Темнохуд Михаил, ПЗПИ-16-2

Лабораторная работа №2

# Задание:

1) Створити 2 кнопки з надписами «лінія» і «багатокутник». Користувач на робочій області натискає кнопку миші кілька разів. Потім натискає одну з кнопок. Залежно від натиснутої кнопки повинна намалюватися або ламана, яка з'єднує всі точки позначені натисканням мишки, або багатокутник (замкнутий контур).

# Код программы:

// lab2.cpp: Определяет точку входа для приложения.

//

#include "stdafx.h"

#include "lab2.h"

#include <vector>

using namespace std;

#define MAX\_LOADSTRING 100

#define ID\_BUTTON\_POLYLINE 1001

#define ID\_BUTTON\_POLYGON 1002

#define POLYGON 0

#define POLYLINE 1

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

vector<POINT> points;

INT currentMode = POLYLINE;

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

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

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: разместите код здесь.

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_LAB2, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LAB2));

MSG msg;

// Цикл основного сообщения:

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

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int)msg.wParam;

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// НАЗНАЧЕНИЕ: регистрирует класс окна.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW 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, MAKEINTRESOURCE(IDI\_LAB2));

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

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

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_LAB2);

wcex.lpszClassName = szWindowClass;

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

return RegisterClassExW(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// НАЗНАЧЕНИЕ: сохраняет обработку экземпляра и создает главное окно.

//

// КОММЕНТАРИИ:

//

// В данной функции дескриптор экземпляра сохраняется в глобальной переменной, а также

// создается и выводится на экран главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить дескриптор экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

// ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// НАЗНАЧЕНИЕ: обрабатывает сообщения в главном окне.

//

// WM\_COMMAND — обработать меню приложения

// WM\_PAINT — отрисовать главное окно

// WM\_DESTROY — отправить сообщение о выходе и вернуться

//

//

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

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case ID\_BUTTON\_POLYGON:

if (currentMode != POLYGON)

points.clear();

currentMode = POLYGON;

//WndProc(hWnd, WM\_PAINT, NULL, NULL);

InvalidateRect(hWnd, NULL, TRUE);

break;

case ID\_BUTTON\_POLYLINE:

if (currentMode != POLYLINE)

points.clear();

currentMode = POLYLINE;

InvalidateRect(hWnd, NULL, true);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

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

}

break;

}

case WM\_LBUTTONDOWN:

{

int xPos = GET\_X\_LPARAM(lParam);

int yPos = GET\_Y\_LPARAM(lParam);

points.push\_back({xPos, yPos});

break;

}

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

if (points.size() < 2)

break;

for (int i = 0; i < points.size() - 1; i++)

{

MoveToEx(hdc, points[i].x, points[i].y, NULL);

LineTo(hdc, points[i + 1].x, points[i + 1].y);

}

POINT lastPoint = points[points.size() - 1];

if (currentMode == POLYGON)

{

MoveToEx(hdc, lastPoint.x, lastPoint.y, NULL);

LineTo(hdc, points[0].x, points[0].y);

}

EndPaint(hWnd, &ps);

break;

}

case WM\_CREATE:

{

HWND buttonPolyLine = CreateWindow(L"button",

L"Draw polyline",

WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

10,

10,

100,

30,

hWnd,

(HMENU)ID\_BUTTON\_POLYLINE,

hInst,

NULL);

HWND buttonPolygon = CreateWindow(L"button",

L"Draw polygon",

WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

120,

10,

100,

30,

hWnd,

(HMENU)ID\_BUTTON\_POLYGON,

hInst,

NULL);

break;

}

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

// Обработчик сообщений для окна "О программе".

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

INT Draw(HWND hWnd)

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

if (currentMode == POLYGON)

{

for (int i = 0; i < points.size(); i++)

{

MoveToEx(hdc, points[i].x, points[i].y, NULL);

LineTo(hdc, points[i].x, points[i].y);

}

if (points.size() > 1)

{

int x = points[0].x;

int y = points[0].y;

MoveToEx(hdc, x, y, NULL);

LineTo(hdc, x, y);

}

}

EndPaint(hWnd, &ps);

return 0;

# Результат выполнение:



