Mateusz Krawczak 241318

Karol Jaskółka 241306

Grupa: Pon P 17:00

Data wykonania ćwiczenia: 10.01.2020

Urządzenia Peryferyjne

Ćwiczenie 11 – Obsługa karty muzycznej z wykorzystaniem DirectSound, API i ActiveX

1. Wstęp

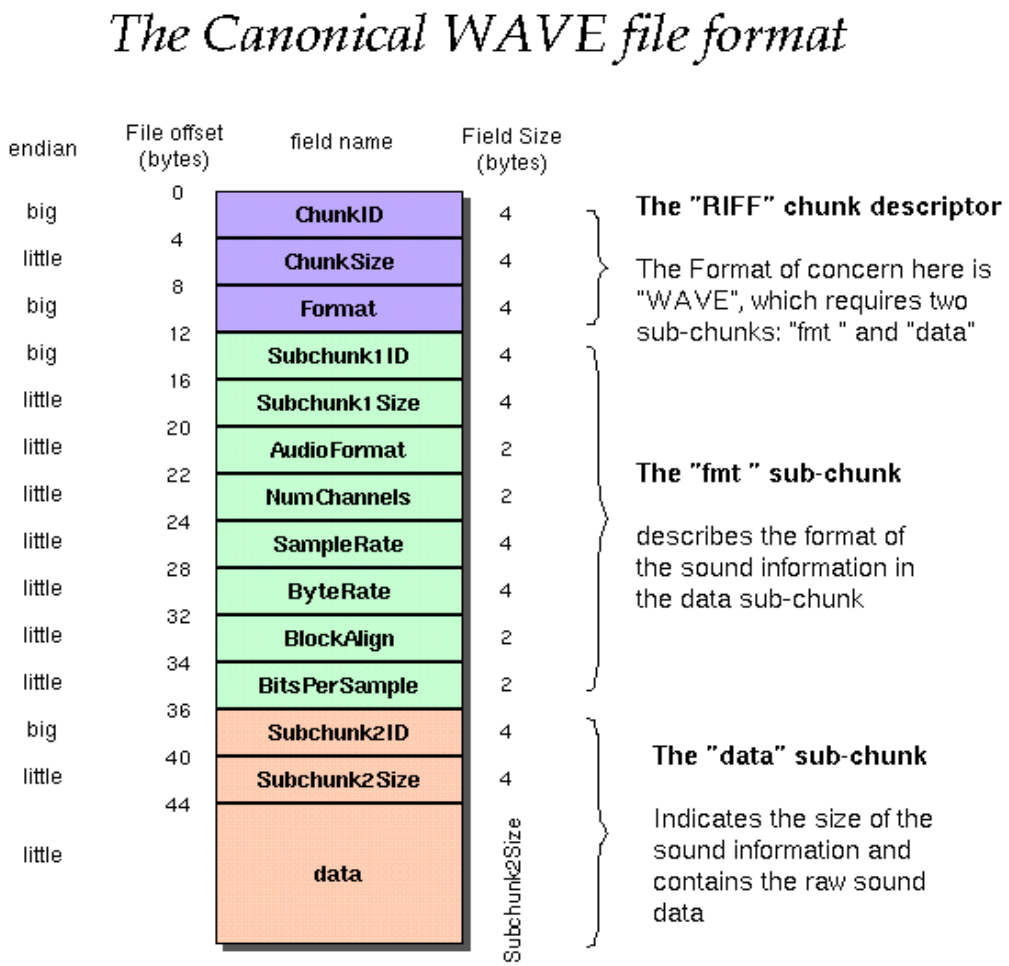
Karta dźwekowa - komputerowa karta rozszerzeń umożliwiająca rejestrację, przetwarzanie i odtwarzanie dźwięku; słuchanie muzyki. Najbardziej znaną grupą kart dźwiękowych jest seria Sound Blaster firmy Creative Labs. Obecnie układy dźwiękowe wystarczające do zastosowań amatorskich są zazwyczaj wbudowywane w płytę główną komputera, a nie stanowią karty rozszerzenia. Z powodów historycznych są jednak określane mianem „zintegrowana karta dźwiękowa". Pojawiły się również zewnętrzne karty dźwiękowe podłączane do komputera przez port USB.

1. Zagadnienia

Plik WAVE

WAV (ang. *wave form audio format*) – format plików dźwiękowych stworzony przez Microsoft oraz IBM. WAVE bazuje na formacie RIFF, poszerzając go o informacje o strumieniu audio, takie jak użyty kodek, częstotliwość próbkowania czy liczba kanałów.

Kanoniczny format WAVE rozpoczyna się od nagłówka RIFF, w którym są przechowywane informacje o ID, rozmiarze pliku i formacie. Następnie w formacie WAVE znajduje się część „fmt" są tam przechowywane informacje o dźwięku np. czy dany utwór jest nagrany w mono czy stereo. Ostatnia część „data" zawiera informacje o rozmiarze danych i dane dźwiękowe. Aby odczytać informacje, która nas interesuje należy odczytać odpowiednie.



SoundPlayer()

Klasa SoundPlayer udostępnia prosty interfejs do ładowania i odtwarzania pliku. wav. Klasa SoundPlayer obsługuje ładowanie pliku WAV z ścieżki pliku, adresu URL, Stream zawierającego plik. wav lub osadzony zasób zawierający plik. wav.

Aby odtworzyć dźwięk przy użyciu klasy SoundPlayer trzeba skonfigurowac SoundPlayer z ścieżką do pliku. wav i wywołać jedną z metod play. Można zidentyfikować plik WAV do odtworzenia przy użyciu jednego z konstruktorów lub ustawiając właściwość SoundLocation lub Stream. Plik może zostać załadowany przed rozpoczęciem korzystania z jednej z metod ładowania lub ładowanie może zostać odroczone do momentu wywołania jednej z metod play. SoundPlayer skonfigurowany do załadowania pliku WAV z Stream lub adres URL musi załadować plik WAV do pamięci przed rozpoczęciem odtwarzania.

2. Kod programu

Cały program znajduję się pod tym linkiem:
https://github.com/matson19/UP/tree/master/Lab%205%20-%20karta%20d%C5%BAwi%C4%99kowa

```
namespace KartaDzwiekowa
{
    public partial class Form1 : Form
    {
        private string filePath = "";
        private bool played;

        SoundPlayer simpleSound;

        NAudio.Wave.IWavePlayer waveOutDevice = new NAudio.Wave.WaveOut();
        NAudio.Wave.AudioFileReader audioFileReader;

        public Form1()
        {
            InitializeComponent();

        }

        private void buttonFile_Click(object sender, EventArgs e)
        {
            OpenFileDialog file = new OpenFileDialog();
            file.Filter = "Audio files (.wav)|*.wav";
            if (file.ShowDialog() == DialogResult.OK)
            {
                filePath = file.FileName;
                SetInfo();
            }
        }

        private void buttonPlay_Click(object sender, EventArgs e)
        {

            if (filePath == String.Empty)
            {
                MessageBox.Show("Wybierz plik!");
            }
            else
            {
                simpleSound = new SoundPlayer(@filePath);
                if (!played)
                {
                    played = !played;
                    simpleSound.Play();
                }
            }
        }

        private void buttonStop_Click(object sender, EventArgs e)
        {
            if (played)
            {
                simpleSound = new SoundPlayer(@filePath);
```

```
                played = !played;
                simpleSound.Stop();
            }

        }

        private void SetInfo()
        {
            if (!string.IsNullOrEmpty(filePath))
            {
                FileStream fileStream = new FileStream(filePath, FileMode.Open,
FileAccess.Read);

                BinaryReader reader = new BinaryReader(fileStream);

                // odczytanie nagłówka pliku Wave
                byte[] wave = reader.ReadBytes(36);

                fileStream.Position = 0;

                int chunkID = reader.ReadInt32();
                int chunkSize = reader.ReadInt32();

                var fileFormat = Encoding.Default.GetString(wave);
                string format = fileFormat.Substring(8, 4);
                string subchunk1ID = fileFormat.Substring(12, 4);
                string chunkIDString = fileFormat.Substring(0, 4);
                int formatInt = reader.ReadInt32();
                int subchunkInt = reader.ReadInt32();

                int subchunk1Size = reader.ReadInt32();
                int audioFormat = reader.ReadInt16();
                int numChannels = reader.ReadInt16();
                int sampleRate = reader.ReadInt32();
                int byteRate = reader.ReadInt32();
                int blockAlign = reader.ReadInt16();
                int bitsPerSample = reader.ReadInt16();

                reader.Close();

                textBoxInfo.Text = "";

                textBoxInfo.Text += "Chunk ID:" + chunkIDString;
                textBoxInfo.Text += "\r\nChunk size:" + chunkSize;
                textBoxInfo.Text += "\r\nFormat: " + format;
                textBoxInfo.Text += "\r\nSubchunk1 ID: " + subchunk1ID;
                textBoxInfo.Text += "\r\nSubchunk1 Size: " + subchunk1Size;
                textBoxInfo.Text += "\r\nAudio Format: " + audioFormat;
                textBoxInfo.Text += "\r\nNum Channels: " + numChannels;
                textBoxInfo.Text += "\r\nSample Rate: " + sampleRate;
                textBoxInfo.Text += "\r\nByte Rate: " + byteRate;
                textBoxInfo.Text += "\r\nBlock Align: " + blockAlign;
                textBoxInfo.Text += "\r\nBitsPerSample: " + bitsPerSample;

                labelFilePath.Text = "Plik: " + filePath;

            }
        }

        private void buttonPlayNAudio_Click(object sender, EventArgs e)
        {
            try
```

```
    {
        CloseWaveOut();
        waveOutDevice = new NAudio.Wave.WaveOut();
        audioFileReader = new NAudio.Wave.AudioFileReader(filePath);
        waveOutDevice.Init(audioFileReader);
        waveOutDevice.Play();
    }
    catch (Exception ex)
    {
        MessageBox.Show(string.Format("Wystąpił błąd: {0}", ex.Message));
    }
}

private void buttonStopNAudio_Click(object sender, EventArgs e)
{
    try
    {
        CloseWaveOut();
        waveOutDevice = new NAudio.Wave.WaveOut();
        audioFileReader = new NAudio.Wave.AudioFileReader(filePath);
        waveOutDevice.Init(audioFileReader);
        waveOutDevice.Stop();
    }
    catch (Exception ex)
    {
        MessageBox.Show(string.Format("Wystąpił błąd: {0}", ex.Message));
    }
}

private void CloseWaveOut()
{
    if (waveOutDevice != null)
    {
        waveOutDevice.Stop();
    }
    if (audioFileReader != null)
    {
        audioFileReader.Dispose();
        audioFileReader = null;
    }
    if (waveOutDevice != null)
    {
        waveOutDevice.Dispose();
        waveOutDevice = null;
    }
}

private void buttonFileMp3_Click(object sender, EventArgs e)
{
    OpenFileDialog fileMp3 = new OpenFileDialog();
    fileMp3.Filter = "Audio files (.mp3)|*.mp3";
    fileMp3.ShowDialog();
    axWindowsMediaPlayer1.URL = fileMp3.FileName;
    labelFilePath.Text = "Plik: " + fileMp3.FileName;
}

private void buttonPlayMp3_Click(object sender, EventArgs e)
{
    axWindowsMediaPlayer1.Ctlcontrols.play();
}

private void buttonStopMp3_Click(object sender, EventArgs e)
```

```
        {
            axWindowsMediaPlayer1.Ctlcontrols.pause();
        }
    }
}
```

3. Podsumowanie

Udało nam się odtworzyć plik muzyczny na trzy sposoby mianowicie była to funkcja SoundPlayer(), Waveform, a także za pomocą Windows Media Player. Program działał poprawnie.