#include <windows.h>

#include <windowsx.h>

#include <tchar.h>

// Объявление функций

BOOL RegClass(WNDPROC, LPCTSTR, HBRUSH);

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

// Описание глобальных переменных

HINSTANCE hInst;

HBRUSH hBrushBackground; // Глобальная переменная для кисти фона

POINT Current = { 500, 325 };

POINT Home = { 500, 325 };

POINT Step = { 30, 30 };

int timerStep = 100;

enum class dirMov {

MS\_BOTTOMRIGHT,

MS\_LEFT,

MS\_TOP,

MS\_RIGHT,

MS\_BOTTOMLEFT

};

int MoveState = 0;

// Функция рисования

void ClientDraw(HWND hwnd, WPARAM wParam, LPARAM lParam) {

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hwnd, &ps);

SetBkMode(hdc, TRANSPARENT);

LPCWSTR text = L"Вывод текста в окно";

TextOut(hdc, 425, 50, text, wcslen(text));

// Структура, описывающая заливку линии

LOGBRUSH lb;

lb.lbHatch = 0;

lb.lbStyle = PS\_SOLID;

lb.lbColor = RGB(0, 0, 0);

HGDIOBJ hPen1 = ExtCreatePen(PS\_GEOMETRIC | PS\_DASH, 4, &lb, 0, NULL);

lb.lbColor = RGB(250, 200, 0);

HGDIOBJ hPen2 = ExtCreatePen(PS\_GEOMETRIC | PS\_DOT, 5, &lb, 0, NULL);

// Выбрали перо 1, старое сохранили в hPenOld

HGDIOBJ hPenOld = SelectObject(hdc, hPen1);

MoveToEx(hdc, Current.x - 250, Current.y, NULL);

LineTo(hdc, Current.x + 250, Current.y);

MoveToEx(hdc, Current.x, Current.y - 150, NULL);

LineTo(hdc, Current.x, Current.y + 150);

// Выбрали перо 2

SelectObject(hdc, hPen2);

SelectObject(hdc, GetStockObject(NULL\_BRUSH));

Ellipse(hdc, Current.x - 235, Current.y - 135, Current.x + 235, Current.y + 135);

// Вернули старое перо, удалили созданные, освободив место в памяти

SelectObject(hdc, hPenOld);

DeleteObject(hPen1);

DeleteObject(hPen2);

EndPaint(hwnd, &ps);

}

void ProcessKey(HWND hwnd, WPARAM wParam, LPARAM lParam) {

switch (wParam) {

case VK\_UP:

Current.y = (Current.y - Step.y + 750) % 750;

break;

case VK\_DOWN:

Current.y = (Current.y + Step.y) % 750;

break;

case VK\_LEFT:

Current.x = (Current.x - Step.x + 1000) % 1000;

break;

case VK\_RIGHT:

Current.x = (Current.x + Step.x) % 1000;

break;

case VK\_HOME:

Current = Home;

break;

default:

break;

}

// Пометка всей рабочей области окна, как требующей отрисовки

InvalidateRect(hwnd, NULL, 1);

}

void ProcessMouse(HWND hwnd, WPARAM wParam, LPARAM lParam) {

if (wParam & MK\_LBUTTON) {

Current.x = lParam % 0x10000;

Current.y = lParam / 0x10000;

InvalidateRect(hwnd, NULL, 1);

}

if (wParam & MK\_RBUTTON) {

Current.x = lParam % 0x10000;

Current.y = lParam / 0x10000;

InvalidateRect(hwnd, NULL, 1);

}

if (wParam & MK\_SHIFT) {

Current.x = lParam % 0x10000;

Current.y = lParam / 0x10000;

InvalidateRect(hwnd, NULL, 1);

}

// В любом случае вне зависимости от кнопок

/\*Current.x = lParam % 0x10000;

Current.y = lParam / 0x10000;

InvalidateRect(hwnd, NULL, 1);\*/

}

void NextMoveStep(HWND hwnd) {

switch (MoveState) {

case 0: //правый низ

if (Current.y > 525 && Current.x > 700)

MoveState = 1;

else {

Current.y += Step.y;

Current.x += Step.x;

}

break;

case 1: //лево

if (Current.x < 275)

MoveState = 2;

else

Current.x -= Step.x;

break;

case 2: //верх

if (Current.y < 200)

MoveState = 3;

else

Current.y -= Step.y;

break;

case 3: //право

if (Current.x > 700)

MoveState = 4;

else

Current.x += Step.x;

break;

case 4: //левый низ

if ((Current.y >= 295 && Current.y <= 355) && (Current.x >= 470 && Current.x <= 530)) {

MoveState = 0;

}

else {

Current.y += Step.y;

Current.x -= Step.x;

}

break;

default:

MoveState = 0;

break;

}

InvalidateRect(hwnd, NULL, 1);

}

// Описание главной функции

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevlnstance, LPSTR lpszCmdLine, int nCmdShow) {

MSG msg;

HWND hwnd;

hInst = hInstance;

hBrushBackground = CreateSolidBrush(RGB(230, 220, 250));

if (!RegClass(WndProc, L"My class", hBrushBackground)) {

return false;

}

hwnd = CreateWindow(L"My class", L"Основы системного программирования, 1 лаба",

WS\_OVERLAPPEDWINDOW | WS\_VISIBLE,

CW\_USEDEFAULT, CW\_USEDEFAULT, 1000, 750, 0, 0, hInstance, NULL);

if (!hwnd) {

return false;

}

SetTimer(hwnd, 0, timerStep, NULL);

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

// Освобождение кисти перед выходом

DeleteObject(hBrushBackground);

KillTimer(hwnd, 0);

return msg.wParam;

}

// Описание функции регистрации классов

BOOL RegClass(WNDPROC Proc, LPCTSTR szName, HBRUSH brBackground) {

WNDCLASS wc;

wc.style = wc.cbClsExtra = wc.cbWndExtra = 0;

wc.lpfnWndProc = Proc;

wc.hInstance = hInst;

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

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

wc.hbrBackground = brBackground;

wc.lpszMenuName = NULL;

wc.lpszClassName = szName;

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

}

// Описание функции окон

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

static HWND hwndChild = NULL;

MSG timermsg;

switch (msg) {

case WM\_PAINT:

ClientDraw(hwnd, wParam, lParam);

break;

case WM\_KEYDOWN:

ProcessKey(hwnd, wParam, lParam);

break;

case WM\_MOUSEMOVE:

ProcessMouse(hwnd, wParam, lParam);

break;

case WM\_TIMER:

/\*if (wParam == 0) {

NextMoveStep(hwnd);

}\*/

break;

case WM\_CLOSE:

PostQuitMessage(0);

break;

case WM\_DESTROY:

break;

default:

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

}

return 0;

}