, int anchoTablero, bool modoDesarrollador)
{
   this->alturaTablero = alturaTablero;
   this->anchoTablero = anchoTablero;
   this->modoDesarrollador = modoDesarrollador;
   int x, y;

   // Inicialización de la matriz de celdas
   for (y = 0; y < this->alturaTablero; y++)
   {
      vector<Celda> fila;
      for (x = 0; x < this->anchoTablero; x++)
      {
         fila.push_back((Celda(x, y, false)));
```

```
        }
        this->contenidoTablero.push_back(fila);
    }
}

// Métodos getters y setters para obtener y modificar atributos privados
int Tablero::getAlturaTablero()
{
    return this->alturaTablero;
}

void Tablero::setAlturaTablero(int alturaTablero)
{
    this->alturaTablero = alturaTablero;
}

int Tablero::getAnchoTablero()
{
    return this->anchoTablero;
}

void Tablero::setAnchoTablero(int anchoTablero)
{
    this->anchoTablero = anchoTablero;
}

bool Tablero::getModoDesarrollador()
{
    return this->modoDesarrollador;
```

```
}

void Tablero::setModoDesarrollador(bool modoDesarrollador)
{
   this->modoDesarrollador = modoDesarrollador;
}

// Devuelve la representación visual de una celda (mina, número o desconocido)
string Tablero::getRepresentacionMina(int coordenadaX, int coordenadaY)
{
   Celda celdaTemporal = this->contenidoTablero.at(coordenadaY).at(coordenadaX);
   if (celdaTemporal.getMinaDescubierta() || this->modoDesarrollador)
   {
      if (celdaTemporal.getMina())
      {
         return "*"; // Representación de una mina
      }
      else
      {
         int cantidadCelda = this->minasCercanas(coordenadaY, coordenadaX);
         stringstream ss;
         ss << cantidadCelda;
         return ss.str(); // Conversión de número a string
      }
   }
   else
   {
      return "?"; // Celda no descubierta
   }
```

```
}

// Cuenta la cantidad de minas cercanas a una celda específica
int Tablero::minasCercanas(int filaTablero, int columnaTablero)
{
   int contadorTablero = 0;
   int filaInicioTablero = max(0, filaTablero - 1);
   int filaFinTablero = min(this->alturaTablero - 1, filaTablero + 1);
   int columnaInicioTablero = max(0, columnaTablero - 1);
   int columnaFinTablero = min(this->anchoTablero - 1, columnaTablero + 1);

   for (int m = filaInicioTablero; m <= filaFinTablero; m++)
   {
      for (int l = columnaInicioTablero; l <= columnaFinTablero; l++)
      {
         if (this->contenidoTablero.at(m).at(l).getMina())
         {
            contadorTablero++;
         }
      }
   }
   return contadorTablero;
}

// Métodos para imprimir la representación visual del tablero
void Tablero::imprimirSeparadorEncabezado()
{
   for (int m = 0; m <= this->anchoTablero; m++)
   {
```

```
        cout << "----";
    }
    cout << "\n";
}

void Tablero::imprimirSeparadorFilas()
{
    for (int m = 0; m <= this->anchoTablero; m++)
    {
        cout << (m == 0 ? "|---" : "+---");
    }
    cout << "+\n";
}

void Tablero::imprimirEncabezado()
{
    this->imprimirSeparadorEncabezado();
    cout << "|    ";
    for (int l = 0; l < this->anchoTablero; l++)
    {
        cout << "| " << l + 1 << " ";
    }
    cout << "|\n";
}

// Imprime el tablero en la consola
void Tablero::imprimir()
{
    this->imprimirEncabezado();
```

```
    this->imprimirSeparadorEncabezado();

    for (int y = 0; y < this->alturaTablero; y++)
    {
        cout << "| " << y + 1 << " ";
        for (int x = 0; x < this->anchoTablero; x++)
        {
            cout << "| " << this->getRepresentacionMina(x, y) << " ";
        }
        cout << "|\n";
        this->imprimirSeparadorFilas();
    }
}

// Coloca una mina en la celda especificada
bool Tablero::colocarMina(int x, int y)
{
    return this->contenidoTablero.at(y).at(x).setMina(true);
}

// Descubre una celda y verifica si contiene una mina
bool Tablero::descubrirMina(int x, int y)
{
    this->contenidoTablero.at(y).at(x).setMinaDescubierta(true);
    return !this->contenidoTablero.at(y).at(x).getMina();
}

// Cuenta la cantidad de celdas sin minas y sin descubrir
int Tablero::contarCeldasSinMinasYSinDescubrir()
```

```
{
   int contador = 0;
   for (int y = 0; y < this->alturaTablero; y++)
   {
      for (int x = 0; x < this->anchoTablero; x++)
      {
         Celda celdaTemporal = this->contenidoTablero.at(y).at(x);
         if (!celdaTemporal.getMinaDescubierta() && !celdaTemporal.getMina())
         {
            contador++;
         }
      }
   }
   return contador;
}
```

main.cpp

```
#include <iostream>
#include <unistd.h> // Para la función getpid()
#include "Juego.h" // Inclusión de la clase Juego
#include "Config.h" // Inclusión de la clase Config

using namespace std;

int main()
{
   // Definición de las constantes para la configuración del juego
   const int FILASTABLERO = 10;  // Número de filas del tablero
   const int COLUMNASTABLERO = 10; // Número de columnas del tablero
   const int MINASENTABLERO = 50; // Número de minas en el tablero
   const bool MODODESARROLLADOR = false; // Modo desarrollador activado o no
   const int VIDASTABLERO = 3; // Número de vidas del jugador

   // Creación del objeto Config con los valores predefinidos
   Config configuracionJuego(FILASTABLERO, COLUMNASTABLERO, MINASENTABLERO,
MODODESARROLLADOR, VIDASTABLERO);

   // Creación del objeto Juego con un tablero y el número de minas
   Juego juego(Tablero(configuracionJuego.getfilasTablero(),
                  configuracionJuego.getcolumnasTablero(),
                  configuracionJuego.getmodoDesarrolladorTablero()),
            configuracionJuego.getminasTablero());

   srand(getpid()); // Inicialización de la semilla de números aleatorios con el ID del proceso
```

```
  int opciones; // Variable para almacenar la opción del menú
  bool repetir = true; // Control del bucle del menú principal

  do
  {
    system("cls"); // Limpiar pantalla

    // Impresión del menú principal
    cout << "\n\n\t\tBUSCA MINAS -Menu-" << endl;
    cout << "\t\t--------------------" << endl;
    cout << "\t\t1. Configuracion del Juego" << endl;
    cout << "\t\t2. Iniciar el Juego" << endl;
    cout << "\t\t3. Salir del Juego" << endl;
    cout << "\n\t\tIngrese una opcion: ";
    cin >> opciones; // Lectura de la opción ingresada por el usuario

    switch (opciones)
    {
    case 1:
      {
        configuracionJuego.menuConfiguracion(); // Llamada al menú de configuración
        break;
      }
    case 2:
      {
        // Creación de un objeto Juego temporal para iniciar una partida con la configuración
actual
        Juego juegoTemporal(Tablero(configuracionJuego.getfilasTablero(),
```

```
                                configuracionJuego.getcolumnasTablero(),
                                configuracionJuego.getmodoDesarrolladorTablero()),
                            configuracionJuego.getminasTablero());
                juegoTemporal.iniciar(); // Llamada al método para iniciar el juego
                system("pause"); // Pausa para que el usuario vea los resultados antes de volver al menú
                break;
            }
        case 3:
            repetir = false; // Salir del bucle y finalizar el programa
            break;
        }
    } while (repetir); // Repetir mientras el usuario no elija salir

    system("cls"); // Limpiar pantalla antes de salir
    return 0;
}
```