ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ МОЛДОВЫ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ, ИНФОРМАТИКИ И МИКРОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЕТ

По лабораторной работе №4

по Програмированию для Windows

Тема: Клавиатура

Выполнила: ст. гр. TI-155 Зверкова К.

Проверил: Скроб С.

Кишинев 2017

1. **Цель лабораторной работы**

Изучение основ и принципов работы с клавиатурой.

1. **Теоретические понятия**

Клавиатура должна разделяться между всеми приложениями, работающими под Windows. Некоторые приложения могут иметь больше одного окна, и клавиатура должна разделяться между этими окнами в рамках одного и того же приложения. Когда на клавиатуре нажата клавиша, только одна оконная процедура может получить сообщение об этом. Окно, которое получает это сообщение клавиатуры, является окном, имеющем "фокус ввода" (input focus). Концепция фокуса ввода тесно связана с концепцией "активного окна". Окно, имеющее фокус ввода — это либо активное окно, либо дочернее окно активного окна. Определить активное окно обычно достаточно просто. Если у активного окна имеется панель заголовка, то Windows выделяет ее. Если у активного окна вместо панели заголовка имеется рамка диалога (это наиболее часто встречается в окнах диалога), то Windows выделяет ее. Если активное окно минимизировано, то Windows выделяет текст заголовка в панели задач.

Наиболее часто дочерними окнами являются кнопки, переключатели, флажки, полосы прокрутки и списки, которые обычно присутствуют в окне диалога. Сами по себе дочерние окна никогда не могут быть активными. Если фокус ввода находится в дочернем окне, то активным является родительское окно этого дочернего окна. То, что фокус ввода находится в дочерних окнах, обычно показывается посредством мигающего курсора или каретки (caret). Если активное окно минимизировано, то окна с фокусом ввода нет. Windows продолжает слать программе

сообщения клавиатуры, но эти сообщения выглядят иначе, чем сообщения, направленные активным и еще не минимизированным окнам. Обрабатывая сообщения WM\_SETFOCUS и WM\_KILLFOCUS, оконная процедура может определить, когда окно имеет фокус ввода. WM\_SETFOCUS показывает, что окно получило фокус ввода, а WM\_KILLFOCUS, что окно

потеряло его.

Параметр *lParam*

Для всех аппаратных сообщений клавиатуры, 32-разрядная переменная *lParam*, передаваемая в оконную процедуру, состоит из шести полей: счетчика повторений (Repeat Count), cкан-кода OEM (Original Equipment Manufacturer Scan Code), флага расширенной клавиатуры (Extended Key Flag), кода контекста (Context Code), флага предыдущего состояния клавиши (Previous Key State) и флага состояния клавиши (Transition State).

Параметр *wParam*.

В этом параметре содержится "виртуальный код клавиши" (virtual key code), идентифицирующий нажатую или отпущенную клавишу. Разработчики Windows попытались определить виртуальные клавиши независимым от аппаратуры способом. По этой причине, некоторые виртуальные коды клавиш не могут вырабатываться на персональных компьютерах IBM и совместимых с ними, но их можно встретить на клавиатурах других производителей.

Скан-код OEM является кодом клавиатуры, генерируемым аппаратурой компьютера. (Скан-код — это код, передаваемый программе в регистре AH при вызове прерывания 16H BIOS.) Приложения Windows обычно игнорируют скан-код OEM, поскольку имеют более совершенные способы расшифровки информации от клавиатуры.

Код контекста устанавливается в 1, если нажата клавиша <Alt>. Этот разряд всегда равен 1 для сообщений

WM\_SYSKEYDOWN и WM\_SYSKEYUP и 0 для сообщений WM\_KEYDOWN и WM\_KEYUP с двумя

исключениями:

• Если активное окно минимизировано, оно не имеет фокус ввода. Все нажатия клавиш вырабатывают

сообщения WM\_SYSKEYDOWN и WM\_SYSKEYUP. Если не нажата клавиша <Alt>, поле кода контекста

устанавливается в 0. (Windows использует SYS сообщения клавиатуры так, чтобы активное окно, которое

минимизировано, не обрабатывало эти сообщения.)

• На некоторых иноязычных клавиатурах некоторые символы генерируются комбинацией клавиш <Shift>, <Ctrl> или <Alt> с другой клавишей. В этих случаях, у переменной *lParam*, которая сопровождает

сообщения WM\_KEYDOWN и WM\_KEYUP, в поле кода контекста ставится 1, но эти сообщения не

являются системными сообщениями клавиатуры.

Вы можете получить текущее состояние любой виртуальной клавиши с помощью функции

*GetKeyState*. Эта функция в основном используется для получения информации о состоянии клавиш сдвига

(<Shift>, <Ctrl> и <Alt>) и клавиш-переключателей (<CapsLock>, <NumLock> и <ScrollLock>).

Например: GetKeyState(VK\_SHIFT); возвращает отрицательное значение (т. е., установлен старший разряд), если клавиша <Shift> нажата. В

возвращаемом функцией: GetKeyState(VK\_CAPITAL); значении установлен младший разряд, если переключатель <CapsLock> включен. Вы также можете получить положение кнопок мыши с помощью виртуальных кодов клавиш VK\_LBUTTON, VK\_MBUTTON и VK\_RBUTTON. Однако, большая часть программ для Windows, которым надо отслеживать сочетание состояний кнопок мыши и клавиш клавиатуры, делают это другим способом — проверяя состояние клавиш клавиатуры при получении сообщения от мыши. Фактически, информация о положении клавиш сдвига и клавиш-переключателей

включается в сообщения от мыши (как вы увидите в следующей главе).

1. **Задание**

Написатть программу с использованем GDI, при запуске которой в окне программы будет выведен графический рисунок. Путём нажатия клавиш (вверх, вниз, вправо, влево) необходимо перемещать фигуры. Также необходимо предусмотреть клавиши для вращения фигур.

1. **Листинг программы**

#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[] = "Lab4";

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,

"Lab4",

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\_DESTROY:

PostQuitMessage(0);

return 0;

}

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

}

1. **Результат работы прогрммы**



**Вывод:**

В результате выполненной работы, были изучены принципы работы с клавиатурой. Клавиатура позволяет пользователю легче работать с программным обеспечением, т.к. мануальное управление является очень удобным инструментом в информационных технологиях. С помощью клавиатуры обработчик обладает практически безграничными возможностями в разработке программного продукта. Несмотря на то, что сейчас развито сенсорное управление, клавиатура остаётся базовым звеном в управлении операционной системы персонального компьютера. В разработке под Windows очень важно освоить базовые навыки работы с клавиатурой. Была создана программа позволяющая перемещать и вращать фигуру, которая доказала нам, что работа с клавиатурой оказалась не сложной и легко усваиваемой.