Paweł Twardawa 235072 Data: 23.10.2018 r.

Aleksandra Wieczorkiewicz 234980

Urządzenia peryferyjne

Ćwiczenie 11

Obsługa karty muzycznej z wykorzystaniem DirectSound, API i ActiveX

Prowadzący:

Dr inż. Jacek Mazurkiewicz

1. **Zadania do wykonania**
   1. Odtworzyć dźwięk za pomocą:

* ActiveX
* Wavefrom
* DirectSound
* PlaySound
  1. Napisać program:
* zczytujący nagłówek WAVa i wyświetlić poszczególne wartości
* odtwarzający wybrany plik w formacie MP3
* zapisujący dane z mikrofonu do pliku

1. **Opis programu**
   1. **Funkcja main()**Główna funkcja programu zawiera pętlę nieskończoną, w której można wybrać czynności do wykonania tj. wyświetlanie nagłówka pliku WAV oraz odtwarzanie i zatrzymywanie pliku. Po wyświetleniu menu programu należy wpisać numer wybranej opcji i zatwierdzić ją klawiszem Enter.

Listing 1.

int main()

{

int ch;

// char nazwa[100];

//char \* nazwa = {"a.wav\0"};

while (true) {

cout << " == DOSTEPNE FUNKCJE ==\n";

cout << "[1] Odtwarzaj PlaySond()\n";

cout << "[2] stop playSound\n";

cout << "[3] Zczytaj naglowek\n";

cout << "Wybor: ";

cin >> ch;

switch (ch)

{

case 1:

PlaySound(TEXT("b.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

break;

case 2:

PlaySound(NULL, NULL, SND\_ASYNC);

break;

case 3:

ZczytajNaglowek("b.wav");

break;

case 4:

//waveForm();

break;

}

}

return 0;

}

* 1. **Odtwarzanie za pomocą PlaySound().**  
     Wykorzystanie funkcji PlaySound() to najprostsza metoda odtwarzania dźwięku. Jako argument funkcji należy podać nazwę pliku. Użycie funkcji zostało przedstawione w listingu 2. Aby zatrzymać odtwarzany dźwięk należy użyć funkcji Stop() przedstawionej w listingu 3.

Listing 2.

void Play(LPCSTR name)

{

PlaySound(name, NULL, SND\_FILENAME | SND\_ASYNC);

}

Listing 3.

void Stop()

{

PlaySound(NULL, NULL, SND\_ASYNC);

}

* 1. **Odtwarzanie DirectSound**Aby odtworzyć dźwięk za pomocą DirectSound należy utworzyć interfejs za pomocą metody DirectSoundCreate() następnie stworzyć obiekt DirectSoundBuffer. Po zapisaniu danych do bufora należy użyć metody Play() która umożliwia odtworzenie dźwięku.  
     Listing 4.

bool odtwarzajDirectSound(LPCSTR nazwa) {

HRESULT result;

CoInitialize(NULL);

result = DirectSoundCreate(NULL, &lpDSO, NULL);

DSBUFFERDESC BufferDesc;

LPDIRECTSOUNDBUFFER DSBuffer;

memset(&BufferDesc, 0, sizeof(DSBUFFERDESC));

BufferDesc.dwSize = sizeof(DSBUFFERDESC);

BufferDesc.dwFlags = DSBCAPS\_PRIMARYBUFFER;

BufferDesc.dwBufferBytes = 0;

BufferDesc.lpwfxFormat = NULL;

lpDSO->SetCooperativeLevel(GetDesktopWindow(), DSSCL\_PRIORITY);

if (DS\_OK != lpDSO->CreateSoundBuffer(&BufferDesc, &DSBuffer, NULL))

cout << "Blad CreateSoundBuffer" << endl;

LPVOID lpvWrite;

DWORD dwLength;

LPVOID lpvWrite2;

DWORD dwLength2;

int tmp = 5;

tmp = DSBuffer->Lock(0, 0, &lpvWrite, &dwLength, NULL, NULL, DSBLOCK\_ENTIREBUFFER);

if (tmp == DS\_OK)

{

dwLength = header.dwBufferLength;

}

else if (tmp == DSERR\_PRIOLEVELNEEDED)

cout << tmp << endl;

DSBuffer->Unlock(lpvWrite, dwLength, NULL, 0);

if (DS\_OK == lpDSO->CreateSoundBuffer(&BufferDesc, &DSBuffer, NULL)) {

}

DSBuffer->SetFormat(wav);

DSBuffer->SetCurrentPosition(0);

DSBuffer->SetVolume(DSBVOLUME\_MAX);

if (DSBuffer->Play(0, 0, DSBPLAY\_LOOPING) != DS\_OK) {

cout << "BLAD" << endl;

}

return true;

}

* 1. **Zczytywanie nagłówka pliku WAV i wyświetlanie jego zawartości.**

Zczytując nagłówek pliku zgodnie z jego strukturą możemy odczytać szczegółowe dane o odtwarzanym pliku tj. liczba kanałów, częstotliwość próbkowania, format, ilość bitów na próbkę. Czytanie nagłówka oraz wyświetlanie go zostało przedstawione w Listingu 4.

Listing 5.

void ZczytajNaglowek(char \* nazwa)

{

FILE\* fptr;

fptr = fopen(nazwa, "r");

struct Wav

{

char ChunkID[4];

int ChunkSize;

char Format[4];

char Subchunk1ID[4];

int Subchunk1Size;

short AudioFormat;

short NumChannels;

int SampleRate;

int ByteRate;

short BlockAlign;

short BitsPerSample;

char Subchunk2ID[4];

int Subchunk2Size;

} HeaderInfo;

fread(&HeaderInfo, sizeof(Wav), 1, fptr);

cout << "Chunk ID: " << HeaderInfo.ChunkID[0] << HeaderInfo.ChunkID[1] << HeaderInfo.ChunkID[2] << HeaderInfo.ChunkID[3] << endl;

cout << "Chunk Size: " << HeaderInfo.ChunkSize << endl;

cout << "Format: " << HeaderInfo.Format[0] << HeaderInfo.Format[1] << HeaderInfo.Format[2] << HeaderInfo.Format[3] << endl;

cout << "Sub-chunk1 ID: " << HeaderInfo.Subchunk1ID[0] << HeaderInfo.Subchunk1ID[1] << HeaderInfo.Subchunk1ID[2] << HeaderInfo.Subchunk1ID[3] << endl;

cout << "Sub-chunk1 Size: " << HeaderInfo.Subchunk1Size << endl;

cout << "Audio Format: " << HeaderInfo.AudioFormat << endl;

cout << "Number of Channels: " << HeaderInfo.NumChannels << endl;

cout << "Sample Rate: " << HeaderInfo.SampleRate << endl;

cout << "Byte Rate: " << HeaderInfo.ByteRate << endl;

cout << "Block Align: " << HeaderInfo.BlockAlign << endl;

cout << "Bits Per Sample: " << HeaderInfo.BitsPerSample << endl;

cout << "Sub-chunk2 ID: " << HeaderInfo.Subchunk2ID[0] << HeaderInfo.Subchunk2ID[1] << HeaderInfo.Subchunk2ID[2] << HeaderInfo.Subchunk2ID[3] << endl;

cout << "Sub-chunk2 Size: " << HeaderInfo.Subchunk2Size << endl << endl;

}

1. **Wnioski**W systemie Windows dostępnych jest wiele metod odtwarzania dźwięku, jednak różnią się sposobem działania oraz stopniem zaawansowania implementacji. Podczas zajęć nie udało się zaimplementować pozostałych metod odtwarzania oraz nagrywania dźwięku