Kurs:
Urządzenia Peryferyjne - Laboratorium

Karta dźwiękowa - zapisywanie i odtwarzanie dźwięku

Autor:
Aleksandra Chrustek, 263900
Jonasz Trocha, 263951

Prowadzący:
dr inż. Dariusz Caban

12.12.2023

1 Wstęp

Celem ćwiczenia było zapoznanie się z zasadami działania karty dźwiękowej oraz napisanie oprogramowania do zapisywania i odtwarzania dźwięku.

2 Informacje teoretyczne

2.1 Zasady zapisywania dźwięku, technologie, parametry zapisu

2.1.1 Próbkowanie (Sampling):

Dźwięk analogowy jest ciągły, ale do zapisu cyfrowego musi być próbkowany w określonych odstępach czasu. Proces ten polega na pobieraniu próbek dźwięku w regularnych interwałach.

2.1.2 Rozdzielczość (Bit Depth):

Określa liczbę bitów używanych do reprezentacji każdej próbki dźwięku. Większa rozdzielczość pozwala na dokładniejsze odwzorowanie dynamiki dźwięku, ale zwiększa rozmiar pliku.

2.1.3 Częstotliwość próbkowania (Sample Rate):

Określa liczbę próbek dźwięku pobieranych na sekundę. Wyższa częstotliwość próbkowania oznacza lepszą jakość dźwięku, ale również większy rozmiar pliku.

2.1.4 Kodowanie dźwięku (Audio Coding):

Dźwięk cyfrowy może być kodowany za pomocą różnych algorytmów kompresji, które zmniejszają rozmiar plików dźwiękowych, zachowując jednocześnie akceptowalną jakość.

2.2 Środowiska programistyczne wspierające zapis, przetwarzanie i odtwarzanie dźwięku

2.2.1 DirectSound:

Technologia opracowana przez Microsoft, umożliwiająca programistom dostęp do funkcji dźwiękowych w systemie Windows.

2.2.2 PlaySound:

Funkcja w środowisku programistycznym Windows API, umożliwiająca proste odtwarzanie plików dźwiękowych.

2.2.3 ActiveX:

Technologia umożliwiająca tworzenie interaktywnych elementów na stronach internetowych, w tym odtwarzaczy dźwięku.

2.2.4 Waveform:

Interfejs programistyczny dla aplikacji Windows umożliwiający dostęp do funkcji związanych z obsługą dźwięku.

2.3 Formaty zapisu informacji dźwiękowej do zbioru

2.3.1 WAV:

Waveform Audio File Format - Format plików dźwiękowych opracowany przez Microsoft i IBM, zachowujący wysoką jakość dźwięku, ale zajmujący duże ilości miejsca.

2.3.2 MP3:

MPEG Audio Layer III - Powszechnie stosowany format kompresji dźwięku, który redukuje rozmiar plików kosztem pewnej straty jakości.

2.3.3 M4A:

MPEG-4 Audio - Rozszerzenie plików dźwiękowych związanych z formatem MPEG-4, charakteryzujące się wysoką jakością dźwięku i umożliwiające korzystanie z zaawansowanych funkcji, takich jak tagi.

2.3.4 PCM:

Pulse Code Modulation - Standardowy format bezstratnej kompresji dźwięku, który zachowuje pełną jakość dźwięku, ale generuje większe pliki w porównaniu do formatów kompresji stratnej.

3 Realizacja zadania

Aplikacja jest zaprojektowana do obsługi karty dźwiękowej. Należało napisać aplikację umożliwiającą odtwarzanie dźwięku z wykorzystaniem komendy PlaySound() oraz ActiveX, odczytanie i wyświetlenie poszczególnych pól nagłówka WAV, odtworzenie dźwięku zapisanego w zbiorze WAV za pomocą DirectSound, WaveForm, zarejestrowanie dźwięku za pomocą mikrofonu i zapisanie dźwięku w zbiorze WAV. W tych celach użyto bibliotek System.Media, NAudio oraz komponentu ActiveX.

Poniżej przedstawione zostały poszczególne metody do obsługi karty dźwiękowej:

1. Form1() - Konstruktor: Inicjalizuje komponenty okna, w tym komponent Windows Media Player (axWindowsMediaPlayer).

```
public Form1()
{
    InitializeComponent();
    mediaPlayer = axWindowsMediaPlayer;
    mediaPlayer.CreateControl();
}
```

2. buttonFile_Click(object sender, EventArgs e) - Obsługa przycisku "Wybierz Plik": Otwiera okno dialogowe do wyboru pliku dźwiękowego. Inicjalizuje odpowiednie obiekty do odtwarzania wybranego pliku.

```
private void buttonFile_Click(object sender, EventArgs e)
{
    OpenFileDialog fileExplorer = new OpenFileDialog();
    fileExplorer.Filter = "Audio files (.wav)|*.wav|Mp3 files (.mp3)|*.mp3";
    if (fileExplorer.ShowDialog() == DialogResult.OK)
    {
        filePath = fileExplorer.FileName;
        FillListBox();
        soundPlayer = new SoundPlayer(@filePath);
        audioFileReader = new AudioFileReader(filePath);
    }
}
```

3. FillListBox() - Wypełnianie ListBox informacjami o pliku dźwiękowym: Otwiera plik dźwiękowy, czyta informacje z nagłówka i wyświetla je w kontrolce ListBox.

```
private void FillListBox()
```

```
{
if (!string.IsNullOrEmpty(filePath))
{
FileStream fileStream = new FileStream(filePath, FileMode.Open, FileAccess.Read);
BinaryReader reader = new BinaryReader(fileStream);
byte[] wave = reader.ReadBytes(24);
fileStream.Position = 0;

int chunkID = reader.ReadInt32();
int fileSize = reader.ReadInt32();
var fileFormat = Encoding.Default.GetString(wave);
string format = fileFormat.Substring(8, 4);
string subchunk1ID = fileFormat.Substring(12, 8);
int subchunk1Size = reader.ReadInt32();

reader.Close();

string chunkIDStr = $"Chunk ID: {chunkID}";
string fileSizeStr = $"Chunk size: {fileSize}";
string fileFormatStr = $"Format: {format}";
string subchunk1IDStr = $"Subchunk ID: {subchunk1ID}";
string subchunk1SizeStr = $"Subchunk Size ID: {subchunk1Size}";

listBoxFileInfo.Items.Clear();
listBoxFileInfo.Items.AddRange(new string[] {
                    "\tNagłówek pliku:",
                    chunkIDStr,
                    fileSizeStr,
                    fileFormatStr,
                    "\tOpis struktury audio:",
                    subchunk1IDStr,
                    subchunk1SizeStr});
    }
}
```

4. buttonPlay_Click(object sender, EventArgs e) - Obsługa przycisku "Odtwórz": Odtwarza dźwięk wybranym sposobem: SoundPlayer, Windows Media Player, NAudio.Wave.WaveOut, lub NAudio.Wave.DirectSoundOut.

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

```
{
    if (filePath == String.Empty)
        MessageBox.Show("Wybierz plik!");
    else
    {
        if (radioButtonSoundPlayer.Checked)
        {
            soundPlayer.Play();
        }
        if (radioButtonMediaPlayer.Checked)
        {
            axWindowsMediaPlayer.Visible = true;
            mediaPlayer.URL = filePath;
            mediaPlayer.Ctlcontrols.play();
        }
        if (radioButtonWaveOut.Checked)
        {
            var waveOutFUNC = new WaveChannel32(new WaveFileReader(filePath));
            waveOut.Init(waveOutFUNC);
            waveOut.Play();
        }
        if (radioButtonDirectSound.Checked)
        {
            directSoundOut.Init(audioFileReader);
            directSoundOut.Play();
        }
    }
}
```

5. buttonStop_Click(object sender, EventArgs e) - Obsługa przycisku "Zatrzymaj": Zatrzymuje odtwarzanie dźwięku dla wybranego sposobu.

```
private void buttonStop_Click(object sender, EventArgs e)
{
    if (radioButtonSoundPlayer.Checked)
    {
        soundPlayer.Stop();
    }

    if (radioButtonMediaPlayer.Checked)
```

```
    {
        mediaPlayer.Ctlcontrols.stop();
    }

    if (radioButtonWaveOut.Checked)
    {
        waveOut.Stop();
    }

    if (radioButtonDirectSound.Checked)
    {
        directSoundOut.Stop();
    }
}
```

6. buttonFindDevice_Click(object sender, EventArgs e) - Obsługa przycisku "Znajdź Urządzenia": Wyświetla dostępne mikrofony w ListBox.

```
private void buttonFindDevice_Click(object sender, EventArgs e)
{
    labelSelectDevice.Visible = true;

    List<WaveInCapabilities> sources = new List<WaveInCapabilities>();

    for (int i = 0; i < WaveIn.DeviceCount; i++)
        sources.Add(WaveIn.GetCapabilities(i));

    listBoxMicrophones.Items.Clear();

    int counter = 0;
    foreach (var source in sources)
    {
        string item = source.ProductName;
        listBoxMicrophones.Items.Add("Microfon " + counter + "->" + item);
        counter++;
    }
}
```

7. sourceStream_DataAvailable(object sender, WaveInEventArgs e) - Obsługa dostępnych danych z mikrofonu: Zapisuje dane dźwiękowe do pliku, jeśli nagrywanie jest aktywowane.

```
private void sourceStream_DataAvailable(object sender, WaveInEventArgs e)
{
    if (waveFileWriter == null)
        return;

    waveFileWriter.Write(e.Buffer, 0, e.BytesRecorded);
    waveFileWriter.Flush();
}
```

8. buttonRecord_Click(object sender, EventArgs e) - Obsługa przycisku "Nagraj": Rozpoczyna nagrywanie dźwięku z wybranego mikrofonu do wybranego pliku.

```
private void buttonRecord_Click(object sender, EventArgs e)
{
    if (listBoxMicrophones.SelectedItems.Count == 0)
        return;
    if (fileSavePath == "")
    {
        MessageBox.Show("Select save file");
    }
    else
    {
        int deviceNumber = listBoxMicrophones.SelectedIndex;

        sourceStream = new WaveIn();
        sourceStream.DeviceNumber = deviceNumber;
        sourceStream.WaveFormat =
        new WaveFormat(44100, WaveIn.GetCapabilities(deviceNumber).Channels);
        sourceStream.DataAvailable +=
        new EventHandler<WaveInEventArgs>(sourceStream_DataAvailable);
        waveFileWriter =
        new WaveFileWriter(fileSavePath, sourceStream.WaveFormat);

        sourceStream.StartRecording();

    }
}
```

9. buttonStopRecord_Click(object sender, EventArgs e) - Obsługa przycisku "Zatrzymaj Nagrywanie": Zatrzymuje proces nagrywania.

```
private void buttonStopRecord_Click(object sender, EventArgs e)
{
    if (soundOut != null)
    {
        soundOut.Stop();
        soundOut.Dispose();
        soundOut = null;
        buttonRecord.Text = "Nagraj";
    }
    if (sourceStream != null)
    {
        sourceStream.StopRecording();
        sourceStream.Dispose();
        sourceStream = null;
        buttonRecord.Text = "Nagraj";
    }
    if (waveFileWriter != null)
    {
        waveFileWriter.Dispose();
        waveFileWriter = null;
        buttonRecord.Text = "Nagraj";
    }
}
```

10. buttonSelectFile_Click(object sender, EventArgs e) - Obsługa przycisku "Wybierz Plik"(dla zapisu nagranego dźwięku): Otwiera okno dialogowe do wyboru pliku do zapisu nagranego dźwięku.

```
private void buttonSelectFile_Click(object sender, EventArgs e)
{
    SaveFileDialog fileExplorer = new SaveFileDialog();
    fileExplorer.Filter = "Audio files (.wav)|*.wav|Mp3 files (.mp3)|*.mp3";
    if (fileExplorer.ShowDialog() == DialogResult.OK)
    {
        fileSavePath = fileExplorer.FileName;
    }
}
```

4 Wnioski

Zadanie miało na celu wprowadzenie w obszar programowania urządzeń peryferyjnych, konkretnie karty dźwiękowej. Realizacja tego zadania okazała się wymagająca, ponieważ wiązała się z koniecznością zaznajomienia się z zasadami odtwarzania i nagrywania dźwięku oraz zrozumieniem, jak obsługa karty dźwiękowej wygląda z perspektywy komputera. W konsekwencji, musieliśmy znaleźć sposoby na efektywne odtwarzanie i nagrywanie dźwięku i przełożenie tych sposobów na konkretne działania na poziomie aplikacji. W trakcie laboratorium udało się zaimplementować wszystkie wymagane przez instrukcję metody. Kod umożliwia obsługę różnych operacji związanych z dźwiękiem, takich jak odtwarzanie plików dźwiękowych, korzystanie z wbudowanego Windows Media Player, oraz nagrywanie dźwięku z dostępnych mikrofonów. W rezultacie, zadanie dostarczyło nam praktycznej wiedzy na temat programowania urządzeń peryferyjnych, a zdobyte umiejętności mogą być użyteczne w projektach obejmujących obsługę różnorodnych urządzeń wejściowych.

Aleksandra Chrustek,
Jonasz Trocha

12.12.2023

Lab 4 - Karta dźwiękowa

1. Stworzyliśmy aplikację do odtwarania dźwięku za pomocą trech sposobów: ActiveX, WaveForm oraz DirectSound.
DirectSound - technologia opracowana przez Microsoft.
ActiveX - technologia umożliwiająca tworenie interaktywnych elemen. na str. internetowych (w tym odtwanaoy dźwięku).
WaveForm - interfejs progranistyczny dla aplikacji Windows, umożliwia dostęp do funkcji związanych z obsługą dźwięku.
PlaySound - funkcja w Winolows API, umożliwiająca proste odtwa- nanie plików dźwiękowych.
2. Dodaliśmy pole, w którym wyświetlamy pola nagłówka WAV.
3. Następnie przeszliśmy do stronenia opcji podłączenia mikrofonu, aby możliwe było zapisanie pliku nagranego z mikrofonu.
4. Stworzyliśmy okienko do obsługi plików, obsługi mikrofonu oraz kontroli odtwarania, które wyświetla też szczegóły nagłów ka WAV oraz dostępne mikrofony.

12.12.2023