Практическая работа№9 Программирование мыши

Задание 1 Отслеживание курсора мышки

Частенько приложению требуется знать координаты курсора мышки. Обычно, это графические программы, которые отслеживают координаты курсора во время рисования какого-нибудь рисунка. Так же отслеживать положение мышки необходимо приложениям работающим с текстом для возможности выделения блоков текста.

Для того, чтобы отслеживать курсор мышки, обычно необходимо обработать три сообщения [WM\_LBUTTONDOWN](javascript:if(confirm('http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_LBUTTONDOWN.asp%20%20\n\nThis%20file%20was%20not%20retrieved%20by%20Teleport%20Pro,%20because%20it%20is%20addressed%20on%20a%20domain%20or%20path%20outside%20the%20boundaries%20set%20for%20its%20Starting%20Address.%20%20\n\nDo%20you%20want%20to%20open%20it%20from%20the%20server?'))window.location='http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_LBUTTONDOWN.asp'), [WM\_MOUSEMOVE](javascript:if(confirm('http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_MOUSEMOVE.asp%20%20\n\nThis%20file%20was%20not%20retrieved%20by%20Teleport%20Pro,%20because%20it%20is%20addressed%20on%20a%20domain%20or%20path%20outside%20the%20boundaries%20set%20for%20its%20Starting%20Address.%20%20\n\nDo%20you%20want%20to%20open%20it%20from%20the%20server?'))window.location='http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_MOUSEMOVE.asp'), и [WM\_LBUTTONUP](javascript:if(confirm('http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_LBUTTONUP.asp%20%20\n\nThis%20file%20was%20not%20retrieved%20by%20Teleport%20Pro,%20because%20it%20is%20addressed%20on%20a%20domain%20or%20path%20outside%20the%20boundaries%20set%20for%20its%20Starting%20Address.%20%20\n\nDo%20you%20want%20to%20open%20it%20from%20the%20server?'))window.location='http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_LBUTTONUP.asp'). Как правило, отслеживание курсора начинается с поступления сообщения WM\_LBUTTONDOWN, в параметре lParam которого записаны координаты курсора. Далее начинается сам процесс отслеживания путём обработки потока сообщений WM\_MOUSEMOVE которые постит само окно при перемещении мышки. Поступление сообщения WM\_LBUTTONUP сигнализирует об окончании процесса отслеживания.

Так же можно использовать функцию [TrackMouseEvent](javascript:if(confirm('http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputFunctions/TrackMouseEvent.asp%20%20\\n\\nThis%20file%20was%20not%20retrieved%20by%20Teleport%20Pro,%20because%20it%20is%20addressed%20on%20a%20domain%20or%20path%20outside%20the%20boundaries%20set%20for%20its%20Starting%20Address.%20%20\\n\\nDo%20you%20want%20to%20open%20it%20from%20the%20server?'))window.location='http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputFunctions/TrackMouseEvent.asp'), чтобы заставить систему посылать другие сообщения необходимые для отслеживания курсора. Сообщение [WM\_MOUSEHOVER](javascript:if(confirm('http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_MOUSEHOVER.asp%20%20\n\nThis%20file%20was%20not%20retrieved%20by%20Teleport%20Pro,%20because%20it%20is%20addressed%20on%20a%20domain%20or%20path%20outside%20the%20boundaries%20set%20for%20its%20Starting%20Address.%20%20\n\nDo%20you%20want%20to%20open%20it%20from%20the%20server?'))window.location='http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_MOUSEHOVER.asp') посылается системой когда мышка попадает в клиентскую область, а сообщение [WM\_MOUSELEAVE](javascript:if(confirm('http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_MOUSELEAVE.asp%20%20\n\nThis%20file%20was%20not%20retrieved%20by%20Teleport%20Pro,%20because%20it%20is%20addressed%20on%20a%20domain%20or%20path%20outside%20the%20boundaries%20set%20for%20its%20Starting%20Address.%20%20\n\nDo%20you%20want%20to%20open%20it%20from%20the%20server?'))window.location='http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_MOUSELEAVE.asp') - когда курсор покидает клиентскую область. Соответственно, сообщения [WM\_NCMOUSEHOVER](javascript:if(confirm('http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_NCMOUSEHOVER.asp%20%20\n\nThis%20file%20was%20not%20retrieved%20by%20Teleport%20Pro,%20because%20it%20is%20addressed%20on%20a%20domain%20or%20path%20outside%20the%20boundaries%20set%20for%20its%20Starting%20Address.%20%20\n\nDo%20you%20want%20to%20open%20it%20from%20the%20server?'))window.location='http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_NCMOUSEHOVER.asp') и [WM\_NCMOUSELEAVE](javascript:if(confirm('http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_NCMOUSELEAVE.asp%20%20\n\nThis%20file%20was%20not%20retrieved%20by%20Teleport%20Pro,%20because%20it%20is%20addressed%20on%20a%20domain%20or%20path%20outside%20the%20boundaries%20set%20for%20its%20Starting%20Address.%20%20\n\nDo%20you%20want%20to%20open%20it%20from%20the%20server?'))window.location='http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/UserInput/MouseInput/MouseInputReference/MouseInputMessages/WM_NCMOUSELEAVE.asp') отвечают за неклиентскую область.

Задание 2 Рисование линий при помощи мышки

В данном разделе статьи представлена часть кода оконной процедуры, которая позволяет пользователю рисовать линии в клиентской области окна путём перетаскивания мышки.

Когда оконная процедура получает сообщение WM\_LBUTTONDOWN, то происходит захват мышки и сохранение координат курсора, используя их как начальную точку линии. При этом используется функция [ClipCursor](javascript:if(confirm('http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/Resources/Cursors/CursorReference/CursorFunctions/ClipCursor.asp%20%20\\n\\nThis%20file%20was%20not%20retrieved%20by%20Teleport%20Pro,%20because%20it%20is%20addressed%20on%20a%20domain%20or%20path%20outside%20the%20boundaries%20set%20for%20its%20Starting%20Address.%20%20\\n\\nDo%20you%20want%20to%20open%20it%20from%20the%20server?'))window.location='http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/Resources/Cursors/CursorReference/CursorFunctions/ClipCursor.asp'), чтобы ограничить перемещение курсора клиентской областью в процессе рисования.

В течение первого сообщения WM\_MOUSEMOVE оконная процедура рисует линию от начальной точки до текущей позиции курсора. В течение последующих сообщений WM\_MOUSEMOVE, оконная процедура стирает предыдущую линию путём рисования повёрх неё линии инверсного цвета. Затем снова рисуется линия от начальной точки до текущих координат курсора.

Поступление сообщения WM\_LBUTTONUP сигнализирует об окончании рисования. Оконная процедура освобождает захват мышки и снимает ограничение движения мышки клиентской областью.

Пример:

LRESULT APIENTRY MainWndProc(HWND hwndMain, UINT uMsg,

WPARAM wParam, LPARAM lParam)

{

HDC hdc; // дескриптор контекста устройства

RECT rcClient; // прямоугольник клиентской области

POINT ptClientUL; // верхний левый угол клиент.области

POINT ptClientLR; // нижний правый угол клиент.области

static POINTS ptsBegin; // начальная точка

static POINTS ptsEnd; // новая конечная точка

static POINTS ptsPrevEnd; // предыдущая конечная точка

static BOOL fPrevLine = FALSE; // флаг предыдущей линии

switch (uMsg)

{

case WM\_LBUTTONDOWN:

// Захватываем мышку.

SetCapture(hwndMain);

// Получаем экранные координаты клиентской области,

// и преобразуем их в клиентские координаты.

GetClientRect(hwndMain, &rcClient);

ptClientUL.x = rcClient.left;

ptClientUL.y = rcClient.top;

// Добавляем один пиксель справа и снизу, так как координаты,

// полученные из GetClientRect не включают левого и

// нижнего пикселей.

ptClientLR.x = rcClient.right + 1;

ptClientLR.y = rcClient.bottom + 1;

ClientToScreen(hwndMain, &ptClientUL);

ClientToScreen(hwndMain, &ptClientLR);

// Копируем клиентские координаты клиентской области

// в структуру rcClient. Ограничиваем курсор мышки клиентской

// областью, передав структуру rcClient в

// функцию ClipCursor.

SetRect(&rcClient, ptClientUL.x, ptClientUL.y,

ptClientLR.x, ptClientLR.y);

ClipCursor(&rcClient);

// Преобразуем координаты курсора для структуры POINTS,

// которая определяет начальную точку рисования линии

// в течение сообщения WM\_MOUSEMOVE.

ptsBegin = MAKEPOINTS(lParam);

return 0;

case WM\_MOUSEMOVE:

// Чтобы рисовалась линия, то при движении мышки

// пользователь должен удерживать нажатой левую кнопку мышки.

if (wParam & MK\_LBUTTON)

{

// Получаем контекст устройства (DC) для клиентской области

hdc = GetDC(hwndMain);

// Следующая функция гарантирует, что пиксели

// предыдущей линии установлены в белый цвет, а

// вновь нарисованной линии - в чёрный.

SetROP2(hdc, R2\_NOTXORPEN);

// Если линия была нарисована в предыдущем WM\_MOUSEMOVE,

// то рисуем поверх неё. Тем самым, установив пиксели

// линии в белый цвет, мы сотрём её.

if (fPrevLine)

{

MoveToEx(hdc, ptsBegin.x, ptsBegin.y, (LPPOINT) NULL);

LineTo(hdc, ptsPrevEnd.x, ptsPrevEnd.y);

}

// Преобразуем текущие координаты курсора в структуру

// POINTS, а затем рисуем новую линию.

ptsEnd = MAKEPOINTS(lParam);

MoveToEx(hdc, ptsBegin.x, ptsBegin.y, (LPPOINT) NULL);

LineTo(hdc, ptsEnd.x, ptsEnd.y);

// Устанавливаем флаг предыдущей линии, сохраняем конечную

// точку новой линии, а затем освобождаем DC.

fPrevLine = TRUE;

ptsPrevEnd = ptsEnd;

ReleaseDC(hwndMain, hdc);

}

break;

case WM\_LBUTTONUP:

// Пользователь закончил рисовать линию. Сбрасываем флаг

// предыдущей линии, освобождаем курсор мышки и

// освобождаем захват мышки.

fPrevLine = FALSE;

ClipCursor(NULL);

ReleaseCapture();

return 0;

case WM\_DESTROY:

PostQuitMessage(0);

break;

// Обрабатываем другие сообщения.

Задание 3 Обработка двойного щелчка

Чтобы получать сообщения о двойном щелчке (double-click messages), класс окна должен содержать стиль [CS\_DBLCLKS](javascript:if(confirm('http://msdn.microsoft.com/library/en-us/winui/winui/windowsuserinterface/windowing/windowclasses/aboutwindow.asp%20%20\n\nThis%20file%20was%20not%20retrieved%20by%20Teleport%20Pro,%20because%20it%20is%20addressed%20on%20a%20domain%20or%20path%20outside%20the%20boundaries%20set%20for%20its%20Starting%20Address.%20%20\n\nDo%20you%20want%20to%20open%20it%20from%20the%20server?'))window.location='http://msdn.microsoft.com/library/en-us/winui/winui/windowsuserinterface/windowing/windowclasses/aboutwindow.asp#class_styles'). Этот стиль устанавливается при регистрации оконного класса, как показано ниже.

Пример:

BOOL InitApplication(HINSTANCE hInstance)

{

WNDCLASS wc;

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

wc.lpfnWndProc = (WNDPROC) MainWndProc;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hInstance = hInstance;

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

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

wc.hbrBackground = GetStockObject(WHITE\_BRUSH);

wc.lpszMenuName = "MainMenu";

wc.lpszClassName = "MainWClass";

return RegisterClass(&wc);

}

Сообщение о двойном щелчке всегда предшествует сообщению о нажатии кнопки.

Задание 4 Выделение строки текста

В данном разделе приведён пример, который был взят из обычного текстового редактора. Он включает код, позволяющий пользователю обычным щелчком устанавливать каретку в любом месте текста, а также выделять (подсвечивать) строку текста двойным щелчком.

Пример:

LRESULT APIENTRY MainWndProc(HWND hwndMain, UINT uMsg,

WPARAM wParam, LPARAM lParam)

{

HDC hdc; // дескриптор контекста устройства

TEXTMETRIC tm; // данные о размере шрифта

int i, j; // счётчики цикла

int cCR = 0; // счётчик возвратов каретки

char ch; // символ из буфера ввода

static int nBegLine; // начало выделенной линии

static int nCurrentLine = 0; // текущая выделенная строка

static int nLastLine = 0; // последняя строка текста

static int nCaretPosX = 0; // x-координата каретки

static int cch = 0; // количество введённых символов

static int nCharWidth = 0; // точная ширина символа

static char szHilite[128]; // строка текста, которая будет выделена

static DWORD dwCharX; // средняя ширина символов

static DWORD dwLineHeight; // высота строки

static POINTS ptsCursor; // координаты курсора мышки

static COLORREF crPrevText; // предыдущий цвет текста

static COLORREF crPrevBk; // предыдущий цвет фона

static PTCHAR pchInputBuf; // указатель на буфер ввода

static BOOL fTextSelected = FALSE; // флаг выделения текста

size\_t \* pcch;

HRESULT hResult;

switch (uMsg)

{

case WM\_CREATE:

// Получаем параметры текущего шрифта.

hdc = GetDC(hwndMain);

GetTextMetrics(hdc, &tm);

ReleaseDC(hwndMain, hdc);

// Сохраняем среднюю ширину и высоту символа.

dwCharX = tm.tmAveCharWidth;

dwLineHeight = tm.tmHeight;

// Выделяем буфер для хранения ввода с клавиатуры.

pchInputBuf = (LPSTR) GlobalAlloc(GPTR,

BUFSIZE \* sizeof(TCHAR));

return 0;

case WM\_CHAR:

switch (wParam)

{

case 0x08: // backspace

case 0x0A: // перевод строки

case 0x1B: // escape

MessageBeep( (UINT) -1);

return 0;

case 0x09: // символ табуляции (tab)

// Преобразуем символы табуляции в четыре пробела.

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

SendMessage(hwndMain, WM\_CHAR, 0x20, 0);

return 0;

case 0x0D: // возврат каретки

// Записываем символ возврата каретки и помещаем каретку

// в начало новой строки.

pchInputBuf[cch++] = 0x0D;

nCaretPosX = 0;

nCurrentLine += 1;

break;

default: // отображаемый символ

ch = (char) wParam;

HideCaret(hwndMain);

// Получаем ширину символа и отображаем его.

hdc = GetDC(hwndMain);

GetCharWidth32(hdc, (UINT) wParam, (UINT) wParam,

&nCharWidth);

TextOut(hdc, nCaretPosX,

nCurrentLine \* dwLineHeight, &ch, 1);

ReleaseDC(hwndMain, hdc);

// Сохраняем символ в буфере.

pchInputBuf[cch++] = ch;

// Вычисляем новую горизонтальную координат каретки.

// Если координата достигла максимума, то вставляем

// перевод каретки и перемещаем каретку

// в начало следующей строки.

nCaretPosX += nCharWidth;

if ((DWORD) nCaretPosX > dwMaxCharX)

{

nCaretPosX = 0;

pchInputBuf[cch++] = 0x0D;

++nCurrentLine;

}

ShowCaret(hwndMain);

break;

}

SetCaretPos(nCaretPosX, nCurrentLine \* dwLineHeight);

nLastLine = max(nLastLine, nCurrentLine);

break;

// Обрабатываем другие сообщения.

case WM\_LBUTTONDOWN:

// Если строка текста уже выделена, то перерисовываем

// текст, чтобы убрать выделение.

if (fTextSelected)

{

hdc = GetDC(hwndMain);

SetTextColor(hdc, crPrevText);

SetBkColor(hdc, crPrevBk);

hResult = StringCchLength(szHilite, 128/sizeof(TCHAR), pcch);

if (FAILED(hResult))

{

// TODO: обработчик ошибки

}

TextOut(hdc, 0, nCurrentLine \* dwLineHeight, szHilite, \*pcch);

ReleaseDC(hwndMain, hdc);

ShowCaret(hwndMain);

fTextSelected = FALSE;

}

// Сохраняем текущие координаты курсора мышки.

ptsCursor = MAKEPOINTS(lParam);

// Определяем, на какой строке находится курсор, и сохраняем

// номер строки. Следим, чтобы номера строк не были больше

// номера последней строки текста. Результат используем

// для установки y-координаты каретки.

nCurrentLine = min((int)(ptsCursor.y / dwLineHeight),

nLastLine);

// Парсим текст буфера ввода, чтобы найти первый символ

// в выделенной строке текста. Каждая строка оканчивается

// символом возврата каретки, поэтому, чтобы найти

// выделенную строку, достаточно сосчитать возвраты каретки.

cCR = 0;

nBegLine = 0;

if (nCurrentLine != 0)

{

for (i = 0; (i < cch) &&

(cCR < nCurrentLine); i++)

{

if (pchInputBuf[i] == 0x0D)

++cCR;

}

nBegLine = i;

}

// Начиная с начала выделенной строки, измеряем ширину

// каждого символа, суммируя с шириной уже измеренного

// символа. Останавливаемся,

// когда сумма больше, чем x-координата курсора.

// Сумма используется для установки x-координаты каретки.

hdc = GetDC(hwndMain);

nCaretPosX = 0;

for (i = nBegLine;

(pchInputBuf[i] != 0x0D) && (i < cch); i++)

{

ch = pchInputBuf[i];

GetCharWidth32(hdc, (int) ch, (int) ch, &nCharWidth);

if ((nCaretPosX + nCharWidth) > ptsCursor.x) break;

else nCaretPosX += nCharWidth;

}

ReleaseDC(hwndMain, hdc);

// Устанавливаем каретку в то место, куда кликнул пользователь.

SetCaretPos(nCaretPosX, nCurrentLine \* dwLineHeight);

break;

case WM\_LBUTTONDBLCLK:

// Копируем выделенную строку в буфер.

for (i = nBegLine, j = 0; (pchInputBuf[i] != 0x0D) &&

(i < cch); i++)

{

szHilite[j++] = pchInputBuf[i];

}

szHilite[j] = '\0';

// Скрываем каретку, инвертируем цвет фона и символов,

// а затем перерисовываем выделенную строку.

HideCaret(hwndMain);

hdc = GetDC(hwndMain);

crPrevText = SetTextColor(hdc, RGB(255, 255, 255));

crPrevBk = SetBkColor(hdc, RGB(0, 0, 0));

hResult = StringCchLength(szHilite, 128/sizeof(TCHAR), pcch);

if (FAILED(hResult))

{

// TODO: обработчик ошибки

}

TextOut(hdc, 0, nCurrentLine \* dwLineHeight, szHilite, \*pcch);

SetTextColor(hdc, crPrevText);

SetBkColor(hdc, crPrevBk);

ReleaseDC(hwndMain, hdc);

fTextSelected = TRUE;

break;

// Обрабатываем другие сообщения.

default:

return DefWindowProc(hwndMain, uMsg, wParam, lParam);

}

return NULL;

}

Задание 5 Использование колеса мышки в документах с встраиваемыми объектами

В этом разделе представлен пример, демонстрирующий работу с документом Microsoft® Word, с различными встраиваемыми объектами:

Таблица Microsoft Excel

Элемент управления list box, который скроллируется в ответ на вращение колёсика

Элемент управления text box, который не реагирует на колёсико

Сообщение MSH\_MOUSEWHEEL всегда посылается главному окну в Microsoft Word, даже если активна встраиваемая таблица экселя. Следующая таблица объясняет, как сообщение MSH\_MOUSEWHEEL обрабатывается в ответ на изменение фокуса.

Фокус на Обрабатывается следующим образом

документе Word Word скроллирует окно документа.

внедрённой таблице Excel Word постит сообщение в Excel. Вы должны решить, должно ли внедрённое приложение реагировать на сообщение или нет.

внедрённом элементе управления Сперва приложение посылает сообщение внедрённому контролу, который имеет фокус, и проверяет код возврата, чтобы узнать, обработал ли это сообщение внедрённый элемент управления. Если элемент управления не обработал сообщение, то приложение начнёт скроллировать окно всего документа. Например, если пользователь кликает по списку (list box), а затем начинает вращать колёсико, то список будет скроллироваться в соответствии с вращением колеса. Если пользователь кликнет в текстовое окно, а затем будет прокручивать колёсико, то будет скроллироваться весь документ.

Следующий пример демонстрирует, как приложение может обработать два сообщения от колёсика.

Пример:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* эта часть кода работает с MSH\_MOUSEWHEEL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "zmouse.h"

//

// Самое главное, это определеить, поддерживается ли в системе

// сообщение WM\_MOUSEWHEEL.

//

#ifndef WM\_MOUSEWHEEL

#define WM\_MOUSEWHEEL WM\_MOUSELAST+1

// ID сообщения для колёсика IntelliMouse

#endif

UINT uMSH\_MOUSEWHEEL = 0; // Значение, возвращённое функцией

// RegisterWindowMessage()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

INT WINAPI WinMain(

HINSTANCE hInst,

HINSTANCE hPrevInst,

LPSTR lpCmdLine,

INT nCmdShow)

{

MSG msg;

BOOL bRet;

if (!InitInstance(hInst, nCmdShow))

return FALSE;

//

// Новые IntelliMouse используют зарегистрированное сообщение

// для передачи информации о вращении колеса. Зарегистрируем!

uMSH\_MOUSEWHEEL =

RegisterWindowMessage(MSH\_MOUSEWHEEL);

if ( !uMSH\_MOUSEWHEEL )

{

MessageBox(NULL,"

RegisterWindowMessag Failed!",

"Error",MB\_OK);

return msg.wParam;

}

while (( bRet = GetMessage(&msg, NULL, 0, 0)) != 0)

{

if (bRet == -1)

{

// обработка ошибки и возможный выход

}

else

{

if (!TranslateAccelerator(ghwndApp,

ghaccelTable,

&msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

}

return msg.wParam;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Следующий код показывает как работать с WM\_MOUSEWHEEL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

LONG APIENTRY MainWndProc(

HWND hwnd,

UINT msg,

WPARAM wParam,

LPARAM lParam)

{

static int nZoom = 0;

switch (msg)

{

case WM\_MOUSEWHEEL:

((short) HIWORD(wParam)< 0) ? nZoom-- : nZoom++;

//

// Как-нибудь работаем с колёсиком...

//

break;

default:

//

// uMSH\_MOUSEWHEEL это сообщение, зарегистрированное

// ддл-кой mswheel в версиях Windows, которые не

// поддерживают новые сообщения в системе.

if( msg == uMSH\_MOUSEWHEEL )

{

((int)wParam < 0) ? nZoom-- : nZoom++;

//

// Как-нибудь работаем с колёсиком...

//

break;

}

return DefWindowProc(hwnd,

msg,

wParam,

lParam);

}

return 0L;

}

Задание 6 Получаем количество строк, проскроллированных колесом мышки

Следующий пример, позволяет узнать количество проскроллированных строк. Для тех операционных систем, которые изначально поддерживают колёсико мышки, такие как Microsoft Windows NT® 4.0 и выше, рекомендуется использовать SystemParametersInfo.

Пример:

/\* SPI\_GETWHEELSCROLLLINES

определена в winuser.h начиная с Windows NT 4.0. Для того, чтобы

иметь возможность узнать кол-во проскроллированных строк была

обновлена функция SystemParametersInfo.

\*/

#ifndef SPI\_GETWHEELSCROLLLINES

#define SPI\_GETWHEELSCROLLLINES 104

#endif

#include "zmouse.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* ФУНКЦИЯ: GetNumScrollLines

\* Описание: Системно-независимый способ получения количества

\* строк, проскроллированных колесом мышки

\* Параметры: нет

\* Возвращает : UINT: Кол-во строк, где WHEEL\_PAGESCROLL

\* указывает на то, что в данный момент идёт скроллирование.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

UINT GetNumScrollLines(void)

{

HWND hdlMsWheel;

UINT ucNumLines=3; // 3 по умолчанию

OSVERSIONINFO osversion;

UINT uiMsh\_MsgScrollLines;

memset(&osversion, 0, sizeof(osversion));

osversion.dwOSVersionInfoSize =sizeof(osversion);

GetVersionEx(&osversion);

// В Windows 9x & Windows NT 3.51, для получения количества строк

// используется MSWheel. В Windows NT 4.0 и выше, для этой цели

// используется SystemParametersInfo.

if ((osversion.dwPlatformId ==

VER\_PLATFORM\_WIN32\_WINDOWS) ||

( (osversion.dwPlatformId ==

VER\_PLATFORM\_WIN32\_NT) &&

(osversion.dwMajorVersion < 4) ) )

{

hdlMsWheel = FindWindow(MSH\_WHEELMODULE\_CLASS,

MSH\_WHEELMODULE\_TITLE);

if (hdlMsWheel)

{

uiMsh\_MsgScrollLines = RegisterWindowMessage

(MSH\_SCROLL\_LINES);

if (uiMsh\_MsgScrollLines)

ucNumLines = (int)SendMessage(hdlMsWheel,

uiMsh\_MsgScrollLines,

0,

0);

}

}

else if ( (osversion.dwPlatformId ==

VER\_PLATFORM\_WIN32\_NT) &&

(osversion.dwMajorVersion >= 4) )

{

SystemParametersInfo(SPI\_GETWHEELSCROLLLINES,

0,

&ucNumLines, 0);

}

return(ucNumLines);

}