

Přehrávač wav souborů s možnostmi úpravy přehrávaného signálu v reálném čase

A .wav file format player with realtime signal processing abilities

Bc. Miroslav Popelka

Diplomová práce
2013

 Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2012/2013

ZADÁNÍ DIPLOMOVÉ PRÁCE (PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Miroslav Popelka**
Osobní číslo: **A11444**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Počítačové a komunikační systémy**
Forma studia: **prezenční**

Téma práce: **Přehrávač wav souborů s možnostmi úpravy
přehrávaného signálu v reálném čase**

Zásady pro vypracování:

1. Nastudujte si problematiku kódování dat ve wav souborech.
2. Naprogramujte přehrávač hudebních souborů wav tak, aby bylo možné přehrávat wav soubory po dílčích blocích. Tyto bloky bude možné předzpracovávat, což umožní úpravu přehrávaného signálu v reálném čase.
3. V jazyce C++ (WxWidgets nebo QT) naprogramujte alespoň jeden blok pro transformaci přehrávaných dat.
4. Zhodnoťte dosažené výsledky a přínos práce.
5. Uvedte další možnosti rozšíření použitelnosti.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. TALBOT-SMITH, Michael. Sound engineering explained. 2nd ed. Oxford: Focal, 2002, xvi, 198 s. ISBN 0-240-51667-2.
2. ALTEN, Stanley R. Audio in media. 8th ed. Belmont, CA: Thomson/Wadsworth, c2008, xxv, 502 s. ISBN 978-0-495-09568-2.
3. POKORNÝ, Pavel. DirectX: začínáme programovat. 1. vyd. Praha: Grada, 2008, 224 s. ISBN 978-80-247-2254-2.
4. BLANCHETTE, Jasmin a Mark SUMMERFIELD. C++ GUI programming with Qt 4. 2nd ed. Upper Saddle River, N.J.: Prentice Hall, c2008, xxi, 718 p. ISBN 9780132354165.
5. MICROSOFT. Microsoft Developer Network [online]. 2013 [cit. 2013-01-31]. Dostupné z: <http://msdn.microsoft.com/cs-cz/ms348103.aspx>

Vedoucí diplomové práce:

Ing. Martin Pospíšilík

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

26. února 2013

Termín odevzdání diplomové práce:

31. května 2013

Ve Zlíně dne 26. února 2013

prof. Ing. Vladimír Vašek, CSc.
děkan

prof. Ing. Karel Vlček, CSc.
ředitel ústavu

ABSTRAKT

Práce se zabývá problematikou zpracování wav souborů v jazyce C++ na platformě Windows ve vývojovém prostředí QT. Je zde popsána struktura samotného wav souboru a vytvořeného programu. Vytvořený program využívá pro přehrávání dvou knihoven - DirectSound a MME. Pro obě metody přehrávání jsou použity úpravy v reálném čase a to změna úrovně a odstupů kanálů zvuku

Klíčová slova: MME, DirectSound, wav soubory, real time, zvuk, C++, QT

ABSTRACT

This thesis deal with processing of wav files in the language C++ on Windows platform in QT framework. In the thesis is described structure of wav file and created application. Created application uses for playing two different libraries - DirectSound and MME. For both playing methods are used real-time changes, which are change in the level and distance of sound channels.

Keywords: MME, DirectSound, wav files, real time, sound, C++, QT

Tímto bych chtěl poděkovat panu Ing. Martinu Pospíšilíkovi za poskytnuté cenné rady a připomínky týkající se zpracování diplomové práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD	10
I TEORETICKÁ ČÁST	11
1 ZVUK	12
1.1 ŠÍŘENÍ ZVUKU	12
1.2 AKUSTICKÝ TLAK	12
1.3 AKUSTICKÁ RYCHLOST	12
1.4 AKUSTICKÝ VÝKON	13
1.5 INTENZITA ZVUKU	13
1.6 VNÍMÁNÍ ZVUKU ČLOVĚKEM	13
2 RIFF	14
2.1 CHUNK	14
3 WAV	15
3.1 WAV CHUNKY	15
3.1.1 Fmt	16
3.1.2 Fact	17
3.1.3 Cue	17
3.1.4 Playlist	19
3.1.5 Assoc-data-list	19
3.1.6 Ltxt	21
3.1.7 Inst	21
3.1.8 Smpl	22
3.1.9 Data	24
3.1.10 Wavl a Slnt	25
3.2 KOMPRESSE	25
3.2.1 PCM	25
3.2.2 DPCM	27
3.2.3 ADPCM	27
3.2.4 A-law a μ -law	27
3.2.5 Uložení zvukových dat	30
4 PROGRAMOVÉ ZPRACOVÁNÍ	31
4.1 WAVEFORMAT	31
4.2 MME	32
4.2.1 WaveOutOpen	32
4.2.2 WaveOutPrepareHeader	33
4.2.3 WaveOutWrite	35
4.2.4 Funkce pro řízení toku přehrávání	35
4.2.5 WaveOutUnprepareHeader	36
4.2.6 WaveOutClose	36

4.3	DIRECTX	37
4.3.1	IDirectSound8	37
4.3.2	IDirectSound8::SetCooperativeLevel	37
4.3.3	IDirectSound8::CreateSoundBuffer	38
4.3.4	IDirectSoundBuffer8::Lock.....	39
4.3.5	IDirectSoundBuffer8::UnLock.....	40
4.3.6	IDirectSoundBuffer8::Play.....	40
4.3.7	IDirectSoundBuffer8::GetCurrentPosition.....	41
5	OSTATNÍ API.....	42
5.1	ASIO.....	42
5.2	GSIF	42
II	PRAKTICKÁ ČÁST	43
6	DOKUMENTACE K SOFTWARE	44
6.1	VÝVOJOVÉ PROSTŘEDÍ QT	44
6.2	INSTALACE A SPUŠTĚNÍ APLIKACE	45
7	GRAFICKÉ ROZHRAŇÍ.....	46
7.1	OVLÁDÁNÍ.....	46
7.2	HLAVNÍ UKAZATEL.....	49
7.3	REGULACE.....	50
7.4	ZOBRAZENÍ.....	51
7.5	STATUS BAR	52
7.6	VYKRESLOVACÍ DIALOG	53
7.7	DIALOG UPOZORNĚNÍ	54
8	TŘÍDY A METODY	55
8.1	MAINWINDOW	55
8.1.1	DejVlakno a DejObjekt.....	56
8.1.2	Otevření_souboru	57
8.1.3	Play_tl.....	57
8.1.4	Pause_tl	57
8.1.5	Stop_tl	57
8.1.6	Set_slider	58
8.1.7	Press_slider	58
8.1.8	Releas_slider	58
8.1.9	Seek_slider	58
8.1.10	Dx_setSstop.....	58
8.1.11	Update_level.....	59
8.1.12	Zobraz_level.....	59
8.1.13	Zprava.....	59
8.1.14	Zobraz_prubeh	59
8.1.15	On_actionDirectSound_triggered.....	59
8.1.16	On_actionWIN_MME_triggered	59
8.1.17	Plus_odstup	60

8.1.18	Minus_odstup.....	60
8.1.19	Reset_eq.....	60
8.1.20	Enable_API.....	60
8.2	DXSOUND.....	60
8.2.1	Dxsound.....	61
8.2.2	DoSetup.....	62
8.2.3	prehraj_DX.....	62
8.2.4	Add_blok_DX.....	63
8.2.5	Zapis_data.....	63
8.2.6	File_setings.....	63
8.2.7	Dx_seeking.....	64
8.2.8	Set_dx_uroven.....	64
8.2.9	Set_dx_odstup.....	64
8.2.10	Dx_pause.....	64
8.2.11	Dx_unpause.....	64
8.2.12	Dx_ukoncení.....	65
8.2.13	Dx_set_pozice.....	65
8.2.14	get_filename.....	65
8.3	WINWAVE.....	65
8.3.1	Winwave.....	66
8.3.2	DoSetup.....	66
8.3.3	File_setings.....	66
8.3.4	Prehraj.....	66
8.3.5	Set_position.....	67
8.4	NACTENÍ.....	67
8.4.1	Nastav_cestu.....	68
8.4.2	Nastav_posuv.....	68
8.4.3	Ano.....	68
8.4.4	Ne.....	68
8.4.5	Zpracuj.....	68
8.4.6	Get_levy.....	69
8.4.7	Get_pravy.....	69
9	POPIS POUŽITÍ.....	70
10	TESTOVÁNÍ A DOSAŽENÉ VÝSLEDKY.....	72
10.1	TESTOVÁNÍ.....	72
10.1.1	Testing.....	72
10.2	DOSAŽENÉ VÝSLEDKY.....	73
	ZÁVĚR.....	74
	ZÁVĚR V ANGLIČTINĚ.....	75
	SEZNAM POUŽITÉ LITERATURY.....	76
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	78
	SEZNAM OBRÁZKŮ.....	79
	SEZNAM TABULEK.....	81

ÚVOD

Cílem práce je vytvoření přehrávače wav souborů s možností úprav v reálném čase. Možnost změny dat ve zvukovém souboru formátu wav v reálném čase je velice zajímavá myšlenka, protože většina známých programů aplikují jednotlivé úpravy na všechna data a mění tak celý soubor. Takovýto postup je z hlediska typu souboru náročný zvláště při delší době nahrávky a nepřítomnosti kódování.

Nabyté poznatky z oblasti zvuku a jeho zpracování během studia i mimo něj mě přivedly k této diplomové práci. Nebyly to pouze znalosti, které zapříčinily výběr tohoto tématu. Dalším faktorem výběru byly mé již dřívější pokusy v oblasti přehrávání zvukových souborů.

Vývojové prostředí QT bylo vybráno z důvodu jeho rozsáhlé a přehledné dokumentace, která je přístupná online.

První část je částí spíše obecnějšího charakteru, jsou zde popsány základní definice týkající se zvuku z fyzikálního pohledu. Další část práce se zaměřuje na rozbor formátu wav. V rozboru jsou popsány jednotlivé bloky “chunky”, které popisují vlastnosti wav souboru. Následuje oddíl zaměřený na použité knihovní funkce pro přehrávání zvuku.

V Praktické části práce je rozdělena do několika oddílů, které postupně popisují vytvořenou aplikaci. Prvním z nich je dokumentace k softwaru, která říká v jakém prostředí byl program vytvářen a jak ho spustit. Poté je uveden popis grafického rozhraní s jednotlivými ovládacími prvky. Prvky v této části jsou nastýlovány pomocí kaskádových stylů a tyto jsou v této části uvedeny. V dalším oddílu jsou popsány všechny třídy a jejich metody využívané při přehrávání. Dále je uveden vývojový diagram popisující životní cyklus programu. Ke konci práce je uvedeno testování vytvořeného programu na jeho stabilitu a výkonnost. A v závěru je práce jsou shrnuty dosažené výsledky

I. TEORETICKÁ ČÁST

1 ZVUK

Zvuk je chápán z fyzikálního hlediska jako podélné mechanické vlnění šířící se různými hmotnými prostředími ve formě tlakové vlny. Díky fyziologickému uspořádání sluchových orgánů je možné tyto tlakové vlny vnímat. [1, s. 238]

Člověk je obecně schopen slyšet zvuková vlnění o frekvencích 16 Hz až 20 000 Hz. Nižší frekvence se označují jako *infrazvuk* a vyšší jako *ultrazvuk*. Sluchem je možné rozlišit v tomto spektru periodické (např. tón, nebo samohláska v řeči) a neperiodické zvuky (např. hluk). [1, s. 238]

Neharmonický periodický zvuk je tvořen z více složek „harmonických“. Právě tyto harmonické jsou příčinou, že tentýž tón zahráný na různé hudební nástroje, má stejnou výšku, ale jinou barvu. Barva je určena právě vyššími harmonickými a výška tou nejnižší harmonickou. [1, s. 240]

1.1 Šíření zvuku

Zvuk se šíří všemi typy látek pevnými, kapalnými a plynnými. V každé se ovšem pohybuje s rozdílnou rychlostí. Obecně platí pro rychlost šíření zvukové vlny v daném prostředí rovnice (1), kde s je dráha a t je čas : [1, s. 240]

$$c = \frac{s}{t} \quad (1)$$

1.2 Akustický tlak

Okamžitá hodnota tlaku v určitém místě a čase. Závisí na typu prostředí a jeho hustotě. Protože se jedná o podélné vlnění, při kterém je amplituda kmitů rovnoběžná se směrem šíření zvukové vlny. [1, s. 243]

1.3 Akustická rychlost

Akustická veličina udávající velikost rychlosti, se kterou se vzduchové částice pohybují při působení akustického tlaku kolem svojí rovnovážné polohy. [4, s. 7]

1.4 Akustický výkon

Akustický výkon určuje energii vyzářenou ze zdroje, nebo energii procházející určitou plochou. Jedná se o veličinu, která popisuje akustické vlastnosti zdroje zvuku. [4, s. 7]

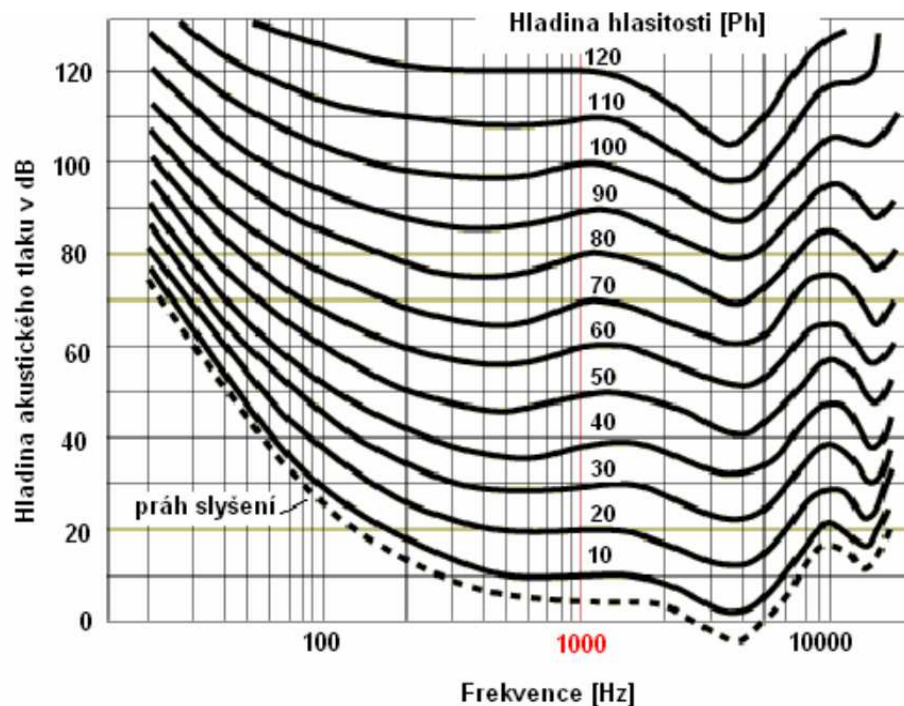
1.5 Intenzita zvuku

Intenzita zvuku je definována jako energie vyzářená zvukovým zdrojem za určitý čas na jednotku plochy, viz rovnice (2): [1, s. 245]

$$I = \frac{\Delta E}{\Delta t \cdot \Delta S} = \frac{\Delta P}{\Delta S} \quad [W \cdot m^{-2}] \quad (2)$$

1.6 Vnímání zvuku člověkem

Zvuk, který je schopný člověk sluchem zachytit musí mít hodnotu akustického tlaku větší než $2 \cdot 10^{-5}$ Pa. Při zkoumání vnímání zvuku lidským sluchem bylo zjištěno, že subjektivní vnímání zvuku závisí na frekvenci, tzn. že lidský sluch reaguje s různou citlivostí na různé frekvence. Na základě tohoto měření bylo slyšitelné pásmo rozděleno do oktávových pásem, které jsou specifikovány svými středními kmitočty. Na obr. 1 je zobrazena závislost vjemu hlasitosti na frekvenci. [1, s. 245]



Obr. 1 - Závislost vjemu hlasitosti na frekvenci [4, s. 15]

2 RIFF

RIFF (Resource Interchange File Format) byl vytvořen firmami IBM a Microsoft v roce 1991. Jeho vnitřní struktura je tvořená různými typy datových struktur (*chunk*). Takto tvořená struktura určuje výhodu tohoto souborového formátu. A to hlavně kvůli široké použitelnosti pro různé druhy multimediálního obsahu. Existuje mnoho multimediálních formátů, které jsou založeny právě na této RIFF struktuře (např. AVI, MIDI, RTF, WAVE.) [7, s. 9-11]

2.1 Chunk

Základní stavební blok všech odvozených RIFF formátů. Každý blok musí obsahovat strukturu a elementy zobrazené na obr. 2.

```
typedef unsigned long DWORD;
typedef unsigned char BYTE;

typedef DWORD FOURCC;           // Four-character code

typedef FOURCC CKID;           // Four-character-code chunk identifier
typedef DWORD CKSIZE;         // 32-bit unsigned size value

typedef struct {               // Chunk structure
    CKID      ckID;             // Chunk type identifier
    CKSIZE    ckSize;          // Chunk size field (size of ckData)
    BYTE      ckData[ckSize];  // Chunk data
} CK;
```

Obr. 2 - Struktura RIFF formátu v jazyce C [7, s. 11]

ckID - Identifikátor o čtyřech znacích, reprezentující data uložená v *ckData*.

ckSize - 32 bitová hodnota určující velikost *ckData*. Když je hodnota lichá přidává se na konec *ckData* nula, aby se dodrželo dvoubytové zarovnání.

ckData - Binární data pevné, nebo proměnné délky. [7, s. 11]

Zarovnání na dva bajty se používá pro zrychlení přístupu do paměti a kvůli kompatibilitě se standardem EA IFF. [7, s. 11]

Podle [7, s. 11] jsou LIST, RIFF jediné dva bloky, které mohou obsahovat podbloky.

3 WAV

Je zobecněním výše zmíněného RIFF formátu. Je určen pro ukládání audia v komprimované i nekomprimované formě. Tab. 1 ukazuje seznam některých registrovaných kompresí pro wav formát. [7, s. 56]

Tab. 1- Tabulka některých typů kompresí použitých ve formátu wav [7, s. 11]

Kód	Komprimace	Společnost
0	WAVE_FORMAT_UNKNOWN	Microsoft Corporation
1	WAVE_FORMAT_ADPCM	Microsoft Corporation
2	WAVE_FORMAT_IEEE_FLOAT	Microsoft Corporation
3	WAVE_FORMAT_VSELP	Compaq Computer Corp.
4	WAVE_FORMAT_IBM_CVSD	IBM Corporation
5	WAVE_FORMAT_ALAW	Microsoft Corporation
6	WAVE_FORMAT_MULAW	Microsoft Corporation
7	WAVE_FORMAT_DTS	Microsoft Corporation
8	WAVE_FORMAT_OKI_ADPCM	OKI
9	WAVE_FORMAT_DVI_ADPCM	Intel Corporation

3.1 WAV Chunky

Stejně jako nadřazený formát obsahuje wav soubor svoje bloky „chunky“. Každý wav soubor by měl obsahovat strukturu zobrazenou na Obr. 3.

```

<WAVE-form> ::=
    RIFF('WAVE'
        <'fmt'-ck>
        [<fact-ck>]
        [<cue-ck>]
        [<playlist-ck>]
        [<assoc-data-list>]
        <wave-data> )
  
```

Obr. 3 - Obecná struktura wav souboru [7, s. 56]

Pro všechny bloky obecně platí již zmíněná RIFF struktura, ve které je nejprve identifikátor a poté velikost bloku. Za tímto je pak umístěna samotná vnitřní struktura bloku. Popis všech bloků je uveden v následujících podkapitolách. [7, s. 56]

3.1.1 Fmt

Tento blok by se dal označit za hlavičku celého wav souboru a podle [7, s. 56] musí být vždy umístěn před samotnými daty.

Fmt blok popisuje uložená data pomocí následující struktury uvedené na Obr. 4

```

<fmt-ck> →      fmt (      <common-fields>
                          <format-specific-fields> )

<common-fields> →
    struct
    {
        WORD    wFormatTag;           // Format category
        WORD    wChannels;           // Number of channels
        DWORD   dwSamplesPerSec;     // Sampling rate
        DWORD   dwAvgBytesPerSec;    // For buffer estimation
        WORD    wBlockAlign;         // Data block size
    }
  
```

Obr. 4 - Struktura fmt bloku [7, s.56]

Ve struktuře „<common- fields>” jsou obsaženy elementy popsané Tab. 2 a Tab. 3.

Tab. 2 - Seznam elementů fmt bloku [7, s. 56]

Název elementu	velikost	offset
	Byte	
<i>wFormatTag</i>	2	0x00
<i>wChannels</i>	2	0x02
<i>dwSamplesPerSec</i>	4	0x04
<i>dwAvgBytesPerSec</i>	4	0x08
<i>wBlockAlign</i>	2	0x0A

Tab. 3 - Popis elementů fmt bloku [7, s. 56]

Element	Popis
wFormatTag	Číslo určující typ komprimace.
wChannels	Počet kanálů zvuku 1- mono, 2-stereo
dwSamplesPerSec	Počet vzorků za sekundu
dwAvgBytesPerSec	Průměrný počet bajtů za sekundu
wBlockAlign	Zarovnání zvukových dat

Format-specific-fields struktura obsahuje doplňkové elementy např. *wBitPerSample*. Elementy v této struktuře jsou přidávány v závislosti na zvoleném kódování. V případě PCM se jedná o element počet bitů na vzorek. Jednotlivé kompresní algoritmy jsou popsány níže. [7, s. 58]

3.1.2 Fact

Obsahuje informace o velikosti dat v jednom vzorku. Využívá se pouze u souborů, které používají nějakou kompresi a obsahují blok wavl blok. U souboru bez komprese (PCM) se tento blok nepoužívá. [7, s. 61]

```
<fact-ck>      →      fact( <dwSampleLength:DWORD> )
```

Obr. 5 - Struktura Fact bloku [7, s. 61]

Na obr. 5 je uvedena vnitřní struktura fact bloku. Podle [8, s. 12] je možné tento blok rozšiřovat o další elementy, které budou nutné pro budoucí wav soubory. V tomto případě se další elementy budou ukládat za již vytvořené elementy.

Dále pak lze z elementu *dwSampleLength* je možné za pomoci *wSamplePerSec* vypočítat délku nahrávky v sekundách.

3.1.3 Cue

Slouží pro ukládání pozic (cue bodů). Cue body jsou významné pozice ve wav souboru jako je začátek, konec souboru popřípadě sloky atd. Jeho struktura je následující a je uvedena na obr. 6. [7, s. 61]

```
<cue-ck> ➡      cue( <dwCuePoints:DWORD>           // Count of cue points
                  <cue-point>... )           // Cue-point
table
<cue-point> ➡   struct {
                                DWORD dwName;
                                DWORD dwPosition;
                                FOURCC fccChunk;
                                DWORD dwChunkStart;
                                DWORD dwBlockStart;
                                DWORD dwSampleOffset;
                                }
}
```

Obr. 6 - Struktura Cue bloku [7, s. 61]

Za číslem *dwCuePoints*, které určuje počet cue-bodů následuje seznam jednotlivých cue-bodů. Každý cue-bod je popsán strukturou uvedenou na obr. 6. V tab. 4 a tab. 5 je uveden seznam a popis elementů uvnitř této struktury s danou velikostí a offsetem každého z nich.[8, s. 13]

Tab. 4 - Seznam elementů Cue-point struktury [8, s. 13]

Element	velikost	offset
	Byte	-
dwName	4	0x00
dwPosition	4	0x04
fccChunk	4	0x08
dwChunkStart	4	0x0C
dwBlockStart	4	0x10
dwSampleOffset	4	0x14
dwChunkStart	4	0x18
dwBlockStart	4	0x1C
dwSampleOffset	4	0x20

Tab. 5 - Popis elementů Cue-point struktury [8, s. 13]

Element	Popis
dwName	Každý cue-bod musí mít tuto hodnotu unikátní
dwPosition	Pozice cue-bodu v dané sekci v závislosti na Plst bloku.
fccChunk	Specifikuje jméno nebo ID bloku obsahující cue-bod.
dwChunkStart	Startovní pozice datového bloku obsahující cue-bod. (0- jeden datový blok)
dwBlockStart	Určuje pozici začátku bloku obsahujícího pozice, nebo také posun bajtů od začátku datové části wavl bloku
dwSampleOffset	Určuje offset vzorku dat cue-bodu od začátku bloku.

3.1.4 Playlist

Určuje pořadí, v jakém se budou cue body z cue bloku přehrávat. Vnitřní struktura je popsána na obr. 7 a v tab. 6 jsou popsány jednotlivé elementy této struktury. [8, s. 13-14]

```

<playlist-ck> ➡      plst(
                        <dwSegments:DWORD> // Count of play
segments              <play-segment>... ) // Play-segment table
<play-segment> ➡    struct {
                        DWORD dwName;
                        DWORD dwLength;
                        DWORD dwLoops;
                        }

```

Obr. 7 - Struktura Plst bloku.[8, s. 14]

Tab. 6 - Popis elementů Plst bloku.[8, s. 14]

Element	Popis
dwName	Definuje jména cue bodů a musí se shodovat s nějakým jménem v seznamu cue-bodů
dwLength	Délka sekce určená počtem vzorků.
dwLoops	Počet opakování dané sekce.

3.1.5 Assoc-data-list

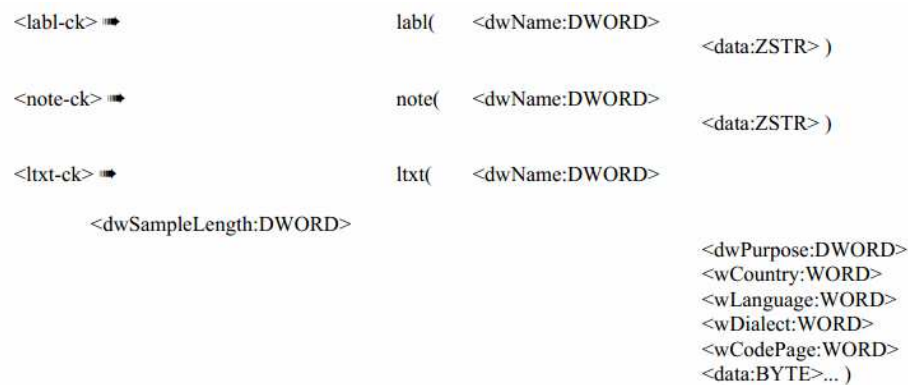
Umožňuje připojit doplňkové informace, jako jsou popisky k určitým sekcím v datovém toku. Struktura bloku je na obr. 8 a obr. 9. [8, s. 14]

```

<assoc-data-list> ➡    LIST( 'adtl'
                        <labl-ck>
                        // Label
                        <note-ck>
                        // Note
                        <ltxt-ck> }
                        // Text with data length

```

Obr. 8 - Struktura Assoc-data-list bloku [8, s. 14]



Obr. 9 - Vnitřní struktura Assoc-data-list bloku[8, s. 14]

Podbloky labl a note jsou podobné svou strukturou a jsou v nich jen nepatrné rozdíly, které jsou popsány níže. Tab. 7 a tab. 8. popisují elementy těchto bloků.

Labl

Labl blok obsahuje popisek, nebo titulek k odpovídajícímu cue-bodu.

Tab. 7 - Popis elementů Labl bloku.[8, s. 15]

Element	Popis
dwName	Definuje jména cue bodů a musí se shodovat s nějakým jménem v seznamu cue-bodů.
data	Textový popisek ukončený \0.

Note

Note blok obsahuje komentář pro jednotlivé cue-body.

Tab. 8 - Popis elementů Note bloku.[8, s. 15]

Element	Popis
dwName	Definuje jména cue bodů a musí se shodovat s nějakým jménem v seznamu cue-bodů.
data	Komentář ukončený \0.

3.1.6 Ltxt

Ltxt blok obsahuje text, který je spojený s datovým segmentem dané délky. Jeho elementy jsou popsány v následující tabulce (tab. 9):[8, s. 15]

Tab. 9 - Popis elementů Ltxt bloku [8, s. 15]

Element	Popis
dwName	Definuje jména cue bodů a musí se shodovat s nějakým jménem v seznamu cue-bodů
dwSampleLength	Určuje počet vzorků v segmentu zvukových dat
dwPurpose	Specifikuje druh, nebo účel textu. (FOURCC)
wCountry	Nastavení kódu země pro text, viz [8, s. 72]
wLanguage	Definuje jazykový kód textu.
wDialect	Definuje dialektový kód, viz [8, s. 73]
wCodePage	např. UTF-8, CP-1251...

3.1.7 Inst

V [8, s. 16] je wav soubor označován za téměř dokonalý souborový formát pro ukládání vzorkovaného zvuku. Tato „dokonalost“ je daná množstvím bloku popisující data (Počet bitů na vzorek, počet kanálů, počet opakování atd.), ale tón vzorku a s tím spojená hlasitost k jinému vzorku v žádném bloku zastoupení nemají.

Tento podblok obsahuje tyto potřebné informace, viz obr. 10 a tab. 10.

```
|<instrument-ck>| ──► inst(
    <bUnshiftedNote:BYTE>
    <chFineTune:CHAR>
    <chGain:CHAR>
    <bLowNote:BYTE>
    <bHighNote:BYTE>
    <bLowVelocity:BYTE>
    <bHighVelocity:BYTE> )
```

Obr. 10 – Struktura Inst bloku.[8, s. 16]

Tab. 10 - Seznam elementů Inst bloku.[8, s. 16]

Element	Popis
bUnshiftedNote	Platná hodnota 0 až 127.
chFineTune	Hodnota určuje, kolik vzorků tónu by měly být změněny, pokud je zvuk přehráván po 1/100 půltónu. Záporná hodnota znamená, že tón by se měl hrát níže a kladná hodnota znamená, že by se měl hrát výše. Platná hodnota od -50 do 50.
chGain	Úroveň hlasitosti v decibelech, kde záporné hodnoty znamenají zeslabení a kladné zesílení amplitudy.
bLowNote and bHighNote	Určuje rozsah MIDI not a nemusí obsahovat neposunuté noty. Platná hodnota od 0 do 127.
bLowVelocity and bHighVelocity	Určuje rozsah MIDI rychlosti. Rychlost určuje jak silně bude nota zahrána.

3.1.8 Smpl

Tento blok popisuje jen minimum nutných informací potřebných pro použití wav souboru ve spojení s vzorkovacími klávesami. Samplery, které vyžadují ukládání více informací, mohou tyto ukládat do podbloku *sampler-specific-data*. Struktura Smpl bloku je na obr. 11 a popis jejich členů je uveden v tab. 11.[8, s. 16]

```

|<sample-ck>| ➔ smpl(
    <dwManufacturer:DWORD>
    <dwProduct:DWORD>
    <dwSamplePeriod:DWORD>
    <dwMIDIUnityNote:DWORD>
    <dwMIDIPitchFraction:DWORD>
    <dwSMPTEFormat:DWORD>
    <dwSMPTEOffset:DWORD>
    <cSampleLoops:DWORD>
    <cbSamplerData:DWORD>
    <sample-loop(s)>
    <sampler-specific-data> )

```

Obr. 11 - Struktura Smpl bloku [8, s. 16]

Tab. 11 - Popis elementů Smpl bloku [8, s. 17]

Element	Popis
dwManufacturer	Specifikuje MMA kód výrobce pro určité cílové zařízení. Nejvyšší byte určuje počet bytů, ve kterých se bude vyskytovat samotný kód (1 nebo 3). Příklad: hodnota bude 0x01000013, kde hodnota 0x13 je kód pro Digidesign. Pokud není vzorek určený pro specifického výrobce, hodnota bude nulová.
dwProduct	Specifikuje kód produktu určitého cílového zařízení. Pokud není vzorek určen pro daný produkt, hodnota by měla být nula.
dwSamplePeriod	Udává periodu jednoho vzorku v nanosekundách (1 / početVzorků ZaSekundu).
dwMIDIUnityNote	Určuje MIDI noty, podle které se přehraje vzorek na originální tónině. Rozsah hodnoty je od 0 do 127 (60 reprezentuje C podle MMA).
dwMIDIPitchFraction	Udává, jaká část půltónu bude přidána k hodnotě dwMIDIUnityNote. Hodnota 0x80000000 je 1/2 půltónu.
dwSMPTEFormat	Určuje SMPTE časový formát použitý v dwSMPTEOffset. Hodnoty mohou být: 0- žádný časový formát a také offset by měl být nula. 24, 25, 29, 30 - rámců za sekundu.
dwSMPTEOffset	Časový offset pro synchronizaci prvního vzorku dat. Formát této hodnoty je 0xhhmmssff.
cSampleLoops	Počet sample-loop záznamů, které jsou obsaženy v tomto bloku. sample-loop jsou uládány za cbSaplerData.
cbSamplerData	Velikost v bajtech volitelného <sampler-specific-data> bloku.

Na následujícím obrázku (obr. 12) a v tabulce (tab. 12) je vyobrazena a popsána struktura podbloku sample-loop.

```

<sample-loop> struct
{
    DWORD      dwIdentifier;
    DWORD      dwType;
    DWORD      dwStart;
    DWORD      dwEnd;
    DWORD      dwFraction;
    DWORD      dwPlayCount;
}

```

Obr. 12 - Vnitřní struktura *sample-loop* [8, s. 18]

Tab. 12 - Popis elementů *sample-loop* bloku [8, s. 18]

Element	Popis
dwIdentifier	Unikátní jméno smyčky. Tato hodnota by měla odpovídat s hodnotou uloženou v cue bloku.
dwType	Určuje typ smyčky: 0 - normal, 1 - Střídání běhu smyčky, 2 - běh smyčky pozpátku, 3 - 32 - rezervováno pro budoucí typy, 32-? - definováno z výroby.
dwStart	Počáteční bod smyčky ve vzorcích.
dwEnd	Koncový bod smyčky ve vzorcích (přehraje se i tento vzorek).
dwFraction	Dovoluje jemné doladění částí smyčky mezi vzorky. Hodnotu je možné nastavit od 0x00000000 do 0xFFFFFFFF. 0x80000000 reprezentuje polovinu délky vzorku.
dwPlayCount	Počet opakování smyčky. 0 - nekonečná udržovací smyčka.

Bloky *Smpl* a *Inst* se v běžné praxi téměř nepoužívají a jsou zde uvedeny jen pro úplnost.

3.1.9 Data

Tento blok obsahuje zvuková data, která jsou přehrávána. Tyto data mohou být komprimována pomocí výše popsaných mechanismů. Struktura je uvedena na obr. 13 [1, s. 60]

```

<wave-data> →      { <data-ck> | <data-list> }
<data-ck> →        data( <wave-data> )
<wave-list> →     LIST( 'wavl' {      <data-ck> |           // Wave
samples                                     <silence-ck> }... ) // Silence
<silence-ck> →    slnt( <dwSamples:DWORD> )           // Count of
                                                         // silent samples

```

Obr. 13 - Struktura data bloku [1, s. 60]

3.1.10 Wavl a Slnt

Wavl blok obsahuje seznam dat a tichých míst. Podblok obsahující tyto tiché vzorky je označován jako Slnt a je používán, když tiché místo v souboru překročí velikosti jednoho vzorku. V struktuře Slnt je jediný element, který obsahuje číslo vyjadřující počet tichých vzorků. [1, s. 60]

3.2 Komprese

Komprimace používané v souborech typu wav jsou následující:

- PCM
- DPCM
- ADPCM
- A-law
- Mu-law

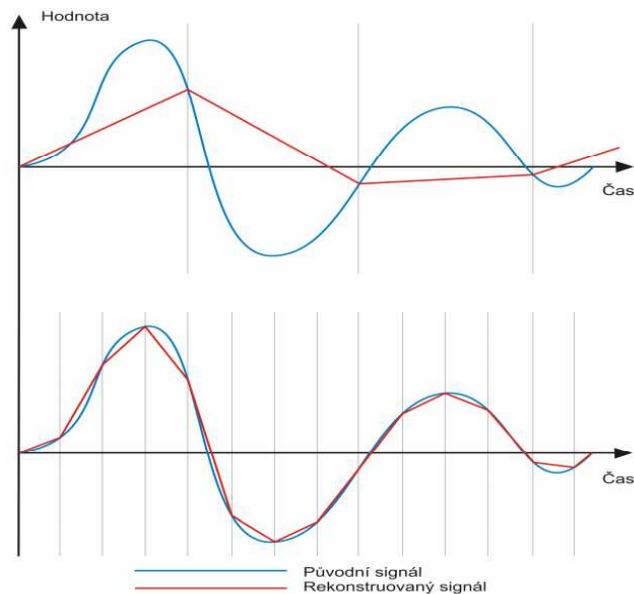
3.2.1 PCM

Jedná se vlastně o pulsně kódovou modulaci, jejímž základním principem je převod analogového signálu na digitální. Tento převod se skládá ze základních operací (vzorkování, kvantování a kódování). PCM by se dala přirovnat svou funkčností k A/D převodníku, který taktéž pořízeným hodnotám vzorků analogového signálu přiřazuje kvantizační hodnoty a k těmto kódové hodnoty. [11, s. 11]

Jako první je nutné provést při digitalizaci vzorkování. Vzorkování je odečtení hodnot analogového signálu v předem stanoveném časovém intervalu. Správný interval odečítání hodnot je definován tzv. Shannon-Nyquist-Kotělnikovův teorém. Tento teorém říká, že

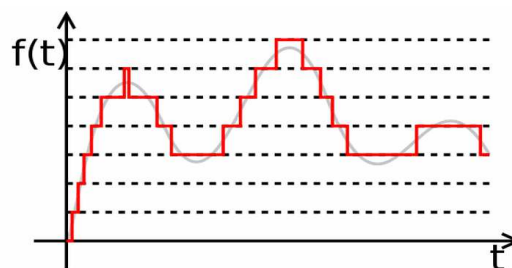
frekvence, s jakou se vzorky budou odečítat, musí být alespoň dvakrát větší, než je maximální frekvence rekonstruovaného signálu. Pokud bude vzorkovací frekvence menší, dojde ke zkreslení digitálního signálu, viz obr 14. [11, s. 11]

Kvantování se provádí tak, že se vzorkovaný signál rozdělí do kvantizačních úrovní a následně se těmto hladinám přidělí jedinečný binární kód.



Obr. 14 - Ukázka špatně a správně provedeného vzorkování [11, s. 12]

U kvantování stejně jako u vzorkování je nutné vzít v potaz tzv. kvantizační šum. Při kvantování se provádí přiřazení vzorkovaných bodů jednotlivým kvantizačním úrovním, viz Obr. 15.



Obr. 15 - Ukázka kvantování [3]

Z obr. 15 je vidět kvantizační šum, který je rozdílem mezi kvantizační úrovní a vzorkovaným signálem. Šum se s počtem kvantizačních úrovní zmenšuje a obvykle se v případě A/D převodníků používá maximálně 24bitová rozlišovací schopnost.

3.2.2 DPCM

Při DPCM se vypočítává rozdíl mezi vzorky. Rozdíl mezi jednotlivými vzorky je podstatně menší než samotná hodnota vzorku. DPCM se provádí se vzorkovaným signálem a to tak, že se první hodnota ukládá nezměněna a další hodnoty jsou určeny rozdílem s předcházející hodnotou vzorku.[14]

Postup při DPCM po kvantizaci vzorků je takový, že se vytváří rámcová struktura pevné délky, do které se ukládá první kvantovaný vzorek nezměněn a následující vzorky jsou vyjádřené jako rozdíl předcházejícího.[14]

Takovýto postup vede ke zlepšení kompresního poměru, ale zvýší se kvantizační chyba. [14]

3.2.3 ADPCM

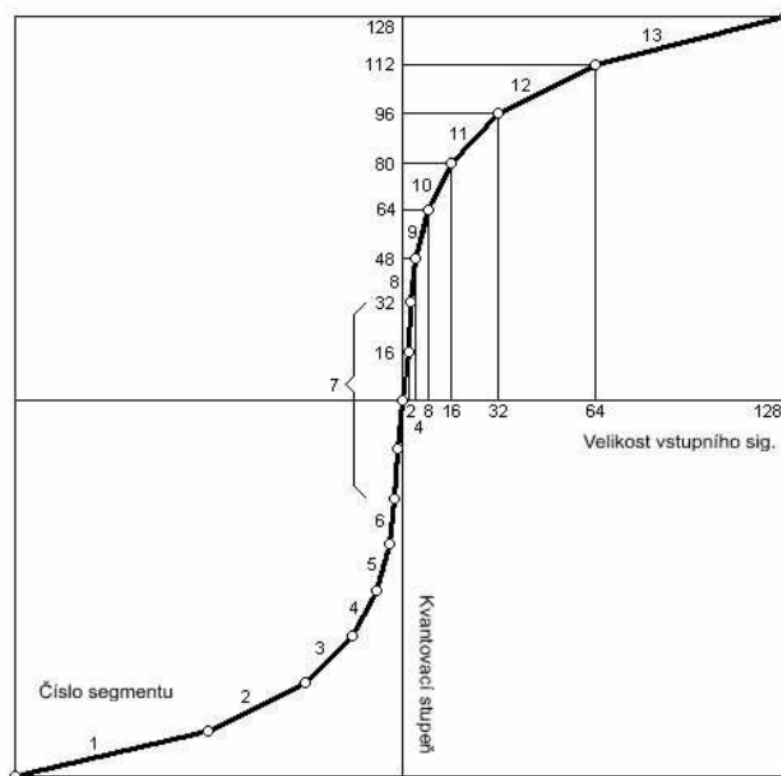
Zde se využívá více kvantizátorů, obvykle 4, ze kterých se vybírá podle rozvržení kvantizačních úrovní v závislosti na rozložení vzorkovaných dat. Data jsou ukládána podobně jako u DPCM. Poprvé je uložen nezměněný první vzorek a následují indexy do kvantizační tabulky vybraného kvantizátoru .[2]

3.2.4 A-law a μ -law

A-law a μ -law jsou zvukové komprimace (kodeky) definované CCITT, které komprimují 13 bitový vzorek na 8 bitů (např. -15e7). Pro Evropu bylo CCITT doporučeno používání 13 bitových A/D převodníků a poté se používá komprese vzorků podle A-křivky (v USA a Japonsku se používají μ -křivky). Při použití těchto kompresí je při rychlosti přenosu 64 kbit/sec chyba kvantování konstantní. A-křivka, viz obr. 16 je složena ze dvou částí takto: [13, s. 40]

$$y = \text{sign}(x) \cdot \frac{A \cdot |x|}{1 + \ln A} \quad \text{pro } 0 \leq |x| < \frac{1}{A} \quad (3)$$

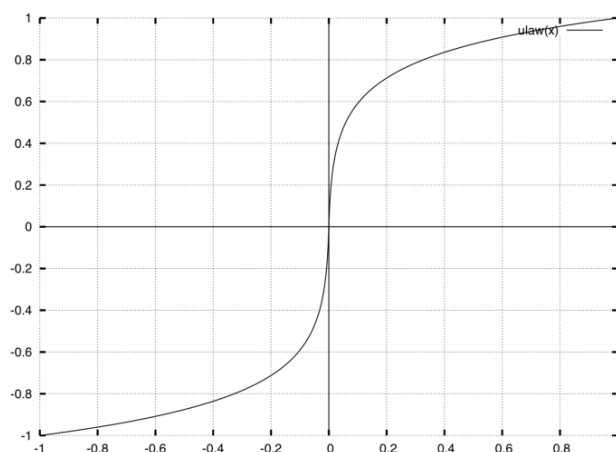
$$y = \text{sign}(x) \cdot \frac{1 + \ln(A \cdot |x|)}{1 + \ln A} \quad \text{pro } \frac{1}{A} \leq |x| < 1 \quad (4)$$



Obr. 16 - A-křivka [11, s. 14]

Hodnota A vyskytující se ve vzorcích 3 a 4 je rovna 87,6. μ -křivka je zobrazena na obr. 17 pro $\mu=255$ a její rovnice (5) [14]

$$y = \text{sign}(x) \cdot \frac{\ln(\mu \cdot |x|)}{\ln(1 + \mu)} \quad \text{pro } 0 \leq |x| < 1 \quad (5)$$



Obr. 17 - μ -křivka [12]

Kompresa a Expanze podle A-křivky

Z A/D převodníku získáme 13 bitový vektor: $\{P \ x_{11} \ x_{12} \ \dots \ x_0\}$ a musíme ho komprimovat na 8 bitový vektor: $\{P \ s_2 \ s_1 \ s_0 \ q_3 \ q_2 \ q_1 \ q_0\}$. V obou vektorech je P neměnicí hodnota určující znaménko.

Bity s_x jsou binární hodnota čísla vypočteného odečtením počtu nulových bitů v 13 bitovém vektoru od čísla 7. Při expanzi se postupuje opačným způsobem. Kompresa je zobrazena v tab. 13 a expanze v tab. 14. [13, s. 40]

Tab. 13 - Kompresa podle A-křivky [13, s. 40]

třináctibitové slovo													osmibitové slovo								
bit	12	11	10	9	8	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	x_{11}	x_{10}	x_9	x_8	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0		s_2	s_1	s_0	q_3	q_2	q_1	q_0	
P	0	0	0	0	0	0	0	0	q3	q2	q1	q0	-	P	0	0	0	q3	q2	q1	q0
P	0	0	0	0	0	0	1	q3	q2	q1	q0	-	P	0	0	1	q3	q2	q1	q0	
P	0	0	0	0	0	1	q3	q2	q1	q0	-	-	P	0	1	0	q3	q2	q1	q0	
P	0	0	0	0	1	q3	q2	q1	q0	-	-	-	P	0	1	1	q3	q2	q1	q0	
P	0	0	0	1	q3	q2	q1	q0	-	-	-	-	P	1	0	0	q3	q2	q1	q0	
P	0	0	1	q3	q2	q1	q0	-	-	-	-	-	P	1	0	1	q3	q2	q1	q0	
P	0	1	q3	q2	q1	q0	-	-	-	-	-	-	P	1	1	0	q3	q2	q1	q0	
P	1	q3	q2	q1	q0	-	-	-	-	-	-	-	P	1	1	1	q3	q2	q1	q0	

bity označené „-“ jsou při kompresi ztraceny.

Tab. 14 - Expanze podle A-křivky [13, s.41]

osmibitové slovo								třináctibitové slovo													
bit	7	6	5	4	3	2	1	0	12	11	10	9	8	7	6	5	4	3	2	1	0
	s_2	s_1	s_0	q_3	q_2	q_1	q_0		x_{11}	x_{10}	x_9	x_8	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	
P	0	0	0	q3	q2	q1	q0	P	0	0	0	0	0	0	0	q3	q2	q1	q0	1	
P	0	0	1	q3	q2	q1	q0	P	0	0	0	0	0	0	1	q3	q2	q1	q0	1	
P	0	1	0	q3	q2	q1	q0	P	0	0	0	0	0	1	q3	q2	q1	q0	1	0	
P	0	1	1	q3	q2	q1	q0	P	0	0	0	0	1	q3	q2	q1	q0	1	0	0	
P	1	0	0	q3	q2	q1	q0	P	0	0	0	1	q3	q2	q1	q0	1	0	0	0	
P	1	0	1	q3	q2	q1	q0	P	0	0	1	q3	q2	q1	q0	1	0	0	0	0	
P	1	1	0	q3	q2	q1	q0	P	0	1	q3	q2	q1	q0	1	0	0	0	0	0	
P	1	1	1	q3	q2	q1	q0	P	1	q3	q2	q1	q0	1	0	0	0	0	0	0	

3.2.5 Uložení zvukových dat

Zvuková data (nekomprimovaná) se ukládají, jak bylo zmíněno výše, do datového bloku. V případě jednoho kanálu jsou vzorky ukládány tak, jak jdou za sebou v čase. Pro dvoukanálová zvuková data se jednotlivé kanály střídají počínaje levým kanálem. U vícekanálového zvuku se jednotlivé kanály prokládají [1, s. 58], viz obr. 18 a obr. 19.

Sample 1	Sample 2	Sample 3	Sample 4
Channel 0	Channel 0	Channel 0	Channel 0

Data Packing for 8-Bit Mono PCM

Sample 1		Sample 2	
Channel 0 (left)	Channel 1 (right)	Channel 0 (left)	Channel 0 (right)

Data Packing for 8-Bit Stereo PCM

Obr. 18 - Uložení 8-bitového mono a stereo zvuku [1, s. 59]

Sample 1		Sample 2	
Channel 0 low-order byte	Channel 0 high-order byte	Channel 0 low-order byte	Channel 0 high-order byte

Data Packing for 16-Bit Mono PCM

Sample 1			
Channel 0 (left) low-order byte	Channel 0 (left) high-order byte	Channel 1 (right) low-order byte	Channel 1 (right) high-order byte

Data Packing for 16-Bit Stereo PCM

Obr. 19 - Uložení 16bitového mono a stereo zvuku [1, s. 59]

Všechny vzorky jsou ukládány v pořadí tzv. LittleEndian. Pokud má vzorek velikost např. 12 bitů tak bude doplněn na 16 bitů nulami a ukládán výše uvedeným způsobem. [1, s. 59].

4 PROGRAMOVÉ ZPRACOVÁNÍ

Pro naprogramování real-time přehrávače jsem využil dvou knihoven od společnosti Microsoft. První je Winmm.dll, ze které jsem používal waveOut* funkce a druhá je Dsound.dll, která je součástí DirectX SDK.

Ke každé z těchto knihoven bylo nutné pro splnění zadání přistupovat rozdílným způsobem, který je uveden v praktické části této práce. V této kapitole jsou uvedeny funkce které byly použity právě v části praktické.

4.1 Waveformat

Tato struktura je klíčová pro obě knihovny. Popisuje základní vlastnosti wav souboru, který se bude přehrávat, viz tab. 15 a je definována v hlavičkovém souboru Mmsystem.h.

Tab. 15 - Struktura waveformat [5]

Název	Velikost	Offset	Popis
wFormatTag	2	0x00	Kód komprese (PCM...).
nChannels	2	0x02	Počet kanálů (1 nebo 2).
nSamplesPerSec	4	0x04	Počet vzorků za sekundu.
nAvgBytesPerSec	4	0x08	Průměrný počet bytů za sekundu (2x2x 44100).
nBlockAlign	2	0x0C	Zarovnání datového bloku. Pro PCM data = počet bytů jednoho vzorku.

Je zřejmé, že tato struktura obsahuje informace obsažené z fmt bloku. Tato struktura byla nahrazena strukturou waveformatex, která je stejná jako předchozí, ale obsahuje navíc následující položky, viz tab. 16.: [5]

Tab. 16 - Přidané parametry ve struktuře waveformatex[5]

wBitsPerSample	2	0x0E	Průměrný počet bytů za sekundu (2x2x 44100)
cbSize	2	0x10	Zarovnání datového bloku. Pro PCM data je počet bytů jednoho vzorku

Strukturu waveformatex je možné použít pouze pro mono nebo stereo zvukové nahrávky s velikostí jednoho vzorku maximálně 16 bitů jak uvádí [5]. Proto byla definována rozšířená struktura waveformatextensible, která obsahuje tuto struktura a přidává potřebné rozšiřující informace, viz tab. 17.

Tab. 17 - Struktura *wavformatextensible* [5]

Název	Velikost	Offset	Popis
Format	18	0x00	WAVEFORMATEX struktura
wValidBitsPerSample	2	0x10	Stejný význam jako počet bitů na vzorek. Počet vzorků za obsaženého v jednom
wSamplesPerBlock	2	0x12	komprimovaném bloku.
wReserved	2	0x14	reservováno, 0.
dwChannelMask	4	0x18	Určení pozic kanálů v datovém toku pro správné určení reproduktoru.
SubFormat		0x1D	Subformát dat.

4.2 MME

Tato knihovna poskytuje nízko-úrovňové zpracování zvuku, které je možné použít v aplikacích pro ovládání audio zařízení. Tyto zařízení a struktury mají prefix „wave“.

Podle [6] se může při převzorkování zvukového toku na jinou vzorkovací frekvenci v některých případech objevit šum. Problém je způsoben tím, že převodník vzorkovací frekvence využívá lineární interpolace při převodu zvukové stopy. A právě toto vytváří šum ve zvukovém souboru, na který je lidské ucho citlivé.

Pro tuto chybu byla vydána oprava, viz [6], ale jen pro Windows 7 a server 2008 R2.

4.2.1 WaveOutOpen

Otevře dané zvukové zařízení pro přehrávání. Syntaxe funkce je následovná:

```
MMRESULT waveOutOpen(
    LPHWAVEOUT phwo,
    UINT uDeviceID,
    LPWAVEFORMATEX pwf,
    DWORD dwCallback,
    DWORD dwInstance,
    DWORD fdwOpen); [5]
```

Parametry předávané této funkci jsou popsány v tab. 18.

Tab. 18 - Popis parametrů funkce *WaveOutOpen* [5]

Parametr	Popis
Phwo	Handler, který identifikuje otevírané zvukové zařízení využívané v dalších funkcích.
uDeviceID	Identifikátor výstupního zvukového zařízení. WAVE_MAPPER je defaultní nastavení.
Pwfx	Ukazatel na waveformatex strukturu, která popisuje formát dat, která se budou na toto zařízení posílat.
dwCallback	Adresa callback funkce, handleru události, okna, nebo identifikátoru vlákna. Tyto jsou volány během přehrávání k poslání zprávy o jeho průběhu. Pokud se nevyužije ani jednoho z těchto mechanismů hodnota by se měla nastavit na 0.
dwInstance	Uživatelské data zpracovaná během callback mechanismu.
fdwOpen	Typ callback mechanismu. CALLBACK_NULL - bez callback funkce.

4.2.2 WaveOutPrepareHeader

Tato funkce jak název napovídá, připravuje strukturu wavehdr pro přehrávání. Struktura vypadá takto a její elementy jsou uvedeny v tab. 19.: [5]

```
typedef struct {
    LPSTR lpData;
    DWORD dwBufferLength;
    DWORD dwBytesRecorded;
    DWORD dwUser;
    DWORD dwFlags;
    DWORD dwLoops;
    struct wavehdr_tag* lpNext;
    DWORD reserved;}WAVEHDR; [5]
```

Tab. 19 - Popis elementů struktury wavehdr [5]

Parametr	Popis
lpData	Ukazatel na bufer, který musí být zarovnaný podle hodnoty nBlockAlign ve struktuře waveformatex.
dwBufferLength	Délka buferu v bajtech.
dwBytesRecorded	Když je hlavička používána pro nahrávání zvuku, zde bude specifikována velikost. V případě přehrávání zde bude hodnota přehrávaných dat z buferu.
dwUser	Uživatelská data.
dwFlags	Specifické vlastnosti o buferu např. WHDR_DONE.
dwLoops	Počet opakování.
lpNext	Používáno zvukovým ovladačem.
reserved	Rezervováno.

Parametry funkce jsou popsány v tab. 20 a syntaxe funkce je tato:

```
MMRESULT waveOutPrepareHeader(
    HWAVEOUT hwo,
    LPWAVEHDR pwh,
    UINT cbwh ); [5]
```

Tab. 20 - Popis parametrů funkce WaveOutPrepareHeader [5]

Parametr	Popis
hwo	Handler audio zařízení z funkce waveOutOpen.
pwh	Ukazatel na wavehdr strukturu.
cbwh	Velikost wavehdr struktury v bajtech

Jednou připravená data by se neměla měnit. Připravení již připravené hlavičky nemá žádný efekt a tato funkce vrátí nulu. A také by neměly dvě různá audio zařízení přistupovat ke stejné hlavičce.[5]

4.2.3 WaveOutWrite

Funkce, která pošle připravený blok dat danému audio zařízení pro přehrání. Funkce má uvedeny používané parametry v tab. 21 a má následující syntaxi:

```
MMRESULT waveOutWrite(
    HWAVEOUT hwo,
    LPWAVEHDR pwh,
    UINT cbwh ); [5]
```

Tab. 21 - Popis parametrů funkce WaveOutWrite [5]

Parametr	Popis
hwo	Handler audio zařízení z funkce waveOutOpen.
pwh	Ukazatel na wavehdr strukturu.
cbwh	Velikost wavehdr struktury v bajtech.

4.2.4 Funkce pro řízení toku přehrávání

WaveOutProc – funkce sloužící jako callback funkce při dokončení přehrávání dat. Adresa této funkce se objevuje ve funkci waveOutOpen. Syntaxe je uvedena níže a její parametry v tab. 22.:

```
void CALLBACK waveOutProc(
    HWAVEOUT hwo,
    UINT uMsg,
    DWORD dwInstance,
    DWORD dwParam1,
    DWORD dwParam2); [5]
```

Tab. 22 - Popis parametrů funkce WaveOutProc [5]

Parametr	Popis
hwo	Handler audio zařízení z funkce waveOutOpen.
uMsg	Zpráva od audio zařízení o průběhu přehrávání.
dwInstance	Uživatelská proměná;
dwParamX	Parametry zprávy.

V [5] jsou uvedeny funkce, které se mohou uvnitř této funkce použít bez vzniku deadlocku (funkce z oblastí kritických sekcí...)

WaveOutPause – funkce pozastaví přehrávání audio zařízení a uloží hodnotu pozice.

```
MMRESULT waveOutPause(  
    HWAVEOUT hwo ); [5]
```

WaveOutRestart – funkce obnoví pozastavené přehrávání audio zařízení. Přehrávání začne na uložené pozici

```
MMRESULT waveOutRestart(  
    HWAVEOUT hwo ); [5]
```

WaveOutReset – funkce zastaví přehrávání audio zařízení a nastaví pozici na 0.

```
MMRESULT waveOutRestart(  
    HWAVEOUT hwo ); [5]
```

4.2.5 WaveOutUnprepareHeader

Jedná se o funkci, která odpřipraví hlavičky po jejich přehrání. Funkce musí být volána před uvolněním buferu.

```
MMRESULT waveOutUnprepareHeader(  
    HWAVEOUT hwo,  
    LPWAVEHDR pwh,  
    UINT cbwh );
```

Funkce má stejnou syntaxi jako její protějšek waveOutPrepare. [5]

4.2.6 WaveOutClose

Funkce uzavře audio zařízení otevřené waveOutOpen.

```
MMRESULT waveOutClose(  
    HWAVEOUT hwo ); [5]
```

4.3 DirectX

Taktéž API od společnosti Microsoft. Poskytuje nejen interface pro ovládání audia, ale jedná se hlavně o funkce pro práci s grafikou. Jednou ze součástí tohoto API je knihovna zaměřená na zvuk, a to dsound.dll. Toto API je mladší a to díky používání grafické části především ve hrách, kde se postupem času vyvíjelo a stále vyvíjí.

4.3.1 IDirectSound8

Interface používaný pro vytvoření buferů, pro správu audio zařízení a nastavení prostředí. Přístup k interface získáme po použití funkce DirectSoundCreate8. Tato funkce má následující syntaxi a popis parametrů funkce je v tab. 23: [5]

```
HRESULT DirectSoundCreate8(
    LPCGUID lpcGuidDevice,
    LPDIRECTSOUND8 * ppDS8,
    LPUNKNOWN pUnkOuter) [5]
```

Tab. 23 - Popis parametrů funkce IDirectSound8 [5]

Parametr	Popis
lpcGuidDevice	Identifikátor audio zařízení. Defaultní hodnota je NULL
ppDS8	Adresa proměnné, která přijme vytvořený interface IDirectSound8.
pUnkOuter	Adresa objektu COM agregace. Musí být NULL.

4.3.2 IDirectSound8::SetCooperativeLevel

Metoda IDirectSound8 musí být volána ihned po vytvoření objektu a získání přístupu k interface. Metoda nastavuje úroveň spolupráce aplikace s audio zařízením. V podstatě se jedná o nastavení přístupu k audio zařízení. [5]

```
HRESULT SetCooperativeLevel(
    HWND hwnd,
    DWORD dwLevel) [5]
```

Parametry metody jsou zobrazeny a popsány v následující tabulce (tab. 24)

Tab. 24 - Parametry metody SetCooperativeLevel [5]

Parametr	Popis
dwLevel	Nastavení přístupu k audio zařízení. Doporučená hodnota je DSSCL_PRIORITY. Viz [5]

4.3.3 IDirectSound8::CreateSoundBuffer

Pro zavolání této metody a tedy jejímu použití musí předcházet vytvoření a nadefinování struktur DSBUFFERDESC a WAVEFORMATEX.

Struktura DSBUFFERDESC vypadá následovně a obsahující elementy jsou v tab. 25:

```
typedef struct DSBUFFERDESC {
    DWORD dwSize;
    DWORD dwFlags;
    DWORD dwBufferBytes;
    DWORD dwReserved;
    LPWAVEFORMATEX lpwfxFormat;
    GUID guid3DAlgorithm;
} DSBUFFERDESC; [5]
```

Tab. 25 - Elementy struktury DSBUFFERDESC [5]

Parametr	Popis
dwSize	Velikost struktury v bajtech.
dwFlags	Specifikuje možnosti buferu, viz [5]
dwBufferBytes	Velikost vytvářeného buferu v bajtech. Pokud se jedná o primární bufer pak musí být hodnota rovna nule.
dwReserved	Reservováno, 0.
lpwfxFormat	Adresa waveformatex struktury určující formát buferu.
Guid3DAlgorithm	Unikátní identifikátor 3d virtualizačního algoritmu. Defaultní hodnota je GUID_NULL .

Funkce pro vytvoření buferu vypadá následovně:

```
HRESULT CreateSoundBuffer(
    LPCDSBUFFERDESC pcDSBufferDesc,
    LPDIRECTSOUNDBUFFER * ppDSBuffer,
    LPUNKNOWN pUnkOuter) [5]
```

A její parametry jsou uvedeny v tab. 26.

Tab. 26 - Parametry metody CreateSoundBuffer [5]

Parametr	Popis
pcDSBufferDesc	Adresa DSBUFFERDESC struktury.
ppDSBuffer	Adresa proměnné, která obdrží interface IDirectSoundBuffer.
pUnkOuter	NULL

Po vytvoření buferu je nutné propojení s WAVEFORMATEX strukturou. Tuto operaci zajišťuje metoda setFormat:

```
HRESULT SetFormat(
    LPCWAVEFORMATEX pcfxFormat) [5]
```

4.3.4 IDirectSoundBuffer8::Lock

Připravuje část buferu pro zápis dat a vrací ukazatel na místo, kam se data mají zapsat. Metoda uzamkne část buferu pro zápis dat, nikdo jiný k těmto datům v tuto chvíli nemá přístup. Metoda má tuto syntaxi:

```
HRESULT Lock(
    DWORD dwOffset,
    DWORD dwBytes,
    LPVOID * ppvAudioPtr1,
    LPDWORD pdwAudioBytes1,
    LPVOID * ppvAudioPtr2,
    LPDWORD pdwAudioBytes2,
    DWORD dwFlags) [5]
```

Parametry metody jsou zobrazeny v tab. 27.

Tab. 27 - Parametry metody Lock [5]

Parametr	Popis
dwOffset	Offset v bajtech, od začátku buferu, kde zámek začíná.
dwBytes	Velikost zamčené oblasti. Bufer je navržen jako kruhový, takže hodnota může přesáhnout počet bajtu do konce buferu.
ppvAudioPtr1	Adresa proměnné, která získá ukazatel na první část zamčené oblasti.
pdwAudioBytes1	Adresa proměnné, která obdrží počet bajtů v bloku ppvAudioPtr1.
ppvAudioPtr2	Adresa proměnné, která získá ukazatel na druhou část zamčené oblasti.
pdwAudioBytes2	Adresa proměnné, která obdrží počet bajtů v bloku ppvAudioPtr2.
dwFlags	Hodnota určující možnosti jak zamčení oblasti provést.

Po provedení této metody se mohou data do buferu. A ihned po této operaci se musí zavolat metoda Unlock.

4.3.5 IDirectSoundBuffer8::Unlock

Odstraní zámek ze zamčené části buferu. Syntaxe je tato:

```
HRESULT Unlock(
    LVOID pvAudioPtr1,
    DWORD dwAudioBytes1,
    LVOID pvAudioPtr2,
    DWORD dwAudioBytes2) [5]
```

Parametry v této metodě jsou shodné s příslušnými parametry vytvořenými metodou Lock. Samozřejmostí je synchronizace v případě využití kruhového buferu se zapisováním dat tak, aby se shodoval počet parametrů v těchto dvou funkcích.[5]

4.3.6 IDirectSoundBuffer8::Play

Metoda zavolaná nad sekundárním buferu, který je naplnění daty, zahájí přehrávání na dané pozici s následujícími parametry, viz tab. 28.

```
void Play(
    int priority,
    BufferPlayFlags flags); [5]
```

Tab. 28 - Parametry metody Play [5]

Parametr	Popis
priority	Číslo určující prioritu zvuku používaného zvukovým managerem. Např. 0.
flags	Určuje chování buferu. Např. přehrávání ve smyčce.

Pro pozastavení přehrávání se zavolá nad buferem metoda *Stop()* a přehrávání se pozastaví. Pro opětovné spuštění je nutné zavolat výše popsanou metodu *Play*.

4.3.7 IDirectSoundBuffer8::GetCurrentPosition

Metoda získává kurzory, podle kterých se řídí přehrávání a zapisování dat do vytvořeného buferu. Syntaxe je uvedena níže společně s tab. 29, která zobrazuje její parametry: [5]

```
public void GetCurrentPosition(
    out int currentPlayPosition,
    out int currentWritePosition); [5]
```

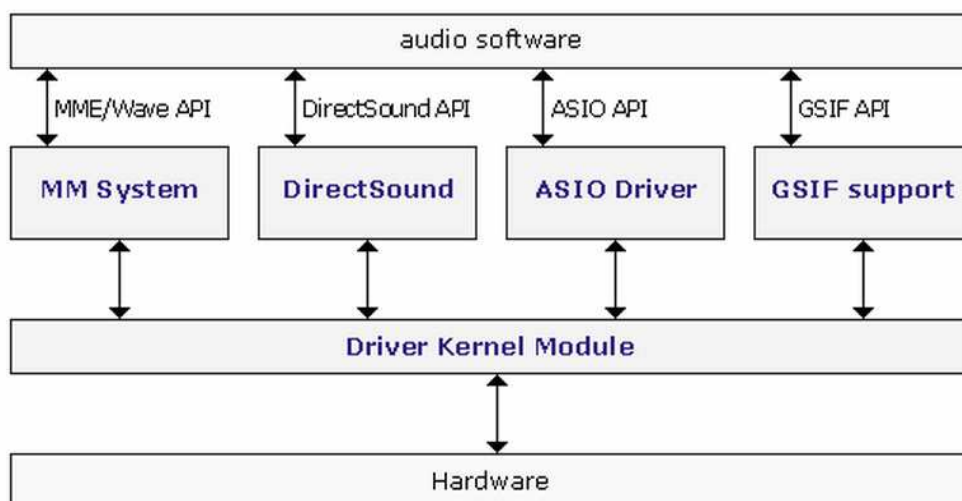
Tab. 29 - Parametry metody GetCurrentPosition [5]

Parametr	Popis
currentPlayPosition	Kursor určující aktuální pozici přehrávání.
currentWritePosition	Kursor určující aktuální pozici zapisového kurzoru.

Vztah mezi zapisovacím a přehrávajícím kurzorem je takový, že přehrávací kurzor je vždy před zapisovacím a na prostor mezi nimi není vhodné zapisovat. [5]

5 OSTATNÍ API

API je prostředek pro komunikaci aplikace a ovladačem. Na obr. 20 jsou zobrazeny 4 hlavní zvuková API, z nichž dvě jsou popsány podrobněji v předcházející kapitole. [10]



Obr. 20 - Zobrazení rozložení a komunikace mezi aplikací a ovladačem [10]

5.1 ASIO

ASIO bylo vytvořeno firmou Steinberg jako součást softwaru Cubase VST 3.5. V první verzi bylo oproti DirectSound velice pomalé a to jak při přehrávání tak i při pořizování záznamu. Hlavním cílem, ale nebylo dosáhnout rychlosti DirectSound, ale rozšířit o možnost vícekanálové zvuku. ASIO se stalo prvním, které umožňovalo vícekanálové zpracování zvuku. Navíc redukuje zatížení CPU při vícenásobném použití I/O kanálů a také zajišťuje synchronizaci mezi rozdílnými fyzickými I/O kanály.[10]

Druhá verze byla uveřejněna později a byly přidány funkce pro monitorování a kontrolu hardwaru. [10]

5.2 GSIF

Byl vytvořen firmou Nemesis (nyní Tescam). Stejně jako ASIO dovoluje přehrávání vícekanálového zvuku na jednom zařízení, ale nahrávání není možné. [10]

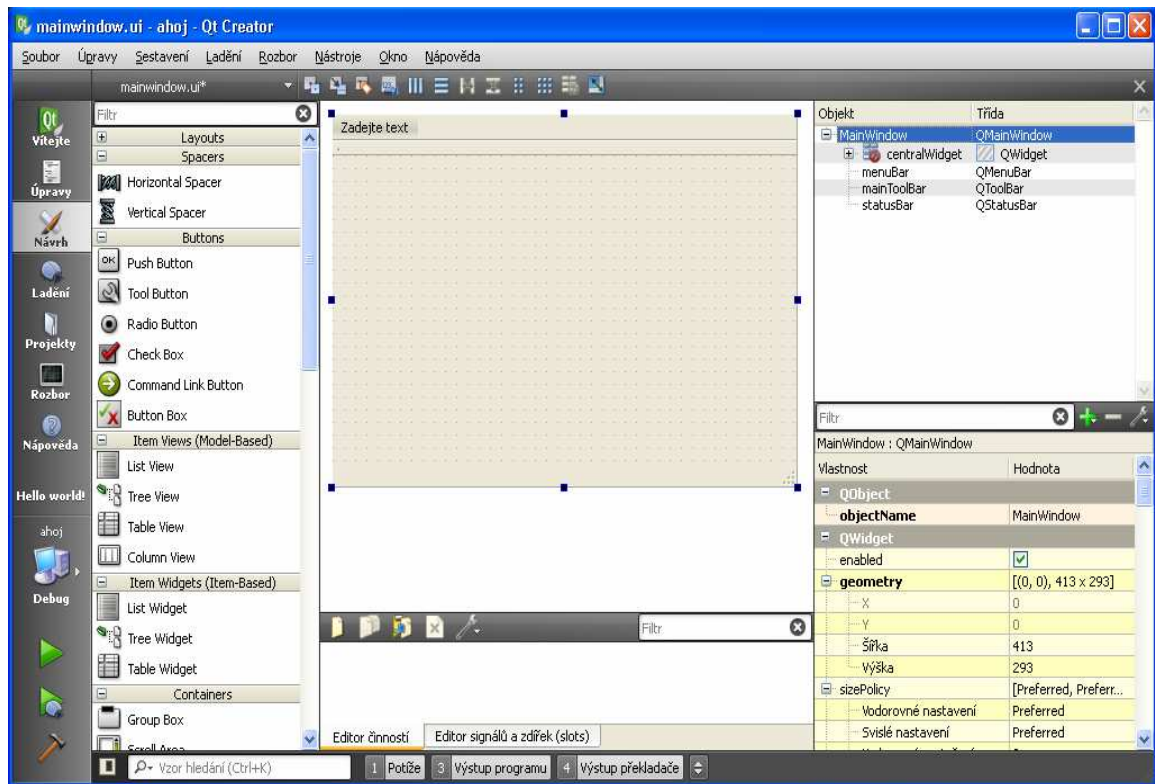
II. PRAKTICKÁ ČÁST

6 DOKUMENTACE K SOFTWARE

Program pro úpravu zvukových souborů typu wav v reálném čase, nazvaný WavePlayer, zpracovává tyto soubory podle daného zadání. Vývojové prostředí bylo zvoleno QT a použitý programovací jazyk C++.

6.1 Vývojové prostředí QT

QT je stále se rozvíjející multiplatformní vývojové prostředí obsahující velké množství integrovaných nástrojů, z nichž je nejpoužívanější tzv. QDesigner, viz obrázek (obr. 21), který je ve spojení s nástrojem QtCreator využíván pro vytvoření grafického rozhraní pro interakci s uživatelem. Používaná verze vývojového prostředí Qt je 5.0.0. a Microsoft Visual Studio 2010 verze 7.1.

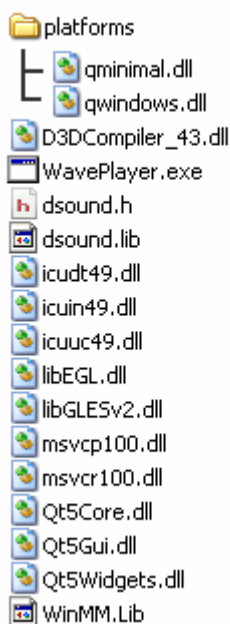


Obr. 21 - Vývojové prostředí QtCreator s otevřeným nástrojem QtDesigner

QT ve spojení s knihovnamy Microsoft Visual Studio je použitelný na platformě Windows. Samozřejmě QT nenabízí pouze podporu Windows, ale i Linuxu a Macintosh. A také podporu dalších programových jazyků (Python, Ruby atd.).

6.2 Instalace a spuštění aplikace

Aplikace WavePlayer je vytvořena pro operační systém Windows a je nutné pro její spuštění mít u samotného programu přiložené knihovny, jejichž seznam a struktura je vyobrazena na obrázku (obr. 22):



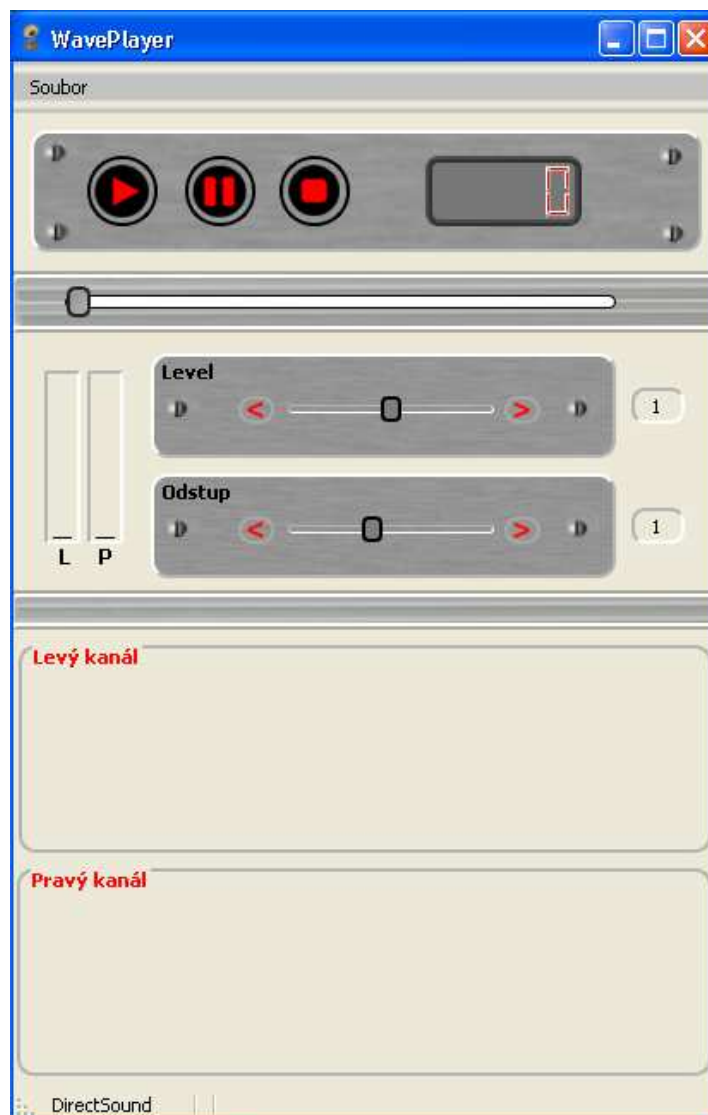
*Obr. 22 - Adresářová
struktura se soubory pro
spuštění aplikace*

Soubor přiložený v této práci s názvem WavePlayer.exe se otevře běžným způsobem a požadovaná adresářová struktura se automaticky vytvoří sama a poté se již spustitelný soubor WavePlayer.exe v takto vytvořené složce spustí a je možné ho používat.

Soubor nesmí být z této složky kopírován na jiná místa na disku. Přesun Aplikace je možný pouze jako celek s celou nadřazenou složkou.

7 GRAFICKÉ ROZHRAŇÍ

Cele programové rozhraní, viz obr. 23 je rozděleno do tří částí, které jsou v této kapitole popsány. Postup popisování je veden směrem dolů. Grafické zpracování jednotlivých prvků je prováděno za pomoci kaskádových stylů. Tuto možnost nabízí QtDesigner téměř každému prvku, se kterým je možné v jeho rámci pracovat, a to pomocí tzv. *StyleSheet*.



Obr. 23 - Grafický interface aplikace

7.1 Ovládaní

V první části je nástrojová lišta, která obsahuje menu *Soubor*, která se dále větví podle obrázku (obr. 24) na *Otevřít...*, *API* a *Zavřít*.



Obr. 24 - Nástrojová lišta s hlavním menu Soubor

Grafické zpracování nástrojové lišty je pomocí CSS následující:

```

QMenuBar { /* nastavení pozadí lišty*/
background-color: qlineargradient(spread:reflect, x1:0.494, y1:0.148,
x2:0.494364, y2:0.49, stop:0.113636 rgba(255, 255, 255, 0),
stop:0.3125 rgba(195, 195, 195, 255));}

QMenuBar::item {/* nastavení jednotlivých položek menu*/
spacing: 3px;
padding: 8px 10px;
background: transparent;
border-radius: 0px;
color: rgb(0, 0, 0);}

QMenuBar::item:pressed {/* nastavení grafiky při stisku položky*/
color: rgb(255, 255, 255);
background-color: rgb(40, 40, 40);
margin: 6px 0;
border:1px;
border-radius:6px;
border-color: rgb(255, 255, 255);
border-style:solid;}

QMenu::item {/* nastavení subpoložky v menu Soubor*/
padding: 2px 25px 2px 20px;
border: 1px solid transparent;}

/* nastavení subpoložky v menu Soubor při jejím vybrání*/
QMenu::item:selected {
border-color:rgb(0, 0, 0);
background: rgba(100, 100, 100, 150);}

```

V hlavní ovládací části jsou tlačítka pro kontrolu přehrávaných dat. Tlačítka Play, Pause a Stop společně s LCD displejem jsou spojena do jednoho bloku pomocí prvku GroupBox a opticky tak uzavírají tuto oblast, viz obr. 25.



Obr. 25 - Ovládání přehrávače s LCD displejem

Pro všechna tři tlačítka je použitý stejný kaskádový styl:

```
QPushButton {
border: 2px;
border-color: rgb(0, 0, 0);
border-style: inset;
border-radius: 20px;
border-style: solid;}

QPushButton: hover { /*nastylování tlačítka při přejetí myší*/
background-color: rgb(181, 181, 181);}

QPushButton: pressed { /*nastylování tlačítka při stisku*/
border: 4px;
border-top-color: rgb(84, 84, 84);
border-left-color: rgb(84, 84, 84);
border-bottom-color: rgb(84, 84, 84);
border-right-color: rgb(84, 84, 84);
border-style: ridge;}
```

Tlačítkům jsou pomocí QtDesigneru přidány ikony. Tyto i všechny další obrázky použité buď jako ikony, nebo pozadí jsou umístěny ve speciálním zdrojovém souboru Qt projektu ikony.qrc. Z tohoto souboru jsou poté linkovány na místa v daném grafickém prvku.

LCD displej je nastýlovaný css stylem takto:

```
QLCDNumber {
background-color: rgb(120,120,120);
color: rgb(255, 0, 0);
border-radius: 8px;
border: 4px;
border-color: rgb(50,50, 50);
border-style: ridge;}
```

7.2 Hlavní ukazatel

Hlavní ukazatel plní funkci ovládání toku dat a hlavně funkci zobrazení aktuální pozice přehrávaných dat. Grafika posuvnému je uvedena na obrázku (obr. 26).



Obr. 26 - Hlavní ukazatel programu

CSS je tento:

```

QSlider::groove:horizontal { /*nastylování drážky posuvníku*/
border:1px solid #000;
height:6px;
background: qlineargradient(x1:0, y1:0, x2:0, y2:1, stop:0 #B1B1B1,
stop:1 #c4c4c4);
margin:0px 0;
border-radius:4px;}

QSlider::handle:horizontal { /*nastylování manipulátoru posuvníku*/
background: qlineargradient(x1:0, y1:0, x2:1, y2:1, stop:0 #8f8f8f,
stop:1 #b4b4b4);
border:2px solid #222;
width: 10px;
margin: -6px 0;
border-radius: 5px;}

/*nastylování drážky před manipulátorem posuvníku*/
QSlider::sub-page:horizontal{
background: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:0,
stop:0 rgba(255, 255, 255, 255), stop:1 rgb(158, 158, 158));
border: 1px solid #000;
border-radius: 4px;}

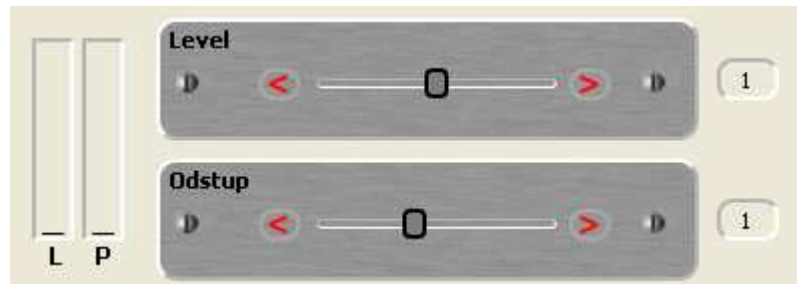
/*nastylování drážky za manipulátorem posuvníku*/
QSlider::add-page:horizontal {
background-color: white;
border: 1px solid #000;
border-radius: 4px;}

```

Posuvník je umístěn stejně jako ovládací prvky do GroupBoxu v němž je nastaveno pozadí na obrázek hlinik_poz.png umístění v projektové složce ikony.qrc.

7.3 Regulace

Regulační část zobrazený na obrázku (obr. 27) obsahuje dva posuvníky, z nichž každému přísluší jeden displej, na kterém se zobrazuje aktuální hodnota regulované veličiny. Pro přesnější regulaci je možné použít postranní tlačítka. Vlevo od posuvníků je umístěn ukazatel aktuální úrovně levého i pravého kanálu



Obr. 27 - Regulační část aplikace

Posuvníky a regulační tlačítka jsou umístěny do GroupBoxu a ten je nastýlován:

```
QGroupBox { /*nastýlování samotného GroupBoxu*/
border: 2px;
border-color: rgb(180, 180, 180);
border-style: outset;
border-radius: 10px;
background-image: url(/pozad_slider.png); }
```

```
QGroupBox::title { /*nastýlování titulku GroupBoxu*/
margin: 4px 6px;
color: rgb(0, 0, 0);
background-color: rgb(0, 0, 0, 0); }
```

Obsah GroupBoxu má vlastní kaskádové styly a to tyto:

```
QSlider::groove:horizontal { /*nastýlování drážky posuvníků*/
border: 1px inset;
border-color: rgb(180, 180, 180);
height: 3px;
border-radius: 2px; }
```

```
QSlider::handle:horizontal { /*nastýlování manipulátoru posuvníku*/
background-color: rgb(127, 127, 127);
border: 2px solid;
border-color: rgb(0, 0, 0);
width: 8px; }
```

```
margin-top: -6px;
margin-bottom: -6px;
border-radius: 4px;}

QPushButton { /*nastylování regulačního tlačítka*/
border: 2px;
border-color: rgb(168, 168, 168);
border-radius: 8px ;
border-style: solid;}

QPushButton: :hover { /*nastylování tlačítka při přejetí myší*/
background-color: rgb(255, 255, 255);}

QPushButton: :pressed { /*nastylování tlačítka při jeho stisku*/
border: 2px;
border-color: rgb(0, 0, 0);
border-style: solid;
background-color: rgb(178, 178, 179);}

```

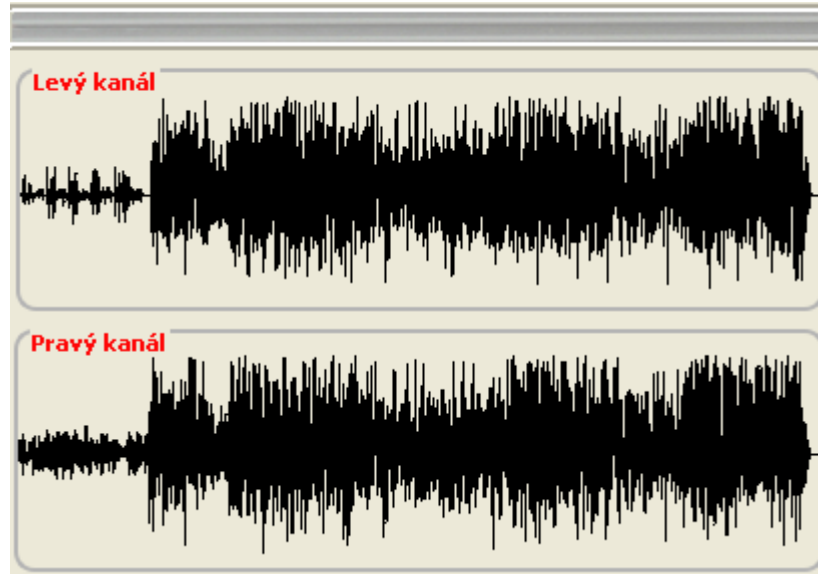
Ostatní prvky v tomto bloku jako je ukazatel úrovně a displeje pro aktuální hodnotu regulované veličiny jsou nastýlovány tímto způsobem:

```
border: 2px;
border-color: rgb(180, 180, 180);
border-style: inset;

```

7.4 Zobrazení

V této části jsou umístěny dva displeje, viz obr. 28, které zobrazují jednotlivé kanály (levý i pravý) jako funkci amplitudy závislou na čase.



Obr. 28 - Vykreslení průběhů jednotlivých kanálů

Styl použitý na těchto dvou GroupBoxech je takovýto:

```

QGroupBox { /*nastýlování samotného GroupBoxu*/
border: 2px;
border-color: rgb(180, 180, 180);
border-style: solid;
border-radius: 10px;
}
QGroupBox::title { /*nastýlování titulku GroupBoxu*/
color: rgb(255, 7, 7);
}

```

GraphicsView uvnitř těchto boxů mají styl:

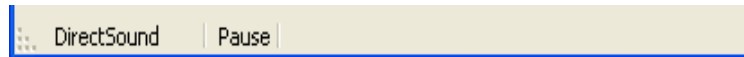
```

background-color: rgba(255, 255, 255, 0);
border-color: rgba(255, 255, 255, 0);
border-style: solid;

```

7.5 Status bar

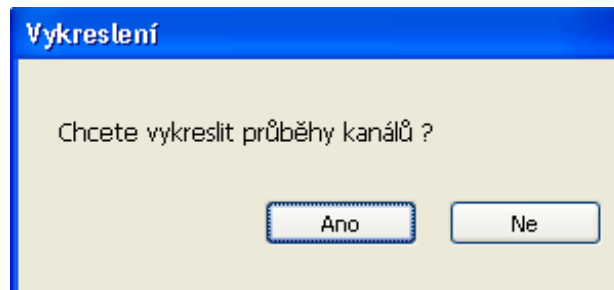
Statusbar hlavního okna obsahuje popisky s aktuálním druhem používaného API (Directsound a MME) a s popisem, který zobrazuje stav, ve kterém se přehrávač v daném času nachází (přehrávání, pause a stopnutí). Na následujícím obrázku (obr. 29) je zobrazen statusbar v jednom z možných stavů.



Obr. 29 - Statusbar

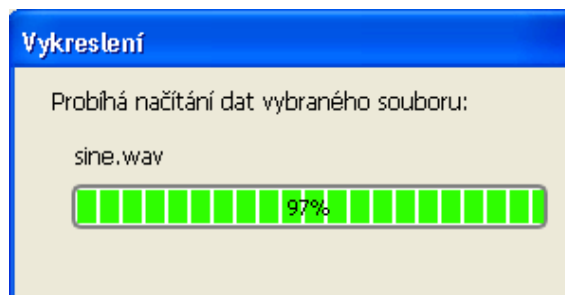
7.6 Vykreslovací dialog

Vykreslovací dialog je zobrazován v okamžiku, kdy je otevírán nový soubor pro přehrání v přehrávači. Dialog má základní rozvržení podle obrázku obr. 30.



Obr. 30 - Vykreslovací dialog v základním stavu

Možným stavem po stisku tlačítka *Ne* je zavření okna dialogu a nevykreslení průběhů. Druhá možnost nastane po stisku *Ano* tlačítka. Po stisku *Ano* tlačítka se přeorganizují prvky uvnitř dialogového okna. Tlačítka zmizí a objeví se progresbar, který zobrazuje, jaká část souboru byla zpracována a je připravena k vykreslení. Přeskupení prvků je zobrazeno na obrázku níže (obr. 31).



Obr. 31 - Přeskupení prvků při načítání dat na vykreslovacím dialogu

Použitý styl na Progresbar ukazatel je :

```

QProgressBar {
border: 2px solid grey;
border-radius: 5px;
text-align: center;}

QProgressBar::chunk {

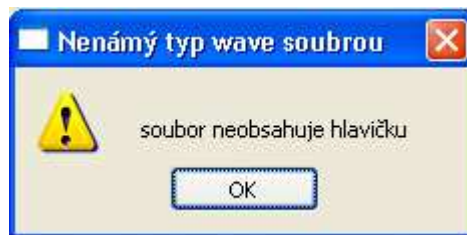
```

```
background-color: rgb(47, 255, 1);  
width: 10px;  
margin: 1px 1px;}
```

Ze záhlaví vykreslovacího dialogového okna jsou odebrány všechny ovládací prvky a byl ponechán pouze titulek, viz obr. 31.

7.7 Dialog upozornění

Dialog, viz obr. 32, se objevuje při otevírání souboru pro přehrávání. Pokud je vybrán soubor nesprávného formátu, anebo se nevybere žádný soubor, tak se zobrazí tento dialog s příslušným textem.



Obr. 32 - Dialog zobrazující upozornění na nesprávný formát vybraného souboru

8 TŘÍDY A METODY

Aplikace se skládá ze čtyř tříd - *Mainwindow*, *dxsound*, *winwave* a *nacteni*. Třída *Mainwindow* popisuje grafické rozhraní, jako jsou tlačítka, posuvníky a displeje. Další dvě třídy jsou třídy, které se starají o přehrávání samotného wav souboru. Poslední třída *nacteni* se stará o zpracování a vykreslování průběhů levého a pravého kanálu.

Program využívá dvou vláken, která běží současně při přehrávání hudebního souboru. Tuto koncepci bylo nutné použít z důvodu reakce tlačítek a všech ovládacích prvků v reálném čase. Bez použití tohoto principu by se veškeré ovládací prvky staly neaktivními a neplnily by svou funkci.

Základní rozdělení z pohledu vláken je tedy takové, že třída *Mainwindow* tvoří tzv. GUI vlákno, které běží neustále a třídy *dxsound* a *winwave* tvoří tzv. Work vlákna. Tyto tzv. Work vlákna jsou přepínána v závislosti na volbě uživatele a jejich rozdíl je uveden v následujících podkapitolách.

Projekt obsahuje soubor *main.cpp*, ve kterém je provedena základní propojení a vytvořeny potřebné instance daných tříd a vláken.

8.1 MainWindow

Třída zajišťující všechno potřebné okolo grafického výstupu z programového hlediska. Jedná se především o zpracování stisknutí tlačítek, zobrazování hodnot na displejích a posouvání manipulátorů do správných pozic. Všechny používané metody a sloty jsou uvedeny na obr. 33, které jsou popsány v následujících podkapitolách

Ve třídě je provedeno dělení metod a proměnných podle work vláken, vláken které se starají o samotné přehrávání zvukových souborů. Metody, které pracují s proměnnými a metodami z *dxsound* vlákna, jsou označeny prefixem “dx_“ v případě, že metoda pracuje s *winwave* metodami a proměnnými, jsou opatřeny žádným prefixem. Stejný postup byl zvolen i v případě, že metoda pracuje s oběma vlákny.

Rozdělení do více metod bylo voleno na základě přehlednosti a možnosti případné změny.

MainWindow	
#	closeEvent(event) :void
+	DejObj() :void
+	DejVlakno() :void
+	dx_DejObj() :void
+	dx_DejVlakno() :void
-	disable_API() :void
-	dx_play_pozice(int) :void
-	dx_enable_API() :void
-	dx_otevreni_souboru() :bool
-	dx_pause_tl() :void
-	dx_play_tl() :void
-	dx_set_slider() :void
-	dx_setSstop() :void
-	dx_slider_pres() :void
-	dx_slider_uvol() :void
-	dx_stop_tl() :void
-	enable_API() :void
-	minus_level() :void
-	minus_odstup() :void
-	on_actionDirectSound_triggered() :void
-	on_actionWIN_MME_triggered() :void
-	on_horizontalSlider_2_valueChanged() :void
-	on_horizontalSlider_3_valueChanged() :void
-	otevreni_souboru() :bool
-	pause_tl() :void
-	play_tl() :void
-	plus_level() :void
-	plus_odstup() :void
-	press_slider() :void
-	releas_slider() :void
-	reset_eq() :void
-	seek_pozice() :void
-	set_odstup(int) :void
-	set_uroven(in) :void
-	setslider() :void
-	spust_cas() :void
-	stop_tl() :void
-	update_level(short, short) :void
-	zobraz_level() :void
-	zobraz_prubeh(QPolygonF, QPolygonF) :void
+	zprava(QString, int) :void

Obr. 33 - Seznam všech metod a slotů v MainWindow třídě

8.1.1 DejVlakno a DejObjekt

Inicializační veřejné metody, které slouží pro předání ukazatelů objektů do GUI vlákna pro komunikace mezi tímto vláknem a work vláknem. Třída obsahuje taktéž „dx_“ ekvivalent plnící stejnou funkci.

8.1.2 Otevření_souboru

Soukromá metoda, starající se jak název napovídá o otevírání souboru. Vyvolává se zde OpenFileDialog s omezujícím filtrem pouze na soubory s koncovkou wav. Vybraný soubor je poté pomocí funkce `set_file` z třídy `winwave` zpracován a podle výsledku se může zahájit spuštění přehrávání. Před spuštěním přehrávání je uživatel dotázán na možnost vykreslení průběhů jednotlivých kanálů. Dále je zde provedeno ošetření některých prvků, např.: vynulování displeje času, resetování ukazatele úrovní, nastavení posuvníku a zobrazení stavu přehrávání.

Třída obsahuje „dx_“ ekvivalent, který pracuje principiálně stejně.

8.1.3 Play_tl

Soukromý slot spouštějící uživatelem vybrané vlákno. Obsahuje taktéž příslušné ošetření některých pomocných proměnných, zejména nulování STL kontejneru uchovávajícího maximální hodnoty přehrávaného bloku a podmínku pro první spouštění.

Třída obsahuje „dx_“ ekvivalent se stejnou funkčností.

8.1.4 Pause_tl

Soukromý slot zajišťující pozastavení a opětovné spuštění přehrávané hudby. Po stisku pause tlačítka se provede kontrola, jestli bylo tlačítko již stisknuté (jestli je přehrávání pozastavené). Po vyhodnocení podmínky se vyvolá signál, který pomocí propojení signal-slot, vyvolá v `dxsound` vlákne příslušnou „dx_“ akci. V případě `winwave` vlákna se provede přímo funkce na pozastavení nebo restartování přehrávání.

8.1.5 Stop_tl

Soukromý slot sloužící pro zastavení přehrávání wav souboru. Zde se stejně jako v případě pozastavení, vyvolává signál spouštějící v `dxsound` vlákne `dx_stop` slot a při použití `winwave` vlákna se volá funkce pro stopnutí přehrávání.

Navíc oproti pause tlačítku se provádí povolení změny používaného vlákna. Toto povolení je možné provést uživatelem z nástrojové lišty.

8.1.6 Set_slider

Soukromý slot, který je připojený na signál vyvolávaný ve *dxsound* vlákne a to vždy, když se blok dat přebral. Slot pro *winwave* vlákno používá funkci pro zjištění aktuální přehrávací pozice a je volán v pravidelných intervalech 40ms pomocí časovače. Slot aktualizuje hlavní posuvník a LCD displej.

V „dx_“ variantě slotu je kontrola pomocné proměnné, která zajišťuje zobrazení aktuální hodnoty manipulátoru posuvníku v době, kdy uživatel drží manipulátor a mění tak pozici, ze které se bude soubor přehrávat po uvolnění tlačítka myši. V druhé variantě je tato situace řešena za pomoci odpojení *set_slider* slotu od časovače a připojení slotu, který se o tuto činnost stará.

8.1.7 Press_slider

Soukromý slot, který při chytnutí manipulátoru posuvníku zajistí, aby se posuvník a LCD displej dále neaktualizovali, ale aby přehrávání ve *work* vlákne dále pokračovalo. Na LCD displeji je zobrazována aktuální hodnota pozice posuvníku přepočítaná na časové jednotky. „Dx_“ ekvivalent je nazván *dx_slider_pres*.

8.1.8 Releas_slider

Soukromý slot, který se provede po uvolnění tlačítka myši při posunu manipulátoru posuvníku. Jeho činností je změna pozice v přehrávaném hudebním souboru. To se provede zavoláním metody z *work* vlákna, která posun v souboru uskuteční.

Třída obsahuje „dx_“ variantu slotu jménem *dx_slider_uvol*.

8.1.9 Seek_slider

Soukromý slot používaný pro zobrazování pozice, když uživatel posouvá s manipulátorem posuvníku. Slot je používaný pouze při činnosti GUI vlákna s *winwave* vlákna.

8.1.10 Dx_setSstop

Soukromý slot, který využívá *dxsound* vlákno pro zobrazení pozice při posouvání manipulátoru uživatelem.

8.1.11 Update_level

Soukromý slot, ukládající maximální hodnoty dvou kanálů, které byly „poslány“ z work vlákna do STL kontejneru pro pozdější zpracování a vykreslení v ukazateli úrovní. Slot využívají obě work vlákna.

8.1.12 Zobraz_level

Soukromý slot, který vybírá z generického seznamu hodnoty a zpracovává je do vhodného tvaru pro zobrazení v ukazateli úrovní. Stejně jako předchozí slot je využíván oběmi work vlákny.

8.1.13 Zprava

Soukromá metoda vytvářející instanci třídy načtení a vytvoření vykreslovacího dialogového okna pro interakci s uživatelem. Metoda volá další metodu `zobraz_prubeh`, které předává dva parametry obsahující maximální hodnoty vrácené po zpracování ve třídě *nacteni*.

8.1.14 Zobraz_prubeh

Soukromá metoda vykreslující předaná data ve svých parametrech. Vykreslení je provedeno za pomoci převodu Polygonu na spojnici jednotlivých dat a její vykreslení na příslušnou vykreslovací plochu.

Dvě posledně uvedené metody jsou využívány jak vláknem *dssound* tak i *winwave*.

8.1.15 On_actionDirectSound_triggered

Slot pro přepnutí work vlákna z *dxsound* na *winwave*. Odpojuje „dx_“ sloty od signálů všech ovládacích prvků a připojuje ke slotům *winwave* vlákna.

8.1.16 On_actionWIN_MME_triggered

Slot, který stejně jako předchozí odpojuje sloty od signálů ovládacích prvků, ale v tomto případě v obráceném pořadí. „dx_“ sloty připojuje a sloty *winwave* vlákna odpojuje

8.1.17 Plus_odstup

Soukromý slot vyvolávaný při stisku tlačítka pro jemné doladění hodnoty odstupu. Slot regulovanou hodnotu zvyšuje

Stejnou funkčnost má soukromý slot *plus_level*.

8.1.18 Minus_odstup

Soukromý slot vyvolávaný při stisku tlačítka pro jemné doladění hodnoty odstupu. Regulovaná hodnota odstupu se zmenšuje.

Stejnou funkčnost má soukromý slot *minus_odstup*.

8.1.19 Reset_eq

Soukromá metoda resetující nastavení ukazatele úrovně kanálů do výchozího stavu pro následné přehrávání.

8.1.20 Enable_API

Soukromá metoda umožňující nastavit zpřístupnění změny API v nabídce *Soubor*→*API* pro *winwave* vlákno.

Ekvivalent s prefixem „dx“ plní stejnou funkci.

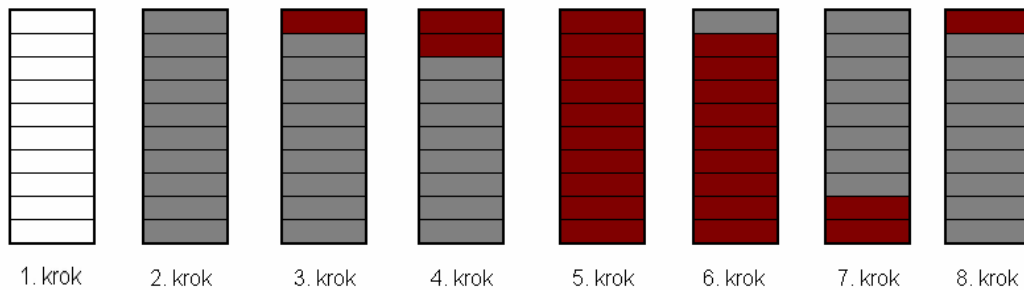
8.2 Dxsound

Pro přehrávání souboru je v této třídě využito knihovny *dsound.h* a funkcí, jež obsahuje. Konstruktor třídy obsahuje základní inicializace všech objektů potřebných pro přehrávání *wav* souboru.

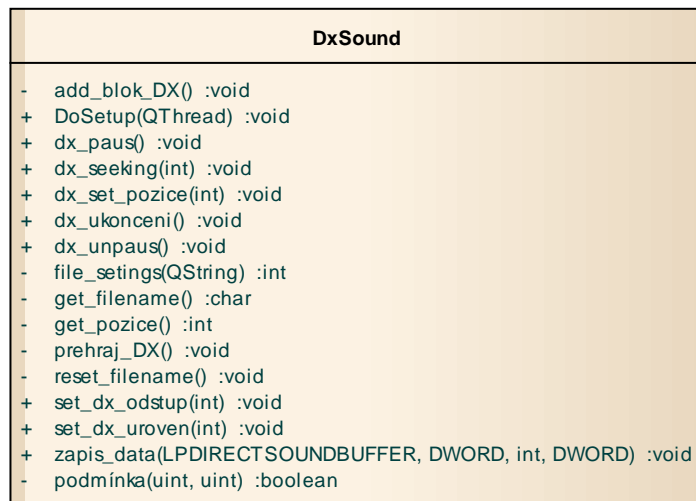
Základní představa o funkčnosti je taková, že po vytvoření nezbytných objektů se po stisku tlačítka pro spuštění přehrávání se zavolá slot *prehraj_DX*. Na jehož konci se spustí časovač s nastaveným intervalem (18ms). V těchto intervalech se opakovaně spouští slot *add_blok_DX*.

Při změně pozice manipulátoru posuvníku se změní přehrávací pozice a taktéž se nejprve zavolá *prehraj_DX* a poté se pokračuje výše popsáním způsobem. Postup přehrávání je popsán níže při popisu slotu *add_blok_DX*.

Hlavní přehrávací smyčka používá kruhový bufer uvedený na následujícím obrázku (obr. 34) pro zápis načítaných bloků data do buferu.



Obr. 34 - Kruhový bufer v jednotlivých krocích, kterými při běhu aplikace prochází



Obr. 35 - Metody a sloty DxSound vlákna

Na předcházejícím obrázku (obr. 35) jsou všechny metody používané při běhu vlákna při přehrávání.

8.2.1 Dxsound

Konstruktor třídy obsahuje inicializaci skládající se z těchto kroků:

- Vytvoření DirectSound objektu
- Nastavení výhradního práva na používání zvukového zařízení
- Vytvoření a nastavení primárního buferu
- Vytvoření sekundárního buferu

8.2.2 DoSetup

Veřejná metoda s účelem propojení signálu spuštění vlákna se slotem *prehraj_dx* a ukončení vlákna při vyvolání signálu *ukončit*.

8.2.3 prehraj_DX

Soukromý slot, který je proveden po vyvolání signálu spuštění vlákna. Obsahem slotu je otevření souboru s nastavenou cestou a načtení dat o velikosti sekundárního buferu. Kontrola velikosti načítaných dat je prováděna z důvodů uživatelských posunů manipulátorem posuvníku. Kontroluje se, jestli velikost načtených dat je menší jako velikost sekundárního buferu. Pokud tomu tak je, provádí se zarovnání nulami na nejbližší násobek velikosti bloku a nastaví se na tento násobek pomocná proměnná, která určuje ukončení cyklického přehrávání sekundárního buferu. Když velikost načítaných dat odpovídá velikosti buferu, potom se nastaví hodnota pomocné proměnné na počet bloků, na které je sekundární buffer alokován.

Data jsou načítána jako 16 bitová v LittleEndian pořadí a jsou dále upravována. První úpravou je násobení každého načteného členu hodnotou proměnné *dx_uroven*:

```
zasob[i]=zasob[i]*dx_uroven;
```

Druhá operace je složitější skládá se z rozdělení načtených dat na levý a pravý kanál a provedení výpočtů:

```
mono=(levy_kanal+pravy_kanal)/2;  
diff =(levy_kanal-pravy_kanal)/2;
```

Vytvořená proměnná *diff* se vynásobí proměnnou *dx_odstup* následuje zpětné přiřazení hodnot do pomocného buferu:

```
levy_kanal = mono+diff;  
pravy_kanal = mono-diff;
```

Upravená data budou zapsána do sekundárního bufferu pro přehrání.

Ve stejném cyklu se z celého bloku načtených dat vyhledává maximum levého a pravého kanálu a „odešle“ se GUI vláknu pro zobrazení na ukazateli úrovní. Podle obrázku (obr. 34), kde jsou uvedeny jednotlivé kroky je možné určit, že tento slot využívá pouze první dva kroky ve svém celém životním cyklu.

8.2.4 Add_blok_DX

Slot pro samotné přehrávání a zapisování dat do sekundárního buferu. Sekundární bufer je vytvořen jako kruhový bufer s předem nastavenou velikostí a podle obrázku (obr. 34) se cyklicky využívají kroky 3 až 8.

Slot je cyklicky volán pomocí časovače a obsahuje funkci pro zjištění pozice přehrávacího a zapisovacího kurzoru, na jejichž základě je prováděno rozhodnutí o dalším zápisu načteného bloku. Rozhodnutí se provádí v metodě *podmínka*, na jejímž výsledku záleží právě zápis dalších dat do buferu. Tímto způsobem se zabraňuje při náhodném vytížení cpu přeskokování zápisu dat na dané místo ve správném časovém okamžiku

Slot je volán pomocí časovače každých 18ms a to kvůli velikosti jednoho bloku. Při každém takovémto zavolání od časovače je provedena kontrola velikosti načtených dat a kontrola pozice přehrávacího kurzoru, jestli již byl přehrán blok. Pokud první blok byl celý přehrán, potom se aplikují stejné úpravy a zobrazovací operace jako v předcházejícím slotu na další blok a blok se zapíše na místo již přehraného bloku.

Po načtení všech dat ze souboru se neustále kontroluje přehrávací kurzor, ale neprovádí se načítání dalších dat. Provádí se pouze dekrementace pomocné proměnné nastavené ve slotu *prehraj_dx*. Když hodnota pomocné proměnné dosáhne hodnoty nula, přehrávání se ukončí.

Při splnění podmínky kruhového buferu je vždy aktualizován posuvník v GUI vlákně pomocí signálu *update_slider*.

8.2.5 Zapis_data

Metoda přebírající načtená a upravená data, jejich velikost ukazatel na sekundární buffer a offset pro zapsání dat na správné místo. Metoda zablokuje bufer, zapíšou se data a bufer odblokuje.

8.2.6 File_setings

Metoda volaná v GUI vlákně slotem *otevřít_soubor* pro zjištění informací z hlavičky souboru a nastavení cesty k souboru, který se bude přehrávat. Metoda ošetřuje případy, kdy soubor neobsahuje hlavičku vůbec nebo soubor je nepodporovaného typu. Když nastane

některá z těchto výjimek, vyvolá se informační okno s příslušným textem upozorňující na tuto skutečnost.

Při manipulaci s bufrem je využíván `MutexLocker` zajišťující jedinečný přístup v daném čase do stejné paměti (`dx_uroven` a `dx_odstup`). Použití je z důvodu předcházení deadlocku a dalších nepříjemností s tím spojených.

8.2.7 `Dx Seeking`

Veřejný slot používaný při posunu manipulatoru posuvníku uživatelem a provádí změnu pozice v souboru, ze které se budou načítat bloky pro další přehrávání. I zde je použit `MutexLocker`, a to ze stejného důvodu jako v předcházejícím případě.

8.2.8 `Set_dx_uroven`

Veřejný slot zajišťující nastavení hodnoty `dx_uroven` na hodnotu, která je slotu předána hodnotou z GUI vlákna pomocí posuvníku.

8.2.9 `Set_dx_odstup`

Veřejný slot zajišťující nastavení hodnoty `dx_odstup` na hodnotu, která je slotu předána hodnotou z GUI vlákna pomocí posuvníku.

Ve slotech `set_dx_odstup` a `set_dx_uroven` je použit `MutexLocker` pro odstranění možnosti výskytu deadlocku při zápisu do těchto proměnných.

8.2.10 `Dx_pause`

Veřejný slot, který pozastaví přehrávání zvukových dat v buferu a vypne časovač starající se o kontrolu a zápis dalších dat.

8.2.11 `Dx_unpause`

Při pozastavení přehrávání se po opětovném stisku tlačítka pause, vyvolá tento soukromý slot, který opět spustí časovač a spustí se přehrávání.

8.2.12 Dx_ukoncení

Slot, který po jeho vyvolání ukončí přehrávání, vypne časovač a uzavře přehrávaný soubor. Po těchto operacích ukončí vlákno zavoláním metody quit.

8.2.13 Dx_set_pozice

Nastavení pomocné proměnné používané při přehrávání. Slot je využíván pro nulování této proměnné.

8.2.14 get_filename

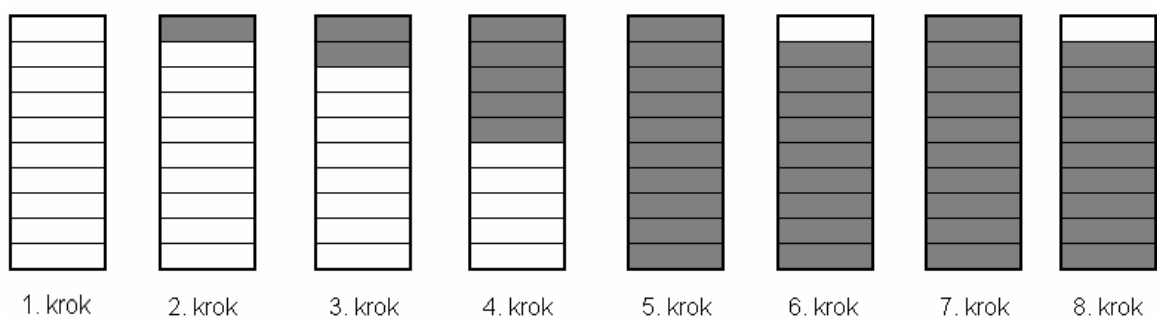
Veřejný slot pro vrácení cesty k přehrávanému souboru.

8.3 Winwave

V této třídě je použita knihovna winmm.dll a funkce s prefixem waveout. Základní algoritmus přehrávání je takovýto:

- Alokování pole o velikosti buferu, který je kruhový, viz obr. 36.
- Načtení prvního bloku dat do vytvořeného buferu.
- Provedení úprav na tomto bloku dat
- Poslání upraveného bloku dat do funkce, která pošle data na zvukové zařízení
- Kontrola plnosti buferu a dokončení přehrátí jednotlivých bloků

Uvedený algoritmus je inspirován postupem, který byl vytvořen v tutoriálu Davida Owertona, viz [9]. Tento postup je v méně srozumitelné formě uveden i na oficiálních stránkách Microsoft, viz [5].



Obr. 36 - Postup naplňování a vyprazdňování kruhového buferu

Seznam použitých metod a slotů třídy je na obr 37.

Winwave	
+	DoSetup(QThread) :void
+	file_setings(QString) :int
+	get_filename() :char
-	Prehraj() :void
+	reset_filename() :void
+	set_position(int) :void
-	waveOutProc(HWAVEOUT, UINT, DWORD, DWORD, DWORD) :void

Obr. 37 - Metody a sloty třídy WinWave

8.3.1 Winwave

Konstruktor třídy neobsahuje žádnou nutnou inicializaci pro přehrávání, ale pouze počáteční nastavení proměnných na výchozí hodnoty.

8.3.2 DoSetup

Veřejná metoda má stejnou funkčnost jako u třídy *dsound*.

8.3.3 File_setings

Veřejná metoda, sloužící při otevření souboru, k jeho prohledání a zjištění informací uložených v hlavičce. Při nepřítomnosti hlavičky, nebo nepodporovaných parametrů se objeví informační okno s upozorněním na danou výjimku. Provádí se zde také nastavení cesty k souboru, který bude přehráván.

8.3.4 Prehraj

Slot pro samotné přehrávání. Na začátku jsou umístěny inicializace všech pomocných proměnných nastavení formátu waveformatex. Následuje otevření souboru a přiřazení do *DataStream* pro možnost čtení dat ne po jednotlivých bajtech, ale rovnou po *LittleEndian* dvoubajtových hodnotách. Alokace paměti pro jednotlivé bloky kruhového buferu.

Na dalších řádcích slotu je přehrávací smyčka, která obsahuje:

- Načítání bloku dat a jeho úpravy – úpravy jsou shodné s úpravami v *dxsound* třídě, tzn. vynásobení každého načteného dvoubajtového prvku proměnnou úroveň a následné rozdělení celého buferu na dva kanály, kde po operaci:

```
mono =(levy_kanal+pravy_kanal)/2;  
diff =(levy_kanal-pravy_kanal)/2;
```

je hodnota diff vynásobená odstup proměnou a hodnoty jsou podle následujícího kódu zpětně přiřazeny do buferu.

```
levy_kanal = mono-diff;  
pravy_kanal = mono+diff;
```

Při rozdělování bloku dat je zároveň hledána maximální hodnota levého a pravého kanálu a výsledek je „poslán“ do GUI vlákna pro uložení, zpracování a zobrazení na ukazateli úrovní.

- Upravená data jsou zkopírována do kruhového buferu a tento je poslán do funkce zajišťující jejich přehrání.
- Kontrola plnosti buferu a dokončení přehrání jednoho bloku je prováděna za pomoci callback funkce WaveOutProc, která je volána právě v případě, když bylo dokončeno přehrávání bloku dat. V této callback funkci je zvyšována pomocná proměnné, podle které je při plnosti kruhového buferu povoleno zapsat na uvolněnou pozici nový blok dat. Pro kontrolu pomocné proměnné je nutné použít ochranu proti vícenásobnému přístupu do paměti. V tomto případě nebylo možné použít mutex, a to z důvodu popsaných v oficiální dokumentaci. Proto byla použita kritická sekce, která plní podobnou funkci jako mutex.

8.3.5 Set_position

Slot sloužící pro nastavení pozice souboru, od kterého se začnou načítat data po posunu manipulátoru posuvníku uživatelem.

8.4 Nacteni

Třída pro zpracování dat hudebního souboru. Z GUI vlákna získává cestu k souboru data získaná z tohoto souboru prochází a hledá v určeném množství maximum a to ukládá do STL kontejneru, který poté předán zpět do Gui vlákna. Metody a sloty třídy jsou graficky znázorněny na obr. 38.

Nacteni
+ get_levy() :QPolygon
+ get_pravy() :QPolygon
+ nastav_cestu(QString) :void
+ nastav_posun(int) :void
+ zpracuj() :void
- Ano() :void
- Ne() :void

Obr. 38 - Metody a sloty

třídy Nacteni

8.4.1 Nastav_cestu

Veřejná metoda nastavující cestu pro soubor v této třídě. Cesta je předána jako parametr z Gui vlákna, kde se vytváří instance této třídy.

8.4.2 Nastav_posuv

Veřejná metoda nastavující posun začátku, ze kterého se budou data zpracovávat. Posun je prováděn, aby data umístěná v hlavičce neovlivňovala vykreslovaný průběh

8.4.3 Ano

Soukromý slot vyvolaný při stisku tlačítka Ano. Přeskupuje prvky zobrazené na vykreslovacím dialogu do podoby, viz obr. 30 a volá metodu na zpracování načtených dat zpracuj().

8.4.4 Ne

Soukromý slot vyvolaný při stisku tlačítka Ne. Uzavírá vykreslovací dialogové okno.

8.4.5 Zpracuj

Soukromá metoda třídy, která otevírá soubor s nastavenou cestou a začíná číst data z nastavené pozice. Velikost souboru je vydělena 20 000 se zaokrouhlením nahoru, aby se dosáhlo počtu výsledných vzorků asi 5000. Dělitel slouží v procházcím cyklu k určení kdy se má maximální hodnota uložit do STL kontejneru a vynulovat pro opakované použití. Maximální hodnota se získává pro levý i pravý kanál hudebního souboru. Maximální hodnoty se násobí konstantou pro lepší zobrazení na vykreslovací ploše.

8.4.6 Get_levy

Veřejná metoda vracející STL kontejner obsahující přibližně 5000 maximálních hodnot levého kanálu získaných při procházející smyčce.

8.4.7 Get_pravy

Veřejná metoda, která vrací STL kontejner obsahující přibližně 5000 maximálních hodnot pravého kanálu, které byly získané při procházení souboru.

9 POPIS POUŽITÍ

Způsob spuštění aplikace je popsán v kapitole uvedené výše. V této kapitole uveden popis použití aplikace. Ovládání aplikace je velice intuitivní a uživatelsky přívětivé. V průběhu testování byl vygenerován testovací soubor s názvem sine.wav o délce 1s, který obsahuje zvukovou nahrávku s hlavičkou a nekomprimovanými daty s těmito parametry: 16 bitové vzorky, 2 kanály a 44100 Hz. Přehrávač podporuje právě takovéto wav soubory.

Po spuštění aplikace se zobrazí hlavní okno s výše popsányými ovládacími prvky pro kontrolu proudu zvuku. Na níže uvedeném obrázku (obr. 39) je uveden vývojový diagram aplikace. Podle obr. 38 má po spuštění aplikace 3 možnosti jak pokračovat:

Změna API (DirectSound/ MME) v nabídce Soubor - Při změně API se pouze přepne kontext (provede se změna vlákna, které se bude starat o přehrávání)

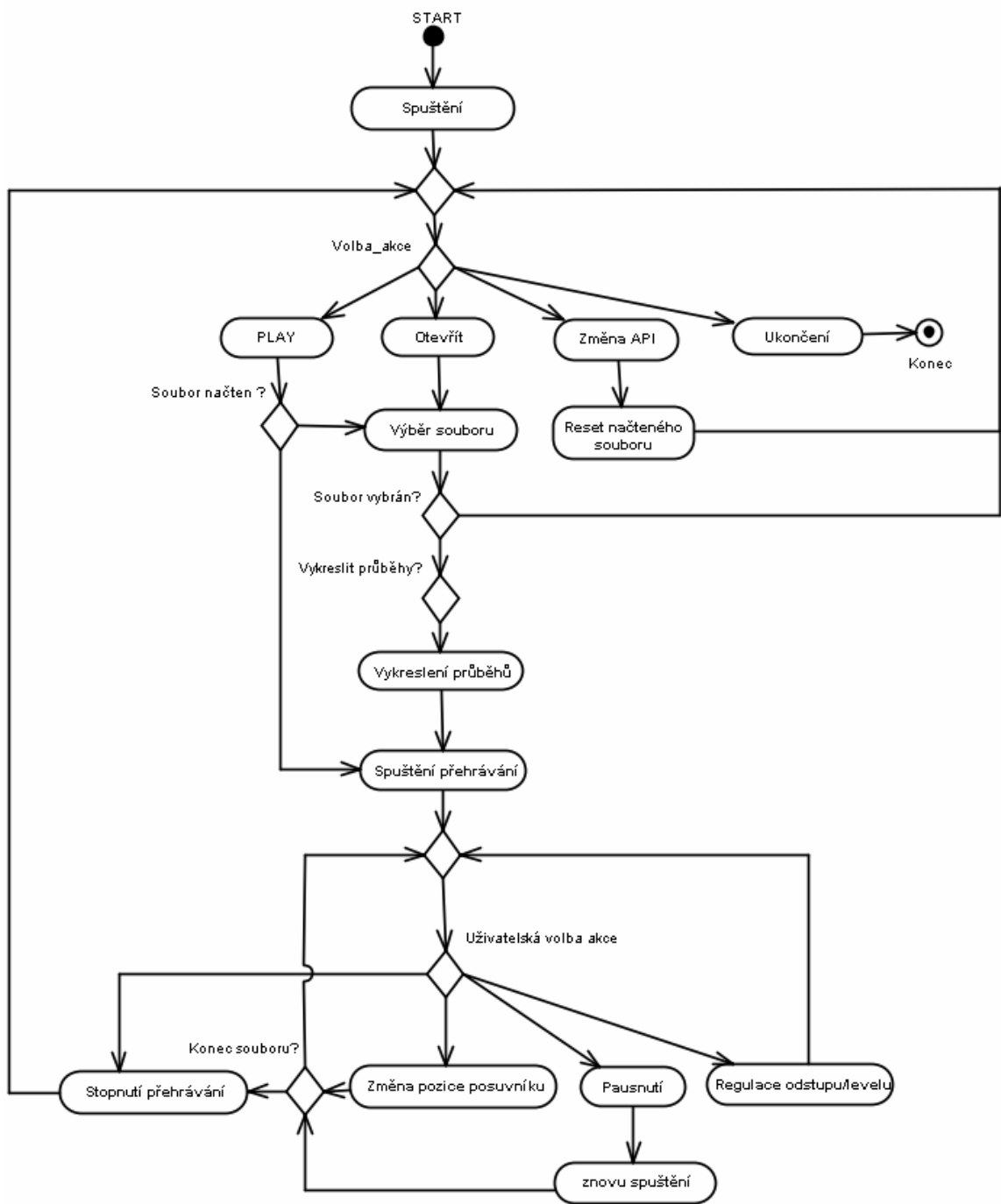
Otevření souboru pro přehrávání v nabídce Soubor – Otevře se OpenFileDialog, pomocí něho je možné vybrat si, jaký soubor bude přehráván. Pokud se soubor nevybere, vrátí se program do předchozího stavu a uživatel musí vybrat soubor znovu.

Stisknutím tlačítka pro spuštění přehrávání – Při stisku se zavolají stejné metody jako v předchozím případě. Toto chování se provede pouze jednou a to vždy, když je vybrán nový soubor.

Po výběru souboru je uživatel dotázán na možnost vykreslení průběhů levého a pravého kanálu. Při otevření souboru pomocí tlačítka play se soubor přímo začne přehrávat a v případě otevření pomocí nabídky *soubor*→*otevří* je nutné spustit přehrávání tlačítkem play.

Při přehrávání je možné regulovat hodnoty odstupu a úrovně zvuku. Dále pak pomocí tlačítka pause a stop pozastavovat a ukončovat přehrávání. V průběhu přehrávání jsou zobrazovány maximální úrovně levého a pravého kanálu.

Hlavním ukazatelem aplikace je posuvník, který společně s LCD displejem zobrazuje aktuální přehrávací pozici přehrávaného souboru. Pomocí toho posuvníku lze taktéž měnit aktuální přehrávanou pozici. Pozici je možné měnit jednak stisknutím manipulátoru posuvníku a přesunutím na požadované místo a jednak s využitím prostředního tlačítka myši. A to tak, že se stiskne prostřední tlačítko myši nad pozicí, na kterou chce uživatel posunout přehrávání.



Obr. 39 - Vývojový diagram aplikace

10 TESTOVÁNÍ A DOSAŽENÉ VÝSLEDKY

V průběhu psaní částí programu byla zařazena testování a kontroly funkčnosti. Takovéto testování funkčnosti bylo prováděno přírůstkovou metodou, které je v těchto případech ideální.

Pro testování byl vytvořen testovací soubor z názvem audio_test.wav, který byl nahrán za pomoci nástroje Záznam zvuku od společnosti Microsoft. Parametry souboru obsahující hlavičku jsou stejné jako v případě vygenerovaného souboru sine.wav.

10.1 Testování

Testování funkčnosti a stability jak jednotlivých funkcí, tak i celého programu byla testována následujícími způsoby:

Přehrávač byl podroben testu pro přehrávání v časovém intervalu 2 hodin nepřetržitého provozu.

Přehrávač byl spuštěn a po dobu 10 minut se provádělo posouvání manipulátoru posuvníku. Tento test se tedy zaměřoval na ověření konzistence a přístupu do sdílené paměti.

Posledním prováděným testem, bylo opakované otevírání, spouštění a stopnutí nahrávky. Během testu se ověřovala funkčnost rozpoznání nesprávného typu wav souboru, možnost vykreslení a správné ukončení nahrávky.

Všechny tyto testy probíhaly za pomoci mnou vytvořeného slotu testing, který je vyvolán za pomoci timeru s danou periodou.

10.1.1 Testing

Soukromý slot, který byl v průběhu testování měněn podle potřeb každého test stejně jako perioda časovače.

První test pro přehrávání po dobu přibližně dvou hodin. Slot vypadal takto:

```
// pro directsound vlákno
dx_play();
// pro winwave vlákno
play();
```

Druhý test pro změnu pozice manipulátoru posuvníku po dobu přibližně deseti minut. Zátěžový testovací slot vypadal takto:

```
if (test){ // pro directsound vlákno
    dx_slider_pres();
    ui->horizontalSlider->setValue(2000);
    dx_slider_uvol();
    test=0;
}else{
    dx_slider_pres();
    ui->horizontalSlider->setValue(3000);
    dx_slider_uvol();
    test=1;
}
if (test){ // pro winwave vlákno
    press_slider();
    ui->horizontalSlider->setValue(2000);
    releas_slider();
    test=0;
}else{
    press_slider();
    ui->horizontalSlider->setValue(3000);
    releas_slider();
    test=1;
}
```

Poslední ze zátěžových testů, test na opakované otevírání, byl prováděn ručně v několika desítkách opakování.

10.2 Dosažené výsledky

Průběh jednotlivých zatěžovacích testů byl sledován a výsledky těchto testů dopadly úspěšně. Ani při jednom z výše uvedených testů se nevyskytl destruktivní problém, při kterém by aplikace neočekávaně ukončila svoji činnost. Při normálním a zatíženém běhu nevznikaly úniky paměti a vytížení procesorů na testovací sestavě bylo v odpovídajícím rozsahu.

Program by mohl být používán nejen jako přehrávač s dalšími úpravami v reálném čase, ale i jako jakýsi „přednástroj“ před použitím sofistikovanějšího a propracovanějšího audio nástroje.

ZÁVĚR

Diplomová práce zahrnuje zpracování a přehrávání zvukového souboru typu wav. Program vytvořený v této práci využívá dvou knihoven DirectSound a MME. Tyto knihovny byly použity pro jejich dobře zpracovanou dokumentaci. Volbu knihovny pro přehrávání provádí uživatel v hlavní nabídce programu. Program využívá tři vlákna, ale v každém okamžiku jsou aktivní pouze dvě. Jedno vlákno je GUI vlákno, které vykresluje ovládací a zobrazovací prvky a druhé přehrává a upravuje zvuk.

Úpravy zvuku v reálném čase jsou aplikovány na načítané bloky dat z vybraného souboru. Samotné přehrávání je prováděno pomocí knihovnických funkcí a načtené bloky dat jsou umísťovány do kruhového buferu. Tento způsob načítání a úprav je vhodný pro další rozšíření funkčnosti. Grafický interface používaný pro interakci s uživatelem a je nastýlovaný pomocí kaskádových stylů.

V teoretické části jsou popsány základní fyzikální veličiny z oboru zvuku a jeho zpracování. V další části je popsána vnitřní struktura wav souboru. Na konci teoretické části jsou zmíněny další používané API pro přehrávání a záznam zvuku s historickým vývojem.

Praktická část začíná popisem vývojového prostředí QT a způsobem spuštění programu. Následuje popis grafického rozhraní s definovanými kaskádovými styly. Dále jsou uvedeny třídy a metody používané v programu. Ke konci praktické části práce je zobrazen vývojový diagram životního cyklu programu. Na závěr byl popsán průběh testování stability a výkonnosti konečného výsledku.

Cílem práce bylo nastudování informací o wav formátu a vytvoření programu, který by prováděl zpracování zvukových bloků dat v reálném čase. Vytvořený program, který jednotlivé bloky předzpracovává, toto tvrzení víceméně potvrzuje. V diplomové práci jsou aplikovány změny úrovně a odstupu kanálů přehrávaného zvuku. Pro další možné úpravy je nutné přidat požadovanou změnu do kódu programu. Další možné úpravy jsou např.: komprese a expanze dynamiky.

Vytvořený program má mnoho dalších využití jako, například by mohl tento program sloužit pro zjištění potřebných hodnot a tyto hodnoty by se poté aplikovali v propracovanějším audio softwaru na celý výstupní soubor, nebo jeho část.

ZÁVĚR V ANGLIČTINĚ

The thesis includes processing and playback of audio WAV files. Program created in this thesis uses two libraries - DirectSound a MME. These libraries were used for their good documentation. Selection of threads is done in the toolbar by user. The program uses three threads, but in every moment are active only two. One thread is the GUI thread that renders controls and displays, and the second one plays and modifies the sound.

Real-time editing of audio are applied to the loaded data blocks from the selected file. Library functions are used for playback of sound and data blocks are stored in circular buffer. This method of loading and modifications are appropriate to further extension of functionality. Graphical interface is used to interact with the user and its styled with CSS.

In the theoretical part are described the basic physical quantity of the sound and its processing. In the next section is described the internal structure of the wav file. At the end of the theoretical part are mentioned another API for playback and recording with their historical development.

The practical part describes the QT framework and a way how to run the created program. Next is a description of the graphical interface with defined CSS styles In this thesis are next the classes and methods used in the program. Towards the end of the practical part is a flowchart of program's life cycle. At the end was described the process of stability testing and performance of the final result.

The aim of this thesis was study information about wav file and create a program, that would processing of audio data in real time. The created program that preprocesses each block, confirms this statement. In the thesis are used changes in the levels and distance of audio channels. For another possible adjustments are necessary make changes in the code. Possible modifications are for example, Dynamic range compression or Dynamic range expansion

Created Program has many uses. For example, this program could be used to determine the required values. These values could be applied in more sophisticated audio software on all data in the output file, or part of it.

SEZNAM POUŽITÉ LITERATURY

- [1] BARČOVÁ, Karla, JANUROVÁ, Milada KOPEČNÁ, Milena KUŠNEROVÁ a Radim UHLÁŘ. Sbíрка úloh z fyziky [online]. I. vydání. Ostrava: VŠB - Technická univerzita Ostrava, 2006 [cit. 2013-05-03]. ISBN 80-248-1197-9. Dostupné z: http://www.studopory.vsb.cz/studijnimaterialy/Sbirka_Fyzika
- [2] BENEŠ, Miroslav. Komprese dat. In: Vysoká škola báňská - Technická univerzita Ostrava [online]. 2003 [cit. 2013-05-03]. Dostupné z: <http://www.cs.vsb.cz/benes/vyuka/pte/texty/kompresse/index.html>
- [3] Kvantování. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-05-03]. Dostupné z: [http://cs.wikipedia.org/wiki/Kvantov%C3%A1n%C3%AD_\(sign%C3%A1l\)](http://cs.wikipedia.org/wiki/Kvantov%C3%A1n%C3%AD_(sign%C3%A1l))
- [4] MEDVECOVÁ, Ivana. PŘÍRUČKA PRO ZAČÁTEČNÍKY: Základy akustiky [online]. 01.11.2011 [cit. 2013-05-03]. Dostupné z: <http://www.greif.cz/download/its075-zaklady-akustiky-prirucka-pro-zacatecniky.pdf>
- [5] MICROSOFT. Windows Desktop Development [online]. 2013 [cit. 2013-05-03]. Dostupné z: <http://msdn.microsoft.com/en-us/library/windows/desktop>
- [6] MICROSOFT. Audio file sounds distorted after you convert the sample rate in Windows 7 or in Windows Server 2008 R2. In: Web podpory Microsoft [online]. 2013 [cit. 2013-05-03]. Dostupné z: <http://support.microsoft.com/kb/2653312/en-us>
- [7] Multimedia Programming Interface and Data Specifications 1.0 [online]. 1991 [cit. 2013-05-03]. Dostupné z: <http://www-mmsp.ece.mcgill.ca/documents/audioformats/wave/Docs/riffmci.pdf>
- [8] *New Multimedia Data Types and Data Techniques* [online]. 15.4.1994 [cit. 2013-05-03]. Dostupné z: <http://www-mmsp.ece.mcgill.ca/documents/audioformats/wave/Docs/RIFFNEW.pdf>
- [9] Playing Audio in Windows using waveOut Interface. In: OVERTON, David. Planet Source Code [online]. 24.7.2002 [cit. 2013-05-03]. Dostupné z: <http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=4422&lngWid=3>
- [10] RIETHMÜLLER, Claus. Windows driver API basics. In: *ST Audio Central* [online]. 01.31.2003 [cit. 2013-05-03]. Dostupné z: <http://www.staudio.de/kb/english/drivers/>

- [11] SVĚTLÍK, Jiří. Digitální modulace a demodulace s podporou hradlovými poli FPGA. Zlín, 2010. Diplomová práce. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky.
- [12] ULawGraph. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-05-03]. Dostupné z: <http://commons.wikimedia.org/wiki/File:ULawGraph.png>
- [13] VLČEK, Karel. Komprese a kódová zabezpečení v multimediálních komunikacích. 2. vyd. Praha: BEN - technická literatura, 2004, 258 s. ISBN 80-7300-134-9.
- [14] Waveform Coding Techniques. CISCO SYSTEMS. Cisco [online]. 2.2.2006 [cit.2013-05-03]. Dostupné z: http://www.cisco.com/en/US/tech/tk1077/technologies_tech_note09186a00801149b3.shtml#t7

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface
FOURCC	Datový typ o velikosti 4 bajty obsahující 4 znaky
GUI	Grafické uživatelské rozhraní
MME	Multimedia extension.
MMA	MIDI Manufacturer's Association
PCM	Pulsně kódová modulace
STL	Standard Template Library.

SEZNAM OBRÁZKŮ

<i>Obr. 1 - Závislost vjemu hlasitosti na frekvenci [4, s. 15]</i>	13
<i>Obr. 2 - Struktura RIFF formátu v jazyce C [7, s. 11]</i>	14
<i>Obr. 3 - Obecná struktura wav souboru [7, s. 56]</i>	15
<i>Obr. 4 - Struktura fmt bloku [7, s.56]</i>	16
<i>Obr. 5 - Struktura Fact bloku [7, s. 61]</i>	17
<i>Obr. 6 - Struktura Cue bloku [7, s. 61]</i>	17
<i>Obr. 7 - Struktura Plst bloku.[8, s. 14]</i>	19
<i>Obr. 8 - Struktura Assoc-data-list bloku [8, s. 14]</i>	19
<i>Obr. 9 - Vnitřní struktura Assoc-data-list bloku[8, s. 14]</i>	20
<i>Obr. 10 - Struktura Inst bloku.[8, s. 16]</i>	21
<i>Obr. 11 - Struktura Smpl bloku [8, s. 16]</i>	22
<i>Obr. 12 - Vnitřní struktura sample-loop [8, s. 18]</i>	24
<i>Obr. 13 - Struktura data bloku [1, s. 60]</i>	25
<i>Obr. 14 - Ukázka špatně a správně provedeného vzorkování [11, s. 12]</i>	26
<i>Obr. 15 - Ukázka kvantování [3]</i>	26
<i>Obr. 16 - A-křivka [11, s. 14]</i>	28
<i>Obr. 17 - μ-křivka [12]</i>	28
<i>Obr. 18 - Uložení 8-bitového mono a stereo zvuku [1, s. 59]</i>	30
<i>Obr. 19 - Uložení 16bitového mono a stereo zvuku [1, s. 59]</i>	30
<i>Obr. 20 - Zobrazení rozložení a komunikace mezi aplikací a ovladačem [10]</i>	42
<i>Obr. 21 - Vývojové prostředí QtCreator s otevřeným nástrojem QtDesigner</i>	44
<i>Obr. 22 - Adresářová struktura se soubory pro spuštění aplikace</i>	45
<i>Obr. 23 - Grafický interface aplikace</i>	46
<i>Obr. 24 - Nástrojová lišta s hlavním menu Soubor</i>	47
<i>Obr. 25 - Ovládání přehrávače s LCD displejem</i>	48
<i>Obr. 26 - Hlavní ukazatel programu</i>	49
<i>Obr. 27 - Regulační část aplikace</i>	50
<i>Obr. 28 - Vykreslení průběhů jednotlivých kanálů</i>	52
<i>Obr. 29 - Statusbar</i>	53
<i>Obr. 30 - Vykreslovací dialog v základním stavu</i>	53
<i>Obr. 31 - Přeskupení prvků při načítání dat na vykreslovacím dialogu</i>	53

<i>Obr. 32 - Dialog zobrazující upozornění na nesprávný formát vybraného souboru.....</i>	<i>54</i>
<i>Obr. 33 - Seznam všech metod a slotů v MainWindow třídě</i>	<i>56</i>
<i>Obr. 34 - Kruhový bufer v jednotlivých krocích, kterými při běhu aplikace prochází</i>	<i>61</i>
<i>Obr. 35 - Metody a sloty DxSound vlákna</i>	<i>61</i>
<i>Obr. 36 - Postup naplňování a vyprazdňování kruhového buferu.....</i>	<i>65</i>
<i>Obr. 37 - Metody a sloty třídy WinWave</i>	<i>66</i>
<i>Obr. 38 - Metody a sloty</i>	<i>68</i>
<i>Obr. 39 - Vývojový diagram aplikace</i>	<i>71</i>

SEZNAM TABULEK

<i>Tab. 1 - Tabulka některých typů kompresí použitých ve formátu wav [7, s. 11].....</i>	15
<i>Tab. 2 - Seznam elementů fmt bloku [7, s. 56]</i>	16
<i>Tab. 3 - Popis elementů fmt bloku [7, s. 56]</i>	16
<i>Tab. 4 - Seznam elementů Cue-point struktury [8, s. 13].....</i>	18
<i>Tab. 5 - Popis elementů Cue-point struktury [8, s. 13].....</i>	18
<i>Tab. 6 - Popis elementů Plst bloku.[8, s. 14]</i>	19
<i>Tab. 7 - Popis elementů Labl bloku.[8, s. 15]</i>	20
<i>Tab. 8 - Popis elementů Note bloku.[8, s. 15].....</i>	20
<i>Tab. 9 - Popis elementů Ltxt bloku [8, s. 15]</i>	21
<i>Tab. 10 - Seznam elementů Inst bloku.[8, s. 16]</i>	22
<i>Tab. 11 - Popis elementů Smpl bloku [8, s. 17]</i>	23
<i>Tab. 12 - Popis elementů sample-loop bloku [8, s. 18]</i>	24
<i>Tab. 13 - Komprese podle A-křivky [13, s. 40]</i>	29
<i>Tab. 14 - Expanze podle A-křivky [13, s.41].....</i>	29
<i>Tab. 15 - Struktura waveformat [5]</i>	31
<i>Tab. 16 - Přidané parametry ve struktuře waveformatex[5]</i>	31
<i>Tab. 17 - Struktura wayformatextensible [5]</i>	32
<i>Tab. 18 - Popis parametrů funkce WaveOutOpen [5]</i>	33
<i>Tab. 19 - Popis elementů struktury wavehdr [5]</i>	34
<i>Tab. 20 - Popis parametrů funkce WaveOutPrepareHeader [5]</i>	34
<i>Tab. 21 - Popis parametrů funkce WaveOutWrite [5]</i>	35
<i>Tab. 22 - Popis parametrů funkce WaveOutProc [5]</i>	35
<i>Tab. 23 - Popis parametrů funkce IDirectSound8 [5]</i>	37
<i>Tab. 24 - Parametry metody SetCooperativeLevel [5]</i>	38
<i>Tab. 25 - Elementy struktury DSBUFFERDESC [5].....</i>	38
<i>Tab. 26 - Parametry metody CreateSoundBuffer [5].....</i>	39
<i>Tab. 27 - Parametry metody Lock [5].....</i>	40
<i>Tab. 28 - Parametry metody Play [5]</i>	41
<i>Tab. 29 - Parametry metody GetCurrentPosition [5].....</i>	41