Michał Muzyk 218332 Data: 8.01.2017 r.

Jakub Jaskóła 218349

Grupa: Wtorek TN 13:15-16:15

Prowadzący: Dr inż. Jan Nikodem

**Urządzenia peryferyjne**

**Obsługa joysticka USB z wykorzystaniem DirectInput**

**1. Zadania do wykonania**

Celem ćwiczenia było napisanie aplikacji, która umożliwi obsługę joysticka podłączonego do komputera za pomocą interfejsu USB.  
  
**2. Wstęp teoretyczny**

**Joystick** to urządzenie wejścia komputera, manipulator służący do sterowania ruchem obiektów na ekranie. W podstawowej wersji składa się z wychylnego drążka zamocowanego na podstawce, którego przechylenie w odpowiednim kierunku powoduje stosowną reakcję sterowanego obiektu, oraz z umieszczonych na drążku i podstawce przycisków uruchamiających przypisane im działania i dodatkowe funkcje sterujące. Pierwsze dżojstiki nie służyły do rozrywki, lecz do sterowania samolotami; nazywano je wtedy „drążkami sterowniczymi”. Joysticki stosuje się również do sterowania robotami i elektrycznymi wózkami inwalidzkimi. W komputerach wykorzystywane są do sterowania kursorem oraz samolotami, samochodami czy postaciami w grach.  
 Do wykonania naszego programu obsługującego joystick wykorzystaliśmy bibliotekę DirectInput.  
 **DirectInput** jest to biblioteka obsługująca urządzenia wejściowe tj. klawiatura, mysz czy joystick, będąca częścią DirectX firmy Microsoft. Zawiera w sobie funkcje odczytujące dane z tych urządzeń w różny sposób (bezpośredni lub buforowany). Ma też możliwość przyporządkowania określonych akcji do konkretnych przycisków (Action Mapping). DirectInput nie przyniesie żadnych dodatkowych korzyści aplikacjom, które używają klawiatury do wprowadzania tekstu i myszy do nawigacji. API jest przeznaczone głównie do tworzenia gier komputerowych, symulacji i wszystkich innych interaktywnych aplikacji dla Windows.  
W projekcie wykorzystywaliśmy funkcje znajdujące się w plikach nagłówkowych:  
\* windows.h – użyty, aby uzyskać dostęp do uchwytów (HWND)  
\* dinput.h – użyty, aby uzyskać dostęp np. do struktury opisującej status joysticka (DIJOYSTATE)  
  
**3. Opisany kod programu**  
  
#include <windows.h>

#include <iostream>

#pragma comment(lib, "dinput8.lib") **//biblioteka directx**

#pragma comment(lib, "dxguid.lib") **//biblioteka directx**

#include <dinput.h>

using namespace std;

LPDIRECTINPUT8 DirectInput;

LPDIRECTINPUTDEVICE8 GamePad;

**// Odpowiedz zwrotna Joysticka**

BOOL CALLBACK enumCallback(const DIDEVICEINSTANCE\* instance, VOID\* context)

{

HRESULT Pom;

**// Tworzenie i inicjalizacja urzadzenia**

Pom = DirectInput->CreateDevice(instance->guidInstance, &GamePad, NULL);

**// Nie udalo sie pobrac GamePada, nie zostal on znaleziony jako urzadzenie podlaczone do komputera**

if (FAILED(Pom)) {

return DIENUM\_CONTINUE;

}

return DIENUM\_STOP;

}

**// Do zwracania wartosci Osi X i Y**

BOOL CALLBACK enumAxesCallback(const DIDEVICEOBJECTINSTANCE\* instance, VOID\* context)

{

HWND hDlg = (HWND)context;

DIPROPRANGE propRange;

propRange.diph.dwSize = sizeof(DIPROPRANGE); **// rozmiar załączonej struktury**

propRange.diph.dwHeaderSize = sizeof(DIPROPHEADER); **// rozmiar struktury**

propRange.diph.dwHow = DIPH\_BYID; **// Identyfikator ten jest zwracany w elemencie dwType struktury DIDEVICEOBJECTINSTANCE**

propRange.diph.dwObj = instance->dwType; **// Wartość ustawiona**

propRange.lMin = -1000; **// dolna granica zasiegu**

propRange.lMax = +1000; **// gorna granica zasiegu**

**// Zakres układu**

if (FAILED(GamePad->SetProperty(DIPROP\_RANGE, &propRange.diph))) {

return DIENUM\_STOP;

}

return DIENUM\_CONTINUE;

}

**// Funkcja poll potrzebna nam jest do uzyskania informacji na temat stanu naszych przyciskow i naszej pozycji(joysticka) w przestrzeni**

HRESULT poll(DIJOYSTATE \*js)

{

HRESULT Pom;

if (GamePad == NULL) {

return S\_OK;

}

**// Odczyt joysticka (czy mozemy go dalej uzywac)**

Pom = GamePad->Poll();

if (FAILED(Pom)) {

**// Przywrocenie polaczenia jesli strumien input zostal przerwany**

Pom = GamePad->Acquire();

while (Pom == DIERR\_INPUTLOST) {

Pom = GamePad->Acquire();

}

**// Na wypadek fatal errora**

if ((Pom == DIERR\_INVALIDPARAM) || (Pom == DIERR\_NOTINITIALIZED)) {

return E\_FAIL;

}

**// Jesli inna aplikacja ma kontrole nad joystickiem, jest ok, polaczenie zostanie przywrocone gdy joystick sie zwolni**

if (Pom == DIERR\_OTHERAPPHASPRIO) {

return S\_OK;

}

}

**// Odczyt stanu urzadzenia**

if (FAILED(Pom = GamePad->GetDeviceState(sizeof(DIJOYSTATE), js))) {

return Pom;

}

return S\_OK;

}

int main() {

HRESULT Pom;

**// Inicjalizacja urzadzenia DirectInput**

if (FAILED(Pom = DirectInput8Create(GetModuleHandle(NULL),

DIRECTINPUT\_VERSION, IID\_IDirectInput8, (VOID\*\*)&DirectInput, NULL))) {

return 1;

}

**// Szukanie pierwszego GamePada**

if (FAILED(Pom = DirectInput->EnumDevices(DI8DEVCLASS\_GAMECTRL, enumCallback,

NULL, DIEDFL\_ATTACHEDONLY))) {

return 1;

}

**// Nie znaleziono GamePada**

if (GamePad == NULL) {

cout<<"Nie znaleziono urzadzenia " << endl;

system("pause");

return E\_FAIL;

}

**// Nazwa Pada i inne informacje o nim dostpne w strukturze DIDEVICEINSTANCE**

else {

cout << "Znaleziono GamePad!" <<endl;

DIDEVICEINSTANCE info;

info.dwSize = sizeof(DIDEVICEINSTANCE);

GamePad->GetDeviceInfo(&info);

wchar\_t\* NazwaPada = info.tszProductName; **// pobranie nazwy joysticka**

wprintf(NazwaPada);

cout << endl;

}

DIDEVCAPS capabilities; **//struktura opisujaca mozliwosci urzadzenia**

if (FAILED(Pom = GamePad->SetDataFormat(&c\_dfDIJoystick))) {

return 1;

}

**// sprawdzanie ile osi ma joystick**

capabilities.dwSize = sizeof(DIDEVCAPS);

if (FAILED(Pom = GamePad->GetCapabilities(&capabilities))) {

return 1;

}

if (FAILED(Pom = GamePad->EnumObjects(enumAxesCallback, NULL, DIDFT\_AXIS))) {

return 1;

}

DIJOYSTATE state;

POINT cursor;

int klik = -1;

**// Oczekiwanie na akcje uzytkownika, czyli klik klawisza, czy zmiana kursora(przesuniecie)**

cout << "Uzyj joystick" <<endl;

int koniec = 0;

BlockInput(true);

while (!poll(&state)) {

for (int i = 0; i < 32; i++) {

if (state.rgbButtons[klik] == 0) {

**// Podniesienie lewego przycisku myszy do gory**

if (klik == 0) {

GetCursorPos(&cursor);

mouse\_event(MOUSEEVENTF\_LEFTUP, cursor.x, cursor.y, 0, 0);

}

klik = -1;

}

if (state.rgbButtons[i] && klik != i) {

klik = i;

if (i == 0) {

**// Wcisniecie i przytrzymanie lewego przycisku myszy**

GetCursorPos(&cursor);

mouse\_event(MOUSEEVENTF\_LEFTDOWN, cursor.x, cursor.y, 0, 0);

}

**// Prawy przycisk Myszy**

else if (i == 1) {

GetCursorPos(&cursor);

mouse\_event(MOUSEEVENTF\_RIGHTUP, cursor.x, cursor.y, 0, 0);

}

**// Wyjscie po wcisnieciu przycisku numer 3**

else if (i == 2) {

koniec++;

}

}

}

GetCursorPos(&cursor);

SetCursorPos(cursor.x + state.lX / 100, cursor.y + state.lY / 100);

**// Czekanie na ponowienie akcji w celu ustalenia stanu i zapewnienia plynnosci**

Sleep(10);

if (koniec != 0) {

break;

}

}

BlockInput(0);

**// Odlaczenie joysticka**

if (GamePad) {

GamePad->Unacquire();

}

system("pause");

return 1;}

**4. Wnioski**W wyniku realizacji zadań laboratoryjnych napisaliśmy program, który umożliwiał: wykrycie aktywnych (podłączonych do komputera) joysticków USB, obsługa drążka oraz przycisków joysticka.