Documentação do Código

Eduardo Hubner Adriano

08/08/2024

jogador.cpp

Descrição

Este arquivo contém a implementação da classe jogador, que inclui métodos para imprimir, apagar e mover navios, além de funções auxiliares para converter coordenadas e obter coordenadas aleatórias.

Código

```cpp
#include "../include/jogador.hpp"
#include "../include/matrizes.hpp"
#include "../include/config.hpp"
#include <cstdlib>
#include <ctime>
#include <conio.h>
#include <iostream>

using namespace std;

/**
 * @file jogador.cpp
 * @brief Implementação da classe jogador.
 */
/**
 * @brief Imprime o navio CARRIER na tela.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::printarCARRIER(int &x, int &y, int c) {
    if (c == 0) {
        gotoxy(x, y); cout << "C";
        gotoxy(x, y + 2); cout << "C";
        gotoxy(x, y + 4); cout << "C";
        gotoxy(x, y + 6); cout << "C";
    }
    if (c == 1) {
        gotoxy(x, y); cout << "C";
        gotoxy(x + 2, y); cout << "C";
```

```
        gotoxy(x + 4, y); cout << "C";
        gotoxy(x + 6, y); cout << "C";
    }
}

/**
 * @brief Apaga o navio CARRIER da tela.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::apagarCARRIER(int &x, int &y, int c) {
    if (c == 0) {
        gotoxy(x, y); cout << " ";
        gotoxy(x, y + 2); cout << " ";
        gotoxy(x, y + 4); cout << " ";
        gotoxy(x, y + 6); cout << " ";
    }
    if (c == 1) {
        gotoxy(x, y); cout << " ";
        gotoxy(x + 2, y); cout << " ";
        gotoxy(x + 4, y); cout << " ";
        gotoxy(x + 6, y); cout << " ";
    }
}

/**
 * @brief Move o navio CARRIER pela tela.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::moverCARRIER(int &x, int &y, int c) {
    if (kbhit()) {
        apagarCARRIER(x, y, c);
        char tecla = getch();
        if (tecla == ESQUERDA && x > 2) x = x - 2;
        if (c == 0) {
            if (tecla == DIREITA && x < 20) x = x + 2;
        }
        if (c == 1) {
            if (tecla == DIREITA && x < 14) x = x + 2;
        }
        if (tecla == CIMA && y > 28) y = y - 2;
        if (c == 0) {
            if (tecla == BAIXO && y < 40) y = y + 2;
        }
        if (c == 1) {
```

```
            if (tecla == BAIXO && y < 46) y = y + 2;
        }
        printarCARRIER(x, y, c);
        x1 = x;
        y1 = y;
    }
}

/**
 * @brief Imprime o navio TANKER na tela.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::printarTANKER(int &x, int &y, int c) {
    if (c == 0) {
        gotoxy(x, y); cout << "T";
        gotoxy(x, y + 2); cout << "T";
        gotoxy(x, y + 4); cout << "T";
    }
    if (c == 1) {
        gotoxy(x, y); cout << "T";
        gotoxy(x + 2, y); cout << "T";
        gotoxy(x + 4, y); cout << "T";
    }
}

/**
 * @brief Apaga o navio TANKER da tela.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::apagarTANKER(int &x, int &y, int c) {
    if (c == 0) {
        gotoxy(x, y); cout << " ";
        gotoxy(x, y + 2); cout << " ";
        gotoxy(x, y + 4); cout << " ";
    }
    if (c == 1) {
        gotoxy(x, y); cout << " ";
        gotoxy(x + 2, y); cout << " ";
        gotoxy(x + 4, y); cout << " ";
    }
}

/**
 * @brief Move o navio TANKER pela tela.
```

```
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::moverTANKER(int &x, int &y, int c) {
    if (kbhit()) {
        apagarTANKER(x, y, c);
        char tecla = getch();
        if (tecla == ESQUERDA && x > 2) x = x - 2;
        if (c == 0) {
            if (tecla == DIREITA && x < 20) x = x + 2;
        }
        if (c == 1) {
            if (tecla == DIREITA && x < 16) x = x + 2;
        }
        if (tecla == CIMA && y > 28) y = y - 2;
        if (c == 0) {
            if (tecla == BAIXO && y < 42) y = y + 2;
        }
        if (c == 1) {
            if (tecla == BAIXO && y < 46) y = y + 2;
        }
        printarTANKER(x, y, c);
        x1 = x;
        y1 = y;
    }
}

/**
 * @brief Imprime o navio DESTROYER na tela.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::printarDESTROYER(int &x, int &y, int c) {
    if (c == 0) {
        gotoxy(x, y); cout << "D";
        gotoxy(x, y + 2); cout << "D";
        gotoxy(x, y + 4); cout << "D";
    }
    if (c == 1) {
        gotoxy(x, y); cout << "D";
        gotoxy(x + 2, y); cout << "D";
        gotoxy(x + 4, y); cout << "D";
    }
}

/**
```

```
 * @brief Apaga o navio DESTROYER da tela.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::apagarDESTROYER(int &x, int &y, int c) {
    if (c == 0) {
        gotoxy(x, y); cout << " ";
        gotoxy(x, y + 2); cout << " ";
        gotoxy(x, y + 4); cout << " ";
    }
    if (c == 1) {
        gotoxy(x, y); cout << " ";
        gotoxy(x + 2, y); cout << " ";
        gotoxy(x + 4, y); cout << " ";
    }
}

/**
 * @brief Move o navio DESTROYER pela tela.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::moverDESTROYER(int &x, int &y, int c) {
    if (kbhit()) {
        apagarDESTROYER(x, y, c);
        char tecla = getch();
        if (tecla == ESQUERDA && x > 2) x = x - 2;
        if (c == 0) {
            if (tecla == DIREITA && x < 20) x = x + 2;
        }
        if (c == 1) {
            if (tecla == DIREITA && x < 16) x = x + 2;
        }
        if (tecla == CIMA && y > 28) y = y - 2;
        if (c == 0) {
            if (tecla == BAIXO && y < 42) y = y + 2;
        }
        if (c == 1) {
            if (tecla == BAIXO && y < 46) y = y + 2;
        }
        printarDESTROYER(x, y, c);
        x1 = x;
        y1 = y;
    }
}
```

```
/**
 * @brief Imprime o navio SUBMARINE na tela.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::printarSUBMARINE(int &x, int &y, int c) {
    (void)c; // Marca o parâmetro c como intencionalmente não
utilizado
    gotoxy(x, y);
    cout << "S";
}

/**
 * @brief Apaga o navio SUBMARINE da tela.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::apagarSUBMARINE(int &x, int &y, int c) {
    (void)c; // Marca o parâmetro c como intencionalmente não
utilizado
    gotoxy(x, y);
    cout << " ";
}

/**
 * @brief Move o navio SUBMARINE pela tela.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::moverSUBMARINE(int &x, int &y, int c) {
    if (kbhit()) {
        apagarSUBMARINE(x, y, c);
        char tecla = getch();
        if (tecla == ESQUERDA && x > 2) x = x - 2;
        if (c == 0) {
            if (tecla == DIREITA && x < 20) x = x + 2;
        }
        if (c == 1) {
            if (tecla == DIREITA && x < 20) x = x + 2;
        }
        if (tecla == CIMA && y > 28) y = y - 2;
        if (c == 0) {
            if (tecla == BAIXO && y < 46) y = y + 2;
        }
```

```
        if (c == 1) {
            if (tecla == BAIXO && y < 46) y = y + 2;
        }
        printarSUBMARINE(x, y, c);
        x1 = x;
        y1 = y;
    }
}

/**
 * @brief Imprime o navio CARRIER permanentemente na matriz JO.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::printarCARRIER_Permanente(int x, int y, int c) {
    printarCARRIER(x, y, c);
    ca = converterMatriz(y);
    if (c == 0) {
        JO[ca][x] = 'C';
        JO[ca + 2][x] = 'C';
        JO[ca + 4][x] = 'C';
        JO[ca + 6][x] = 'C';
    }
    if (c == 1) {
        JO[ca][x] = 'C';
        JO[ca][x + 2] = 'C';
        JO[ca][x + 4] = 'C';
        JO[ca][x + 6] = 'C';
    }
}

/**
 * @brief Imprime o navio TANKER permanentemente na matriz JO.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::pintarTANKER_Permanente(int x, int y, int c) {
    printarTANKER(x, y, c);
    ca = converterMatriz(y);
    if (c == 0) {
        JO[ca][x] = 'T';
        JO[ca + 2][x] = 'T';
        JO[ca + 4][x] = 'T';
    }
    if (c == 1) {
        JO[ca][x] = 'T';
```

```
        JO[ca][x + 2] = 'T';
        JO[ca][x + 4] = 'T';
    }
}

/**
 * @brief Imprime o navio DESTROYER permanentemente na matriz JO.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::printarDESTROYER_Permanente(int x, int y, int c) {
    printarDESTROYER(x, y, c);
    ca = converterMatriz(y);
    if (c == 0) {
        JO[ca][x] = 'D';
        JO[ca + 2][x] = 'D';
        JO[ca + 4][x] = 'D';
    }
    if (c == 1) {
        JO[ca][x] = 'D';
        JO[ca][x + 2] = 'D';
        JO[ca][x + 4] = 'D';
    }
}

/**
 * @brief Imprime o navio SUBMARINE permanentemente na matriz JO.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param c Direção do navio (0 para vertical, 1 para horizontal).
 */
void jogador::printarSUBMARINE_Permanente(int x, int y, int c) {
    printarSUBMARINE(x, y, c);
    ca = converterMatriz(y);
    JO[ca][x] = 'S';
}

/**
 * @brief Apaga um navio da tela.
 *
 * @param x Coordenada x do navio.
 * @param y Coordenada y do navio.
 * @param cam Direção do navio (0 para vertical, 1 para horizontal).
 * @param tipo Tipo do navio.
 */
void jogador::apagarNavio(int x, int y, int cam, int tipo) {
    switch(tipo) {
```

```
        case 1: apagarCARRIER(x, y, cam); break;
        case 2: apagarTANKER(x, y, cam); break;
        case 3: apagarDESTROYER(x, y, cam); break;
        case 4: apagarSUBMARINE(x, y, cam); break;
    }
}

/**
 * @brief Converte coordenadas da matriz para coordenadas do
tabuleiro.
 *
 * @param y Coordenada na matriz.
 * @return Coordenada correspondente no tabuleiro.
 */
int jogador::converterMatriz(int y) {
    while (true) {
        if (mapa == y) {
            matriz1 = matriz;
            mapa = 28;
            matriz = 2;
            return matriz1;
        } else {
            matriz = matriz + 2;
            mapa = mapa + 2;
        }
    }
}

/**
 * @brief Converte coordenadas do tabuleiro para coordenadas da
matriz.
 *
 * @param y Coordenada no tabuleiro.
 * @return Coordenada correspondente na matriz.
 */
int jogador::converterTabuleiro(int y) {
    while (true) {
        if (matriz == y) {
            mapa1 = mapa;
            mapa = 28;
            matriz = 2;
            return mapa1;
        } else {
            matriz = matriz + 2;
            mapa = mapa + 2;
        }
    }
}

/**
```

```
 * @brief Imprime o tabuleiro do jogador (JO) na tela.
 */
void jogador::printarTabuleiroJO() {
    for (int i = 0; i < 22; i++) {
        cout << endl;
        for (int j = 0; j < 22; j++) {
            cout << JO[i][j];
        }
    }
}

/**
 * @brief Obtém uma coordenada aleatória par entre 2 e 20.
 *
 * @return Coordenada aleatória par entre 2 e 20.
 */
int jogador::obter() {
    while (true) {
        f = rand() % 19 + 2;
        if (f % 2 == 0) return f;
    }
}

/**
 * @brief IA: Obtém coordenadas para jogada automática.
 *
 * @return Coordenada gerada automaticamente.
 */
int jogador::IAobter() {
    while (true) {
        x2 = obter();
        y2 = obter();
        f1 = converterTabuleiro(y2);

        if (JO[y2][x2] != 'X' && JO[y2][x2] != 'O') {
            if (JO[y2][x2] == 'C' || JO[y2][x2] == 'T' || JO[y2][x2]
== 'D' || JO[y2][x2] == 'S') {
                if (JO[y2][x2] == 'C') dato = 'C';
                if (JO[y2][x2] == 'T') dato = 'T';
                if (JO[y2][x2] == 'D') dato = 'D';
                if (JO[y2][x2] == 'S') dato = 'S';
                JO[y2][x2] = 'O';
                turno1 = 2;
                vidaJO = vidaJO + 1;
                return 1;
            } else {
                JO[y2][x2] = 'X';
                turno1 = 1;
                return 2;
            }
```

```
        }
    }
}

/**
 * @brief IA: Verifica se um ataque atingiu um navio.
 *
 * @param x Coordenada x do ataque.
 * @param y Coordenada y do ataque.
 * @return Resultado da verificação.
 */
int jogador::IAverificar(int x, int y) {
    if (JO[y][x] == 'C' || JO[y][x] == 'T' || JO[y][x] == 'D' || JO[y]
[x] == 'S') {
        JO[y][x] = 'O';
        vidaJO = vidaJO + 1;
        return 1;
    }
    return 0;
}

/**
 * @brief IA: Executa jogadas alternando entre jogadas aleatórias e
direcionadas.
 *
 * @return Resultado da jogada.
 */
int jogador::IAgeneral() {
    if (turno1 == 2) turno = 2;
    if (turno1 == 1) turno = 1;
    if (turno == 1) {
        IAobter();
        return 0;
    }

    if (turno == 2) {
        int ver = IAdificuldade(dato);
        if (ver == 1) return IAverificar(x3, y3);
        if (ver == 2) {
            IAobter();
            turno1 = 1;
            return 0;
        }
    }
    return 0;
}

/**
 * @brief IA: Define a dificuldade da jogada.
 *
```

```
 * @param A Dificuldade da jogada.
 * @return Resultado da definição de dificuldade.
 */
int jogador::IAdificuldade(char A) {
    do {
        for (int i = 2; i < 21; i++) {
            for (int j = 2; j < 21; j++) {
                if (JO[i][j] == A) {
                    x3 = i;
                    y3 = j;
                    JO[i][j] = 'O';
                    return 1;
                }
            }
        }
        return 2;
    } while (r == 1);
}

/**
 * @brief Registra um disparo no tabuleiro do jogador na posição (x,
y).
 *
 * @param x Coordenada x do disparo.
 * @param y Coordenada y do disparo.
 */
void jogador::disparo(int x, int y) {
    y = y - 1;
    if (JO[y][x] == 'C' || JO[y][x] == 'T' || JO[y][x] == 'D' || JO[y]
[x] == 'S' || JO[y][x] == 'O') {
        JO[y][x] = 'O';
        y = y + 1;
        gotoxy(x, y); std::cout << "O";
        vidaJO = vidaJO + 1;
    } else {
        JO[y][x] = 'X';
        y = y + 1;
        gotoxy(x, y); std::cout << "X";
    }
}
```