Urządzenia peryferyjne - laboratorium

Karol Kulawiec 241281
Jakub Kalina 241346

13.01.2020

1 Wstęp

Celem wykonanego ćwiczenia było napisanie programu, za pomocą którego będzie można obsługiwać kartę dźwiękową przy pomocy DirectSound, Waveform i ActiveX.

Karta dźwiękowa to komputerowa karta rozszerzeń umożliwiająca rejestrację, przetwarzanie i odtwarzanie dźwięku. Składają się z:

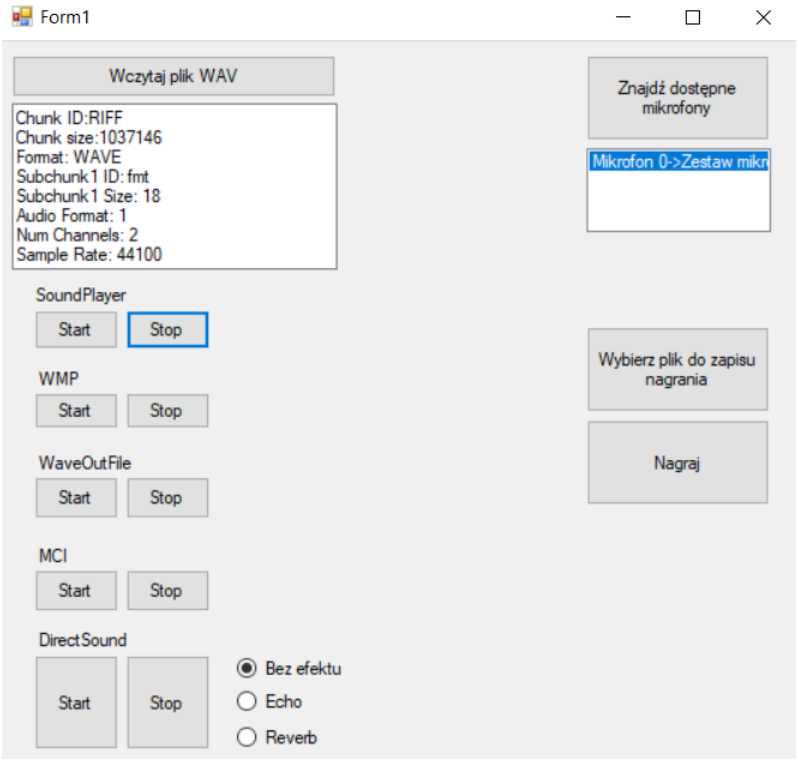
- generatorów dźwięku, który służył do sprzętowego generowania dźwięków za pomocą modulacji i łączenia fal oraz szumu,
- przetworników A/C oraz C/A, które zmieniają sposób reprezentacji sygnału z analogowy na cyfrowy i odwrotnie,
- buforu, czyli małej pamięci RAM, do której dźwięk jest zapisywany i odczytywany przez procesor główny,
- miksera dźwięku, odpowiedzialnego za łączenie sygnałów dźwięku z generatorów dźwięku, przetworników C/A i innych źródeł,
- wzmacniacza sygnałów wyjściowych,
- złącz i interfejsów służących do połączeń i komunikacji z innymi urządzeniami.

Dźwięk naturalny jest próbkowany i przenoszony na komputer jako cyfrowy. Aby jak najlepiej oddać oryginalne brzmienie, niezbędna jest wysoka rozdzielczość digitalizacji oraz duża częstotliwość próbkowania. Starsze karty zapisują dźwięk w trybie 8 bitowym, pozwalającym na rozróżnienie 256 różnych poziomów amplitudy sygnału, natomiast nagrania stereofoniczne na 4 bajtach, co daje 65536 różnych poziomów amplitudy sygnału. Częstotliwość, jaka odpowiada jakości rozmowy telefonicznej, wynosi 8kHz, jakości płytom CD - 44,1kHz, co jest rekonstruowane do 22kHz, ponieważ standardowa górna granica słyszalności przez człowieka wynosi 20kHz.

2 Opis aplikacji

Program został napisany w technologii *C#*, wykorzystując biblioteki WMPLib, SharpDX, które umożliwiają odtwarzanie dźwięku w różnych formatach.

Aplikacja prezentuje się w następujący sposób:



Rysunek 1: Interface aplikacji

Po lewej stronie aplikacji, znajdują się następujące funkcjonalności:

- przycisk do wczytania pliku .wav,
- okienko do wyświetlenia wartości nagłówka
- lista sposobów, na które można odtworzyć uprzednio wybrany plik, wraz z przyciskami start i stop.
- przyciski opcji, pozwalające zmienić dźwięk DirectSound.

Po prawej stronie aplikacji, znajdują się następujące funkcjonalności:

- przycisk znajdujący dostępne mikrofony,
- okienko do wyświetlenia i wybrania dostępnego mikrofonu,
- przycisk uruchamiający procedurę wybrania pliku do zapisu nagrania,
- przyciski rozpoczynający i kończący nagrywanie.

3 Opis ważniejszych funkcji

- Wyświetlenie informacji o pliku:

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

```
private void displayFileInfo()
{
    if (!string.IsNullOrEmpty(filePath))
    {
        try
        {
            FileStream fileStream = new FileStream(filePath,
                FileMode.Open, FileAccess.Read);
            BinaryReader reader = new
                BinaryReader(fileStream);
            // Odczytanie naglowka
            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();

            // Wyczyszczenie pola do wyswietlania informacji
            textBoxFIleInfo.Text = "";

            // Wyswietlenie danych pliku
            textBoxFIleInfo.Text += "Chunk ID:" +
                chunkIDString;
            textBoxFIleInfo.Text += "\r\nChunk size:" +
                chunkSize;
            textBoxFIleInfo.Text += "\r\nFormat: " + format;
            textBoxFIleInfo.Text += "\r\nSubchunk1 ID: " +
                subchunk1ID;
            textBoxFIleInfo.Text += "\r\nSubchunk1 Size: " +
                subchunk1Size;
            textBoxFIleInfo.Text += "\r\nAudio Format: " +
                audioFormat;
            textBoxFIleInfo.Text += "\r\nNum Channels: " +
                numChannels;
            textBoxFIleInfo.Text += "\r\nSample Rate: " +
                sampleRate;
            textBoxFIleInfo.Text += "\r\nByte Rate: " +
                byteRate;
            textBoxFIleInfo.Text += "\r\nBlock Align: " +
                blockAlign;
            textBoxFIleInfo.Text += "\r\nBitsPerSample: " +
                bitsPerSample;
        }
        catch(Exception)
        {
            MessageBox.Show("Podano nieprawidlowy format
                pliku");
```

```
            }
        }
    }
```

Plik WAV wczytujemy korzystając z OpenFileDialog, ustawiając filtr plików na rozszerzenie .wav. Pobieramy ścieżkę pliku i uruchamiamy metodę wyświetlającą informacje o tym pliku. W metodzie pobieramy strumień pliku oraz obiekt BinaryReader, z którego będziemy wyczytywać informacje, a następnie sczytujemy z niego w określonych momentach konkretne informacje.

- SoundPlayer

```
SoundPlayer soundPlayer;
private void buttonStartSoundPlayer_Click(object sender,
    EventArgs e)
{
    soundPlayer = new SoundPlayer(filePath);
    soundPlayer.Play();
}
private void buttonStopSoundPlayer_Click(object sender,
    EventArgs e)
{
    soundPlayer.Stop();
}
```

Przy pomocy obiektu SoundPlayer, podczas startu odtwarzania muzyki, pobieramy plik z ścieżki i uruchamiamy na nim Start(), a podczas zatrzymania, uruchamiamy metodę Stop().

- WindowsMediaPlayer

```
WindowsMediaPlayer wmp = new WMPLib.WindowsMediaPlayer();
private void buttonStartWmp_Click(object sender, EventArgs e)
{
    try
    {
        wmp.controls.stop();
    }
    catch (Exception) { }

    wmp.URL = filePath;
    wmp.settings.setMode("loop", true);
    wmp.controls.play();
}
private void buttonStopWmp_Click(object sender, EventArgs e)
{
    wmp.controls.stop();
}
```

Mechanizm WindowsMediaPlayer działa podobnie co SoundPlayer.

- WaveOutFiles

```
[DllImport("winmm.DLL", EntryPoint = "PlaySound",
    SetLastError = true, CharSet = CharSet.Unicode,
    ThrowOnUnmappableChar = true)]
private static extern bool PlaySound(string szSound,
    System.IntPtr hMod, PlaySoundFlags flags);

[System.Flags]
public enum PlaySoundFlags : int
{
    SND_SYNC = 0x0000,
    SND_ASYNC = 0x0001,
    SND_NODEFAULT = 0x0002,
```

```
        SND_LOOP = 0x0008,
        SND_NOSTOP = 0x0010,
        SND_NOWAIT = 0x00002000,
        SND_FILENAME = 0x00020000,
        SND_RESOURCE = 0x00040004
    }
    private void buttonStartWaveOutFile_Click(object sender,
        EventArgs e)
    {
        PlaySound(filePath, new System.IntPtr(),
            PlaySoundFlags.SND_ASYNC);
    }
    private void buttonStopWaveOutFile_Click(object sender,
        EventArgs e)
    {
        PlaySound(null, new System.IntPtr(),
            PlaySoundFlags.SND_ASYNC);
    }
```

W przypadku WaveOutFiles, potrzebujemy zaimportować odpowiednią bibliotekę w specyficzny sposób,zewnętrzną metodę PlaySound(), do której należy przekazać parametry, w tym Systemowe Flagi o określonych numerach.

- MCI

```
    string Pcom;
    [DllImport("winmm.dll", EntryPoint = "mciSendStringA",
        CharSet = CharSet.Ansi, SetLastError = true, ExactSpelling
        = true)]
    private static extern int mciSendString(string strCommand,
        StringBuilder strReturn, int iReturnLength, IntPtr
        hwndCallback);
    private void buttonStartMci_Click(object sender, EventArgs e)
    {
        int error = 0;
        Pcom = "close MediaFile";
        error = mciSendString(Pcom, null, 0, IntPtr.Zero);

        Pcom = "open \"" + filePath + "\" type mpegvideo alias
            MediaFile";
        error = mciSendString(Pcom, null, 0, IntPtr.Zero);

        Pcom = "play MediaFile";
        error = mciSendString(Pcom, null, 0, IntPtr.Zero);
    }
    private void buttonStopMci_Click(object sender, EventArgs e)
    {
        Pcom = "close MediaFile";
        int error = mciSendString(Pcom, null, 0, IntPtr.Zero);
    }
```

Korzystanie z MCI wymaga podobnego sposobu importu biblioteki oraz zewnętrznej funkcji. Tutaj w przypadku uruchomienia muzyki, należy zamknąć MediaFile, otworzyć plik z naszą muzyką i go uruchomić.

- DirectSound

```
    device = new XAudio2();
    device.StartEngine();
    masteringVoice = new MasteringVoice(device);

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

```
{
    stream = new SoundStream(File.OpenRead(filePath));
    var waveFormat = stream.Format;
    var buffer = new AudioBuffer
    {
        Stream = stream.ToDataStream(),
        AudioBytes = (int)stream.Length,
        Flags = SharpDX.XAudio2.BufferFlags.EndOfStream
    };
    stream.Close();
    sourceVoice = new SourceVoice(device, waveFormat, true);

    sourceVoice.SubmitSourceBuffer(buffer,
       stream.DecodedPacketsInfo);
    sourceVoice.Start();

    switch (chosenOption)
    {
        case 1:
            SharpDX.XAPO.Fx.Echo echo = new
               SharpDX.XAPO.Fx.Echo(device);
            EffectDescriptor effectDescriptorEcho = new
               EffectDescriptor(echo);
            sourceVoice.SetEffectChain(effectDescriptorEcho);
            sourceVoice.EnableEffect(0);
            break;
        case 2:
            SharpDX.XAPO.Fx.Reverb reverb = new
               SharpDX.XAPO.Fx.Reverb(device);
            EffectDescriptor effectDescriptorReverb = new
               EffectDescriptor(reverb);
            sourceVoice.SetEffectChain(effectDescriptorReverb);
            sourceVoice.EnableEffect(0);
            break;
    }
}
private void buttonStopDirectSound_Click(object sender,
   EventArgs e)
{
    sourceVoice.Stop();
}
```

DirectSound wymaga ustawienia strumienia, formatu strumienia, bufferu ręcznie. Następnie tworzony jest obiekt SourceVoice, a po ustawieniu parametrów, przy użyciu metody Start(), rozpoczyna się włączenie dźwięku.
W przypadku, gdy chcielibyśmy dodać do dźwięku jakiś efekt, np echo lub pogłos, należy wybrać tą opcję przy pomocy przycisku, a program, zależnie od wybranej opcji, doda efekt. Zakończenie odtwarzania dźwięku, wykonuje metoda Stop().

- Nagranie dźwięku

```
private void buttonStartStopRecording_Click(object sender,
   EventArgs e)
{
    if (wasRecored == false)
    {
        if (listBoxAvailableDevices.SelectedItems.Count == 0)
            return;

        if (fileRecordPath == "")
        {
```

```
                MessageBox.Show("Wybierz miejsce w ktorym chcesz
                    zapisac plik!");
            }
            else
            {

                int deviceNumber =
                    listBoxAvailableDevices.SelectedIndex;

                sourceStream = new NAudio.Wave.WaveIn();
                sourceStream.DeviceNumber = deviceNumber;
                sourceStream.WaveFormat = new
                    NAudio.Wave.WaveFormat(44100,
                    NAudio.Wave.WaveIn.GetCapabilities(deviceNumber).Channels);
                    // nadanie czestotliwosci nagrywania, i
                    standardu mono czy stereo wynikajacego z
                    urzadzenia

                sourceStream.DataAvailable += new
                    EventHandler<NAudio.Wave.WaveInEventArgs>(sourceStream_DataAva
                waveFileWriter = new
                    NAudio.Wave.WaveFileWriter(fileRecordPath,
                    sourceStream.WaveFormat);

                sourceStream.StartRecording();

                buttonStartStopRecording.Text = "W trakcie
                    nagrywania";
                wasRecored = true;
            }
        }
        else if (wasRecored == true)
        {
            if (soundOut != null)
            {
                soundOut.Stop();
                soundOut.Dispose();
                soundOut = null;
                buttonStartStopRecording.Text = "Nagraj";
            }
            if (sourceStream != null)
            {
                sourceStream.StopRecording();
                sourceStream.Dispose();
                sourceStream = null;
                buttonStartStopRecording.Text = "Nagraj";
            }
            if (waveFileWriter != null)
            {
                waveFileWriter.Dispose();
                waveFileWriter = null;
                buttonStartStopRecording.Text = "Nagraj";
            }
        }
    }
```

Przycisk posiada 2 funkcjonalności. Pierwsza z nich rozpoczyna nagrywanie, druga zatrzymuje. Działają naprzemiennie. Aby rozpocząć nagrywanie, należy uprzednio wybrać urządzenie oraz plik, miejsce zapisu pliku. Następnie ustawia się parametry strumienia, format Wave, dostępne dane oraz plik zapisu. Po tych ustawieniach rozpoczynamy nagranie metodą Star-

tRecording(), a przycisk zmienia swoją funkcjonalność. Podczas zatrzymania nagrywania, wyjście dźwięku jest zatrzymywane, strumień również, a plik zamykany.

4 Wnioski

Dzięki bibliotekom dostępnym w C# można zbudować własny odtwarzacz audio, dzięki któremu od środka można zobaczyć, jakie opcje są potrzebne, aby wykorzystać różne interfejsy odtwarzające audio. https://github.com/JakubKalina/UP-Karta_Muzyczna.