Системное программирование

Дедов Никита, 251003

Лабораторная №1

#include <windows.h>

#include <math.h>

#define M\_PI 3.14159265358979323846

struct Square {

int x;

int y;

int width;

int height;

};

struct Square square = { 100, 100, 50, 50 };

float angle = 0.0f;

POINT points[4];

void ChangePoints(int x, int y, int alpha) {

float rad = alpha \* M\_PI / 180.0f;

float rotMatrix[2][2] = { {cos(rad), -sin(rad)}, {sin(rad), cos(rad)} };

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

points[i].x += x;

points[i].y += y;

}

float CenterX = (points[0].x + points[1].x + points[2].x + points[3].x) / 4;

float CenterY = (points[0].y + points[1].y + points[2].y + points[3].y) / 4;

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

POINT p = { 0, 0 };

points[i].x -= CenterX;

points[i].y -= CenterY;

p.x = points[i].x \* rotMatrix[0][0] + points[i].y \* rotMatrix[0][1];

p.y = points[i].x \* rotMatrix[1][0] + points[i].y \* rotMatrix[1][1];

points[i].x = p.x + CenterX;

points[i].y = p.y + CenterY;

}

}

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

switch (message) {

case WM\_PAINT: {

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

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

SelectObject(hdc, brush);

points[0].x = square.x;

points[0].y = square.y;

points[1].x = square.x + square.width;

points[1].y = square.y;

points[2].x = square.x + square.width;

points[2].y = square.y + square.height;

points[3].x = square.x;

points[3].y = square.y + square.height;

ChangePoints(0, 0, angle);

Polygon(hdc, points, 4);

DeleteObject(brush);

EndPaint(hWnd, &ps);

return 0;

}

case WM\_KEYDOWN: {

switch (wParam) {

case VK\_LEFT:

case 'A':

if (square.x >= 10)

square.x -= 10; break;

case VK\_RIGHT:

case 'D':

if (square.x <= 420)

square.x += 10; break;

case VK\_UP:

case 'W':

if (square.y >= 10)

square.y -= 10; break;

case VK\_DOWN:

case 'S':

if (square.y <= 400)

square.y += 10; break;

case 'Q':

angle -= 10; break;

case 'E':

angle += 10; break;

}

InvalidateRect(hWnd, NULL, TRUE);

return 0;

}

case WM\_MOUSEWHEEL: {

int delta = GET\_WHEEL\_DELTA\_WPARAM(wParam);

if (delta > 0) {

if (GetKeyState(VK\_MENU) & 0x8000)

{

if (square.x <= 420)

square.x += 10;

}

else

{

if (square.y >= 10)

square.y -= 10;

}

}

else if (delta < 0) {

if (GetKeyState(VK\_MENU) & 0x8000)

{

if (square.x >= 10)

square.x -= 10;

}

else

{

if (square.y <= 400)

square.y += 10;

}

}

InvalidateRect(hWnd, NULL, TRUE);

return 0;

}

case WM\_SYSKEYDOWN:

if (wParam == 'Q') {

int result = MessageBox(hWnd, L"Are you sure you want to quit?", L"Quitting", MB\_YESNO | MB\_ICONQUESTION);

if (result == IDYES) {

PostQuitMessage(0);

}

}

return 0;

case WM\_CLOSE: {

int result = MessageBox(hWnd, L"Are you sure you want to quit?", L"Quitting", MB\_YESNO | MB\_ICONQUESTION);

if (result == IDYES) {

PostQuitMessage(0);

}

return 0;

}

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

default:

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

}

}

int APIENTRY WinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPTSTR lpCmdLine,

int nCmdShow) {

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(NULL, IDI\_APPLICATION);

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

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

wcex.lpszMenuName = NULL;

wcex.lpszClassName = L"MyWindowClass";

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

if (!RegisterClassEx(&wcex)) {

MessageBox(NULL, L"Не удалось зарегистрировать класс окна", L"Ошибка", MB\_OK | MB\_ICONERROR);

return 1;

}

HWND hWnd = CreateWindowEx(

0,

L"MyWindowClass",

L"SP Lab1",

WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT,

500, 500,

NULL, NULL, hInstance, NULL

);

if (!hWnd) {

MessageBox(NULL, L"Не удалось создать окно", L"Ошибка", MB\_OK | MB\_ICONERROR);

return 1;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

MSG msg;

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return (int)msg.wParam;

}