Politechnika Wrocławska
Katedra Informatyki Technicznej

Urządzenia Peryferyjne

Obsługa karty muzycznej

Magdalena Biernat
Michał Bojzan

Prowadzący
dr inż. Jan Nikodem

18 stycznia 2018

1 Wstęp

Karta muzyczna lub karta dzwiękowa jest to karta kmputerowa, która umożliwia rejestrację, odtwarzanie oraz modyfikowanie dzwięku.
Obecnie karty muzyczne zapisują dzwięk w trybie 16-bitowym. W przypadku nagrań stereofonicznych każdy pojedynczy dźwięk (próbka) jest więc zapisywany na 4 bajtach. Takie rozwiązanie pozwala na rozróżnienie 65536 różnych poziomów amplitudy dla każdego kanału stereo, dzięki czemu generowany dźwięk ma już naturalne brzmienie o jakości hi-fi.
Równie istotna jest szybkość próbkowania (samplingu), czyli częstotliwość z jaką generowane są kolejne 16-bitowe sekwencje. Im częściej jest próbkowany oryginalny dźwięk, tym wyższa jest maksymalna częstotliwość dźwięku uzyskiwanego nagrania. Częstotliwość samplingu rzędu 8 kHz odpowiada w przybliżeniu poziomowi jakości rozmowy telefonicznej natomiast do uzyskania jakości płyty CD potrzebna jest częstotliwość 44,1 kHz (pozwala to na rekonstrukcję dźwięku aż do częstotliwości 22 kHz(co wynika z czestotliwosci Nyquista), co jest powyżej górnej granicy słyszalności dźwięków u człowieka, tj. około 20 kHz)
Częstotliwość Nyquista – maksymalna częstotliwość składowych widmowych sygnału poddawanego procesowi próbkowania, które mogą zostać odtworzone z ciągu próbek bez zniekształceń.
Częstotliwość Nyquista jest równa połowie częstotliwości próbkowania. Przykładowo dla częstotliwości próbkowania 44,1 kHz stosowanej na płytach CD częstotliwość Nyquista wynosi 22,05 kHz. Jeśli w sygnale analogowym obecne są składowe o częstotliwości wyższej od częstotliwości Nyquista, spowoduje to powstanie błędów próbkowania (aliasing)

1.1 Kod programu

```
namespace karta_Biernat_Bojzan
{
public partial class Form1 : Form
{
private string filePath = "";
private SoundPlayer soundPlayer;
private bool wasPlayed;
NAudio.Wave.WaveIn sourceStream = null;
```

```
NAudio.Wave.DirectSoundOut waveOut = null;
NAudio.Wave.WaveFileWriter waveWriter = null;
public Form1()
{
InitializeComponent();
}

private void button2_Click(object sender, EventArgs e)
{

OpenFileDialog fileDialog = new OpenFileDialog();
fileDialog.Filter = "Audio files (.wav)|*.wav";
if (fileDialog.ShowDialog() == DialogResult.OK)
{
filePath = fileDialog.FileName;

FillListBox();
}
}

private void button1_Click(object sender, EventArgs e)
{
if (filePath == String.Empty)
MessageBox.Show("Wybierz plik!");
else
{
soundPlayer = new SoundPlayer(filePath);
if (!wasPlayed)
{
button1.Text = "Zatrzymaj";
wasPlayed = !wasPlayed;
soundPlayer.Play();
}
else
```

```
{
button1.Text = "Odwtórz";
wasPlayed = !wasPlayed;
soundPlayer.Stop();
}
}
}
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();
int riffType = reader.ReadInt32();

var fileFormat = Encoding.Default.GetString(wave);
string format = fileFormat.Substring(8, 4);


int fmtID = reader.ReadInt32();
int fmtSize = reader.ReadInt32();
int fmtCode = reader.ReadInt16();
int channels = reader.ReadInt16();
int sampleRate = reader.ReadInt32();
int byteRate = reader.ReadInt32();
int fmtBlockAlign = reader.ReadInt16();
```

```
int bitDepth = reader.ReadInt16();

reader.Close();

string chunkIDStr = $"Chunk ID: {chunkID}";
string fileSizeStr = $"File size: {fileSize}";
string riffTypeStr = $"Riff type: {riffType}";
string fileFormatStr = $"File format: {format}";
string fmtIDStr = $"Fmt ID: {fmtID}";
string fmtSizeStr = $"Fmt size: {fmtSize}";
string fmtCodeStr = $"File code: {fmtCode}";
string channelsStr = $"Channels: {channels}";
string sampleRateStr = $"Sample rate: {sampleRate}";
string byteRateStr = $"Byte rate: {byteRate}";
string fmtBlockAlignStr = $"Fmt block align: {fmtBlockAlign}";
string bitDepthStr = $"Bit depth: {bitDepth}";

listBox1.Items.Clear();
listBox1.Items.AddRange(new string[]
{
chunkIDStr, fileSizeStr, riffTypeStr, fileFormatStr, fmtIDStr, fmtSizeStr, fmtCodeStr
channelsStr, sampleRateStr, byteRateStr, fmtBlockAlignStr, bitDepthStr
});
}
}

private void buttonRecord_Click(object sender, EventArgs e)
{
if (listBox2.SelectedItems.Count == 0) return;

SaveFileDialog save = new SaveFileDialog();
save.Filter = "Wave File (*.wav)|*.wav;";
if (save.ShowDialog() != DialogResult.OK) return;
```

```
int deviceNumber = listBox2.SelectedIndex;

sourceStream = new NAudio.Wave.WaveIn();
sourceStream.DeviceNumber = deviceNumber;
sourceStream.WaveFormat = new NAudio.Wave.WaveFormat(44100, NAudio.Wave.WaveIn.GetCapa

sourceStream.DataAvailable += new EventHandler<NAudio.Wave.WaveInEventArgs>(sourceStre
waveWriter = new NAudio.Wave.WaveFileWriter(save.FileName, sourceStream.WaveFormat);

sourceStream.StartRecording();
sourceStream.GetMixerLine();
label1.Text = "Nagrywanie...";
}
private void sourceStream_DataAvailable(object sender, NAudio.Wave.WaveInEventArgs e)
{
if (waveWriter == null) return;

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

private void button3_Click(object sender, EventArgs e)
{
List<NAudio.Wave.WaveInCapabilities> sources = new List<NAudio.Wave.WaveInCapabilities

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

listBox2.Items.Clear();

foreach (var source in sources)
{
string item = source.ProductName;
listBox2.Items.Add(item);
```

```
}
}

private void button4_Click(object sender, EventArgs e)
{
if (waveOut != null)
{
waveOut.Stop();
waveOut.Dispose();
waveOut = null;
}
if (sourceStream != null)
{
sourceStream.StopRecording();
sourceStream.Dispose();
sourceStream = null;
}
if (waveWriter != null)
{
waveWriter.Dispose();
waveWriter = null;
}
label1.Text = "";
}
}

}
```

2 Objaśnienia

Chunk ID: Identyfikator pliku RIFF
File Size: Liczba określająca długość danych w pliku w bajtach z pominięciem pierwszych 8 bajtów nagłówka
File format: Format pliku
Fmt Id: Początek części opisowej pliku
Fmt size: Rozmiar części opisowej, dla fmt wynosi zwykle 16
File code: Rodzaj kompresji. 1 - bez kompresji, modulacja PCM.
Channels: liczba kanałów- 1 - mono, 2 - stereo
Sample rate-Częstotliwość próbkowania w Hz: 44100 co wynika z częstotliwości Nyquista
Byte rate: Częstotliwość bajtów: 176400 bajtów/s =1411200 bitów/s =1.41 Mb/s