ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ МОЛДОВЫ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ, ИНФОРМАТИКИ И МИКРОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЕТ

По лабораторной работе №5

по Програмированию для Windows

Тема: Мышь

Выполнила: ст. гр. TI-155 Зверкова К.

Проверил: Скроб С.

Кишинев 2017

1. **Цель лабораторной работы**

Изучение основ и принципов работы с мышью.

1. **Теоретические понятия**

Вы можете определить наличие мыши с помощью функции *GetSystemMetrics*:

fMouse = GetSystemMetrics(SM\_MOUSEPRESENT);

Значение *fMouse* будет равным TRUE (ненулевым), если мышь установлена. Для определения количества кнопок установленной мыши используйте следующий вызов:

cButtons = GetSystemMetrics(SM\_CMOUSEBUTTONS);

Если мышь не инсталлирована, то возвращаемым значением этой функции будет 0.

В драйвере дисплея содержатся несколько ранее определенных курсоров мыши, которые могут использоваться в программах. Наиболее типичным курсором является наклонная стрелка, которая называется IDC\_ARROW и определяется в заголовочных файлах Windows. Вершина — это конец стрелки. Курсор IDC\_CROSS (используемый в приведенных в этой главе программах BLOKOUT) имеет вершину в центре крестообразного шаблона. Курсор IDC\_WAIT в виде песочных часов обычно используется программами для индикации того, что они чем-то заняты.

Программисты также могут спроектировать свои собственные курсоры (как это делается в главе 9). Курсор, устанавливаемый по умолчанию, для конкретного окна задается при определении структуры класса окна.

Например:

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

Для всех этих сообщений значение параметра *lParam* содержит положение мыши. Младшее слово — это координата *х*, а старшее слово — координата *y* относительно верхнего левого угла рабочей области окна. Вы можете извлечь координаты *х* и *y* из параметра *lParam* с помощью макросов LOWORD и HIWORD, определенных в заголовочных файлах Windows. Значение параметра *wParam* показывает состояние кнопок мыши и клавиш <Shift> и <Ctrl>. Вы можете проверить параметр *wParam* с помощью битовых масок, определенных в заголовочных файлах. Префикс MK означает "клавиша мыши" (mouse key).

Двойным щелчком мыши называются два, следующих один за другим в быстром темпе, щелчка мыши. Для того, чтобы два последовательных щелчка мыши считались двойным щелчком, они должны произойти в течение очень короткого промежутка времени, который называется "временем двойного щелчка" (double-click time). Если вы хотите, чтобы ваша оконная процедура получала сообщения двойного щелчка мыши, то вы должны включить идентификатор CS\_DBLCLKS при задании стиля окна в классе окна перед вызовом функции *RegisterClassEx* :

wndclass.style = CS\_HREDRAW | CS\_VREDRAW | CS\_DBLCLKS;

Если вы не включите CS\_DBLCLKS в стиль окна, и пользователь дважды в быстром темпе щелкнет левой кнопкой мыши, то ваша оконная процедура получит следующие сообщения: WM\_LBUTTONDOWN, WM\_LBUTTONUP, WM\_LBUTTONDOWN и WM\_LBUTTONUP. (Оконная процедура вполне может между этими сообщениями от кнопок мыши получать и другие сообщения.) Если вы хотите реализовать собственную логику обработки двойного

щелчка мыши, то для получения относительного времени сообщений WM\_LBUTTONDOWN, вы можете использовать функцию Windows *GetMessageTime*. Более подробно об этой функции рассказывается в главе 7. Если вы включаете в свой класс окна идентификатор CS\_DBLCLKS, то оконная процедура при двойном щелчке мыши получает следующие сообщения: WM\_LBUTTONDOWN, WM\_LBUTTONUP, WM\_LBUTTONDBLCLK и WM\_LBUTTONUP. Сообщение WM\_LBUTTONDBLCLK просто заменяет второе сообщение WM\_LBUTTONDOWN.

1. **Задание**
2. Модифицировать лабораторную работу 3 так, чтобы опорные точки кривой Безье можно было перемещать при помощи мыши, изменяя сплайн в реальном времени.
3. Изменить программу, созданную в лабораторной работе 4 таким образом, чтобы можно было перемещать рисунок с помощью мыши.
4. **Листинг программы**

1)

#include <windows.h>

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

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine, int iCmdShow) {

static char szAppName[] = "Bezier";

HWND hwnd;

MSG msg;

WNDCLASSEX wndclass;

wndclass.cbSize = sizeof(wndclass);

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

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;

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

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

wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);

wndclass.lpszMenuName = NULL;

wndclass.lpszClassName = (LPCWSTR)szAppName;

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

RegisterClassEx(&wndclass);

hwnd = CreateWindow((LPCWSTR)szAppName, "Bezier Splines",

WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT,

CW\_USEDEFAULT, CW\_USEDEFAULT,

NULL, NULL, hInstance, NULL);

ShowWindow(hwnd, iCmdShow);

UpdateWindow(hwnd);

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

void DrawBezier(HDC hdc, POINT pt[])

{

PolyBezier(hdc, pt, 4);

MoveToEx(hdc, pt[0].x, pt[0].y, NULL);

LineTo(hdc, pt[1].x, pt[1].y);

MoveToEx(hdc, pt[2].x, pt[2].y, NULL);

LineTo(hdc, pt[3].x, pt[3].y);

}

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

static POINT pt[4];

static POINT pt1[4];

HDC hdc;

PAINTSTRUCT ps;

HPEN hpen;

int cxClient, cyClient;

switch (iMsg) {

case WM\_CREATE:

hpen = CreatePen(PS\_SOLID, 2, RGB(255, 0, 0));

pt[0].x = 100;

pt[0].y = 100;

pt[1].x = 300;

pt[1].y = 200;

pt[2].x = 400;

pt[2].y = 300;

pt[3].x = 600;

pt[3].y = 700;

return 0;

case WM\_MOUSEMOVE:

hpen = CreatePen(PS\_SOLID, 2, RGB(255, 0, 0));

if (wParam & MK\_LBUTTON || wParam & MK\_RBUTTON)

{

hdc = GetDC(hwnd);

SelectObject(hdc, GetStockObject(WHITE\_PEN));

DrawBezier(hdc, pt);

if (wParam & MK\_LBUTTON)

{

pt[1].x = LOWORD(lParam);

pt[1].y = HIWORD(lParam);

}

if (wParam & MK\_RBUTTON)

{

pt[2].x = LOWORD(lParam);

pt[2].y = HIWORD(lParam);

}

SelectObject(hdc, GetStockObject(BLACK\_PEN));

DrawBezier(hdc, pt);

ReleaseDC(hwnd, hdc);

}

return 0;

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

}

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

}

2)

#include <windows.h>

#include <math.h>

#include <stdbool.h>

#define PI 3.14159265

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

HBITMAP hBitmap;

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine, int iCmdShow)

{

static char szAppName[] = "Lab5";

HWND hwnd;

MSG msg;

WNDCLASSEX wndclass;

wndclass.cbSize = sizeof(wndclass);

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

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;

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

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

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

wndclass.lpszMenuName = NULL;

wndclass.lpszClassName = szAppName;

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

RegisterClassEx(&wndclass);

hwnd = CreateWindow(

szAppName,

"Lab5",

WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

CW\_USEDEFAULT,

NULL,

NULL,

hInstance,

NULL

);

hBitmap = (HBITMAP)LoadImage(hInstance, "rudolph.bmp", IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);

ShowWindow(hwnd, iCmdShow);

UpdateWindow(hwnd);

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

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

{

HDC hdc;

PAINTSTRUCT ps;

RECT rect;

HDC hdcMem;

HGDIOBJ Bitmap;

HGDIOBJ hBrush;

static float angle, d;

static POINT p[3];

static int x, y;

switch (iMsg)

{

case WM\_CREATE:

d = sqrt(256 \* 256 + 256 \* 256) / 2;

x = 800;

y = 150;

angle = 0;

return 0;

case WM\_PAINT:

hdc = BeginPaint(hwnd, &ps);

GetClientRect(hwnd, &rect);

p[0].x = x + d\*cos(-0.75\*PI + angle);

p[0].y = y + d\*sin(-0.75\*PI + angle);

p[1].x = x + d\*cos(-0.25\*PI + angle);

p[1].y = y + d\*sin(-0.25\*PI + angle);

p[2].x = x + d\*cos(-1.25\*PI + angle);

p[2].y = y + d\*sin(-1.25\*PI + angle);

hdcMem = CreateCompatibleDC(hdc);

Bitmap = SelectObject(hdcMem, hBitmap);

PlgBlt(hdc, p, hdcMem, 0, 0, 256, 256, NULL, 0, 0);

SelectObject(hdcMem, Bitmap);

DeleteDC(hdcMem);

EndPaint(hwnd, &ps);

return 0;

case WM\_KEYDOWN:

switch (wParam) {

case VK\_F1:

angle -= PI / 18;

break;

case VK\_F2:

angle += PI / 18;

break;

case VK\_UP:

y -= 50;

break;

case VK\_DOWN:

y += 50;

break;

case VK\_LEFT:

x -= 50;

break;

case VK\_RIGHT:

x += 50;

break;

}

InvalidateRect(hwnd, NULL, true);

return 0;

case WM\_MOUSEMOVE:

if (wParam & MK\_LBUTTON)

{

if (wParam & MK\_LBUTTON)

{

x = LOWORD(lParam);

y = HIWORD(lParam);

}

InvalidateRect(hwnd, NULL, TRUE);

}

return 0;

case WM\_DESTROY:

PostQuitMessage(0);

return 0;

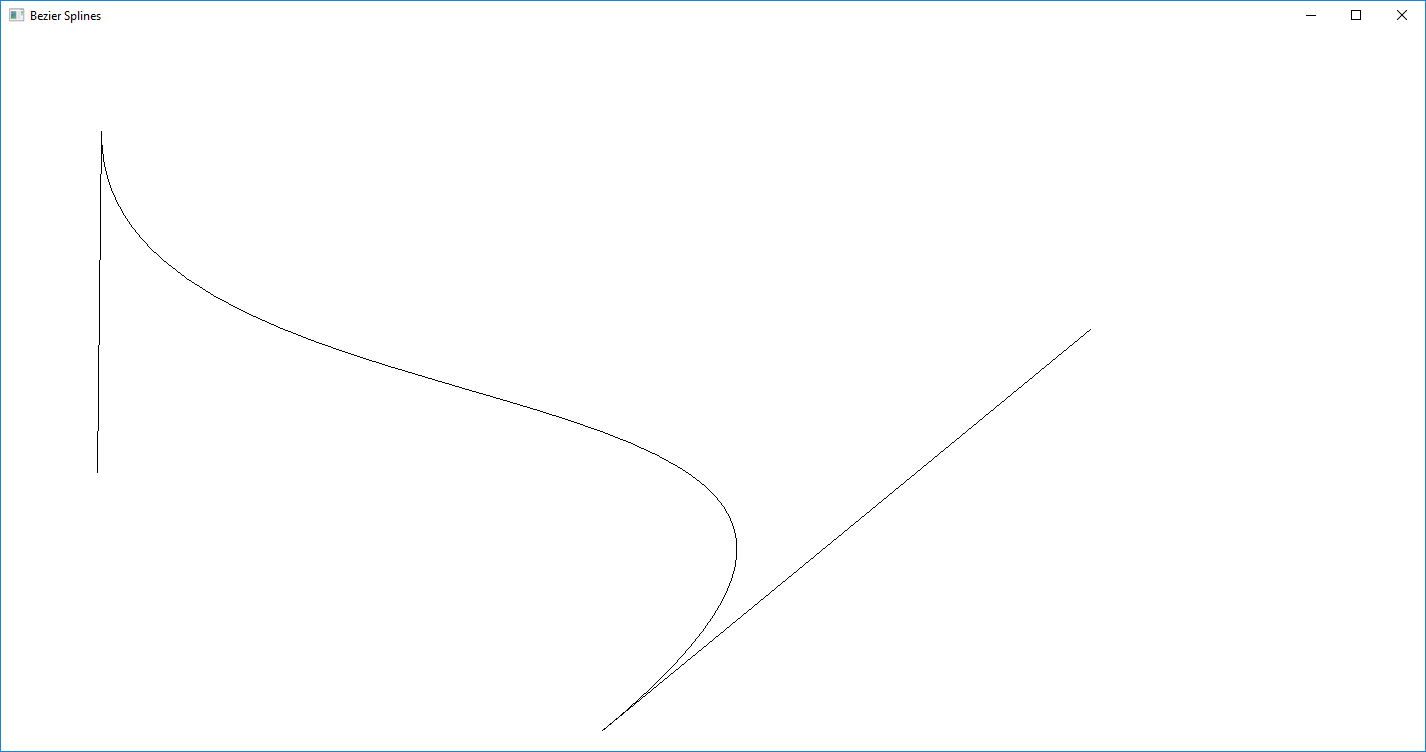
}

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

}

1. **Результат работы прогрммы**

1)



2)



**Вывод:**

В результате выполненной работы, были изучены принципы работы с мышью. Мышь - крайне важное устройство в составе ПК, поскольку вместе с клавиатурой постоянно используется для ввода информации и управления ею внутри ПК. Следовательно, удобство мыши, ее соответствие задачам пользователя имеет большое значение. В разработке под Windows очень важно освоить базовые навыки работы с мышью. Были созданы программы, позволяющие перемещать фигуру и перемещать опорные точки кривой Безье, которые доказали нам, что работа с мышью является не сложной и легко усваиваемой.