#include <SDL.h>

#include <SDL\_image.h>

#include <SDL\_mixer.h>

#include <stdio.h>

#include <string>

#include <cstdlib>

#include <iostream>

#include <ctime>

#include <vector>

#include <SDL\_ttf.h>

#include <sstream>

#include <fstream>

#include <map>

#include <cmath>

using namespace std;

const int SCREEN\_WIDTH = 720;

const int SCREEN\_HEIGHT = 720;

const char\* WINDOW\_TITLE = "2048 GAME";

const int TILE\_SIZE = 100;

const int TILE\_MARGIN = 10;

const int ROWS = 4;

const int COLUMNS = 4;

SDL\_Window\* window;

SDL\_Renderer\* renderer;

SDL\_Surface\* surface;

TTF\_Font\* font;

SDL\_Color FONT\_COLOR = { 0, 0, 0, 255 };

map <int, SDL\_Texture\*> textures;

int board[ROWS][COLUMNS];

void logErrorAndExit(const char\* msg, const char\* error)

{

SDL\_LogMessage(SDL\_LOG\_CATEGORY\_APPLICATION, SDL\_LOG\_PRIORITY\_ERROR, "%s: %s", msg, error);

SDL\_Quit();

}

bool check\_board\_full()

{

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

if (!board[i][j]) return 0;

}

}

return 1;

}

bool check\_vaild()

{

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

if (board[i][j] == 0) return 1;

else if (i < 4 - 1 && board[i][j] == board[i + 1][j]) return 1;

else if (j < 4 - 1 && board[i][j] == board[i][j + 1]) return 1;

}

}

return 0;

}

bool GameOver() {

return (check\_board\_full() && !check\_vaild());

}

bool Buttons(SDL\_Renderer\* renderer, TTF\_Font\* font, SDL\_Event& event, bool& resetGame) {

SDL\_Rect ChoiLai = { SCREEN\_WIDTH / 2 - 100, SCREEN\_HEIGHT / 2 + 160, 200, 50 };

SDL\_Rect KetThuc = { SCREEN\_WIDTH / 2 - 100, SCREEN\_HEIGHT / 2 + 220, 200, 50 };

SDL\_SetRenderDrawColor(renderer, 100, 100, 255, 255);

SDL\_RenderFillRect(renderer, &ChoiLai);

SDL\_SetRenderDrawColor(renderer, 0, 0, 0, 255);

SDL\_RenderDrawRect(renderer, &ChoiLai);

SDL\_SetRenderDrawColor(renderer, 255, 100, 100, 255);

SDL\_RenderFillRect(renderer, &KetThuc);

SDL\_SetRenderDrawColor(renderer, 0, 0, 0, 255);

SDL\_RenderDrawRect(renderer, &KetThuc);

SDL\_Color white = { 255, 255, 255 };

SDL\_Surface\* replaySurf = TTF\_RenderText\_Solid(font, "Try Again.", white);

SDL\_Texture\* replayText = SDL\_CreateTextureFromSurface(renderer, replaySurf);

SDL\_Rect replayTextRect = {

ChoiLai.x + (ChoiLai.w - replaySurf->w) / 2,

ChoiLai.y + (ChoiLai.h - replaySurf->h) / 2,

replaySurf->w,

replaySurf->h

};

SDL\_RenderCopy(renderer, replayText, NULL, &replayTextRect);

SDL\_Surface\* quitSurf = TTF\_RenderText\_Solid(font, "Quit Game.", { 255, 255, 255 });

SDL\_Texture\* quitText = SDL\_CreateTextureFromSurface(renderer, quitSurf);

SDL\_Rect quitTextRect = {

KetThuc.x + (KetThuc.w - quitSurf->w) / 2,

KetThuc.y + (KetThuc.h - quitSurf->h) / 2,

quitSurf->w,

quitSurf->h

};

SDL\_RenderCopy(renderer, quitText, NULL, &quitTextRect);

SDL\_FreeSurface(replaySurf); SDL\_DestroyTexture(replayText);

SDL\_FreeSurface(quitSurf); SDL\_DestroyTexture(quitText);

if (event.type == SDL\_MOUSEBUTTONDOWN) {

int x = event.button.x;

int y = event.button.y;

// Click vào nút "Chơi lại"

if (x >= ChoiLai.x && x <= ChoiLai.x + ChoiLai.w && y >= ChoiLai.y && y <= ChoiLai.y + ChoiLai.h) {

resetGame = true;

return false; // không thoát game

}

// Click vào nút "Kết thúc"

if (x >= KetThuc.x && x <= KetThuc.x + KetThuc.w && y >= KetThuc.y && y <= KetThuc.y + KetThuc.h) {

return true;

}

}

return false;

}

int CinHighScore(const string& filename) {

ifstream file(filename);

int highscore = 0;

if (file.is\_open()) {

file >> highscore;

file.close();

}

return highscore;

}

void CoutHighScore(const string& filename, int highscore) {

ofstream file(filename);

if (file.is\_open()) {

file << highscore;

file.close();

}

}

void LoadTextures(int value)

{

char\* basePath = SDL\_GetBasePath();

string name = string(basePath) + to\_string(value) + ".png";

SDL\_free(basePath);

cout << "Dang tai anh tu: " << name << endl;

SDL\_Surface\* surface = IMG\_Load(name.c\_str());

if (surface) {

textures[value] = SDL\_CreateTextureFromSurface(renderer, surface);

SDL\_FreeSurface(surface);

}

else {

cout << "Khong the load anh. Loi :" << IMG\_GetError() << endl;

}

}

void DrawScore(int score, int highscore, TTF\_Font\* font)

{

string ScoreString = "SCORE: " + to\_string(score);

SDL\_Surface\* scoreSurface = TTF\_RenderText\_Solid(font, ScoreString.c\_str(), FONT\_COLOR);

if (!scoreSurface) {

SDL\_Log("Failed to render score text surface: %s", TTF\_GetError());

return;

}

// tạo score nói riêng

SDL\_Texture\* scoreTexture = SDL\_CreateTextureFromSurface(renderer, scoreSurface);

int scoreWidth = scoreSurface->w;

int scoreHeight = scoreSurface->h;

SDL\_FreeSurface(scoreSurface);

SDL\_Rect ScoreRect = { 10, 10, scoreWidth, scoreHeight };

SDL\_RenderCopy(renderer, scoreTexture, NULL, &ScoreRect);

SDL\_DestroyTexture(scoreTexture);

// tạo highscore

string HighScoreString = "HIGH SCORE: " + to\_string(highscore);

SDL\_Surface\* HighScoreSurface = TTF\_RenderText\_Solid(font, HighScoreString.c\_str(), FONT\_COLOR);

if (!HighScoreSurface) {

SDL\_Log("Failed to render high score text surface: %s", TTF\_GetError());

return;

}

SDL\_Texture\* highTexture = SDL\_CreateTextureFromSurface(renderer, HighScoreSurface);

int highWidth = HighScoreSurface->w;

int highHeight = HighScoreSurface->h;

SDL\_FreeSurface(HighScoreSurface);

SDL\_Rect HighScoreRect = { SCREEN\_WIDTH - highWidth - 10 , 10 , highWidth, highHeight };

SDL\_RenderCopy(renderer, highTexture, NULL, &HighScoreRect);

SDL\_DestroyTexture(highTexture);

}

void DrawGameOverMessage(int score)

{

std::string msg = "Diem cua ban la: " + to\_string(score) + ". Nhan ENTER de choi lai.Nhan Q de ket thuc";

SDL\_Surface\* surface = TTF\_RenderText\_Solid(font, msg.c\_str(), FONT\_COLOR);

if (!surface) {

SDL\_Log("Failed to render game over message: %s", TTF\_GetError());

return;

}

SDL\_Texture\* texture = SDL\_CreateTextureFromSurface(renderer, surface);

if (!texture) {

SDL\_Log("Failed to create texture from surface: %s", SDL\_GetError());

SDL\_FreeSurface(surface);

return;

}

// Hiển thị ở góc dưới bên trái

int textWidth = surface->w;

int textHeight = surface->h;

SDL\_FreeSurface(surface);

int x = 10; // Lề trái

int y = SCREEN\_HEIGHT - textHeight - 10; // Lề dưới

SDL\_Rect dstRect = { x, y, textWidth, textHeight };

SDL\_RenderCopy(renderer, texture, NULL, &dstRect);

SDL\_DestroyTexture(texture);

}

void DrawBackground(SDL\_Renderer\* renderer){

SDL\_SetRenderDrawColor(renderer, 237, 224, 200, 255); // màu nền (background)

SDL\_RenderClear(renderer); // tô màu toàn bộ window

}

void render(int score, int highscore, TTF\_Font\* font)

{

DrawBackground(renderer);

SDL\_RenderClear(renderer);

SDL\_Rect tile;

int board\_width = COLUMNS \* TILE\_SIZE + (COLUMNS + 1) \* TILE\_MARGIN;

int board\_height = ROWS \* TILE\_SIZE + (ROWS + 1) \* TILE\_MARGIN;

int offsetX = (SCREEN\_WIDTH - board\_width) / 2;

int offsetY = (SCREEN\_HEIGHT - board\_height) / 2 - 80 ;

for (int i = 0; i < ROWS; i++) {

for (int j = 0; j < COLUMNS; j++) {

tile.x = offsetX + TILE\_MARGIN + j \* (TILE\_SIZE + TILE\_MARGIN);

tile.y = offsetY + TILE\_MARGIN + i \* (TILE\_SIZE + TILE\_MARGIN);

tile.w = TILE\_SIZE;

tile.h = TILE\_SIZE;

int val = board[i][j];

if (textures.count(val)) {

SDL\_RenderCopy(renderer, textures[val], NULL, &tile);

}

}

}

DrawScore(score, highscore, font);

if (GameOver()) {

DrawGameOverMessage(score);

bool dummy = false;

SDL\_Event dummyEvent;

Buttons(renderer, font, dummyEvent, dummy);

}

SDL\_RenderPresent(renderer);

SDL\_Delay(50);

}

void init()

{

srand(static\_cast<unsigned int>(time(NULL)));

SDL\_Init(SDL\_INIT\_EVERYTHING);

IMG\_Init(IMG\_INIT\_PNG);

TTF\_Init();

font = TTF\_OpenFont("font/NunitoSans-Italic.ttf", 20);

window = SDL\_CreateWindow(WINDOW\_TITLE, SDL\_WINDOWPOS\_CENTERED, SDL\_WINDOWPOS\_CENTERED, SCREEN\_WIDTH, SCREEN\_HEIGHT, SDL\_WINDOW\_SHOWN);

renderer = SDL\_CreateRenderer(window, -1, SDL\_RENDERER\_ACCELERATED);

int values[] = { 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048 };

int size = sizeof(values) / sizeof(values[0]);

for (int i = 0; i < size; i++) {

int val = values[i];

LoadTextures(val);

}

}

void spawn\_number() {

if (check\_board\_full()) return;

// Đếm số ô trống

int empty\_count = 0;

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

if (board[i][j] == 0) empty\_count++;

}

}

if (empty\_count == 0) return;

// Chọn ngẫu nhiên một ô trống (từ 1 đến empty\_count)

int target = rand() % empty\_count + 1;

int current = 0;

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

if (board[i][j] == 0) {

current++;

if (current == target) {

board[i][j] = (rand() % 100 < 20) ? 4 : 2;

return;

}

}

}

}

}

void spawn\_initial\_numbers() {

// Đếm số ô trống

int empty\_cells = 0;

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

if (board[i][j] == 0) empty\_cells++;

}

}

// Xác định số lượng số cần spawn (tối đa 3)

int numbers\_to\_spawn = (empty\_cells < 3) ? empty\_cells : 3;

// Spawn số vào các ô trống

for (int k = 0; k < 1; k++) {

spawn\_number();

}

}

void quitSDL(SDL\_Window\* window, SDL\_Renderer\* renderer)

{

// Giải phóng font nếu đã load

if (font) {

TTF\_CloseFont(font);

font = nullptr;

}

TTF\_Quit(); // Đóng SDL\_ttf

// Clean SDL...

SDL\_DestroyRenderer(renderer);

SDL\_DestroyWindow(window);

SDL\_Quit();

}

void play\_2048()

{

bool out = false;

bool keydown = false;

bool valid\_move;

string filename = "high\_score.txt";

int score = 0, highscore = CinHighScore(filename);

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

board[i][j] = 0;

}

}

spawn\_initial\_numbers();

SDL\_Event event;

bool enteredGameOver = false;

while (!out) {

render(score, highscore, font);

bool resetGame = false;

if (GameOver()) {

if (!enteredGameOver) {

enteredGameOver = true; // Đánh dấu vừa mới vào trạng thái gameover

SDL\_SetWindowTitle(window, "Tro choi ket thuc! Nhan ENTER de choi lai. Nhan Q de ket thuc");

if (score > highscore) {

highscore = score;

CoutHighScore(filename, highscore);

}

}

bool quitGameByButton = false;

SDL\_Event gameOverEvent;

while (SDL\_PollEvent(&gameOverEvent)) {

if (gameOverEvent.type == SDL\_QUIT) {

out = true;

break;

}

else if (gameOverEvent.type == SDL\_MOUSEBUTTONDOWN) {

bool tempQuit = Buttons(renderer, font, gameOverEvent, resetGame);

if (tempQuit) {

quitGameByButton = true;

break; // Thoát khỏi vòng lặp sự kiện vì đã xử lý click chuột thoát

}

}

else if (gameOverEvent.type == SDL\_KEYDOWN) {

switch (gameOverEvent.key.keysym.sym){

case SDLK\_q:

out = true;

break;

case SDLK\_RETURN:

SDL\_SetWindowTitle(window, WINDOW\_TITLE);

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

board[i][j] = 0;

}

}

score = 0;

spawn\_initial\_numbers();

break;

}

}

}

if (quitGameByButton) {

out = true;

}

else if (resetGame) {

score = 0;

for (int r = 0; r < 4; r++) {

for (int c = 0; c < 4; c++) {

board[r][c] = 0;

}

}

enteredGameOver = false;

spawn\_initial\_numbers();

SDL\_SetWindowTitle(window, WINDOW\_TITLE);

}

}

else {

keydown = false;

while (!keydown) {

while (SDL\_PollEvent(&event)) {

if (event.type == SDL\_QUIT) {

keydown = 1;

out = 1;

break;

}

else if (event.type == SDL\_KEYDOWN) {

keydown = 1;

switch (event.key.keysym.sym) {

case SDLK\_q:

out = 1;

break;

case SDLK\_UP:

valid\_move = 0;

// Kiểm tra có nước đi hợp lệ không

for (int col = 0; col < 4; col++) {

for (int row = 0; row < 3; row++) {

if ((board[row][col] == board[row + 1][col] && board[row][col] != 0) ||

(board[row][col] == 0 && board[row + 1][col] != 0)) {

valid\_move = 1;

break;

}

}

if (valid\_move) break;

}

if (valid\_move) {

for (int col = 0; col < 4; col++) {

int temp[4] = { 0 };

int index = 0;

// Bước 1: Dồn ô khác 0 (từ TRÊN xuống)

for (int row = 0; row < 4; row++) {

if (board[row][col] != 0) {

temp[index++] = board[row][col];

}

}

// Bước 2: Gộp ô giống nhau

for (int i = 0; i < 3; i++) {

if (temp[i] == temp[i + 1] && temp[i] != 0) {

temp[i] \*= 2;

score += temp[i];

temp[i + 1] = 0;

}

}

// Bước 3: Dồn lại sau khi gộp

index = 0;

int merged[4] = { 0 };

for (int i = 0; i < 4; i++) {

if (temp[i] != 0) {

merged[index++] = temp[i];

}

}

// Bước 4: Cập nhật lại cột (đẩy LÊN đỉnh)

for (int row = 0; row < 4; row++) {

board[row][col] = (row < index) ? merged[row] : 0;

}

}

}

spawn\_initial\_numbers();

break;

case SDLK\_DOWN:

valid\_move = 0;

// Kiểm tra có nước đi hợp lệ không

for (int col = 0; col < 4; col++) {

for (int row = 0; row < 3; row++) {

if ((board[row][col] == board[row + 1][col] && board[row][col] != 0) ||

(board[row][col] != 0 && board[row + 1][col] == 0)) {

valid\_move = 1;

break;

}

}

if (valid\_move) break;

}

if (valid\_move) {

for (int col = 0; col < 4; col++) {

int temp[4] = { 0 }, index = 0;

// B1: Dồn ô khác 0 vào temp (từ DƯỚI lên)

for (int row = 3; row >= 0; row--) {

if (board[row][col] != 0) {

temp[index++] = board[row][col];

}

}

// B2: Gộp ô giống nhau (ví dụ: [2, 2, 4, 4] → [4, 0, 8, 0])

for (int i = 0; i < 3; i++) {

if (temp[i] == temp[i + 1] && temp[i] != 0) {

temp[i] \*= 2;

score += temp[i]; // Cộng điểm

temp[i + 1] = 0;

}

}

// B3: Dồn lại sau khi gộp (để không có ô trống ở giữa)

index = 0;

int merged[4] = { 0 };

for (int i = 0; i < 4; i++) {

if (temp[i] != 0) {

merged[index++] = temp[i];

}

}

// B4: Cập nhật lại cột (đẩy XUỐNG đáy)

for (int row = 0; row < 4; row++) {

board[row][col] = (3 - row < index) ? merged[3 - row] : 0;

}

}

}

spawn\_initial\_numbers();

break;

case SDLK\_LEFT:

valid\_move = 0;

// Kiểm tra có nước đi hợp lệ không

for (int row = 0; row < 4; row++) {

for (int col = 0; col < 3; col++) {

if ((board[row][col] == board[row][col + 1] && board[row][col] != 0) ||

(board[row][col] == 0 && board[row][col + 1] != 0)) {

valid\_move = 1;

break;

}

}

if (valid\_move) break;

}

if (valid\_move) {

for (int row = 0; row < 4; row++) {

int temp[4] = { 0 };

int index = 0;

// Bước 1: Dồn các ô khác 0 về bên trái

for (int col = 0; col < 4; col++) {

if (board[row][col] != 0) {

temp[index++] = board[row][col];

}

}

// Bước 2: Gộp các ô giống nhau

for (int i = 0; i < 3; i++) {

if (temp[i] == temp[i + 1] && temp[i] != 0) {

temp[i] \*= 2;

score += temp[i]; // Cộng điểm

temp[i + 1] = 0;

}

}

// Bước 3: Dồn lại sau khi gộp

index = 0;

for (int i = 0; i < 4; i++) {

if (temp[i] != 0) {

board[row][index++] = temp[i];

}

}

// Bước 4: Điền các ô trống bên phải

while (index < 4) {

board[row][index++] = 0;

}

}

}

spawn\_initial\_numbers();

break;

case SDLK\_RIGHT:

valid\_move = 0;

// Kiểm tra có nước đi hợp lệ không

for (int row = 0; row < 4; row++) {

for (int col = 0; col < 3; col++) {

if ((board[row][col] == board[row][col + 1] && board[row][col] != 0) ||

(board[row][col + 1] == 0 && board[row][col] != 0)) {

valid\_move = 1;

break;

}

}

if (valid\_move) break;

}

if (valid\_move) {

for (int row = 0; row < 4; row++) {

int temp[4] = { 0 };

int index = 0;

// B1: Dồn ô khác 0 (từ PHẢI sang trái)

for (int col = 3; col >= 0; col--) {

if (board[row][col] != 0) {

temp[index++] = board[row][col];

}

}

// B2: Gộp ô giống nhau

for (int i = 0; i < 3; i++) {

if (temp[i] == temp[i + 1] && temp[i] != 0) {

temp[i] \*= 2;

score += temp[i];

temp[i + 1] = 0;

}

}

// B3: Dồn lại sau khi gộp

index = 0;

int merged[4] = { 0 };

for (int i = 0; i < 4; i++) {

if (temp[i] != 0) {

merged[index++] = temp[i];

}

}

// B4: Cập nhật lại hàng (đẩy sang PHẢI)

for (int col = 0; col < 4; col++) {

board[row][3 - col] = (col < index) ? merged[col] : 0;

}

}

}

spawn\_initial\_numbers();

break;

case SDLK\_r:

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

board[i][j] = 0;

}

}

spawn\_initial\_numbers();

score = 0;

break;

}

}

SDL\_Delay(16);

}

SDL\_Delay(50);

}

}

}

SDL\_RenderPresent(renderer);

SDL\_Delay(16);

}

int main(int argc, char\* argv[])

{

init();

play\_2048();

quitSDL(window, renderer);

return 0;

}