

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN KỸ THUẬT LẬP TRÌNH

Game CARO trên Console Windows

NHÓM 13

| | | |
|------------------|---|----------|
| Đặng Duy Lâm | - | 22120182 |
| Nguyễn Nhật Long | - | 22120194 |
| Hoàng Thanh Mẫn | - | 22120200 |
| Lý Trường Nam | - | 22120218 |
| Trần Thảo Ngân | - | 22120225 |

Giáo viên hướng dẫn: TS. Trương Toàn Thịnh

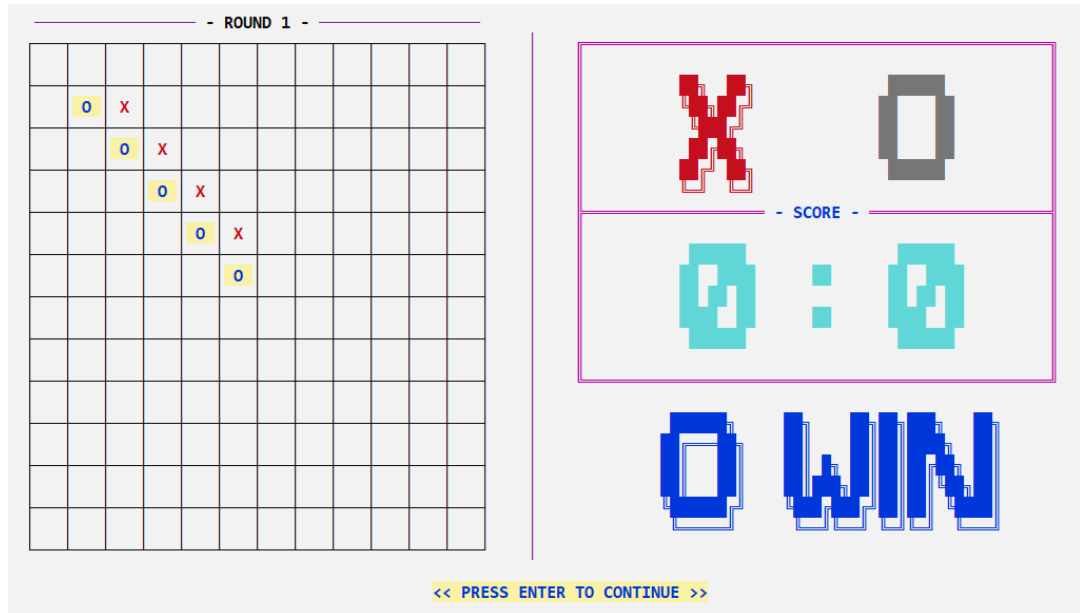
Năm học: 2022 - 2023

MỤC LỤC

| | |
|------------------------------------|----|
| I. Giới thiệu..... | 2 |
| II. Xây dựng trò chơi | 3 |
| 1. File Data.h | 3 |
| 2. Nhóm hàm View | 4 |
| a. Các hàm công cụ | 4 |
| b. Các hàm vẽ giao diện game | 10 |
| 3. Nhóm hàm Model..... | 33 |
| a. Các hàm xử lý trong game | 33 |
| b. Các hàm khác | 38 |
| 4. Nhóm hàm Control..... | 41 |
| III. Lời kết | 43 |
| IV. Tài liệu tham khảo..... | 44 |

I. Giới thiệu

- Bản báo cáo cho đồ án game Caro trên màn hình console do nhóm 13 thực hiện.
- Game bao gồm các chức năng chính như:
 - Bắt đầu trò chơi
 - Save/load game
 - Bật/tắt âm thanh
- Cách chơi: người chơi sẽ dùng các phím W/A/S/D để di chuyển, ENTER để đánh một ô và ESC để tạm dừng trò chơi.
- Một vài hình ảnh trong game:



II. Xây dựng trò chơi

1. File Data.h

- Đây là file chứa các hằng số, các kiểu struct và các thư viện cần thiết phục vụ cho trò chơi.

| | |
|----|---|
| 1 | #pragma once |
| 2 | #pragma comment(lib, "Winmm.lib") |
| 3 | |
| 4 | #define BGM 0 |
| 5 | #define CLICK_SFX 1 |
| 6 | #define SOUND_PATH string("save/sound.txt") |
| 7 | |
| 8 | // Included libraries |
| 9 | #include <iostream> |
| 10 | #include <cctype> |
| 11 | #include <Windows.h> |
| 12 | #include <conio.h> |
| 13 | #include <fstream> |
| 14 | #include <vector> |
| 15 | #include <math.h> |
| 16 | #include <string> |
| 17 | |
| 18 | // Namespace |
| 19 | using namespace std; |
| 20 | |
| 21 | // Console size |
| 22 | #define WIDTH 120 |
| 23 | #define HEIGHT 30 |
| 24 | |
| 25 | // Board size |
| | #define B_SIZE 12 |
| 26 | #define BOARD_X 5 |
| 27 | #define BOARD_Y 2 |
| 28 | |
| 29 | // Color code |
| 30 | #define BLACK ((15 << 4) |
| 31 | #define BLUE ((15 << 4) 1) |
| 32 | #define GREEN ((15 << 4) 2) |
| 33 | #define CYAN ((15 << 4) 3) |
| 34 | #define RED ((15 << 4) 4) |
| 35 | #define MAGENTA ((15 << 4) 5) |
| 36 | #define YELLOW ((15 << 4) 6) |
| 37 | #define WHITE ((15 << 4) 7) |
| 38 | #define GRAY ((15 << 4) 8) |
| 39 | #define LIGHT_BLUE ((15 << 4) 9) |
| 40 | #define LIGHT_GREEN ((15 << 4) 10) |
| 41 | #define LIGHT_CYAN ((15 << 4) 11) |
| 42 | #define LIGHT_RED ((15 << 4) 12) |
| 43 | #define LIGHT_MAGENTA ((15 << 4) 13) |
| 44 | #define LIGHT_YELLOW ((15 << 4) 14) |
| 45 | #define BRIGHT_WHITE ((15 << 4) 15) |
| 46 | #define X_COLOR RED |
| 47 | #define O_COLOR BLUE |
| 48 | #define BACKGROUND_YELLOW (14 << 4) |
| 49 | |
| 50 | // ASCII code |
| 51 | // special key |
| 52 | #define ENTER 13 |
| 53 | #define ESC 27 |
| 54 | #define BACK_SPACE 8 |
| 55 | |
| 56 | // for drawing the caro table |
| 57 | #define H_LINE (char)196 |

| | |
|-----|--|
| 58 | #define V_LINE (char)179 |
| 59 | #define CROSS (char)197 |
| 60 | #define TOP_LEFT (char)218 |
| 61 | #define TOP_RIGHT (char)191 |
| 62 | #define BOTTOM_LEFT (char)192 |
| 63 | #define BOTTOM_RIGHT (char)217 |
| 64 | #define TOP_CROSS (char)194 |
| 65 | #define BOTTOM_CROSS (char)193 |
| 66 | #define |
| 67 | #define RIGHT_CROSS (char)180 |
| 68 | #define L_TRIANGLE ((char)16) |
| 69 | #define R_TRIANGLE ((char)17) |
| 70 | // for drawing boxes (menu, etc.) |
| 71 | #define BOX_TOP_LEFT (char)201 |
| 72 | #define BOX_TOP_RIGHT (char)187 |
| 73 | #define BOX_BOTTOM_LEFT (char)200 |
| 74 | #define BOX_BOTTOM_RIGHT (char)188 |
| 75 | #define BOX_RIGHT (char)185 |
| 76 | #define BOX_LEFT (char)204 |
| 77 | #define BOX_V_LINE (char)186 |
| 78 | #define BOX_H_LINE (char)205 |
| 79 | #define BOX_X WIDTH / 2 - 20 |
| 80 | #define BOX_Y HEIGHT / 2 - 13 |
| 81 | #define BOX_W 40 |
| 82 | #define BOX_H 20 |
| 83 | // moving keys |
| 84 | #define W 87 |
| 85 | #define A 65 |
| 86 | #define S 83 |
| 87 | #define D 68 |
| 88 | // others |
| 89 | #define SPACE (char)32 |
| 90 | #define key_none -1 |
| 91 | |
| 92 | // Data types |
| 93 | struct _POINT { |
| 94 | int x, y, c; |
| 95 | // x, y: coordinate |
| 96 | // c -> {0, 1, 2}, 0: vacant, 1: player X, 2: player O |
| 97 | }; |
| 98 | |
| 99 | struct _BUTTON { |
| 100 | int x, y; |
| 101 | string data; |
| 102 | }; |
| 103 | |
| 104 | struct WinningPos { |
| 105 | int x, y; |
| 106 | }; |

2. Nhóm hàm View

a. Các hàm công cụ

Là nhóm các hàm cực kỳ quan trọng, dùng để tạo nên các thành phần giao diện chính của game

- FixConsoleWindow: Hàm cố định của sổ console để người dùng không thể thay đổi kích thước console

| | |
|---|---|
| 1 | void FixConsoleWindow() { |
| 2 | HWND consoleWindow = GetConsoleWindow(); |
| 3 | LONG style = GetWindowLong(consoleWindow, GWL_STYLE); |
| 4 | style = style & ~(WS_MAXIMIZEBOX) & ~(WS_THICKFRAME); |
| 5 | SetWindowLong(consoleWindow, GWL_STYLE, style); |

| | |
|---|---|
| 6 | } |
|---|---|

- changeFont: hàm thay đổi kích thước cửa sổ console

| | |
|----|--|
| 1 | void changeFont(int x) |
| 2 | { |
| 3 | CONSOLE_FONT_INFOEX cfi; |
| 4 | cfi.cbSize = sizeof(cfi); |
| 5 | cfi.nFont = 0; |
| 6 | cfi.dwFontSize.X = 0; |
| 7 | cfi.dwFontSize.Y = x; |
| 8 | cfi.FontFamily = FF_DONTCARE; |
| 9 | cfi.FontWeight = FW_BOLD; |
| 10 | wcscpy_s(cfi.FaceName, L"Consolas"); |
| 11 | SetCurrentConsoleFontEx(GetStdHandle(STD_OUTPUT_HANDLE), FALSE, &cfi); |
| 12 | } |

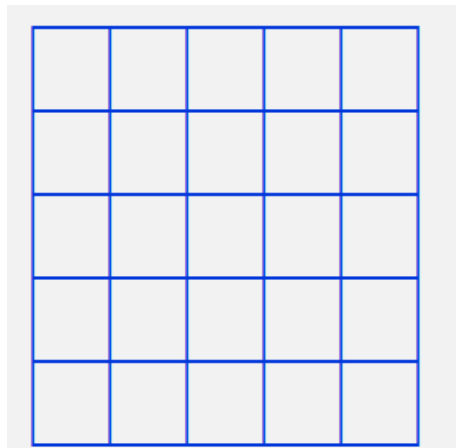
- HideCursor: hàm dùng để ẩn/hiện con trỏ trên màn hình console

| | |
|---|---|
| 1 | void HideCursor(bool ok) { |
| 2 | HANDLE consoleHandle = GetStdHandle(STD_OUTPUT_HANDLE); |
| 3 | CONSOLE_CURSOR_INFO info; |
| 4 | info.dwSize = 100; |
| 5 | info.bVisible = !ok; |
| 6 | SetConsoleCursorInfo(consoleHandle, &info); |
| 7 | } |

- DrawBoard: Vẽ bàn cờ caro, với tham số truyền vào lần lượt là số hàng, số cột, tọa độ x, y và màu sắc

| | |
|----|---|
| 1 | void DrawBoard(int r, int c, int x, int y, int color) { |
| 2 | int tmp = GetCurrentColor(); |
| 3 | GotoXY(x, y); |
| 4 | TextColor(color); |
| 5 | // top row |
| 6 | cout << TOP_LEFT << H_LINE << H_LINE << H_LINE; |
| 7 | for (int i = 1; i < c; i++) |
| 8 | cout << TOP_CROSS << H_LINE << H_LINE << H_LINE; |
| 9 | cout << TOP_RIGHT; |
| 10 | GotoXY(x, y + 1); |
| 11 | for (int i = 0; i <= c; i++) |
| 12 | cout << V_LINE << SPACE << SPACE << SPACE; |
| 13 | // all the middle row |
| 14 | for (int i = 1; i < r; i++) { |
| 15 | GotoXY(x, y + i * 2); |
| 16 | cout << LEFT_CROSS << H_LINE << H_LINE << H_LINE; |
| 17 | for (int j = 1; j < c; j++) |
| 18 | cout << CROSS << H_LINE << H_LINE << H_LINE; |
| 19 | cout << RIGHT_CROSS; |
| 20 | GotoXY(x, y + i * 2 + 1); |
| 21 | for (int j = 0; j <= c; j++) |
| 22 | cout << V_LINE << SPACE << SPACE << SPACE; |
| 23 | } |
| 24 | // bottom row |
| 25 | GotoXY(x, y + 2 + 2 * (r - 1)); |
| 26 | cout << BOTTOM_LEFT << H_LINE << H_LINE << H_LINE; |
| 27 | for (int i = 1; i < c; i++) |
| 28 | cout << BOTTOM_CROSS << H_LINE << H_LINE << H_LINE; |

| | |
|----|--|
| 29 | <code>cout << BOTTOM_RIGHT;</code> |
| 30 | <code>TextColor(tmp);</code> |
| 31 | <code>}</code> |



bàn cờ 5x5 được vẽ bởi hàm DrawBoard

- DrawBox() và DrawBoxMini(): Hàm vẽ các bảng với tham số truyền vào lần lượt là độ rộng, độ dài bảng, tọa độ x, y và màu sắc của bảng, với sự khác nhau về đường viền. Hàm DrawBox có thêm tham số time – thời gian để tạo hiệu ứng xuất hiện cho bảng. Đây là 2 hàm quan trọng và được sử dụng rất nhiều trong quá trình thiết kế giao diện

| | |
|----|--|
| 1 | <code>void DrawBox(int w, int h, int x, int y, int color, int Time) {</code> |
| 2 | <code>int tmp = GetCurrentColor();</code> |
| 3 | <code>TextColor(color);</code> |
| 4 | <code>for (int i = 0; i < w / 2; i++) {</code> |
| 5 | <code>GotoXY(x + w / 2 - i, y);</code> |
| 6 | <code>cout << BOX_H_LINE;</code> |
| 7 | <code>GotoXY(x + w / 2 + i, y);</code> |
| 8 | <code>cout << BOX_H_LINE;</code> |
| 9 | <code>Sleep(Time);</code> |
| 10 | <code>}</code> |
| 11 | <code>GotoXY(x, y);</code> |
| 12 | <code>cout << BOX_TOP_LEFT;</code> |
| 13 | <code>GotoXY(x + w, y);</code> |
| 14 | <code>cout << BOX_TOP_RIGHT;</code> |
| 15 | <code>GotoXY(x, y + i * 2);</code> |
| 16 | <code>for (int i = 1; i < h - 1; i++) {</code> |
| 17 | <code>GotoXY(x, y + i);</code> |
| 18 | <code>cout << BOX_V_LINE;</code> |
| 19 | <code>for (int j = 1; j < w; j++)</code> |
| 20 | <code>cout << SPACE;</code> |
| 21 | <code>cout << BOX_V_LINE;</code> |
| 22 | <code>Sleep(Time);</code> |
| 23 | <code>}</code> |
| 24 | <code>GotoXY(x, y + h - 1);</code> |
| 25 | <code>cout << BOX_BOTTOM_LEFT;</code> |
| 26 | <code>GotoXY(x + w, y + h - 1);</code> |
| 27 | <code>cout << BOX_BOTTOM_RIGHT;</code> |
| 28 | <code>for (int i = w / 2 - 1; i >= 0; i--) {</code> |
| 29 | <code>GotoXY(x + w / 2 - i, y + h - 1);</code> |
| 30 | <code>cout << BOX_H_LINE;</code> |
| 31 | <code>GotoXY(x + w / 2 + i, y + h - 1);</code> |

| | |
|---|--|
| 3 | GotoXY(x, y); |
| 4 | unsigned char Note[] = { 179, 32, 80, 114, 111, 106, 101, 99, 116, 32, 98, 121, 32, 71, 114, 111, 117, 112, 32, 49, 51, 32, 45, 32, 50, 50, 67, 84, 84, 52, 32, 72, 67, 77, 85, 83, 32, 179 }; |
| 5 | for (int i = 0; i < 38; i++) { |
| 6 | cout << Note[i]; |
| 7 | Sleep(10); } |
| 8 | } |

- DrawPattern_Col(): Hàm vẽ các họa tiết theo chiều dọc, với tham số truyền vào là tọa độ x, y, màu sắc (color), giá trị của họa tiết (theo bảng mã Ascii) và khoảng cách giữa các họa tiết

| | |
|---|---|
| 1 | void DrawPattern_Col(int x, int y, int color, int pt, int kc) { |
| 2 | TextColor(color); |
| 3 | unsigned char pattern = pt; |
| 4 | for (int i = 0; i < 30; i = i + kc) { |
| 5 | GotoXY(y, x + i); |
| 6 | putchar(pattern); } |
| 7 | } |

- getFileContents() dùng để đọc dữ liệu từ file, đọc lần lượt các dòng và lưu vào 1 chuỗi sau đó trả về (hàm này chủ yếu dùng để lưu vào 1 string hỗ trợ các hàm Draw, DrawAsciiFile để in ra màn hình console)

| | |
|----|--|
| 1 | string getFileContents(ifstream& File) |
| 2 | { |
| 3 | string Lines = ""; //All lines |
| 4 | |
| 5 | if (File) //Check if everything is |
| 6 | good |
| 7 | { |
| 8 | while (File.good()) |
| 9 | { |
| 10 | string TempLine; //Temp |
| 11 | getline(File, TempLine); //Get temp |
| 12 | TempLine += "\n"; //Add |
| 13 | newline character |
| 14 | Lines += TempLine; //Add |
| 15 | newline |
| 16 | } |
| 17 | return Lines; |
| 18 | } |
| 19 | else //Return error |
| 20 | { |
| 21 | return "ERROR File does not exist."; |
| 22 | } |

- Hàm Draw() sẽ đi đôi với hàm getFileContents() để in ra dữ liệu từ 1 tệp tin ra màn hình console

| | |
|---|---|
| 1 | void Draw(int x, int y, string nameFile, int color) { |
| 2 | TextColor(color); |

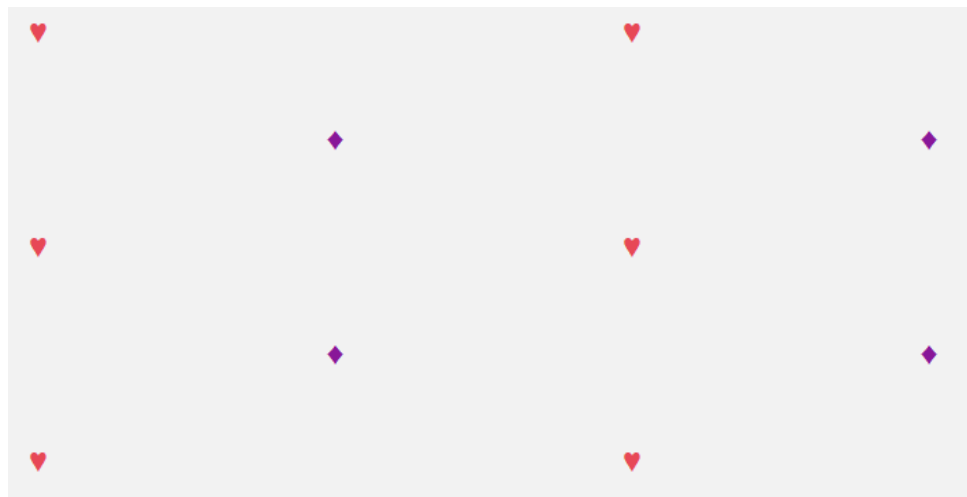
| | | |
|----|--|------------------------|
| 3 | SetConsoleOutputCP(65001); | //Chọn bộ hóa để |
| | uft-8 | |
| 4 | ifstream Read("assets/UI/" + nameFile + ".txt"); | |
| 5 | string line = ""; | |
| 6 | for (int i = 0; Read.good(); i++) { | |
| 7 | getline(Read, line); | |
| 8 | GotoXY(x - (i == 0), y + i); | |
| 9 | cout << line; | |
| 10 | } | |
| 11 | Read.close(); | |
| 12 | SetConsoleOutputCP(437); | //Đưa bộ mã hóa về mặc |
| | định của console | |
| 13 | } | |

b. Các hàm vẽ giao diện game

Là các hàm sử dụng các hàm công cụ để tạo nên giao diện cho trò chơi

- Background(): Tổng hợp hàm DrawPattern_Col() kết hợp với nhau để tạo nên hình nền cho giao diện game

| | |
|---|---|
| 1 | void Background() { |
| 2 | DrawPattern_Col(1, 6, LIGHT_RED, 3, 6); |
| 3 | DrawPattern_Col(4, 24, MAGENTA, 4, 6); |
| 4 | DrawPattern_Col(1, 42, LIGHT_RED, 3, 6); |
| 5 | DrawPattern_Col(4, 60, MAGENTA, 4, 6); |
| 6 | DrawPattern_Col(1, 78, LIGHT_RED, 3, 6); |
| 7 | DrawPattern_Col(4, 96, MAGENTA, 4, 6); |
| 8 | DrawPattern_Col(1, 114, LIGHT_RED, 3, 6); |
| 9 | } |



- DrawButton(): Ứng dụng hàm DrawBoxMini() để vẽ các nút ấn “PLAY”, “LOAD”, “HELP”, “SETTING”, “EXIT”

| | |
|---|----------------------------------|
| 1 | void DrawButton() { |
| 2 | DrawBoxMini(14, 3, 89, 6, BLUE); |
| 3 | GotoXY(94, 7); |
| 4 | TextColor(BLUE); |
| 5 | cout << "PLAY"; |
| 6 | Sleep(200); |

| | |
|----|-----------------------------------|
| 7 | DrawBoxMini(14, 3, 89, 9, BLUE); |
| 8 | GotoXY(94, 10); |
| 9 | TextColor(BLUE); |
| 10 | cout << "LOAD"; |
| 11 | Sleep(200); |
| 12 | DrawBoxMini(14, 3, 89, 12, BLUE); |
| 13 | GotoXY(94, 13); |
| 14 | TextColor(BLUE); |
| 15 | cout << "HELP"; |
| 16 | Sleep(200); |
| 17 | DrawBoxMini(14, 3, 89, 15, BLUE); |
| 18 | GotoXY(93, 16); |
| 19 | TextColor(BLUE); |
| 20 | cout << "SETTING"; |
| 21 | Sleep(200); |
| 22 | DrawBoxMini(14, 3, 89, 18, BLUE); |
| 23 | GotoXY(94, 19); |
| 24 | TextColor(BLUE); |
| 25 | cout << "EXIT"; |
| 26 | Sleep(200); |
| 27 | } |

- DrawCaroBox(): Hàm vẽ bảng kết hợp hiệu ứng xuất hiện, kèm theo các hàm CaroAnimation(), PrintNote() và câu lệnh Sleep() để tạo sự xuất hiện cho giao diện Menu một cách bắt mắt

| | |
|----|--|
| 1 | void DrawCaroBox(int w, int h, int x, int y, int Time) { |
| 2 | TextColor(GREEN); |
| 3 | GotoXY(x + w / 2, y); |
| 4 | for (int i = 0; i < w / 2; i++) { |
| 5 | GotoXY(x + w / 2 + i, y); |
| 6 | cout << BOX_H_LINE; |
| 7 | GotoXY(x + w / 2 - i, y); |
| 8 | cout << BOX_H_LINE; |
| 9 | Sleep(Time); |
| 10 | } |
| 11 | GotoXY(x, y); |
| 12 | cout << BOX_TOP_LEFT; |
| 13 | GotoXY(x + w, y); |
| 14 | cout << BOX_TOP_RIGHT; |
| 15 | for (int i = 1; i < h - 1; i++) { |
| 16 | GotoXY(x, i + y); |
| 17 | cout << BOX_V_LINE; |
| 18 | for (int j = 1; j < w; j++) |
| 19 | cout << SPACE; |
| 20 | cout << BOX_V_LINE; |
| 21 | Sleep(Time); |
| 22 | } |
| 23 | GotoXY(x, h + y - 1); |
| 24 | cout << BOX_BOTTOM_LEFT; |
| 25 | GotoXY(x + w, h + y - 1); |
| 26 | cout << BOX_BOTTOM_RIGHT; |
| 27 | GotoXY(x, y + h - 5); |
| 28 | cout << BOX_LEFT; |
| 29 | GotoXY(x + w, y + h - 5); |
| 30 | cout << BOX_RIGHT; |
| 31 | for (int i = 1; i <= w / 2; i++) { |
| 32 | GotoXY(x + i, y + h - 1); |

| | |
|----|---|
| 33 | cout << BOX_H_LINE; |
| 34 | GotoXY(x + w - i, y + h - 1); |
| 35 | cout << BOX_H_LINE; |
| 36 | GotoXY(x + i, y + h - 5); |
| 37 | cout << BOX_H_LINE; |
| 38 | GotoXY(x + w - i, y + h - 5); |
| 39 | cout << BOX_H_LINE; |
| 40 | Sleep(Time); |
| 41 | } |
| 42 | DrawBoxMini(59, 13, 14, 6, LIGHT_YELLOW); |
| 43 | CaroAnimation(); |
| 44 | Sleep(500); |
| 45 | PrintNote(24, 21, BLACK); |
| 46 | } |

- mainScreen(): tổng hợp các hàm BackGround(), DrawCaroBox(), DrawButton() để thiết kế hoàn chỉnh giao diện Menu chính

| | |
|---|--------------------------------|
| 1 | void mainScreen() { |
| 2 | BackGround(); |
| 3 | DrawCaroBox(62, 19, 12, 5, 5); |
| 4 | Sleep(500); |
| 5 | DrawButton(); |
| 6 | } |

- MainMenu(): một trong những hàm chính trong game, hàm này sẽ gọi hàm mainScreen để vẽ giao diện và dùng một vòng while lặp vô hạn cho người dùng nhập vào các ký tự điều khiển con trỏ để chọn các nút chức năng tương ứng

| | |
|----|--|
| 1 | void MainMenu(_POINT _A[B_SIZE][B_SIZE], bool& _TURN, int& _COMMAND, bool sound[], int& _X, int& _Y, int& cX, int& cY, int& cntX, int& cnt0, int& cntWin0, int& cntLose0, int& cntDraw, int& saveTurn, int& cntRound, string& NamePlayer_0, string& NamePlayer_X, float& remain, WinningPos WP[5]) |
| 2 | { |
| 3 | Sleep(50); |
| | vẽ Menu ban đầu: |
| 4 | SetConsoleBlank(); |
| 5 | MainScreen(); |
| 6 | |
| 7 | int x = 94; |
| 8 | int y = 7; |
| 9 | DrawBoxMini(14, 3, 89, y - 1, RED); |
| 10 | GotoXY(x, y); // Đưa cursor tới phím chức năng đầu tiên. |
| 11 | TextColor(RED); |
| 12 | cout << "PLAY"; |
| | thay đổi màu sắc phím của Menu |
| 13 | while (true) |
| 14 | { |
| 15 | unsigned char c = toupper(_getch()); |
| 16 | if (sound[CLICK_SFX]) PlayAudio(CLICK_SFX); |
| 17 | if (c == 'W' c == 'S') |
| 18 | { |
| | Nếu chọn đi lên |
| 19 | if (c == 'W') |
| 20 | { |
| 21 | // Đưa lại nút cũ về màu xanh |
| 22 | DrawBoxMini(14, 3, 89, y - 1, BLUE); |
| 23 | TextColor(BLUE); |

| | |
|--|--|
| 24 | <code>if (y == 7)</code> |
| 25 | <code>{</code> |
| 26 | <code> GotoXY(x, y);</code> |
| 27 | <code> cout << "PLAY";</code> |
| 28 | <code>}</code> |
| 29 | <code>if (y == 10)</code> |
| 30 | <code>{</code> |
| 31 | <code> GotoXY(x, y);</code> |
| 32 | <code> cout << "LOAD";</code> |
| 33 | <code>}</code> |
| 34 | <code>if (y == 13)</code> |
| 35 | <code>{</code> |
| 36 | <code> GotoXY(x, y);</code> |
| 37 | <code> cout << "HELP";</code> |
| 38 | <code>}</code> |
| 39 | <code>if (y == 16)</code> |
| 40 | <code>{</code> |
| 41 | <code> GotoXY(x - 1, y);</code> |
| 42 | <code> cout << "SETTING";</code> |
| 43 | <code>}</code> |
| 44 | <code>if (y == 19)</code> |
| 45 | <code>{</code> |
| 46 | <code> GotoXY(x, y);</code> |
| 47 | <code> cout << "EXIT";</code> |
| 48 | <code>}</code> |
| 49 | <code>y = y - 3;</code> |
| 50 | <code>if (y < 7)</code> |
| 51 | <code> y = 19;</code> |
| Biển nút đang trở vào thành màu đỏ | |
| 52 | <code> DrawBoxMini(14, 3, 89, y - 1, RED);</code> |
| 53 | <code> TextColor(RED);</code> |
| 54 | <code>if (y == 7)</code> |
| 55 | <code>{</code> |
| 56 | <code> GotoXY(x, y);</code> |
| 57 | <code> cout << "PLAY";</code> |
| 58 | <code>}</code> |
| 59 | <code>if (y == 10)</code> |
| 60 | <code>{</code> |
| 61 | <code> GotoXY(x, y);</code> |
| 62 | <code> cout << "LOAD";</code> |
| 63 | <code>}</code> |
| 64 | <code>if (y == 13)</code> |
| 65 | <code>{</code> |
| 66 | <code> GotoXY(x, y);</code> |
| 67 | <code> cout << "HELP";</code> |
| 68 | <code>}</code> |
| 69 | <code>if (y == 16)</code> |
| 70 | <code>{</code> |
| 71 | <code> GotoXY(x - 1, y);</code> |
| 72 | <code> cout << "SETTING";</code> |
| 73 | <code>}</code> |
| 74 | <code>if (y == 19)</code> |
| 75 | <code>{</code> |
| 76 | <code> GotoXY(x, y);</code> |
| 77 | <code> cout << "EXIT";</code> |
| 78 | <code>}</code> |
| 79 | <code>}</code> |
| Nếu chọn đi xuống | |
| 80 | <code> else if (c == 'S')</code> |
| 81 | <code>{</code> |
| Đưa lại nút cũ về màu xanh trước khi chuyển sang nút mới | |

| | |
|------------------------------------|--------------------------------------|
| 82 | DrawBoxMini(14, 3, 89, y - 1, BLUE); |
| 83 | TextColor(BLUE); |
| 84 | if (y == 7) |
| 85 | { |
| 86 | GotoXY(x, y); |
| 87 | cout << "PLAY"; |
| 88 | } |
| 89 | if (y == 10) |
| 90 | { |
| 91 | GotoXY(x, y); |
| 92 | cout << "LOAD"; |
| 93 | } |
| 94 | if (y == 13) |
| 95 | { |
| 96 | GotoXY(x, y); |
| 97 | cout << "HELP"; |
| 98 | } |
| 99 | if (y == 16) |
| 100 | { |
| 101 | GotoXY(x - 1, y); |
| 102 | cout << "SETTING"; |
| 103 | } |
| 104 | if (y == 19) |
| 105 | { |
| 106 | GotoXY(x, y); |
| 107 | cout << "EXIT"; |
| 108 | } |
| 109 | y = y + 3; |
| 110 | if (y > 19) |
| 111 | y = 7; |
| Biến nút đang trở vào thành màu đỏ | |
| 112 | DrawBoxMini(14, 3, 89, y - 1, RED); |
| 113 | TextColor(RED); |
| 114 | |
| 115 | if (y == 7) |
| 116 | { |
| 117 | GotoXY(x, y); |
| 118 | cout << "PLAY"; |
| 119 | } |
| 120 | if (y == 10) |
| 121 | { |
| 122 | GotoXY(x, y); |
| 123 | cout << "LOAD"; |
| 124 | } |
| 125 | if (y == 13) |
| 126 | { |
| 127 | GotoXY(x, y); |
| 128 | cout << "HELP"; |
| 129 | } |
| 130 | if (y == 16) |
| 131 | { |
| 132 | GotoXY(x - 1, y); |
| 133 | cout << "SETTING"; |
| 134 | } |
| 135 | if (y == 19) |
| 136 | { |
| 137 | GotoXY(x, y); |
| 138 | cout << "EXIT"; |
| 139 | } |
| 140 | } |

| | |
|-----|---|
| 141 | } |
| | thao tác với phím chức năng của Menu |
| 142 | else if ((c == 32) (c == 13)) // Tương tác với phím bằng cách bấm Enter hoặc Space |
| 143 | { |
| 144 | if (y == 7) |
| 145 | { |
| 146 | EnterNamePlayer(NamePlayer_0, NamePlayer_X); |
| 147 | AskTurn(_TURN, sound, NamePlayer_0, NamePlayer_X); |
| 148 | cntWin0 = cntLose0 = cntDraw = 0; |
| 149 | cntRound = 1; |
| 150 | StartGame(_A, 1, _TURN, _COMMAND, sound, _X, _Y, cX, cY, cntX, cnt0, cntWin0, cntLose0, cntDraw, saveTurn, cntRound, NamePlayer_0, NamePlayer_X, remain, WP); |
| 151 | return; |
| 152 | } |
| 153 | if (y == 10) |
| 154 | { |
| 155 | LoadGameMenu(_A, _TURN, _COMMAND, sound, _X, _Y, cX, cY, cntX, cnt0, cntWin0, cntLose0, cntDraw, saveTurn, cntRound, NamePlayer_0, NamePlayer_X, remain, WP); |
| 156 | return; |
| 157 | } |
| 158 | if (y == 13) |
| 159 | { |
| 160 | HelpScreen(sound); |
| 161 | return; |
| 162 | } |
| 163 | if (y == 16) |
| 164 | { |
| 165 | SettingMenu(sound); |
| 166 | return; |
| 167 | } |
| 168 | if (y == 19) |
| 169 | { |
| 170 | ExitGame(); |
| 171 | } |
| 172 | } |
| 173 | } |
| 174 | } |

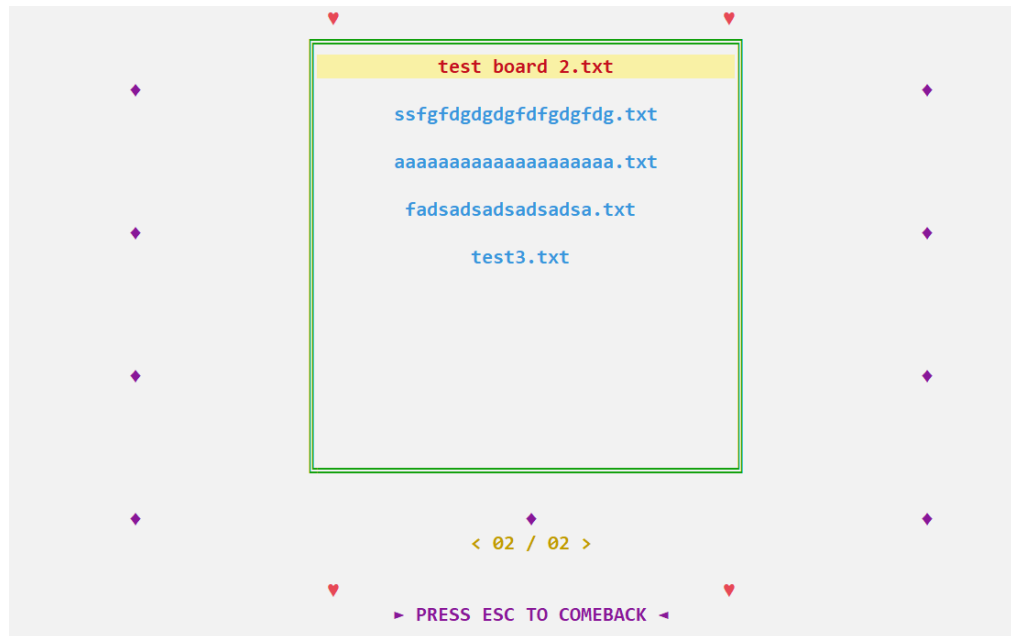


- LoadGameMenu(): để đọc các file game đã lưu và xuất ra màn hình cho người chơi lựa chọn

| | |
|---|--|
| 1 | void LoadGameMenu(_POINT _A[B_SIZE][B_SIZE], bool& _TURN, int& _COMMAND, bool sound[], int& _X, int& _Y, int& cX, int& cY, int& cntX, int& cnt0, int& cntWin0, int& cntLose0, int& cntDraw, int& saveTurn, int& cntRound, string& NamePlayer_0, string& NamePlayer_X, float& remain, WinningPos WP[5]) { |
| 2 | // Open the file that contains all the name of the saved games. |
| 3 | fstream inp; |
| 4 | inp.open("save/all_save.txt", ios::in); |
| 5 | if (inp.fail()) { |
| 6 | cout << "Can't open file\n"; |

| | |
|----|---|
| 7 | return; |
| 8 | } |
| 9 | |
| 10 | SetConsoleBlank(); |
| 11 | BackGround(); |
| 12 | DrawBox(BOX_W, BOX_H - 1, BOX_X, BOX_Y, GREEN, 1); |
| 13 | TextColor(CYAN); |
| 14 | |
| 15 | // Get the current save data |
| 16 | string nameFile; |
| 17 | int filesPerPage = 9; |
| 18 | vector<_BUTTON> v; |
| 19 | while (inp.good()) { |
| 20 | getline(inp, nameFile); |
| 21 | if (v.size() == 1 && v[0].data == "") |
| 22 | v[0].data = nameFile; |
| 23 | else |
| 24 | v.push_back({ 0, 0, nameFile }); |
| 25 | } |
| 26 | inp.close(); |
| 27 | GotoXY(47, 26); |
| 28 | TextColor(MAGENTA); |
| 29 | cout << " " << L_TRIANGLE << " PRESS ESC TO COMEBACK " << R_TRIANGLE << " "; |
| 30 | if (v.size() == 1 && v[0].data == "") { |
| 31 | GotoXY(53, 11); |
| 32 | cout << "< EMPTY DATA >"; |
| 33 | while (true) { |
| 34 | char c = toupper(_getch()); |
| 35 | if (sound[CLICK_SFX]) PlayAudio(CLICK_SFX); |
| 36 | if (c == ESC) |
| 37 | return; |
| 38 | } |
| 39 | } |
| 40 | for (int i = 0, cnt = 1; i < v.size(); i++, cnt = cnt % filesPerPage + 1) { |
| 41 | //GotoXY(BOX_X + 1, BOX_Y + 2 * cnt - 1); |
| 42 | v[i].x = BOX_X + 1; |
| 43 | v[i].y = BOX_Y + 2 * cnt - 1; |
| 44 | int x = (BOX_W - 2 - v[i].data.size()) / 2; |
| 45 | if (v[i].data.size() % 2 == 1) |
| 46 | v[i].data += " "; |
| 47 | while (x--) |
| 48 | v[i].data = " " + v[i].data, v[i].data += " "; |
| 49 | //v[i].data += " "; |
| 50 | } |
| 51 | int nFiles = v.size(); |
| 52 | int nPages = ceil(1.0 * nFiles / filesPerPage); |
| 53 | DrawSaveFilesPage(v, 1, filesPerPage); |
| 54 | int curFile = 0, prvFile = -1, curPage = 1, lastFile = 0; |
| 55 | HoverButton(v[curFile]); |
| 56 | while (true) { |
| 57 | int _COMMAND = toupper(_getch()); |
| 58 | if (sound[CLICK_SFX]) PlayAudio(CLICK_SFX); |
| 59 | lastFile = (curPage == nPages ? (nFiles - 1) % filesPerPage : filesPerPage - 1); |
| 60 | if (_COMMAND == ESC) { |
| 61 | return; |
| 62 | } |
| 63 | else if (_COMMAND == W) { |

| | |
|-----|---|
| 64 | prvFile = curFile--; |
| 65 | if (curFile < 0) |
| 66 | curFile = lastFile; |
| 67 | } |
| 68 | else if (_COMMAND == S) { |
| 69 | prvFile = curFile++; |
| 70 | if (curFile > lastFile) |
| 71 | curFile = 0; |
| 72 | } |
| 73 | else if (_COMMAND == A) { |
| 74 | if (--curPage < 1) |
| 75 | curPage = nPages; |
| 76 | prvFile = curFile; |
| 77 | DrawSaveFilesPage(v, curPage, filesPerPage); |
| 78 | } |
| 79 | else if (_COMMAND == D) { |
| 80 | curPage++; |
| 81 | if (curPage > nPages) |
| 82 | curPage = 1; |
| 83 | prvFile = curFile; |
| 84 | DrawSaveFilesPage(v, curPage, filesPerPage); |
| 85 | } |
| 86 | else if (_COMMAND == ENTER) { |
| 87 | LoadData(_A, _TURN, _COMMAND, _X, _Y, cX, cY, cntX, cnt0, cntWin0, cntLose0, cntDraw, saveTurn, cntRound, NamePlayer_0, NamePlayer_X, CleanFileName(v[(curPage - 1) * filesPerPage + curFile].data), remain); |
| 88 | LoadingScreen(BLUE, GREEN, LIGHT_CYAN); |
| 89 | StartGame(_A, 0, _TURN, _COMMAND, sound, _X, _Y, cX, cY, cntX, cnt0, cntWin0, cntLose0, cntDraw, saveTurn, cntRound, NamePlayer_0, NamePlayer_X, remain, WP); |
| 90 | return; |
| 91 | } |
| 92 | |
| 93 | lastFile = (curPage == nPages ? (nFiles - 1) % filesPerPage : filesPerPage - 1); |
| 94 | |
| 95 | if (curFile + filesPerPage * (curPage - 1) >= nFiles) |
| 96 | prvFile = curFile = lastFile; |
| 97 | UnhoverButton(v[prvFile + filesPerPage * (curPage - 1)], CYAN); |
| 98 | HoverButton(v[curFile + filesPerPage * (curPage - 1)]); |
| 99 | } |
| 100 | } |



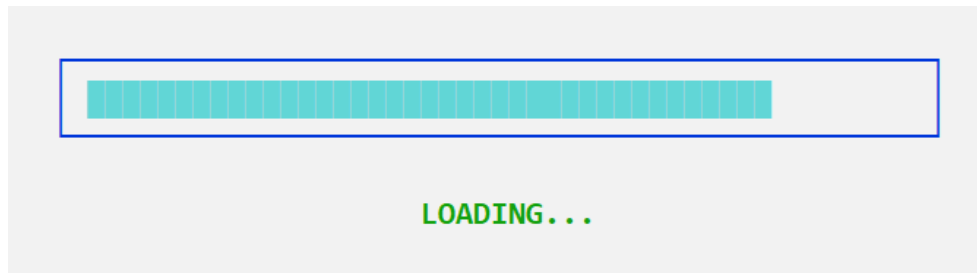
- HelpScreen(): Vẽ khung và ghi các thông tin cần thiết, bao gồm luật chơi, cách di chuyển và thông tin Project trong trang Help

| | |
|----|--|
| 1 | <code>void HelpScreen(bool sound[]) {</code> |
| 2 | <code> SetConsoleBlank();</code> |
| 3 | <code> DrawBox(98, 25, 11, 2, CYAN, 0);</code> |
| 4 | <code> DrawBoxMini(94, 23, 13, 3, LIGHT_CYAN);</code> |
| 5 | <code> for (int i = 0; i < 21; i++) {</code> |
| 6 | <code> TextColor(LIGHT_CYAN);</code> |
| 7 | <code> GotoXY(34, 4 + i);</code> |
| 8 | <code> cout << V_LINE; }</code> |
| 9 | <code> TextColor(RED);</code> |
| 10 | <code> GotoXY(20, 7);</code> |
| 11 | <code> cout << L_TRIANGLE << " RULE " << R_TRIANGLE;</code> |
| 12 | <code> GotoXY(40, 5);</code> |
| 13 | <code> cout << "The game is played on a board with 12 rows and</code> |
| 14 | <code>12 columns." << endl;</code> |
| 15 | <code> GotoXY(40, 7);</code> |
| 16 | <code> cout << "To win the round, player needs to make a line</code> |
| 17 | <code>consisting of 5" << endl;</code> |
| 18 | <code> GotoXY(40, 8);</code> |
| 19 | <code> cout << "consecutive pieces horizontally, vertically or</code> |
| 20 | <code>diagonally." << endl;</code> |
| 21 | <code> GotoXY(40, 10);</code> |
| 22 | <code> cout << "If more than 90% of the board has been marked,</code> |
| 23 | <code>the result will ";</code> |
| 24 | <code> GotoXY(40, 11);</code> |
| 25 | <code> cout << "be a draw.";</code> |
| 26 | <code> TextColor(GREEN);</code> |
| 27 | <code> GotoXY(20, 14);</code> |
| 28 | <code> cout << L_TRIANGLE << " MOVE " << R_TRIANGLE;</code> |
| 29 | <code> GotoXY(40, 14);</code> |
| 30 | <code> cout << "Up: W";</code> |
| | <code> GotoXY(40, 15);</code> |
| | <code> cout << "Down: S";</code> |
| | <code> GotoXY(55, 14);</code> |
| | <code> cout << "Left: A";</code> |

| | |
|----|--|
| 31 | GotoXY(55, 15); |
| 32 | cout << "Right: D"; |
| 33 | GotoXY(70, 14); |
| 34 | cout << "Choose: Enter"; |
| 35 | TextColor(MAGENTA); |
| 36 | GotoXY(18, 20); |
| 37 | cout << L_TRIANGLE << " ABOUT US " << R_TRIANGLE; |
| 38 | GotoXY(76, 19); |
| 39 | cout << "Teacher: Truong Toan Thinh"; |
| 40 | GotoXY(76, 20); |
| 41 | cout << "Group: 13"; |
| 42 | GotoXY(76, 21); |
| 43 | cout << "22CTT4 - HCMUS"; |
| 44 | GotoXY(40, 18); |
| 45 | cout << "Dang Duy Lan - 22120182"; |
| 46 | GotoXY(40, 19); |
| 47 | cout << "Nguyen Nhat Long - 22120194"; |
| 48 | GotoXY(40, 20); |
| 49 | cout << "Hoang Thanh Man - 22120200"; |
| 50 | GotoXY(40, 21); |
| 51 | cout << "Ly Truong Nam - 22120218"; |
| 52 | GotoXY(40, 22); |
| 53 | cout << "Tran Thao Ngan - 22120225"; |
| 54 | TextColor(LIGHT_RED); |
| 55 | GotoXY(47, 26); |
| 56 | cout << " " << L_TRIANGLE << " PRESS ESC TO COMEBACK " << R_TRIANGLE << " "; |
| 57 | while (true) { |
| 58 | char c = _getch(); |
| 59 | if (sound[CLICK_SFX]) PlayAudio(CLICK_SFX); |
| 60 | if (c == ESC) |
| 61 | return; |
| 62 | } |
| 63 | } |

- LoadingScreen(): Hàm vẽ trang Loading với tham số màu sắc truyền vào như ghi chú

| | |
|----|--|
| 1 | void LoadingScreen(int color1, int color2, int color3) { //color 1 là màu khung, 2 là màu chữ, 3 là màu cái thanh ở giữa |
| 2 | SetConsoleBlank(); |
| 3 | unsigned char s = { 219 }; |
| 4 | DrawBoxMini(51, 3, 34, 13, color1); |
| 5 | GotoXY(55, 17); |
| 6 | TextColor(color2); |
| 7 | cout << "LOADING..."; |
| 8 | GotoXY(36, 14); |
| 9 | for (int i = 0; i < 34; i++) { |
| 10 | TextColor(color3); |
| 11 | cout << s; |
| 12 | Sleep(10); } |
| 13 | Sleep(200); |
| 14 | GotoXY(70, 14); |
| 15 | for (int i = 0; i < 13; i++) { |
| 16 | TextColor(color3); |
| 17 | cout << s; |
| 18 | Sleep(10); } |
| 19 | } |



- ExitScreen(): Hàm vẽ trang Exit khi muốn thoát game

| | |
|----|--|
| 1 | <code>void ExitScreen()</code> |
| 2 | <code>{</code> |
| 3 | <code> SetConsoleBlank();</code> |
| 4 | <code> DrawAsciiFile(0, 0, "PatternBG", LIGHT_CYAN);</code> |
| 5 | <code> do {</code> |
| 6 | <code> DrawBoxMini(88, 12, 15, 8, LIGHT_CYAN);</code> |
| 7 | <code> DrawAsciiFile(24, 10, "DrawGoodbye", RED);</code> |
| 8 | <code> Sleep(300);</code> |
| 9 | <code> DrawAsciiFile(24, 10, "DrawGoodbye", GREEN);</code> |
| 10 | <code> Sleep(300);</code> |
| 11 | <code> DrawAsciiFile(24, 10, "DrawGoodbye", LIGHT_BLUE);</code> |
| 12 | <code> Sleep(300);</code> |
| 13 | <code> DrawAsciiFile(24, 10, "DrawGoodbye", MAGENTA);</code> |
| 14 | <code> Sleep(300);</code> |
| 15 | <code> } while (!_kbhit());</code> |
| 16 | <code> GotoXY(0, 25);</code> |
| 16 | <code>}</code> |

- Hàm EnterNamePlayer dùng để vẽ các ô cho người chơi có thể nhập tên

| | |
|----|---|
| 1 | <code>void EnterNamePlayer(string& NamePlayer_0, string& NamePlayer_X) {</code> |
| 2 | <code> unsigned char x = 16;</code> |
| 3 | <code> SetConsoleBlank();</code> |
| 4 | <code> HideCursor(0);</code> |
| 5 | <code> DrawBanner(15, 2, GREEN);</code> |
| 6 | <code> DrawEnterName(30, 3, MAGENTA);</code> |
| 7 | <code> TextColor(BLACK);</code> |
| 8 | <code> GotoXY(50, 27);</code> |
| 9 | <code> cout << "< 20 character max >";</code> |
| 10 | <code> DrawBoxMini(49, 3, 35, 14, GREEN);</code> |
| 11 | <code> TextColor(CYAN);</code> |
| 12 | <code> GotoXY(51, 13); cout << " Player 1's Name ";</code> |
| 13 | <code> TextColor(RED);</code> |
| 14 | <code> GotoXY(48, 15); cout << x << ' ';</code> |
| 15 | <code> NamePlayer_0 = NamePlayer_X = "";</code> |
| 16 | <code> EnterName(NamePlayer_0, 20);</code> |
| 17 | <code> DrawBoxMini(49, 3, 35, 19, GREEN);</code> |
| 18 | <code> TextColor(CYAN);</code> |
| 19 | <code> GotoXY(51, 18); cout << " Player 2's Name ";</code> |
| 20 | <code> TextColor(RED);</code> |
| 21 | <code> GotoXY(48, 20); cout << x << ' ';</code> |
| 22 | <code> EnterName(NamePlayer_X, 20);</code> |
| 23 | <code> HideCursor(1);</code> |
| 24 | <code>}</code> |

- Hàm AskTurn(): Hàm này có chức năng chính là dùng để hỏi xem người nào chọn O (hoặc X) để đánh trước. Và vẽ cái khung hiển thị tên, chọn tên. Vẽ hai hình X và

O, bấm 'A' hoặc 'D' để di chuyển sang trái sang phải để chọn turn. Khi con trỏ di chuyển sang ô nào thì ô đó sẽ nổi bật lên và sẽ chọn ô đó nếu ta bấm enter

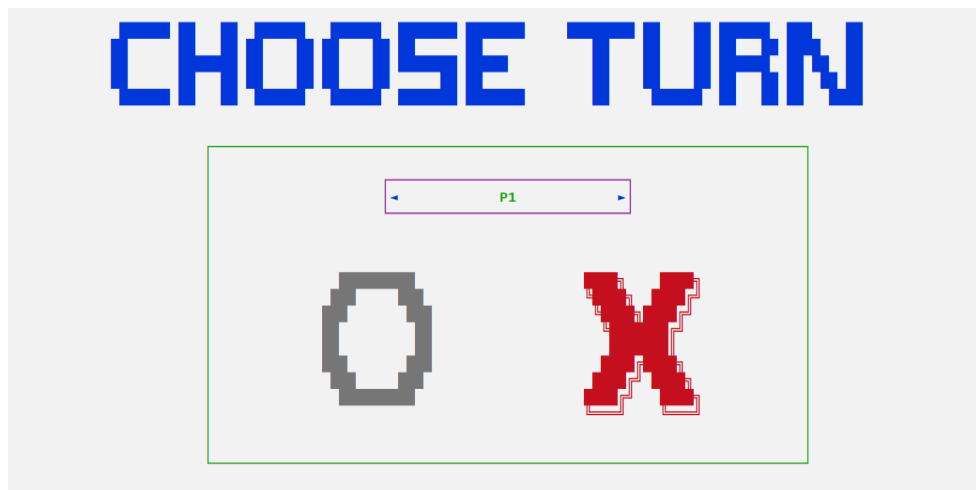
```

1 void AskTurn(bool& _TURN, bool sound[], string& NamePlayer_0,
2 string& NamePlayer_X) {
3     SetConsoleBlank();
4     DrawBoxMini(72, 20, 27, 8, GREEN); //vẽ khung to
5     // Vẽ hai ô hiển thị O vs X
6     int wide = 20;
7     int high = 10;
8     int x = 20;
9     int y = 10;
10    DrawAsciiFile(0, 1, "DrawTurn", BLUE);
11    Draw(41, 15, "OTurn", GRAY);
12    Draw(72, 15, "XTurn", GRAY);
13    // Vẽ ô hiển thị tên
14    int WideBoxName = 30;
15    int HighBoxName = 3;
16    int X_BoxName = 48;
17    int Y_BoxName = 10;
18    DrawBoxMini(WideBoxName, HighBoxName, X_BoxName,
19    Y_BoxName, MAGENTA);
20    GotoXY(X_BoxName + WideBoxName / 2 - NamePlayer_0.size() /
21    2, Y_BoxName + 1);
22    TextColor(GREEN);
23    cout << NamePlayer_0;
24    GotoXY(X_BoxName + 1, Y_BoxName + 1);
25    TextColor(BLUE);
26    cout << (unsigned char)17;
27    GotoXY(X_BoxName + WideBoxName - 2, Y_BoxName + 1);
28    TextColor(BLUE);
29    cout << (unsigned char)16;
30    TextColor(RED);
31    bool check = true;
32    while (true) {
33        char press = toupper(_getch());
34        if (sound[CLICK_SFX]) PlayAudio(CLICK_SFX);
35        if (press == 'D' || press == 'A') {
36            if (press == 'D')
37            {
38                GotoXY(X_BoxName + WideBoxName - 2, Y_BoxName
39                + 1);
40                TextColor(YELLOW);
41                cout << (unsigned char)16;
42                Sleep(200);
43                GotoXY(X_BoxName + WideBoxName - 2, Y_BoxName
44                + 1);
45                TextColor(BLUE);
46                cout << (unsigned char)16;
47            }
48            else if (press == 'A') {
49                GotoXY(X_BoxName + 1, Y_BoxName + 1);
50                TextColor(YELLOW);
51                cout << (unsigned char)17;
52                Sleep(200);
53                GotoXY(X_BoxName + 1, Y_BoxName + 1);
54                TextColor(BLUE);
55                cout << (unsigned char)17;
56            }
57        }
58        if (check == true)
59        {

```

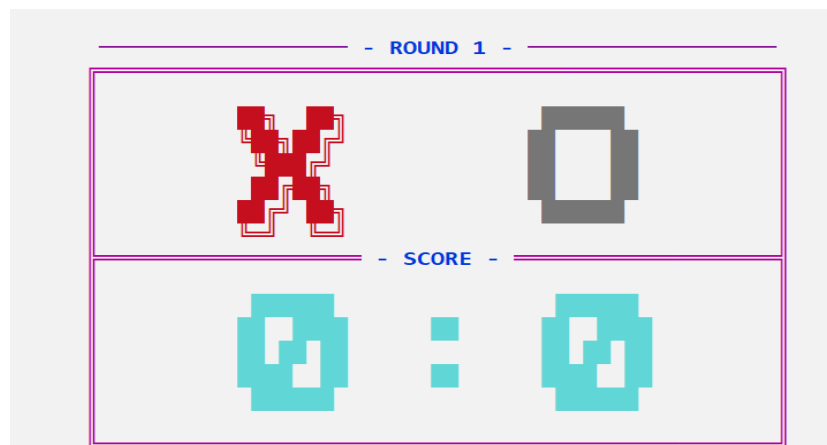
| | |
|-----|---|
| 54 | GotoXY(X_BoxName + WideBoxName / 2 - NamePlayer_0.size() / 2, Y_BoxName + 1); |
| 55 | for (int i = 0; i < NamePlayer_0.size(); i++) |
| 56 | cout << ' '; |
| 57 | GotoXY(X_BoxName + WideBoxName / 2 - NamePlayer_X.size() / 2, Y_BoxName + 1); |
| 58 | TextColor(GREEN); |
| 59 | cout << NamePlayer_X; |
| 60 | check = false; |
| 61 | } |
| 62 | else if (check == false) { |
| 63 | GotoXY(X_BoxName + WideBoxName / 2 - NamePlayer_X.size() / 2, Y_BoxName + 1); |
| 64 | for (int i = 0; i < NamePlayer_X.size(); i++) |
| 65 | cout << ' '; |
| 66 | GotoXY(X_BoxName + WideBoxName / 2 - NamePlayer_0.size() / 2, Y_BoxName + 1); |
| 67 | TextColor(GREEN); |
| 68 | cout << NamePlayer_0; |
| 69 | check = true; |
| 70 | } |
| 71 | } |
| 72 | else if (press == ENTER) { |
| 73 | break; |
| 74 | } |
| 75 | } |
| 76 | GotoXY(x, y); |
| 77 | //DrawAsciiFile(x + wide / 2, y + high / 2 - 2, "DrawTurn0", BLUE); |
| 78 | //VeO2(43, 17, 0_COLOR); |
| 79 | Draw(41, 15, "OTurnC", 0_COLOR); |
| 80 | while (true) { |
| 81 | unsigned char c = toupper(_getch()); |
| 82 | if (sound[CLICK_SFX]) PlayAudio(CLICK_SFX); |
| 83 | if (c == 'A' c == 'D') { |
| 84 | if (c == 'A') { |
| 85 | if (x == 50) { |
| 86 | Draw(72, 15, "XTurn", GRAY); |
| 87 | Draw(41, 15, "OTurnC", 0_COLOR); |
| 88 | x -= 30; |
| 89 | } |
| 90 | } |
| 91 | else if (c == 'D') { |
| 92 | if (x == 20) { |
| 93 | Draw(72, 15, "XTurnC", X_COLOR); |
| 94 | Draw(41, 15, "OTurn", GRAY); |
| 95 | x += 30; |
| 96 | } |
| 97 | } |
| 98 | } |
| 99 | else if (c == ENTER) { |
| 100 | if (x == 20) { |
| 101 | _TURN = true; // true là lượt của 0 |
| 102 | if (check == false) |
| 103 | swap(NamePlayer_0, NamePlayer_X); |
| 104 | return; |
| 105 | } |
| 106 | else if (x == 50) { |
| 107 | _TURN = false; // false là lượt của X |
| 108 | if (check == true) |
| 109 | swap(NamePlayer_0, NamePlayer_X); |

| | |
|-----|----------------------|
| 110 | <code>return;</code> |
| 111 | <code>}</code> |
| 112 | <code>}</code> |
| 113 | <code>}</code> |
| 114 | <code>}</code> |



- Hàm ShowTurn(): Hàm này dùng để hiển thị lượt đánh khi đang chơi. Nếu hiện X (hoặc O) thì lượt đánh hiện tại sẽ là của X (hoặc O). Đồng thời gọi hai hàm TurnX() hoặc TurnO() để hiển thị X và O đẹp hơn

| | |
|----|---|
| 1 | <code>void ShowTurn(int _X, int _Y, bool _TURN, bool validEnter)</code> |
| 2 | <code>{</code> |
| 3 | <code>int tmp = GetCurrentColor();</code> |
| 4 | <code>TextColor(CYAN);</code> |
| 5 | <code>if (_TURN == true && validEnter == true)</code> |
| 6 | <code>{</code> |
| 7 | <code>TurnX();</code> |
| 8 | <code>}</code> |
| 9 | <code>else if (_TURN == false && validEnter == true)</code> |
| 10 | <code>{</code> |
| 11 | <code>TurnO();</code> |
| 12 | <code>}</code> |
| 13 | <code>GotoXY(_X, _Y);</code> |
| 14 | <code>TextColor(tmp);</code> |
| 15 | <code>}</code> |



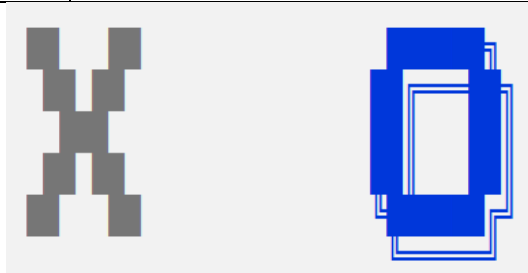
- Hàm VeO1(), VeX1(), VeO2(), VeX2() tương ứng với các hàm vẽ cho từng file text lần lượt là VeO1.txt, VeX1.txt, VeO2.txt, VeX2.txt

| | |
|----|---|
| 1 | <code>void VeO1(int x, int y, int color) {</code> |
| 2 | <code> TextColor(color);</code> |
| 3 | <code> SetConsoleOutputCP(65001);</code> |
| 4 | <code> ifstream Read("assets/UI/VeO1.txt");</code> |
| 5 | <code> string line = "";</code> |
| 6 | <code> for (int i = 0; Read.good(); i++) {</code> |
| 7 | <code> getline(Read, line);</code> |
| 8 | <code> GotoXY(x, y + i);</code> |
| 9 | <code> cout << line;</code> |
| 10 | <code> }</code> |
| 11 | <code> Read.close();</code> |
| 12 | <code> SetConsoleOutputCP(437);</code> |
| 13 | <code>}</code> |
| 14 | <code>void VeX1(int x, int y, int color) {</code> |
| 15 | <code> TextColor(color);</code> |
| 16 | <code> SetConsoleOutputCP(65001);</code> |
| 17 | <code> ifstream Read("assets/UI/VeX1.txt");</code> |
| 18 | <code> string line = "";</code> |
| 19 | <code> for (int i = 0; Read.good(); i++) {</code> |
| 20 | <code> getline(Read, line);</code> |
| 21 | <code> GotoXY(x, y + i);</code> |
| 22 | <code> cout << line;</code> |
| 23 | <code> }</code> |
| 24 | <code> Read.close();</code> |
| 25 | <code> SetConsoleOutputCP(437);</code> |
| 26 | <code>}</code> |
| 27 | <code>void VeO2(int x, int y, int color) {</code> |
| 28 | <code> TextColor(color);</code> |
| 29 | <code> SetConsoleOutputCP(65001);</code> |
| 30 | <code> ifstream Read("assets/UI/VeO2.txt");</code> |
| 31 | <code> string line = "";</code> |
| 32 | <code> for (int i = 0; Read.good(); i++) {</code> |
| 33 | <code> getline(Read, line);</code> |
| 34 | <code> GotoXY(x, y + i);</code> |
| 35 | <code> cout << line;</code> |
| 36 | <code> }</code> |
| 37 | <code> Read.close();</code> |
| 38 | <code> SetConsoleOutputCP(437);</code> |
| 39 | <code>}</code> |
| 40 | <code>void VeX2(int x, int y, int color) {</code> |
| 41 | <code> TextColor(color);</code> |
| 42 | <code> SetConsoleOutputCP(65001);</code> |
| 43 | <code> ifstream Read("assets/UI/VeX2.txt");</code> |
| 44 | <code> string line = "";</code> |
| 45 | <code> for (int i = 0; Read.good(); i++) {</code> |
| 46 | <code> getline(Read, line);</code> |
| 47 | <code> GotoXY(x, y + i);</code> |
| 48 | <code> cout << line;</code> |
| 49 | <code> }</code> |
| 50 | <code> Read.close();</code> |
| 51 | <code> SetConsoleOutputCP(437);</code> |
| 52 | <code>}</code> |

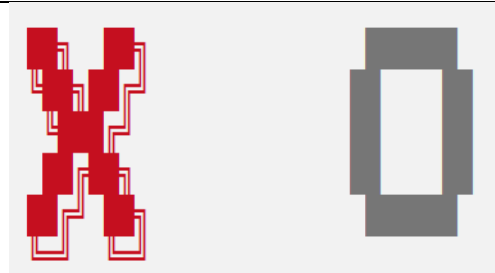
- Hàm TurnX() và TurnO() dùng để gọi các hàm VeO1(), VeX1(), ...

| | |
|---|-------------------------------------|
| 1 | <code>void TurnX() {</code> |
| 2 | <code> VeO1(95, 5, GRAY);</code> |
| 3 | <code> VeX2(74, 5, RED);</code> |

| | |
|---|--------------------|
| 4 | } |
| 5 | void TurnO() { |
| 6 | VeO2(95, 5, BLUE); |
| 7 | VeX1(74, 5, GRAY); |
| 8 | } |



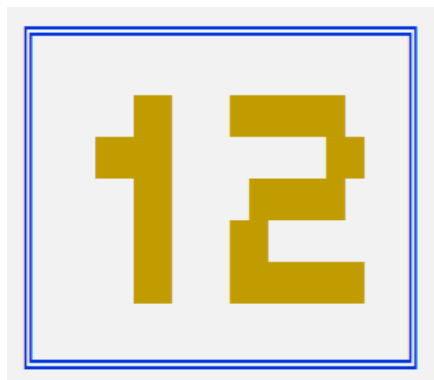
hàm TurnO()



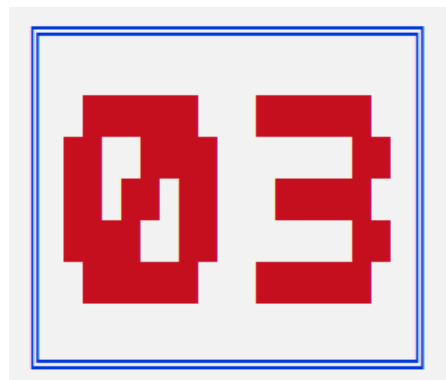
hàm TurnX()

- Hàm DrawTimer: dùng để vẽ đồng hồ đếm ngược thời gian trong game

| | |
|---|--|
| 1 | void DrawTimer(float time, bool _TURN) { |
| 2 | time = float(int(time * 100)) / 100; |
| 3 | int num[2] = { 0 }; |
| 4 | num[0] = (int)ceil(time) / 10; |
| 5 | num[1] = (int)ceil(time) % 10; |
| 6 | for (int i = 0; i < 2; i++) |
| 7 | DrawNumber(80 + i * 10 - (i == 1 && (num[0] == 1 num[1] == 9)), 21, num[i], (num[0] == 0 && num[1] <= 5 ? RED : YELLOW)); |
| 8 | } |



thời gian còn lại cho người chơi hiện tại đánh là 12s



nếu còn dưới 5s thì đồng hồ sẽ hiển thị bằng màu đỏ

- Hàm PauseGame(): dùng để dừng trò chơi, vẽ ra menu phụ cho người chơi lựa chọn các chức năng như save/load/option

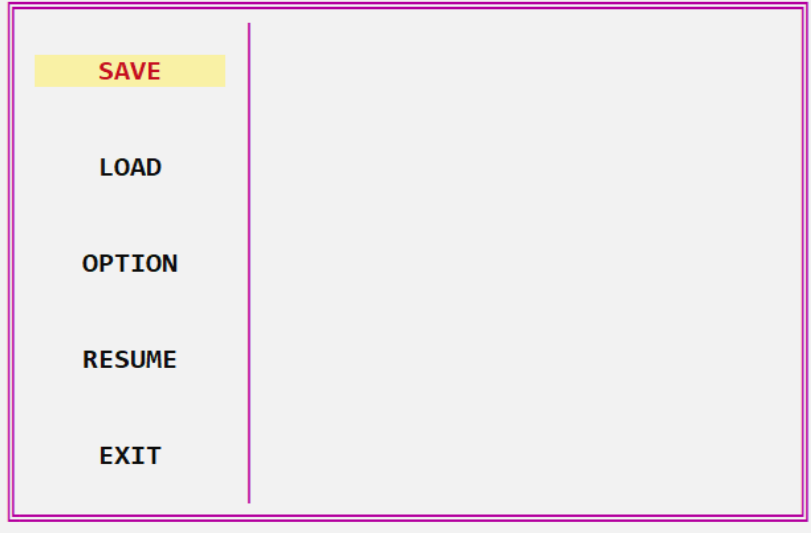
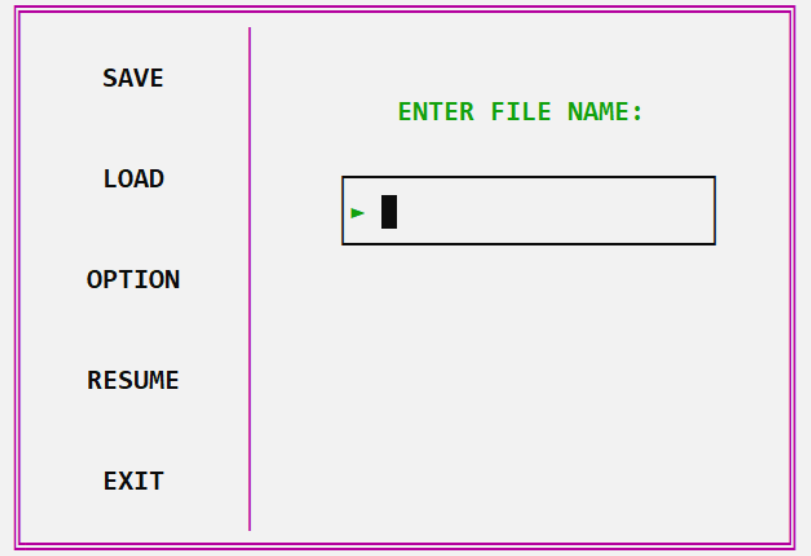
| | |
|---|---|
| 1 | bool PauseGame(_POINT _A[B_SIZE][B_SIZE], bool& _TURN, int& _COMMAND, bool sound[], int& _X, int& _Y, int& cX, int& cY, int& cntX, int& cntO, int& cntWinO, int& cntLoseO, int& cntDraw, int& saveTurn, int& cntRound, string& NamePlayer_O, string& NamePlayer_X, float& remain, float& lastPressed, WinningPos WP[5]) { |
| 2 | remain += lastPressed - clock(); |
| 3 | ClearBox(49, 15, 64, 4); |
| 4 | TextColor(BLACK); |
| 5 | GotoXY(23, 1); |
| 6 | cout << "PAUSED"; |
| 7 | TextColor(LIGHT_MAGENTA); |
| 8 | GotoXY(63, 11); |

| | |
|----|---|
| 9 | cout << BOX_V_LINE; |
| 10 | GotoXY(113, 11); |
| 11 | cout << BOX_V_LINE; |
| 12 | for (int i = 0; i < 15; i++) { |
| 13 | GotoXY(78, 4 + i); |
| 14 | cout << V_LINE; |
| 15 | } |
| 16 | int x = 65, y = 5; |
| 17 | vector <_BUTTON> button; |
| 18 | button.resize(5); |
| 19 | button[0].data = " SAVE "; |
| 20 | button[1].data = " LOAD "; |
| 21 | button[2].data = " OPTION "; |
| 22 | button[3].data = " RESUME "; |
| 23 | button[4].data = " EXIT "; |
| 24 | for (int i = 0; i < button.size(); i++) { |
| 25 | button[i].x = x; |
| 26 | button[i].y = y + 3 * i; |
| 27 | } |
| 28 | TextColor(BLACK); |
| 29 | int n = button.size(); |
| 30 | for (int i = 0; i < n; i++) { |
| 31 | GotoXY(button[i].x, button[i].y); |
| 32 | cout << button[i].data; |
| 33 | } |
| 34 | char c; |
| 35 | int cur = 0, prv = -1; |
| 36 | bool ok = 0; |
| 37 | HoverButton(button[cur]); |
| 38 | while (true) { |
| 39 | c = toupper(_getch()); |
| 40 | if (sound[CLICK_SFX]) PlayAudio(CLICK_SFX); |
| 41 | if (c == ESC) |
| 42 | break; |
| 43 | else if (c == W c == S) { |
| 44 | if (c == W) |
| 45 | prv = cur--; |
| 46 | else |
| 47 | prv = cur++; |
| 48 | if (cur < 0) |
| 49 | cur = n - 1; |
| 50 | if (cur >= n) |
| 51 | cur = 0; |
| 52 | UnhoverButton(button[prv], BLACK); |
| 53 | HoverButton(button[cur]); |
| 54 | } |
| 55 | else if (c == ENTER) { |
| 56 | if (cur == 4) { |
| 57 | ok = 1; |
| 58 | break; |
| 59 | } |
| 60 | if (cur == 3) |
| 61 | break; |
| 62 | if (cur == 1) { |
| 63 | UnhoverButton(button[cur], BLACK); |
| 64 | bool ok = LoadGameInPauseMenu(_A, _TURN, _COMMAND, sound, _X, _Y, cX, cY, cntX, cnt0, cntWin0, cntLose0, cntDraw, saveTurn, cntRound, NamePlayer_0, NamePlayer_X, remain, lastPressed, WP); |
| 65 | if (ok) |
| 66 | return 1; |

| | |
|-----|--|
| 67 | ClearBox(34, 15, 79, 4); |
| 68 | HoverButton(button[cur]); |
| 69 | } |
| 70 | else if (cur == 0) { |
| 71 | UnhoverButton(button[cur], BLACK); |
| 72 | string fileName = ""; |
| 73 | GotoXY(88, 6); |
| 74 | TextColor(GREEN); |
| 75 | cout << "ENTER FILE NAME: "; |
| 76 | DrawBoxMini(25, 3, 84, 8, BLACK); |
| 77 | GotoXY(85, 9); |
| 78 | cout << L_TRIANGLE << ' '; |
| 79 | HideCursor(0); |
| 80 | bool ok = EnterName(fileName, 20); |
| 81 | if (ok) { |
| 82 | fileName += ".txt"; |
| 83 | SaveData(_A, _TURN, _COMMAND, _X, _Y, cX, cY, cntX, cnt0, cntWin0, cntLose0, cntDraw, saveTurn, cntRound, NamePlayer_0, NamePlayer_X, fileName, remain); |
| 84 | GotoXY(87, 13); |
| 85 | cout << "SAVED SUCCESSFULLY!"; |
| 86 | } |
| 87 | HideCursor(1); |
| 88 | while (ok) { |
| 89 | char c = toupper(_getch()); |
| 90 | if (sound[CLICK_SFX]) |
| 91 | PlayAudio(CLICK_SFX); |
| 92 | break; |
| 93 | } |
| 94 | ClearBox(34, 15, 79, 4); |
| 95 | HoverButton(button[cur]); |
| 96 | } |
| 97 | else { |
| 98 | UnhoverButton(button[cur], BLACK); |
| 99 | GotoXY(83, 5); cout << "MUSIC"; |
| 100 | GotoXY(83, 12); cout << "CLICK_SFX"; |
| 101 | TextColor(BLACK); |
| 102 | GotoXY(86, 7); cout << "TURN ON"; |
| 103 | DrawBoard(1, 1, 98, 6, BLACK); |
| 104 | GotoXY(86, 10); cout << "TURN OFF"; |
| 105 | DrawBoard(1, 1, 98, 9, BLACK); |
| 106 | GotoXY(86, 14); cout << "TURN ON"; |
| 107 | DrawBoard(1, 1, 98, 13, BLACK); |
| 108 | GotoXY(86, 17); cout << "TURN OFF"; |
| 109 | DrawBoard(1, 1, 98, 16, BLACK); |
| 110 | int n = 4, i = 0; |
| 111 | _POINT a[4]; |
| 112 | a[0] = { 99, 7, 0 }; |
| 113 | a[1] = { 99, 10, 0 }; |
| 114 | a[2] = { 99, 14, 0 }; |
| 115 | a[3] = { 99, 17, 0 }; |
| 116 | GotoXY(a[i].x, a[i].y); cout << L_TRIANGLE; |
| 117 | GotoXY(a[i].x + 2, a[i].y); cout << R_TRIANGLE; |
| 118 | if (sound[BGM]) |
| 119 | GotoXY(a[0].x + 1, a[0].y), cout << "X"; |
| 120 | else |
| 121 | GotoXY(a[1].x, a[1].y), cout << " X "; |
| 122 | if (sound[CLICK_SFX]) |

| | |
|-----|--|
| 123 | GotoXY(a[2].x, a[2].y), cout << " X "; |
| 124 | else |
| 125 | GotoXY(a[3].x, a[3].y), cout << " X "; |
| 126 | while (true) { |
| 127 | char c = toupper(_getch()); |
| 128 | if (c == ESC) { |
| 129 | if (sound[CLICK_SFX]) |
| 130 | PlayAudio(CLICK_SFX); |
| 131 | break; |
| 132 | int newI = i; |
| 133 | if (c == ENTER) { |
| 134 | if (i == 0) { |
| 135 | GotoXY(a[0].x + 1, a[0].y); |
| 136 | cout << "X"; |
| 137 | GotoXY(a[1].x + 1, a[1].y); |
| 138 | cout << " "; |
| 139 | SetSound(sound, BGM, 1); |
| 140 | } |
| 141 | else if (i == 1) { |
| 142 | GotoXY(a[1].x + 1, a[1].y); |
| 143 | cout << "X"; |
| 144 | GotoXY(a[0].x + 1, a[0].y); |
| 145 | cout << " "; |
| 146 | SetSound(sound, BGM, 0); |
| 147 | } |
| 148 | else if (i == 2) { |
| 149 | GotoXY(a[2].x + 1, a[2].y); |
| 150 | cout << "X"; |
| 151 | GotoXY(a[3].x + 1, a[3].y); |
| 152 | cout << " "; |
| 153 | SetSound(sound, CLICK_SFX, 1); |
| 154 | } |
| 155 | else { |
| 156 | GotoXY(a[3].x + 1, a[3].y); |
| 157 | cout << "X"; |
| 158 | GotoXY(a[2].x + 1, a[2].y); |
| 159 | cout << " "; |
| 160 | SetSound(sound, CLICK_SFX, 0); |
| 161 | } |
| 162 | if (sound[CLICK_SFX]) |
| 163 | PlayAudio(CLICK_SFX); |
| 164 | continue; |
| 165 | else if (c == W) |
| 166 | newI = (--newI + n) % n; |
| 167 | else if (c == S) |
| 168 | newI = ++newI % n; |
| 169 | GotoXY(a[i].x, a[i].y); cout << ' '; |
| 170 | GotoXY(a[i].x + 2, a[i].y); cout << ' '; |
| 171 | i = newI; |
| 172 | GotoXY(a[i].x, a[i].y); cout << |
| 173 | L_TRIANGLE; |
| 174 | GotoXY(a[i].x + 2, a[i].y); cout << |
| 175 | R_TRIANGLE; |
| 176 | if (sound[CLICK_SFX]) |
| 177 | PlayAudio(CLICK_SFX); |
| 178 | } |

| | |
|-----|------------------------|
| 179 | } |
| 180 | } |
| 181 | TextColor(BLACK); |
| 182 | GotoXY(23, 1); |
| 183 | cout << " "; |
| 184 | lastPressed = clock(); |
| 185 | return ok; |
| 186 | } |

| | |
|--|------------------------|
|  | giao diện của menu phụ |
|  | chức năng save |

| | |
|---|------------------|
| <div> <div>SAVE</div> <div>LOAD</div> <div>OPTION</div> <div>RESUME</div> <div>EXIT</div> </div> <div> <div>das.txt</div> <div>ok.txt</div> <div>test.txt</div> <div>0123456789.txt</div> <div>22120182.txt</div> <div>das123.txt</div> <div>< 01 / 03 ></div> </div> | chức năng load |
| <div> <div>SAVE</div> <div>LOAD</div> <div>OPTION</div> <div>RESUME</div> <div>EXIT</div> </div> <div> <div>MUSIC</div> <div>TURN ON <input type="checkbox"/></div> <div>TURN OFF <input checked="" type="checkbox"/></div> <div>CLICK_SFX</div> <div>TURN ON <input checked="" type="checkbox"/></div> <div>TURN OFF <input type="checkbox"/></div> </div> | chức năng option |

- Hàm SetOWin và SetXWin: khi có một người chơi đã thắng cuộc thì sẽ tìm ra các đường đi đó và highlight lên cho người chơi thấy

| | |
|----|---|
| 1 | <code>void SetOWin(string NamePlayer_0, WinningPos WP[5], _POINT</code> |
| 2 | <code>_A[B_SIZE][B_SIZE]) {</code> |
| 3 | <code>for (int i = 0; i < 5; i++) {</code> |
| 4 | <code>GotoXY(_A[WP[i].x][WP[i].y].x,</code> |
| 5 | <code>_A[WP[i].x][WP[i].y].y);</code> |
| 6 | <code>TextColor(BLUE);</code> |
| 7 | <code>Sleep(80);</code> |
| 8 | <code>cout << "O";</code> |
| 9 | <code>GotoXY(_A[WP[i].x][WP[i].y].x,</code> |
| 10 | <code>_A[WP[i].x][WP[i].y].y);</code> |
| 11 | <code>TextColor(RED);</code> |
| 12 | <code>Sleep(80);</code> |
| 13 | <code>cout << "O";</code> |
| 14 | <code>GotoXY(_A[WP[i].x][WP[i].y].x,</code> |
| | <code>_A[WP[i].x][WP[i].y].y);</code> |
| | <code>TextColor(GREEN);</code> |
| | <code>Sleep(80);</code> |
| | <code>cout << "O";</code> |
| | <code>}</code> |

| | |
|----|--|
| 15 | GotoXY(_A[WP[i].x][WP[i].y].x - 1, _A[WP[i].x][WP[i].y].y); |
| 16 | TextColor(0_COLOR & 15 BACKGROUND_YELLOW); |
| 17 | Sleep(80); |
| 18 | cout << " 0 "; |
| 19 | } |
| 20 | GotoXY(48, 28); |
| 21 | cout << "<< PRESS ENTER TO CONTINUE >>"; |
| 22 | int x = 3; |
| 23 | int y = 2; |
| 24 | long long a[] = { RED,BLUE,GREEN,YELLOW,GRAY,BLUE }; |
| 25 | int n = sizeof(a) / sizeof(a[0]); |
| 26 | ClearBox(25, 9, 76, 19); |
| 27 | while (true) { |
| 28 | unsigned char c; |
| 29 | if (_kbhit()) { |
| 30 | unsigned char c = _getch(); |
| 31 | if (c == ENTER) { |
| 32 | SetConsoleBlank(); |
| 33 | Sleep(100); |
| 34 | ascii_art("congratulation", x, y, RED); |
| 35 | ascii_art(NamePlayer_0, x + 15 * 7 / 2 - NamePlayer_0.size() * 7 / 2, y + 6, BLUE); |
| 36 | Sleep(100); |
| 37 | ascii_art("congratulation", x, y, GREEN); |
| 38 | ascii_art(NamePlayer_0, x + 15 * 7 / 2 - NamePlayer_0.size() * 7 / 2, y + 6, BLUE); |
| 39 | Sleep(100); |
| 40 | ascii_art("congratulation", x, y, YELLOW); |
| 41 | ascii_art(NamePlayer_0, x + 15 * 7 / 2 - NamePlayer_0.size() * 7 / 2, y + 6, BLUE); |
| 42 | Sleep(100); |
| 43 | ascii_art("congratulation", x, y, BLUE); |
| 44 | ascii_art(NamePlayer_0, x + 15 * 7 / 2 - NamePlayer_0.size() * 7 / 2, y + 6, BLUE); |
| 45 | Sleep(100); |
| 46 | ascii_art("congratulation", x, y, RED); |
| 47 | ascii_art(NamePlayer_0, x + 15 * 7 / 2 - NamePlayer_0.size() * 7 / 2, y + 6, BLUE); |
| 48 | break; |
| 49 | } |
| 50 | } |
| 51 | for (int i = 0; i < n; i++) { |
| 52 | if (_kbhit()) |
| 53 | c = _getch(); |
| 54 | TextColor((a[i] & 15) BACKGROUND_YELLOW); |
| 55 | Sleep(100); |
| 56 | GotoXY(_A[WP[0].x][WP[0].y].x, _A[WP[0].x][WP[0].y].y); |
| 57 | cout << "0"; |
| 58 | GotoXY(_A[WP[1].x][WP[1].y].x, _A[WP[1].x][WP[1].y].y); |
| 59 | cout << "0"; |
| 60 | GotoXY(_A[WP[2].x][WP[2].y].x, _A[WP[2].x][WP[2].y].y); |
| 61 | cout << "0"; |
| 62 | GotoXY(_A[WP[3].x][WP[3].y].x, _A[WP[3].x][WP[3].y].y); |
| 63 | cout << "0"; |
| 64 | GotoXY(_A[WP[4].x][WP[4].y].x, _A[WP[4].x][WP[4].y].y); |

| | |
|-----|--|
| 65 | cout << "0"; |
| 66 | } |
| 67 | DrawOWin(); |
| 68 | } |
| 69 | } |
| 70 | void SetXWin(string NamePlayer_X, WinningPos WP[5], _POINT _A[B_SIZE][B_SIZE]) { |
| 71 | for (int i = 0; i < 5; i++) { |
| 72 | GotoXY(_A[WP[i].x][WP[i].y].x, _A[WP[i].x][WP[i].y].y); |
| 73 | TextColor(BLUE); |
| 74 | Sleep(80); |
| 75 | cout << "X"; |
| 76 | GotoXY(_A[WP[i].x][WP[i].y].x, _A[WP[i].x][WP[i].y].y); |
| 77 | TextColor(RED); |
| 78 | Sleep(80); |
| 79 | cout << "X"; |
| 80 | GotoXY(_A[WP[i].x][WP[i].y].x, _A[WP[i].x][WP[i].y].y); |
| 81 | TextColor(GREEN); |
| 82 | Sleep(80); |
| 83 | cout << "X"; |
| 84 | GotoXY(_A[WP[i].x][WP[i].y].x - 1, _A[WP[i].x][WP[i].y].y); |
| 85 | TextColor(X_COLOR & 15 BACKGROUND_YELLOW); |
| 86 | Sleep(80); |
| 87 | cout << " X "; |
| 88 | } |
| 89 | GotoXY(48, 28); |
| 90 | cout << "<< PRESS ENTER TO CONTINUE >>"; |
| 91 | int x = 3; |
| 92 | int y = 2; |
| 93 | long long a[] = { RED, BLUE, GREEN, YELLOW, GRAY, RED }; |
| 94 | int n = sizeof(a) / sizeof(a[0]); |
| 95 | ClearBox(25, 9, 76, 19); |
| 96 | while (true) { |
| 97 | unsigned char c; |
| 98 | if (_kbhit()) { |
| 99 | c = _getch(); |
| 100 | if (c == ENTER) { |
| 101 | SetConsoleBlank(); |
| 102 | Sleep(100); |
| 103 | ascii_art("congratulation", x, y, RED); |
| 104 | ascii_art(NamePlayer_X, x + 15 * 7 / 2 - NamePlayer_X.size() * 7 / 2, y + 6, BLUE); |
| 105 | Sleep(100); |
| 106 | ascii_art("congratulation", x, y, GREEN); |
| 107 | ascii_art(NamePlayer_X, x + 15 * 7 / 2 - NamePlayer_X.size() * 7 / 2, y + 6, BLUE); |
| 108 | Sleep(100); |
| 109 | ascii_art("congratulation", x, y, YELLOW); |
| 110 | ascii_art(NamePlayer_X, x + 15 * 7 / 2 - NamePlayer_X.size() * 7 / 2, y + 6, BLUE); |
| 111 | Sleep(100); |
| 112 | ascii_art("congratulation", x, y, BLUE); |
| 113 | ascii_art(NamePlayer_X, x + 15 * 7 / 2 - NamePlayer_X.size() * 7 / 2, y + 6, BLUE); |
| 114 | Sleep(100); |
| 115 | ascii_art("congratulation", x, y, RED); |

| | |
|-----|---|
| 116 | ascii_art(NamePlayer_X, x + 15 * 7 / 2 - NamePlayer_X.size() * 7 / 2, y + 6, BLUE); |
| 117 | break; |
| 118 | } |
| 119 | } |
| 120 | else { |
| 121 | for (int i = 0; i < n; i++) { |
| 122 | if (_kbhit()) |
| 123 | c = _getch(); |
| 124 | TextColor((a[i] & 15) BACKGROUND_YELLOW); |
| 125 | Sleep(100); |
| 126 | GotoXY(_A[WP[0].x][WP[0].y].x, _A[WP[0].x][WP[0].y].y); |
| 127 | cout << "X"; |
| 128 | GotoXY(_A[WP[1].x][WP[1].y].x, _A[WP[1].x][WP[1].y].y); |
| 129 | cout << "X"; |
| 130 | GotoXY(_A[WP[2].x][WP[2].y].x, _A[WP[2].x][WP[2].y].y); |
| 131 | cout << "X"; |
| 132 | GotoXY(_A[WP[3].x][WP[3].y].x, _A[WP[3].x][WP[3].y].y); |
| 133 | cout << "X"; |
| 134 | GotoXY(_A[WP[4].x][WP[4].y].x, _A[WP[4].x][WP[4].y].y); |
| 135 | cout << "X"; |
| 136 | } |
| 137 | DrawXWin(); |
| 138 | } |
| 139 | } |
| 140 | } |

| | | | | | | |
|--|---|---|---|---|---|--|
| | | | | | | |
| | O | X | | | | |
| | | O | X | | | |
| | | | O | X | | |
| | | | | O | X | |
| | | | | | O | |
| | | | | | | |

Hàm SetOWin()

| | | | | | | |
|--|---|---|---|---|---|--|
| | | | | | | |
| | X | X | X | X | X | |
| | O | O | O | O | | |
| | | | | | | |

Hàm SetXWin()

3. Nhóm hàm Model

a. Các hàm xử lý trong game

- Hàm StartGame(): hàm này dùng để bắt đầu trò chơi, phần quan trọng nhất ở hàm này là một vòng while lặp vô tận, người chơi sẽ nhấn các nút để di chuyển và đánh cờ

| | |
|----|--|
| 1 | <code>void StartGame(_POINT _A[B_SIZE][B_SIZE], bool reset, bool& _TURN, int& _COMMAND, bool sound[], int& _X, int& _Y, int& cX, int& cY, int& cntX, int& cntO, int& cntWinO, int& cntLoseO, int& cntDraw, int& saveTurn, int& cntRound, string& NamePlayer_O, string& NamePlayer_X, float& remain, WinningPos WP[5]) {</code> |
| 2 | <code> SetupGame(_A, reset, _TURN, _COMMAND, _X, _Y, cX, cY, cntX, cntO, cntWinO, cntLoseO, cntDraw, cntRound, NamePlayer_O, NamePlayer_X, remain);</code> |
| 3 | <code> bool validEnter = true;</code> |
| 4 | <code> float lastPressed = clock();</code> |
| 5 | <code> while (true) {</code> |
| 6 | <code> if (((lastPressed + remain - clock()) / 1000) < 0) {</code> |
| 7 | <code> ShowTurn(_X, _Y, _TURN, 1);</code> |
| 8 | <code> _TURN = !_TURN;</code> |
| 9 | <code> DrawTimerBox(_TURN);</code> |
| 10 | <code> remain = 15e3;</code> |
| 11 | <code> lastPressed = clock();</code> |
| 12 | <code> }</code> |
| 13 | <code> DrawTimer((lastPressed + remain - clock()) / 1000.0, _TURN);</code> |
| 14 | <code> if (_kbhit()) {</code> |
| 15 | <code> _COMMAND = toupper(_getch());</code> |
| 16 | <code> if (sound[CLICK_SFX]) PlayAudio(CLICK_SFX);</code> |
| 17 | <code> if (_COMMAND == ESC) {</code> |
| 18 | <code> bool ok = PauseGame(_A, _TURN, _COMMAND, sound, _X, _Y, cX, cY, cntX, cntO, cntWinO, cntLoseO, cntDraw, saveTurn, cntRound, NamePlayer_O, NamePlayer_X, remain, lastPressed, WP);</code> |
| 19 | <code> if (ok)</code> |
| 20 | <code> return;</code> |
| 21 | <code> DrawScoreBoard(_TURN, _X, _Y, cntWinO, cntLoseO, cntRound);</code> |
| 22 | <code> continue;</code> |
| 23 | <code> }</code> |
| 24 | <code> // Điều khiển</code> |
| 25 | <code> int tmp = 0;</code> |
| 26 | <code> if (_COMMAND == ENTER) {</code> |
| 27 | <code> switch (CheckBoard(_A, _TURN, _X, _Y)) {</code> |
| 28 | <code> case -1:</code> |
| 29 | <code> GotoXY(_X, _Y);</code> |
| 30 | <code> tmp = GetCurrentColor();</code> |
| 31 | <code> TextColor(X_COLOR);</code> |
| 32 | <code> cout << "X";</code> |
| 33 | <code> TextColor(tmp);</code> |
| 34 | <code> break;</code> |
| 35 | <code> case 1:</code> |
| 36 | <code> GotoXY(_X, _Y);</code> |
| 37 | <code> tmp = GetCurrentColor();</code> |
| 38 | <code> TextColor(O_COLOR);</code> |
| 39 | <code> cout << "O";</code> |
| 40 | <code> TextColor(tmp);</code> |
| 41 | <code> break;</code> |
| 42 | <code> case 0:</code> |
| 43 | <code> validEnter = false;</code> |
| 44 | <code> break;</code> |
| 45 | <code> }</code> |
| 46 | <code> ShowTurn(_X, _Y, _TURN, validEnter);</code> |
| 47 | <code> CntTurn(_TURN, cntX, cntO, validEnter);</code> |
| 48 | <code> if (validEnter == true) {</code> |

| | |
|----|--|
| 49 | <code>switch (ProcessFinish(_A, _X, _Y, _TURN, TestBoard(_A, saveTurn, cntWin0, cntLose0, cntDraw, cntRound, CheckWinLose(_A, saveTurn, cX, cY, WP)), NamePlayer_0, NamePlayer_X, WP)) {</code> |
| 50 | <code>case -1:</code> |
| 51 | <code>case 1:</code> |
| 52 | <code>case 0:</code> |
| 53 | <code>if (AskContinue(_A) != 'Y') {</code> |
| 54 | <code>return;</code> |
| 55 | <code>}</code> |
| 56 | <code>else {</code> |
| 57 | <code>SetupGame(_A, 1, _TURN, _COMMAND, _X, _Y, cX, cY, cntX, cnt0, cntWin0, cntLose0, cntDraw, cntRound, NamePlayer_0, NamePlayer_X, remain);</code> |
| 58 | <code>}</code> |
| 59 | <code>}</code> |
| 60 | <code>DrawTimerBox(_TURN);</code> |
| 61 | <code>remain = 15e3;</code> |
| 62 | <code>lastPressed = clock();</code> |
| 63 | <code>}</code> |
| 64 | <code>validEnter = true;</code> |
| 65 | <code>}</code> |
| 66 | <code>else if (_COMMAND == 'A') MoveLeft(_A, _X, _Y, cX, cY);</code> |
| 67 | <code>else if (_COMMAND == 'W') MoveUp(_A, _X, _Y, cX, cY);</code> |
| 68 | <code>else if (_COMMAND == 'S') MoveDown(_A, _X, _Y, cX, cY);</code> |
| 69 | <code>else if (_COMMAND == 'D') MoveRight(_A, _X, _Y, cX, cY);</code> |
| 70 | <code>}</code> |
| 71 | <code>}</code> |
| 72 | <code>}</code> |
| 73 | <code>void StartGame(_POINT _A[B_SIZE][B_SIZE], bool reset, bool& _TURN, int& _COMMAND, bool sound[], int& _X, int& _Y, int& cX, int& cY, int& cntX, int& cnt0, int& cntWin0, int& cntLose0, int& cntDraw, int& saveTurn, int& cntRound, string& NamePlayer_0, string& NamePlayer_X, float& remain, WinningPos WP[5]) {</code> |

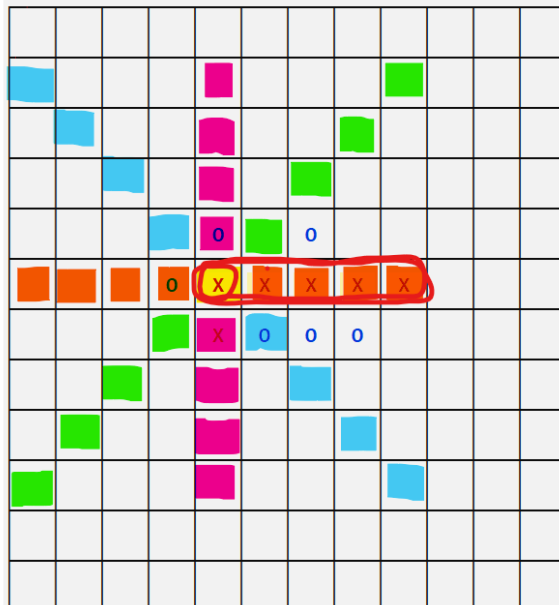
- Hàm CheckFullBoard(): Hàm này sẽ duyệt qua tất cả các ô trong bàn cờ và kiểm tra xem vị trí tại đó đã có người đánh hay chưa. Nếu đã có người đánh thì sẽ tăng giá trị của biến count lên. Nếu giá trị của biến count lớn hơn hoặc bằng 90% tổng số ô trong bảng thì nó sẽ trả về true (tức là sẽ hòa) và nếu không thì nó sẽ trả về false (người chơi tiếp tục đánh tiếp)

| | |
|----|---|
| 1 | <code>bool CheckFullBoard(_POINT _A[B_SIZE][B_SIZE]) {</code> |
| 2 | <code>int cnt=0;</code> |
| 3 | <code>for (int i = 0; i < B_SIZE; i++) {</code> |
| 4 | <code>for (int j = 0; j < B_SIZE; j++) {</code> |
| 5 | <code>if (_A[i][j].c != 0)</code> |
| 6 | <code>cnt++;</code> |
| 7 | <code>}</code> |
| 8 | <code>}</code> |
| 9 | <code>if(cnt>=B_SIZE*B_SIZE*0.9)</code> |
| 10 | <code>return true;</code> |
| 11 | <code>return false;</code> |
| 12 | <code>}</code> |

- Hàm CheckWin(): Hàm này kiểm tra xem trên bàn cờ có người chơi nào thắng hay chưa. Bằng cách duyệt tại tọa độ đang xét (vị trí màu vàng). Hàm này sẽ có 4 trường hợp được duyệt và nó sẽ duyệt theo bán kính 5 ô tính từ tọa độ hiện tại (vị trí lần lượt theo đường dọc, đường ngang, đường chéo phụ và cuối cùng là đường chéo chính. Và ta phải duyệt 360 độ tức là mỗi trường hợp ta phải duyệt tất cả là 9 ô. Nếu mỗi trường hợp đếm được 5 lần vị trí X đánh (hoặc O đánh) thì sẽ đánh dấu lại các vị trí đó và sẽ trả về true nếu X (hoặc O) thắng nếu mọi trường hợp đều không thỏa mãn thì sẽ trả về false và người chơi sẽ tiếp tục đánh

| | |
|----|--|
| 1 | bool CheckWin(_POINT _A[B_SIZE][B_SIZE], int x, int y, WinningPos WP[5]) { |
| 2 | int count = 0; |
| 3 | int i, j; |
| 4 | int n = B_SIZE; |
| 5 | if (_A[y][x].c == 0) |
| 6 | return false; |
| 7 | // Kiểm tra hàng ngang |
| 8 | for (i = x - 4; i <= x; i++) { |
| 9 | int k = 0; |
| 10 | if (i < 0 i + 4 >= n) continue; |
| 11 | count = 0; |
| 12 | for (j = i; j <= i + 4; j++) { |
| 13 | if (_A[y][j].c == _A[y][x].c) count++; |
| 14 | } |
| 15 | if (count == 5) { |
| 16 | for (j = i; j <= i + 4; j++) { |
| 17 | WP[k].x = y; |
| 18 | WP[k].y = j; |
| 19 | k++; |
| 20 | } |
| 21 | return true; |
| 22 | } |
| 23 | } |
| 24 | |
| 25 | // Kiểm tra hàng dọc |
| 26 | for (i = y - 4; i <= y; i++) { |
| 27 | int k = 0; |
| 28 | if (i < 0 i + 4 >= n) continue; |
| 29 | count = 0; |
| 30 | for (j = i; j <= i + 4; j++) { |
| 31 | if (_A[j][x].c == _A[y][x].c) count++; |
| 32 | } |
| 33 | if (count == 5) |
| 34 | { |
| 35 | for (j = i; j <= i + 4; j++) { |
| 36 | WP[k].x = j; |
| 37 | WP[k].y = x; |
| 38 | k++; |
| 39 | } |
| 40 | return true; |
| 41 | } |
| 42 | } |
| 43 | |
| 44 | // kiểm tra đường chéo chính |
| 45 | for (i = x - 4, j = y - 4; i <= x && j <= y; i++, j++) { |
| 46 | int l = 0; |
| 47 | if (i < 0 i + 4 >= n j < 0 j + 4 >= n) |
| | continue; |

| | |
|----|--|
| 48 | count = 0; |
| 49 | for (int k = 0; k < 5; k++) { |
| 50 | if (_A[j + k][i + k].c == _A[y][x].c) |
| | count++; |
| 51 | } |
| 52 | if (count == 5) |
| 53 | { |
| 54 | for (int k = 0; k < 5; k++) { |
| 55 | WP[l].x = j + k; |
| 56 | WP[l].y = i + k; |
| 57 | l++; |
| 58 | } |
| 59 | return true; |
| 60 | } |
| 61 | } |
| 62 | |
| 63 | // Kiểm tra đường chéo phụ |
| 64 | for (i = x - 4, j = y + 4; i <= x && j >= y; i++, j--) { |
| 65 | int l = 0; |
| 66 | if (i < 0 i + 4 >= n j >= n j - 4 < 0) |
| | continue; |
| 67 | count = 0; |
| 68 | for (int k = 0; k < 5; k++) { |
| 69 | if (_A[j - k][i + k].c == _A[y][x].c) |
| | count++; |
| 70 | } |
| 71 | if (count == 5) |
| 72 | { |
| 73 | for (int k = 0; k < 5; k++) { |
| 74 | WP[l].x = j - k; |
| 75 | WP[l].y = i + k; |
| 76 | l++; |
| 77 | } |
| 78 | return true; |
| 79 | } |
| 80 | } |
| 81 | |
| 82 | return false; |
| 83 | } |



- Hàm CheckWinLose(): Hàm này sẽ kiểm tra thắng thua sau đó lưu lượt chơi chơi người chơi thắng vào biến saveTurn. Để sau đó sẽ có hàm gọi lại biến saveTurn và thông báo X (hoặc O) thắng. Và hàm này sẽ trả về true nếu có X (hoặc O) thắng, sẽ trả về false nếu không có ai thắng cả và sẽ tiếp tục trò chơi.

| | |
|---|--|
| 1 | <code>bool CheckWinLose(_POINT _A[B_SIZE][B_SIZE], int& saveTurn, int cX, int cY, WinningPos WP[5])</code> |
| 2 | <code>{</code> |
| 3 | <code>if (CheckWin(_A, cX, cY, WP) == true)</code> |
| 4 | <code>{</code> |
| 5 | <code>saveTurn = _A[cY][cX].c;</code> |
| 6 | <code>return true;</code> |
| 7 | <code>}</code> |
| 8 | <code>return false;</code> |
| 9 | <code>}</code> |

- Hàm TestBoard(): Hàm này kiểm tra xem board có người nào thắng hay chưa và trả về giá trị tương ứng kiểu (int) để gọi hàm processFinish() và hiển thị ra thông tin người đó thắng. Nếu chưa có ai thắng thì sẽ trả về 2 (tức là sẽ tiếp tục đánh)

| | |
|----|--|
| 1 | <code>int TestBoard(_POINT _A[B_SIZE][B_SIZE], int& saveTurn, int& cntWin0, int& cntLose0, int& cntDraw, int& cntRound, bool check) {</code> |
| 2 | <code>if (check == true && saveTurn == 1)</code> |
| 3 | <code>{</code> |
| 4 | <code>if (++cntRound > 9)</code> |
| 5 | <code>cntRound = 1;</code> |
| 6 | <code>cntWin0 = (cntWin0 + 1) % 10;</code> |
| 7 | <code>return 1; // trả về 1 nếu O thắng</code> |
| 8 | <code>}</code> |
| 9 | <code>else if (check == true && saveTurn == -1)</code> |
| 10 | <code>{</code> |
| 11 | <code>if (++cntRound > 9)</code> |
| 12 | <code>cntRound = 1;</code> |
| 13 | <code>cntLose0 = (cntLose0 + 1) % 10;</code> |
| 14 | <code>return -1; // trả về -1 nếu X thắng</code> |
| 15 | <code>}</code> |
| 16 | <code>else if (check == false && CheckFullBoard(_A) == true)</code> |
| 17 | <code>{</code> |
| 18 | <code>if (++cntRound > 9)</code> |
| 19 | <code>cntRound = 1;</code> |
| 20 | <code>cntDraw = (cntDraw + 1) % 10;</code> |
| 21 | <code>return 0; // trả về không nếu không có người nào thắng và đã đánh full board</code> |
| 22 | <code>}</code> |
| 23 | <code>return 2;</code> |
| 24 | <code>}</code> |

b. Các hàm khác

- Hàm ResetData(): khởi tạo lại giá trị của các biến cần thiết để bắt đầu một màn chơi mới

| | |
|---|---|
| 1 | <code>void ResetData(_POINT _A[B_SIZE][B_SIZE], bool& _TURN, int& _COMMAND, int& _X, int& _Y, int& cX, int& cY, int& cntX, int& cntO, float& remain)</code> |
|---|---|

| | |
|----|------------------------------------|
| 2 | { |
| 3 | for (int i = 0; i < B_SIZE; i++) { |
| 4 | for (int j = 0; j < B_SIZE; j++) { |
| 5 | _A[i][j].y = i * 2 + 1 + BOARD_Y; |
| 6 | _A[i][j].x = j * 4 + 2 + BOARD_X; |
| 7 | _A[i][j].c = 0; |
| 8 | } |
| 9 | } |
| 10 | _COMMAND = -1; |
| 11 | _X = _A[0][0].x; |
| 12 | _Y = _A[0][0].y; |
| 13 | cX = cY = 0; |
| 14 | cntX = cntO = 0; |
| 15 | remain = 15e3; |
| 16 | } |

- Hàm LoadData(): lấy dữ liệu game từ một file người chơi đã chọn

| | |
|----|--|
| 1 | bool LoadData(_POINT _A[B_SIZE][B_SIZE], bool& _TURN, int& _COMMAND, int& _X, int& _Y, int& cX, int& cY, int& cntX, int& cntO, int& cntWinO, int& cntLoseO, int& cntDraw, int& saveTurn, int& cntRound, string& NamePlayer_O, string& NamePlayer_X, string& FileName, float& remain) { |
| 2 | fstream inp; |
| 3 | inp.open("save/data/" + FileName, ios::in); |
| 4 | if (inp.fail()) { |
| 5 | inp.close(); |
| 6 | return 0; |
| 7 | } |
| 8 | for (int i = 0; i < B_SIZE; i++) |
| 9 | for (int j = 0; j < B_SIZE; j++) { |
| 10 | inp >> _A[i][j].c; |
| 11 | _A[i][j].y = i * 2 + 1 + BOARD_Y; |
| 12 | _A[i][j].x = j * 4 + 2 + BOARD_X; |
| 13 | } |
| 14 | inp >> _TURN >> _COMMAND >> _X >> _Y >> cX >> cY; |
| 15 | inp >> cntX >> cntO >> cntWinO >> cntLoseO >> cntDraw >> saveTurn >> cntRound >> remain; |
| 16 | inp.ignore(); |
| 17 | getline(inp, NamePlayer_O); |
| 18 | getline(inp, NamePlayer_X); |
| 19 | inp.close(); |
| 20 | return 1; |
| 21 | } |

- Hàm SaveData(): lưu dữ liệu game vào một file người chơi nhập vào, các file người chơi lưu sẽ được lưu tên lại trong đường dẫn "save/all_save.txt" để có thể truy xuất tên file một cách nhanh chóng trong các hàm Load game

| | |
|---|--|
| 1 | bool SaveData(_POINT _A[B_SIZE][B_SIZE], bool& _TURN, int& _COMMAND, int& _X, int& _Y, int& cX, int& cY, int& cntX, int& cntO, int& cntWinO, int& cntLoseO, int& cntDraw, int& saveTurn, int& cntRound, string& NamePlayer_O, string& NamePlayer_X, string& FileName, float& remain) { |
|---|--|

| | |
|----|--|
| 2 | <code>fstream out;</code> |
| 3 | <code>out.open("save/data/" + FileName, ios::out);</code> |
| 4 | <code>if (out.fail()) {</code> |
| 5 | <code> out.close();</code> |
| 6 | <code> return 0;</code> |
| 7 | <code>}</code> |
| 8 | <code>for (int i = 0; i < B_SIZE; i++) {</code> |
| 9 | <code> for (int j = 0; j < B_SIZE; j++)</code> |
| 10 | <code> out << _A[i][j].c << ' ';</code> |
| 11 | <code> out << endl;</code> |
| 12 | <code> }</code> |
| 13 | <code> out << _TURN << ' ' << _COMMAND << ' ' << _X << ' ' << _Y << ' ' << cX << ' ' << cY << endl;</code> |
| 14 | <code> out << cntX << ' ' << cnt0 << ' ' << cntWin0 << ' ' << cntLose0 << ' ' << cntDraw << ' ' << saveTurn << ' ' << cntRound << ' ' << remain << endl;</code> |
| 15 | <code> out << NamePlayer_0 << endl << NamePlayer_X;</code> |
| 16 | <code> out.close();</code> |
| 17 | <code> out.open("save/all_save.txt", ios::app);</code> |
| 18 | <code> string s = "";</code> |
| 19 | <code> bool ok = 0;</code> |
| 20 | <code> fstream inp;</code> |
| 21 | <code> inp.open("save/all_save.txt", ios::in);</code> |
| 22 | <code> while (inp >> s)</code> |
| 23 | <code> ok = (s == FileName);</code> |
| 24 | <code> if (!ok)</code> |
| 25 | <code> out << endl << FileName;</code> |
| 26 | <code> out.close();</code> |
| 27 | <code> return 1;</code> |
| 28 | <code>}</code> |

- Hàm LoadSound(): lấy thông tin về các âm thanh trong game và lưu vào mảng sound

| | |
|----|---|
| 1 | <code>void LoadSound(bool sound[]) {</code> |
| 2 | <code> fstream inp;</code> |
| 3 | <code> inp.open(SOUND_PATH, ios::in);</code> |
| 4 | <code> if (inp.fail()) {</code> |
| 5 | <code> cout << "Can't open file";</code> |
| 6 | <code> return;</code> |
| 7 | <code> }</code> |
| 8 | <code> int n = 2;</code> |
| 9 | <code> for (int i = 0; i < n; i++)</code> |
| 10 | <code> inp >> sound[i];</code> |
| 11 | <code> if (sound[BGM])</code> |
| 12 | <code> PlayAudio(BGM);</code> |
| 13 | <code> else</code> |
| 14 | <code> StopSound(BGM);</code> |
| 15 | <code> inp.close();</code> |
| 16 | <code>}</code> |

- Hàm SetSound() điều chỉnh giá trị của mảng sound để bật/tắt nhạc

| | |
|---|--|
| 1 | <code>void SetSound(bool sound[], int type, bool value) {</code> |
| 2 | <code> fstream out;</code> |
| 3 | <code> out.open(SOUND_PATH, ios::out);</code> |

| | |
|----|--|
| 4 | <code>if (out.fail()) {</code> |
| 5 | <code> cout << "Can't open file";</code> |
| 6 | <code> return;</code> |
| 7 | <code>}</code> |
| 8 | <code>sound[type] = value;</code> |
| 9 | <code>int n = 2;</code> |
| 10 | <code>for (int i = 0; i < n; i++)</code> |
| 11 | <code> out << sound[i] << ' ';</code> |
| 12 | <code>LoadSound(sound);</code> |
| 13 | <code>out.close();</code> |
| 14 | <code>}</code> |

- Hàm PlayAudio(): dùng để chạy file âm thanh dựa trên tham số đầu vào tương ứng

| | |
|---|--|
| 1 | <code>void PlayAudio(int type) {</code> |
| 2 | <code> if (type == BGM) {</code> |
| 3 | <code> DWORD error = mciSendString(L"play</code> <code>\"assets/sounds/bgm.wav\" ", NULL, 0, NULL);</code> |
| 4 | <code> }</code> |
| 5 | <code> else if (type == CLICK_SFX)</code> |
| 6 | <code> PlaySound(TEXT("assets/sounds/click_sfx.wav"), NULL,</code> <code>SND_ASYNC);</code> |
| 7 | <code>}</code> |

- Hàm StopSound(): dừng nhạc nền

| | |
|---|---|
| 1 | <code>void StopSound(int type) {</code> |
| 2 | <code> if (type == BGM)</code> |
| 3 | <code> mciSendString(L"stop assets/sounds/bgm.wav",</code> <code>NULL, 0, NULL);</code> |
| 4 | <code>}</code> |

4. Nhóm hàm Control

Các hàm này dùng để điều khiển vị trí và các thứ liên quan đến con trỏ trên màn hình console

- Hàm GotoXY() dùng để đưa con trỏ tới vị trí mà người dùng truyền vào

| | |
|---|--|
| 1 | <code>void GotoXY(int column, int line)</code> |
| 2 | <code>{</code> |
| 3 | <code> COORD coord;</code> |
| 4 | <code> coord.X = column;</code> |
| 5 | <code> coord.Y = line;</code> |
| 6 | <code> SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),</code> <code>coord);</code> |
| 7 | <code>}</code> |

- Hàm whereX() và whereY() trả về vị trí của con trỏ hiện tại trên màn hình console

| | |
|---|--|
| 1 | <code>int whereX()</code> |
| 2 | <code>{</code> |
| 3 | <code> CONSOLE_SCREEN_BUFFER_INFO csbi;</code> |
| 4 | <code> if</code> <code>(GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE),</code> <code>&csbi))</code> |
| 5 | <code> return csbi.dwCursorPosition.X;</code> |
| 6 | <code> return -1;</code> |
| 7 | <code>}</code> |
| 8 | <code>int whereY()</code> |
| 9 | <code>{</code> |

| | |
|----|---|
| 10 | CONSOLE_SCREEN_BUFFER_INFO csbi; |
| 11 | if (GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE), &csbi)) |
| 12 | return csbi.dwCursorPosition.Y; |
| 13 | return -1; |
| 14 | } |

- Các hàm MoveUp, MoveDown, MoveLeft, MoveRight dùng để di chuyển con trỏ trên bàn cờ khi người dùng nhấn vào các phím W/S/A/D tương ứng

| | |
|----|--|
| 1 | void MoveRight(_POINT _A[B_SIZE][B_SIZE], int& _X, int& _Y, int& cX, int& cY) |
| 2 | { |
| 3 | if (_X < _A[B_SIZE - 1][B_SIZE - 1].x) |
| 4 | { |
| 5 | UnHoverCell(_A, cY, cX); |
| 6 | _X += 4; |
| 7 | cX++; |
| 8 | HoverCell(_A, cY, cX); |
| 9 | } |
| 10 | } |
| 11 | void MoveLeft(_POINT _A[B_SIZE][B_SIZE], int& _X, int& _Y, int& cX, int& cY) { |
| 12 | if (_X > _A[0][0].x) |
| 13 | { |
| 14 | UnHoverCell(_A, cY, cX); |
| 15 | _X -= 4; |
| 16 | cX--; |
| 17 | HoverCell(_A, cY, cX); |
| 18 | } |
| 19 | } |
| 20 | void MoveUp(_POINT _A[B_SIZE][B_SIZE], int& _X, int& _Y, int& cX, int& cY) { |
| 21 | if (_Y > _A[0][0].y) |
| 22 | { |
| 23 | UnHoverCell(_A, cY, cX); |
| 24 | _Y -= 2; |
| 25 | cY--; |
| 26 | HoverCell(_A, cY, cX); |
| 27 | } |
| 28 | } |
| 29 | void MoveDown(_POINT _A[B_SIZE][B_SIZE], int& _X, int& _Y, int& cX, int& cY) { |
| 30 | if (_Y < _A[B_SIZE - 1][B_SIZE - 1].y) |
| 31 | { |
| 32 | UnHoverCell(_A, cY, cX); |
| 33 | _Y += 2; |
| 34 | cY++; |
| 35 | HoverCell(_A, cY, cX); |
| 36 | } |
| 37 | } |

III. Lời kết

- Trong quá trình làm việc, cả nhóm đã cùng góp ý và giúp đỡ nhau từ việc tìm kiếm các hàm, cách cải thiện các đoạn code để có hiệu năng tốt hơn cũng như các cải tiến về giao diện trong game.
- Tuy còn vài chỗ code khá vụng về, cũng như còn dài, lặp lại và khá tối nghĩa, nhưng các thành viên trong nhóm đã cố hết sức để giao tiếp với nhau và làm ra được sản phẩm cuối cùng.

IV. Tài liệu tham khảo

- Giáo trình Nhập môn lập trình và Kỹ thuật lập trình của thầy Trần Đan Thu
- DoAnCaro.pdf của thầy Trương Toàn Thịnh
- Text to ASCII Art Generator (TAAG): <http://patorjk.com/software/taag/>