Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

**ОТЧЕТ**

к лабораторной работе № 2

на тему «Расширенное использование оконного интерфейса Win 32 и GDI.

Формирование сложных изображений, создание и использование элементов управления, обработка различных сообщений, механизм перехвата сообщений (winhook)»

Выполнил:

студент гр 153504

Шишков В.В.

Проверил:

Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc147928064)

[2 Теоретические сведения по gdi+ и winhook 4](#_Toc147928065)

[3 Реализация программного продукта 6](#_Toc147928066)

[4 Результат выполнения программы 7](#_Toc147928067)

[Заключение 9](#_Toc147928068)

[Список использованных источников 10](#_Toc147928069)

[Приложение А 11](#_Toc147928070)

# **1 ЦЕЛЬ РАБОТЫ**

Задачей этой лабораторной работы является изучение и практическое применение расширенных возможностей оконного интерфейса Win32 и библиотеки GDI+ для создания сложных изображений, интеграции элементов управления, обработки сообщений, и реализации механизма перехвата сообщений с использованием WinHook.

# **2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ ПО GDI+ И WINHOOK**

Графический интерфейс приложений, разработанных с использованием технологии Win32 API, является одним из важнейших элементов современных операционных систем Microsoft Windows. Для обработки графики в таких приложениях широко применяется библиотека GDI+ (Graphics Device Interface Plus), предоставляющая мощные инструменты для рисования, обработки изображений и взаимодействия с графическими ресурсами. GDI+ предоставляет программистам возможность создания сложных графических элементов, включая растровую и векторную графику, текст, анимацию и другие визуальные элементы.

Одной из ключевых особенностей GDI+ является поддержка альфа-каналов, что позволяет создавать полупрозрачные и прозрачные изображения. Это важно для разработки интерактивных пользовательских интерфейсов, где элементы могут накладываться друг на друга с соблюдением прозрачности.[1]

WinHook - это механизм в операционной системе Windows, позволяющий приложениям перехватывать и обрабатывать различные сообщения, отправляемые окнам. Этот механизм чрезвычайно полезен для расширения функциональности приложений и реагирования на события, происходящие в системе. WinHook может использоваться для перехвата клавиш, мыши, системных событий и других важных событий, что делает его мощным инструментом для контроля над приложением.[2]

В контексте лабораторной работы, объединение GDI+ и WinHook позволяет создавать интерактивные пользовательские интерфейсы с богатой графикой и возможностью обработки пользовательских взаимодействий. Это позволяет разработчикам создавать приложения с более сложными и привлекательными графическими элементами, реагирующими на действия пользователя.

В лабораторной работе по использованию WinHook и GDI+ в оконных приложениях Win32, оба эти механизма используются совместно для расширения функциональности и улучшения визуального интерфейса. Программа разрабатывается с целью создания интерактивного клонсера "Сапер", который позволяет пользователю открывать клетки на игровом поле, устанавливать и удалять флаги, а также обрабатывать события нажатия мыши для интерактивных взаимодействий.

В коде лабораторной работы, WinHook используется для перехвата событий нажатия кнопок мыши, что позволяет программе реагировать на действия пользователя. Например, с помощью WinHook обрабатываются нажатия левой и правой кнопок мыши для открытия клеток и установки флагов. Это позволяет создать интерактивный пользовательский опыт, в котором пользователь может взаимодействовать с игровым полем.

С использованием GDI+, в коде лабораторной работы реализовано отображение сложных изображений на игровом поле. Например, изображения мин и флагов загружаются и рисуются с использованием GDI+, что обогащает визуальный интерфейс и делает игровое поле более привлекательным. GDI+ также позволяет работать с альфа-каналами, что позволяет создавать полупрозрачные изображения, что полезно для отображения визуальных состояний элементов, таких как флаги.

В конечном итоге, совместное использование WinHook и GDI+ в коде лабораторной работы создает возможность для создания сложных интерактивных приложений с богатыми графическими элементами. Пользователь может взаимодействовать с приложением, нажимая на клетки и устанавливая флаги, а GDI+ обеспечивает качественное отображение изображений, что делает приложение привлекательным и информативным.

# **3 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА**

Реализация игры "Сапёр" включает в себя использование WinHook и GDI+ для обеспечения интерактивности и качественной графики в оконном приложении Win32.

В первую очередь, WinHook был применен для обработки сообщений от мыши и обеспечения реакции на действия пользователя. Нажатия левой и правой кнопок мыши обрабатываются с использованием WinHook. Например, в обработчике WM\_LBUTTONDOWN, при нажатии левой кнопки мыши, реализована логика игры, включая проверку состояния игры и взаимодействие с ячейками игрового поля. Таким образом, WinHook обеспечивает возможность пользователю взаимодействовать с игрой через мышь.

Во-вторых, GDI+ был задействован для улучшения графического представления игры. С использованием GDI+, в игру были добавлены изображения мины и флага. Эти изображения загружаются из файлов и отображаются на игровом поле, что значительно повышает визуальное восприятие. Например, функция DrawImage из GDI+ позволяет отображать загруженные изображения на определенных координатах игрового поля.

В результате использования WinHook и GDI+ было реализовано взаимодействие с игровым полем с помощью мыши. Пользователь может кликать по клеткам поля и взаимодействовать с ними, нажимая левую и правую кнопки мыши. Также было улучшено графическое представление. Загруженные изображения могут представлять различные элементы игры, такие как мины и флаги, что делает графику более наглядной и привлекательной.

В игре создано меню, предоставляющее пользователю опции выбора сложности: легко, средне и сложно. Эти опции позволяют настраивать параметры игры, такие как размер игрового поля и количество мин.

Реализация меню начинается с создания радиокнопок для каждого уровня сложности. С использованием GDI+, красочные кнопки с надписями "Легко," "Средне" и "Сложно" отображаются на экране.

Обработчик сообщения WM\_COMMAND обрабатывает выбор уровня сложности, в зависимости от выбранной радиокнопки. Например, при выборе "Легко," размер игрового поля и количество мин устанавливаются на определенные значения, и игра пересоздается с этими параметрами. Это обеспечивает возможность настройки сложности игры.

Эти технологии совместно позволяют создать интерактивную игру "Сапёр" с улучшенным визуальным оформлением и возможностью взаимодействия с игровым полем с помощью мыши.

# **4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

В данной лабораторной было перемещено игровое поле, также свое положение и оформление изменили кнопки завершения игры и начала новой игры. Было добавлено меню выбора сложности. (рисунок 4.1).

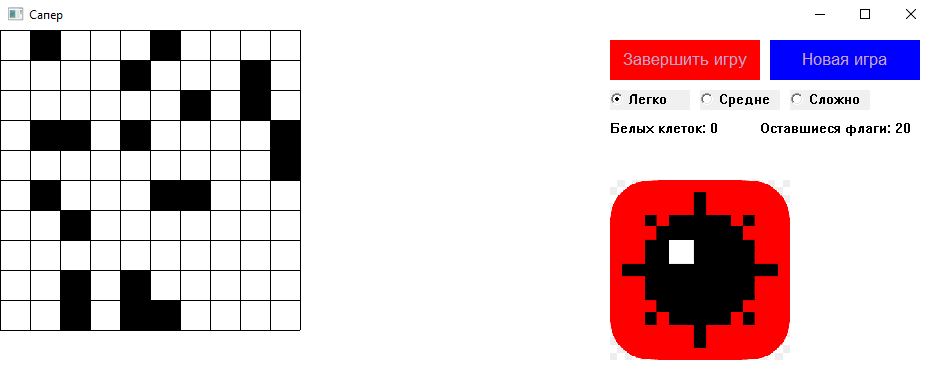


Рисунок 4.1 – Главное окно приложения

Также добавлено изображение игры “Сапер”, при нажатии на которое выведет сообщение (рисунок 4.2).

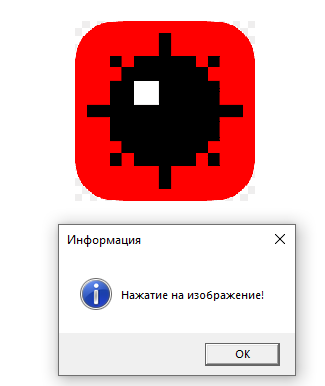


Рисунок 4.2 – Информация о нажатии на изображение

Теперь в приложении имеется возможность выбора сложности. По умолчанию идет легкий уровень, но можно выбрать средний и сложный. При выборе других уровней сложности поле меняет размер и количество мин. (рисунок 4.3, рисунок 4.4).

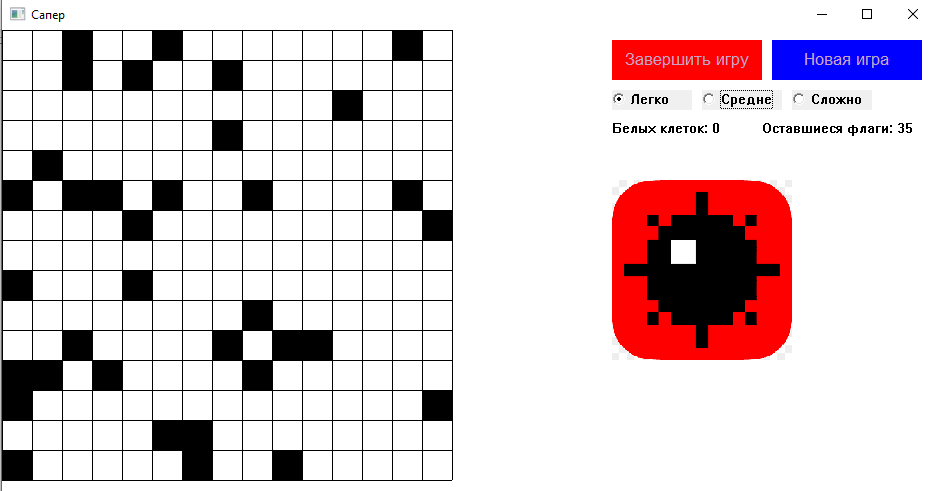


Рисунок 4.3 – Средний уровень

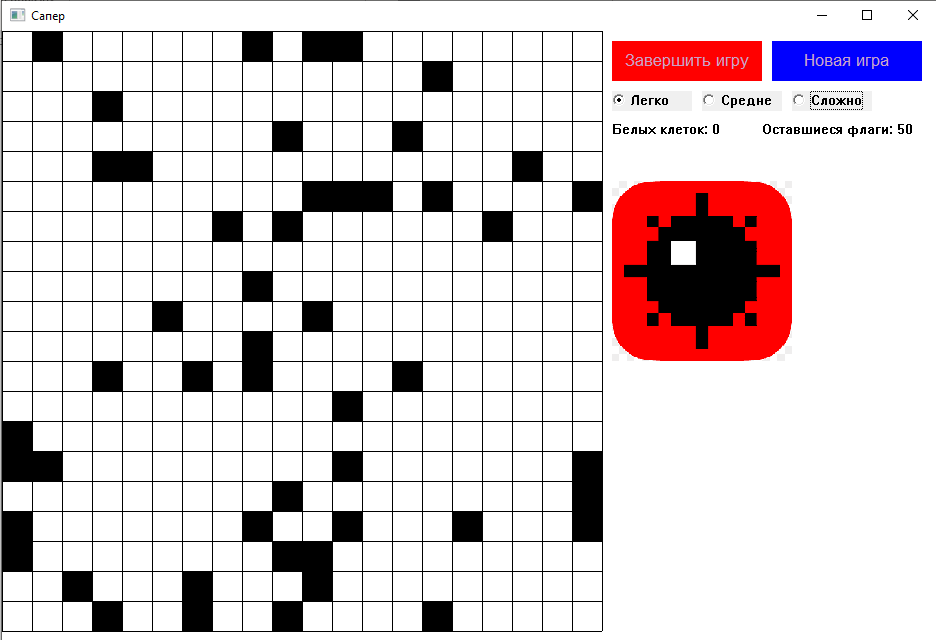


Рисунок 4.4 – Сложный уровень

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения лабораторной работы были рассмотрены и применены две важные технологии для создания оконных приложений в среде Win32: WinHook и GDI+. Эти технологии позволили усовершенствовать графический интерфейс приложения и реализовать различные функции, делая игру "Сапёр" более интересной и функциональной.

WinHook предоставил возможность перехватывать и обрабатывать сообщения, что позволило эффективно реагировать на события в приложении. Это было особенно полезно при обработке нажатий мыши для установки флагов на минных клетках и раскрытия нулевых областей в игровом поле.

GDI+ был использован для улучшения графического интерфейса игры. Он позволил загружать и отображать изображения, такие как изображение флага, а также улучшить отрисовку элементов управления, таких как радиокнопки для выбора уровня сложности.

Введение меню сложности в игру стало важным элементом для улучшения пользовательского опыта. Пользователь может выбирать между разными уровнями сложности, меняя размер поля и количество мин, что делает игру более настраиваемой и увлекательной.

Использование WinHook и GDI+ позволило создать более интерактивное и визуально привлекательное приложение. Полученный опыт в разработке оконных приложений на платформе Win32 позволит легче реализовывать сложные функции и улучшать интерфейс в будущих проектах.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] GDI+ [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/gdiplus/-gdiplus-gdi-start

[2] Windows hook: просто о сложном [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/icl\_group/articles/324718/

# **ПРИЛОЖЕНИЕ А**

include <windows.h>

#include <tchar.h>

#include <ctime>

#include <sstream>

#include <stack>

#include <gdiplus.h>

#include "window.h"

#include <fstream>

using namespace std;

#pragma comment (lib, "Gdiplus.lib")

using namespace Gdiplus;

#define ID\_BUTTON\_ENDGAME 1001

#define ID\_BUTTON\_NEWGAME 1002

enum Difficulty { EASY, MEDIUM, HARD };

Difficulty currentDifficulty = EASY;

int selectedDifficulty = 0;

HHOOK g\_mouseHook = NULL;

HWND hWndEasy;

HWND hWndMedium;

HWND hWndHard;

HINSTANCE hInst;

HWND hWndMain;

HWND hWndButtonEndGame;

HWND hWndButtonNewGame;

int cellSize = 30; // Размер ячейки

int numRows = 10; // Количество строк

int numCols = 10; // Количество столбцов

bool\*\* cellClicked = nullptr; // Массив для отслеживания, была ли ячейка нажата

bool\*\* cellChecked = nullptr; // Массив для отслеживания, была ли ячейка проверена

int\*\* cellCount = nullptr; // Массив для счетчиков

int totalWhiteCells = 0; // Общее количество белых клеток

int score = 0; // Счетчик белых клеток

bool gameOver = false; // Флаг, указывающий на завершение игры

int numMines = 20; // Количество мин на поле

int flagsPlaced = 0; // Количество установленных флагов

bool\*\* flags = nullptr; // Массив для отслеживания установленных флагов

bool revealMinesAlways = true; // Переменная для определения, всегда ли отображать черные клетки

int remainingFlags = numMines; // Счетчик оставшихся флагов

UINT\_PTR timerID = 1; // Идентификатор таймера

const UINT timerInterval = 100; // Интервал таймера в миллисекундах (100 миллисекунд)

HBITMAP hFlagBitmap = nullptr;

bool isExitButtonHovered = false;

bool isNewGameButtonHovered = false;

// Генерирует случайные черные точки на поле

void GenerateRandomBoard(int numBlackPoints) {

srand(static\_cast<unsigned int>(time(nullptr))); // Инициализируем генератор случайных чисел текущим временем

// Заполняем поле случайными черными точками и подсчитываем белые клетки

totalWhiteCells = numRows \* numCols - numBlackPoints;

// Инициализируем массивы

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

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

cellClicked[i][j] = false;

cellChecked[i][j] = false;

cellCount[i][j] = 0;

}

}

while (numBlackPoints > 0) {

int x = rand() % numCols;

int y = rand() % numRows;

if (!cellClicked[y][x]) {

cellClicked[y][x] = true;

numBlackPoints--;

}

}

// Вычисляем счетчики для белых клеток на основе мин в радиусе 1 от клетки

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

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

if (!cellClicked[i][j]) {

// Проверяем восемь соседних клеток

for (int dx = -1; dx <= 1; dx++) {

for (int dy = -1; dy <= 1; dy++) {

int nx = j + dx;

int ny = i + dy;

if (nx >= 0 && nx < numCols && ny >= 0 && ny < numRows && cellClicked[ny][nx]) {

cellCount[i][j]++;

}

}

}

}

}

}

// Инициализируем массив для отслеживания установленных флагов

flags = new bool\* [numRows];

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

flags[i] = new bool[numCols]();

}

}

// Функция отрисовки игрового поля

void DrawBoard(HDC hdc) {

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

int y = i \* cellSize;

MoveToEx(hdc, 0, y, NULL);

LineTo(hdc, numCols \* cellSize, y);

}

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

int x = i \* cellSize;

MoveToEx(hdc, x, 0, NULL);

LineTo(hdc, x, numRows \* cellSize);

}

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

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

RECT cellRect = { j \* cellSize, i \* cellSize, (j + 1) \* cellSize, (i + 1) \* cellSize };

if (gameOver || revealMinesAlways) {

if (cellClicked[i][j]) {

HBRUSH hBlackBrush = CreateSolidBrush(RGB(0, 0, 0));

FillRect(hdc, &cellRect, hBlackBrush);

DeleteObject(hBlackBrush);

}

}

if (!cellClicked[i][j] && cellChecked[i][j]) {

if (cellCount[i][j] > 0) {

std::wstringstream ss;

ss << cellCount[i][j];

TextOut(hdc, cellRect.left + 4, cellRect.top + 4, ss.str().c\_str(), static\_cast<int>(ss.str().length()));

}

else {

// Отрисовываем "0" для белых клеток без черных соседей

TextOut(hdc, cellRect.left + 4, cellRect.top + 4, L"0", 1);

}

}

else if (flags[i][j]) {

HBRUSH hRedBrush = CreateSolidBrush(RGB(255, 0, 0));

HBRUSH hOldBrush = static\_cast<HBRUSH>(SelectObject(hdc, hRedBrush));

POINT points[3];

points[0] = { cellRect.left, cellRect.top };

points[1] = { cellRect.right, cellRect.top };

points[2] = { (cellRect.left + cellRect.right) / 2, cellRect.bottom };

Polygon(hdc, points, 3);

SelectObject(hdc, hOldBrush);

DeleteObject(hRedBrush);

}

}

}

// Отрисовываем счетчик белых клеток

std::wstringstream ss;

ss << L"Белых клеток: " << score;

TextOut(hdc, 610, 90, ss.str().c\_str(), static\_cast<int>(ss.str().length()));

// Отрисовываем счетчик оставшихся флагов

std::wstringstream flagsCounter;

flagsCounter << L"Оставшиеся флаги: " << remainingFlags;

TextOut(hdc, 760, 90, flagsCounter.str().c\_str(), static\_cast<int>(flagsCounter.str().length()));

}

// Функция проверки условия победы в игре

bool CheckWin() {

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

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

if (cellClicked[i][j] && cellCount[i][j] != -1 && !flags[i][j]) {

return false; // Если есть открытая не-мина без флага, пользователь не выиграл

}

}

}

return true; // Если все открытые ячейки - мины или с флагами, пользователь выиграл

}

// Функция проверки условия победы и вывод этого

void CheckForWin() {

if (!gameOver) {

bool allFlagsOnMines = true; // Дополнительная переменная для проверки

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

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

if (cellClicked[i][j] && cellCount[i][j] != -1 && !flags[i][j]) {

return; // Если есть открытая не-мина без флага, выходим

}

if (!cellClicked[i][j] && !cellChecked[i][j]) {

allFlagsOnMines = false; // Если есть непроверенная клетка, выставляем флаг

}

}

}

if (allFlagsOnMines) { // Если все флаги находятся на минах

MessageBox(hWndMain, \_T("Поздравляем! Вы выиграли!"), \_T("Победа"), MB\_OK | MB\_ICONINFORMATION);

gameOver = true;

//Разблокировка кнопок

EnableWindow(hWndButtonEndGame, TRUE);

EnableWindow(hWndButtonNewGame, TRUE);

}

}

}

bool IsInImageArea(POINT pt) {

int imageX = 650;

int imageY = 150;

int imageWidth = 180;

int imageHeight = 180;

// Проверяем, попадает ли точка в область изображения

if (pt.x >= imageX && pt.x <= imageX + imageWidth && pt.y >= imageY && pt.y <= imageY + imageHeight) {

return true;

}

return false;

}

LRESULT CALLBACK MouseHookProc(int nCode, WPARAM wParam, LPARAM lParam) {

if (nCode == HC\_ACTION) {

MOUSEHOOKSTRUCT\* pMouseStruct = (MOUSEHOOKSTRUCT\*)lParam;

if (pMouseStruct != NULL && wParam == WM\_LBUTTONDOWN) {

// Проверяем, был ли клик выполнен в области изображения

if (IsInImageArea(pMouseStruct->pt)) {

// Если клик был в области изображения, выведите текст в окне

MessageBox(NULL, \_T("Нажатие на изображение!"), \_T("Информация"), MB\_OK | MB\_ICONINFORMATION);

}

}

}

// Передаем управление следующему обработчику в цепочке

return CallNextHookEx(g\_mouseHook, nCode, wParam, lParam);

}

bool SetMouseHook() {

g\_mouseHook = SetWindowsHookEx(WH\_MOUSE\_LL, MouseHookProc, GetModuleHandle(NULL), 0);

return g\_mouseHook != NULL;

}

// Функция для удаления глобального хука

void UnhookMouse() {

if (g\_mouseHook != NULL) {

UnhookWindowsHookEx(g\_mouseHook);

g\_mouseHook = NULL;

}

}

// Функция отображения всех мин на игровом поле при завершении игры

void RevealAllMines(HDC hdc) {

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

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

if (cellClicked[i][j]) {

HBRUSH hBlackBrush = CreateSolidBrush(RGB(0, 0, 0));

HBRUSH hOldBrush = static\_cast<HBRUSH>(SelectObject(hdc, hBlackBrush));

RECT cellRect = { j \* cellSize, i \* cellSize, (j + 1) \* cellSize, (i + 1) \* cellSize };

Ellipse(hdc, cellRect.left, cellRect.top, cellRect.right, cellRect.bottom);

SelectObject(hdc, hOldBrush);

DeleteObject(hBlackBrush);

}

}

}

}

// Функция отображения всех мин на игровом поле при завершении игры

void SpreadZeros(int x, int y) {

std::stack<std::pair<int, int>> zeroStack;

zeroStack.push(std::make\_pair(x, y));

while (!zeroStack.empty()) {

std::pair<int, int> current = zeroStack.top();

zeroStack.pop();

int cx = current.first;

int cy = current.second;

// Проверяем, что счетчик текущей белой клетки равен 0

if (cellCount[cy][cx] == 0) {

// Отображаем "0" для текущей белой клетки

cellChecked[cy][cx] = true;

score++;

// Перерисовываем окно

InvalidateRect(hWndMain, NULL, TRUE);

// Перебираем восемь соседних клеток

for (int dx = -1; dx <= 1; dx++) {

for (int dy = -1; dy <= 1; dy++) {

int nx = cx + dx;

int ny = cy + dy;

if (nx >= 0 && nx < numCols && ny >= 0 && ny < numRows) {

if (!cellChecked[ny][nx]) {

// Добавляем соседнюю клетку с нулевым счетчиком в стек

zeroStack.push(std::make\_pair(nx, ny));

}

}

}

}

}

}

}

void ResetGame() {

// Освобождаем память для массивов cellClicked, cellChecked, cellCount и flags

if (cellClicked != nullptr) {

delete[] cellClicked;

delete[] cellChecked;

delete[] cellCount;

}

if (flags != nullptr) {

delete[] flags;

}

cellClicked = new bool\* [numRows];

cellChecked = new bool\* [numRows];

cellCount = new int\* [numRows];

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

cellClicked[i] = new bool[numCols]();

cellChecked[i] = new bool[numCols]();

cellCount[i] = new int[numCols]();

}

GenerateRandomBoard(numMines); // Генерируем случайные черные точки

score = 0; // Сбрасываем счетчик белых клеток

// Инициализируем массив flags заново

flags = new bool\* [numRows];

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

flags[i] = new bool[numCols]();

}

flagsPlaced = 0;

remainingFlags = numMines;

// Перерисовываем окно

InvalidateRect(hWndMain, NULL, TRUE);

}

// Функция обработки сообщений окна

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {

GdiplusStartupInput gdiplusStartupInput;

ULONG\_PTR gdiplusToken;

GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, NULL);

switch (message) {

case WM\_CREATE:

break;

case WM\_TIMER:

// Проверяем победу каждые 100 миллисекунд

CheckForWin();

break;

case WM\_PAINT: {

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

// Отрисовываем игровое поле

DrawBoard(hdc);

// Создаем объект Graphics и загружаем изображение

Graphics graphics(hdc);

Bitmap bmp(L"C:\\plusi\\img\\mine.png");

int x = 610; // X-координата

int y = 150; // Y-координата

int width = bmp.GetWidth(); // Ширина изображения

int height = bmp.GetHeight(); // Высота изображения

// Выводим изображение на экран

graphics.DrawImage(&bmp, x, y, width, height);

EndPaint(hWnd, &ps);

break;

}

case WM\_DESTROY:

// Освобождаем память для массивов cellClicked, cellChecked и cellCount

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

delete[] cellClicked[i];

delete[] cellChecked[i];

delete[] cellCount[i];

}

delete[] cellClicked;

delete[] cellChecked;

delete[] cellCount;

KillTimer(hWnd, timerID);

PostQuitMessage(0);

GdiplusShutdown(gdiplusToken);

PostQuitMessage(0);

break;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case ID\_BUTTON\_ENDGAME:

// Обработка нажатия кнопки "Завершить игру"

PostQuitMessage(0);

break;

case ID\_BUTTON\_NEWGAME:

// Обработка нажатия кнопки "Новая игра"

ResetGame();

gameOver = false;

break;

case ID\_DIFFICULTY\_EASY:

CheckRadioButton(hWndMain, ID\_DIFFICULTY\_EASY, ID\_DIFFICULTY\_HARD, ID\_DIFFICULTY\_EASY);

numRows = 10;

numCols = 10;

numMines = 20;

ResetGame();

break;

case ID\_DIFFICULTY\_MEDIUM:

CheckRadioButton(hWndMain, ID\_DIFFICULTY\_EASY, ID\_DIFFICULTY\_HARD, ID\_DIFFICULTY\_MEDIUM);

numRows = 15;

numCols = 15;

numMines = 35;

ResetGame();

break;

case ID\_DIFFICULTY\_HARD:

CheckRadioButton(hWndMain, ID\_DIFFICULTY\_EASY, ID\_DIFFICULTY\_HARD, ID\_DIFFICULTY\_HARD);

numRows = 20;

numCols = 20;

numMines = 50;

ResetGame();

break;

}

break;

case WM\_LBUTTONDOWN:

POINT pt;

pt.x = LOWORD(lParam);

pt.y = HIWORD(lParam);

if (IsInImageArea(pt)) {

MessageBox(hWnd, L"Нажатие на изображение!", L"Информация", MB\_OK | MB\_ICONINFORMATION);

}

break;

if (!gameOver) {

int x = LOWORD(lParam) / cellSize;

int y = HIWORD(lParam) / cellSize;

// Проверяем, что нажатие произошло в пределах игрового поля

if (x >= 0 && x < numCols && y >= 0 && y < numRows) {

if (cellClicked[y][x]) {

gameOver = true;

EnableWindow(hWndButtonEndGame, TRUE);

EnableWindow(hWndButtonNewGame, TRUE);

HDC hdc = GetDC(hWnd);

RevealAllMines(hdc);

ReleaseDC(hWnd, hdc);

MessageBox(hWnd, \_T("Game Over"), \_T("Сообщение"), MB\_OK | MB\_ICONERROR);

}

else if (!cellChecked[y][x]) {

if (cellCount[y][x] == 0) {

// Распространение нулей

SpreadZeros(x, y);

}

else {

// Отображаем счетчик для белых клеток

cellChecked[y][x] = true;

score++;

// Перерисовываем окно

HDC hdc = GetDC(hWnd);

DrawBoard(hdc); // Перерисовываем поле

ReleaseDC(hWnd, hdc);

}

}

}

}

break;

case WM\_DRAWITEM: {

LPDRAWITEMSTRUCT lpDIS = (LPDRAWITEMSTRUCT)lParam;

HDC hdcButton = lpDIS->hDC;

if (lpDIS->CtlID == ID\_BUTTON\_ENDGAME) {

HBRUSH hRedBrush = CreateSolidBrush(RGB(255, 0, 0)); // Желаемый цвет фона

FillRect(hdcButton, &lpDIS->rcItem, hRedBrush); // Закрашиваем фон красным цветом

HFONT hFont = CreateFont(20, 0, 0, 0, FW\_NORMAL, FALSE, FALSE, FALSE, DEFAULT\_CHARSET, OUT\_DEFAULT\_PRECIS,

CLIP\_DEFAULT\_PRECIS, DEFAULT\_QUALITY, DEFAULT\_PITCH | FF\_DONTCARE, \_T("Arial"));

HFONT hOldFont = (HFONT)SelectObject(hdcButton, hFont);

SetTextColor(hdcButton, RGB(200, 162, 200)); // Устанавливаем цвет текста

SetBkMode(hdcButton, TRANSPARENT); // Устанавливаем прозрачный фон текста

RECT textRect = lpDIS->rcItem;

DrawText(hdcButton, \_T("Завершить игру"), -1, &textRect, DT\_SINGLELINE | DT\_CENTER | DT\_VCENTER);

// Освобождаем созданную кисть

DeleteObject(hRedBrush);

}

else if (lpDIS->CtlID == ID\_BUTTON\_NEWGAME) {

HBRUSH hBlueBrush = CreateSolidBrush(RGB(0, 0, 255)); // Желаемый цвет фона

FillRect(hdcButton, &lpDIS->rcItem, hBlueBrush); // Закрашиваем фон синим цветом

HFONT hFont = CreateFont(20, 0, 0, 0, FW\_NORMAL, FALSE, FALSE, FALSE, DEFAULT\_CHARSET, OUT\_DEFAULT\_PRECIS,

CLIP\_DEFAULT\_PRECIS, DEFAULT\_QUALITY, DEFAULT\_PITCH | FF\_DONTCARE, \_T("Arial"));

HFONT hOldFont = (HFONT)SelectObject(hdcButton, hFont);

SetTextColor(hdcButton, RGB(200, 162, 200)); // Устанавливаем цвет текста

SetBkMode(hdcButton, TRANSPARENT); // Устанавливаем прозрачный фон текста

RECT textRect = lpDIS->rcItem;

DrawText(hdcButton, \_T("Новая игра"), -1, &textRect, DT\_SINGLELINE | DT\_CENTER | DT\_VCENTER);

DeleteObject(hBlueBrush);

}

return TRUE;

}

case WM\_RBUTTONDOWN:

if (!gameOver) {

int x = LOWORD(lParam) / cellSize;

int y = HIWORD(lParam) / cellSize;

// Проверяем, что нажатие произошло в пределах игрового поля

if (x >= 0 && x < numCols && y >= 0 && y < numRows) {

// Если флаг уже установлен, убираем его и увеличиваем счетчик оставшихся флагов

if (!cellChecked[y][x]) {

if (flags[y][x]) {

flags[y][x] = false;

flagsPlaced--;

remainingFlags++;

}

else if (remainingFlags > 0) {

flags[y][x] = true;

flagsPlaced++;

remainingFlags--;

}

// Перерисовываем окно

InvalidateRect(hWnd, NULL, TRUE);

}

}

// Проверяем, выиграл ли игрок после каждого хода

if (flagsPlaced == numMines && CheckWin()) {

MessageBox(hWnd, \_T("Поздравляем! Вы выиграли!"), \_T("Победа"), MB\_OK | MB\_ICONINFORMATION);

gameOver = true;

EnableWindow(hWndButtonEndGame, TRUE);

EnableWindow(hWndButtonNewGame, TRUE);

}

}

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

// Функция создания и инициализации окна

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow) {

hInst = hInstance;

// Инициализируем массивы для отслеживания нажатых ячеек, проверенных ячеек и счетчиков

cellClicked = new bool\* [numRows];

cellChecked = new bool\* [numRows];

cellCount = new int\* [numRows];

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

cellClicked[i] = new bool[numCols]();

cellChecked[i] = new bool[numCols]();

cellCount[i] = new int[numCols]();

}

GenerateRandomBoard(numMines); // Генерируем случайные черные точки

timerID = SetTimer(hWndMain, 1, timerInterval, NULL);

if (timerID == 0) {

MessageBox(hWndMain, \_T("Не удалось создать таймер."), \_T("Ошибка"), MB\_OK | MB\_ICONERROR);

return FALSE;

}

hWndMain = CreateWindow(

\_T("SapperApp"), \_T("Сапер"), WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT, static\_cast<int>(numCols \* cellSize \* 1.5) + 500, numRows \* cellSize + 600, NULL, NULL, hInstance, NULL);

hWndButtonEndGame = CreateWindow(

\_T("BUTTON"), \_T("Завершить игру"), WS\_CHILD | WS\_VISIBLE | BS\_OWNERDRAW,

610, 10, 150, 40, hWndMain, (HMENU)ID\_BUTTON\_ENDGAME, hInstance, NULL);

hWndButtonNewGame = CreateWindow(

\_T("BUTTON"), \_T("Новая игра"), WS\_CHILD | WS\_VISIBLE | BS\_OWNERDRAW,

770, 10, 150, 40, hWndMain, (HMENU)ID\_BUTTON\_NEWGAME, hInstance, NULL);

HWND hWndEasy = CreateWindow(

\_T("BUTTON"), \_T("Легко"), WS\_CHILD | WS\_VISIBLE | BS\_RADIOBUTTON,

610, 60, 80, 20, hWndMain, (HMENU)ID\_DIFFICULTY\_EASY, hInstance, NULL);

HWND hWndMedium = CreateWindow(

\_T("BUTTON"), \_T("Средне"), WS\_CHILD | WS\_VISIBLE | BS\_RADIOBUTTON,

700, 60, 80, 20, hWndMain, (HMENU)ID\_DIFFICULTY\_MEDIUM, hInstance, NULL);

HWND hWndHard = CreateWindow(

\_T("BUTTON"), \_T("Сложно"), WS\_CHILD | WS\_VISIBLE | BS\_RADIOBUTTON,

790, 60, 80, 20, hWndMain, (HMENU)ID\_DIFFICULTY\_HARD, hInstance, NULL);

if (!hWndMain) {

return FALSE;

}

ShowWindow(hWndMain, nCmdShow);

UpdateWindow(hWndMain);

return TRUE;

}

// Точка входа в приложение

int APIENTRY \_tWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPTSTR lpCmdLine, int nCmdShow) {

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// Регистрируем класс окна

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, IDI\_APPLICATION);

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = NULL;

wcex.lpszClassName = \_T("SapperApp");

wcex.hIconSm = LoadIcon(wcex.hInstance, IDI\_APPLICATION);

if (!RegisterClassEx(&wcex)) {

return FALSE;

}

// Инициализируем главное окно приложения

if (!InitInstance(hInstance, nCmdShow)) {

return FALSE;

}

GdiplusStartupInput gdiplusStartupInput;

ULONG\_PTR gdiplusToken;

GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, nullptr);

// Основной цикл обработки сообщений

MSG msg;

while (GetMessage(&msg, NULL, 0, 0)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return (int)msg.wParam;

}