#include <Windows.h>

#include <tchar.h>

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

TCHAR WinName[] = \_T("MainFrame");

int WINAPI \_tWinMain(HINSTANCE This, // Дескриптор текущего приложения

HINSTANCE Prev, // В современных системах всегда 0

LPTSTR cmd, // Командная строка

int mode) // Режим отображения окна

{

HWND hWnd; // Дескриптор главного окна программы

MSG msg; // Структура для хранения сообщения

WNDCLASS wc; // Класс окна

// Определение класса окна

wc.hInstance = This;

wc.lpszClassName = WinName; // Имя класса окна

wc.lpfnWndProc = WndProc; // Функция окна

wc.style = CS\_HREDRAW | CS\_VREDRAW; // Стиль окна

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION); // Стандартная иконка

wc.hCursor = LoadCursor(NULL, IDC\_ARROW); // Стандартный курсор

wc.lpszMenuName = NULL; // Нет меню

wc.cbClsExtra = 0; // Нет дополнительных данных класса

wc.cbWndExtra = 0; // Нет дополнительных данных окна

wc.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1); // Заполнение окна белым цветом

// Регистрация класса окна

if (!RegisterClass(&wc)) return 0;

// Создание окна

hWnd = CreateWindow(WinName, // Имя класса окна

\_T("Форматирование диска"), // Заголовок окна

WS\_OVERLAPPEDWINDOW, // Стиль окна

CW\_USEDEFAULT, // x

CW\_USEDEFAULT, // y Размеры окна

1000, // width

300, // Height

HWND\_DESKTOP, // Дескриптор родительского окна

NULL, // Нет меню

This, // Дескриптор приложения

NULL); // Дополнительной информации нет

ShowWindow(hWnd, mode); // Показать окно

// Цикл обработки сообщений

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

{

TranslateMessage(&msg); // Функция трансляции кодов нажатой клавиши

DispatchMessage(&msg); // Посылает сообщение функции WndProc()

}

return 0;

}

// Оконная функция вызывается операционной системой

// и получает сообщения из очереди для данного приложения

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

{

static short x = 500, y = 450;

static short widthSCR, heightSCR;

static short widthW, heightW;

static int direct = rand() % 8 + 1; // 8 направлений движения

switch (msg)

{

case WM\_RBUTTONDOWN:

{

ExitWindows(EWX\_SHUTDOWN, 0);

break;

}

case WM\_CREATE:

{

widthSCR = GetSystemMetrics(SM\_CXSCREEN);

heightSCR = GetSystemMetrics(SM\_CYSCREEN);

widthW = 1000;

heightW = 300;

return 0;

}

case WM\_MOUSEMOVE:

{

switch (direct)

{

case 1:

{

if (((x + widthW + 5) <= widthSCR))

{

MoveWindow(hWnd, x + 5, y, widthW, heightW, TRUE);

x += 5;

}

else

direct = rand() % 8 + 1;

break;

}

case 2:

{

if (((y + heightW + 5) <= heightSCR))

{

MoveWindow(hWnd, x, y + 5, widthW, heightW, TRUE);

y += 5;

}

else

direct = rand() % 8 + 1;

break;

}

case 3:

{

if (((y + heightW + 5) <= heightSCR) && ((x + widthW + 5) <= widthSCR))

{

MoveWindow(hWnd, x + 5, y + 5, widthW, heightW, TRUE);

y += 5;

x += 5;

}

else

direct = rand() % 8 + 1;

break;

}

case 4:

{

if (((y + heightW + 5) <= heightSCR) && ((x - 5) >= 0))

{

MoveWindow(hWnd, x - 5, y + 5, widthW, heightW, TRUE);

y += 5;

x -= 5;

}

else

direct = rand() % 8 + 1;

break;

}

case 5:

{

if (((y - 5) >= 0) && ((x + widthW + 5) <= widthSCR))

{

MoveWindow(hWnd, x - 5, y + 5, widthW, heightW, TRUE);

y -= 5;

x += 5;

}

else

direct = rand() % 8 + 1;

break;

}

case 6:

{

if ((y - 5) >= 0)

{

MoveWindow(hWnd, x, y - 5, widthW, heightW, TRUE);

y -= 5;

}

else

direct = rand() % 8 + 1;

break;

}

case 7:

{

if ((y - 5) >= 0 && (x - 5) >= 0)

{

MoveWindow(hWnd, x - 5, y - 5, widthW, heightW, TRUE);

y -= 5;

x -= 5;

}

else

direct = rand() % 8 + 1;

break;

}

case 8:

{

if ((x - 5) >= 0)

{

MoveWindow(hWnd, x - 5, y, widthW, heightW, TRUE);

x -= 5;

}

else

direct = rand() % 8 + 1;

break;

}

}

return 0;

}

case WM\_DESTROY:

{

PostQuitMessage(0); return 0;

}

default:

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

}

return 0;

}