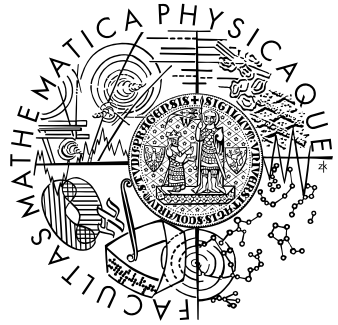
Univerzita Karlova v Prahe
Matematicko-fyzikálna fakulta

BAKALÁRSKA PRÁCA



Matej Marko

Hudobné efekty

Katedra aplikovanej matematiky

Vedúci bakalárskej práce: Mgr. Martin Bálek

Študijný program: Informatika

2010

Ďakujem Mgr. Martinovi Bálkovi za vedenie ročníkového projektu a bakalárskej práce, svojej rodine a priateľom za podporu.

Prehlasujem, že som svoju bakalársku prácu vypracoval samostatne a výhradne s použitím uvedených prameňov. Súhlasím so zapožičiavaním práce.

V Prahe dňa 26. 5. 2010 Matej Marko

Obsah

Názov práce: Hudobné efekty
Autor: Matej Marko
Katedra (ústav): Katedra aplikovanej matematiky
Vedúci baklárskej práce: Mgr. Martin Bálek
e-mail vedúceho: Martin.Balek@mff.cuni.cz

Abstrakt: Efekty zohrávajú v procese tvorby hudby významnú úlohu. Pomáhajú doplniť a obohatiť aranžmán skladby. Cieľom práce je vyvinúť aplikáciu, ktorá umožní užívateľovi prehrať zvukový súbor a v reálnom čase na prehrávaný obsah aplikovať zvolené efekty. Každá zmena v nastavení efektov sa tak ihneď prejaví na výslednom zvuku a užívateľ ju môže okamžite posúdiť. Konkrétne nastavenie môže užívateľ aplikovať na celý vstupný súbor a tento výsledok uložiť. Súčasťou aplikácie je tiež implementácia vybraných efektov. V texte práce sa potom zameriavame na postupy, ktoré sú použité pri implementovaní týchto efektov.

Kľúčové slová: efekty, hudba, zvuk

Title: Music effects
Author: Matej Marko
Department: Department of applied mathematics
Supervisor: Mgr. Martin Bálek
Supervisor's e-mail address: Martin.Balek@mff.cuni.cz

Abstract: The effects have crucial position in the proccess of creating music. They enrich song's arrangement and make it all more colorful. Aim of the work is to create application that allows the user to play a sound file and apply chosen effets on it's content in real-time. All changes in effects' settings are imidiatelly transformed into the hearable results. Certain settings can be applied on the whole file and the results afterwards exported to the file. Part of the application is naturally implementation of chosen effects. Main topic of the work's text are methods used to implement the effects.

Keywords: effects, music, sound

Kapitola 1

Úvod

Za hudobný efekt môžeme vo všeobecnosti považovať ľubovoľné zariadenia, ktoré modifikujú vstupný signál spôsobom, ktorý je kreatívne využiteľný pri tvorbe hudby. Z čisto teoretického hľadiska pritom môže ísť o zariadenie, ktoré vstupný signál rôzne deformuje, či poškodzuje. Ako príklad uveďme rôzne skreslovacie efekty, ktoré využívajú práve nelineárnu charakteristiku prebudených tranzistorov. Ako hudobný efekt teda môže slúžiť akékoľvek zariadenie, pokiaľ znie „dostatočne zaujímavo“ pri dotvorení a obohacovaní aranžmánu skladby. Efekty sa používajú počas nahrávacieho procesu, ale tiež pri živom hraní hudby.

Pôvodne boli efekty implementované pomocou jednoúčelových analógových zariadení. Funkcia efektu závisela od konkrétneho elektronického obvodu a nebolo možné ju modifikovať. Z vývojom digitálnych technológií začali niektoré analógové zariadenia nahrádzať digitálne. Tie pracujú s diskrétnym, vzorkovaným signálom. Ich výhodou je hlavne stálosť vlastností pri rôznych vonkajších podmienkach (teplota) a tiež možnosť neskoršej úpravy fungovania efektu zmenou programu, ktorý zariadenie vykonáva.

V poslednom období nastal opäť posun v realizácii efektov. Namiesto samostatných zariadení sa čoraz viac presadzujú čisto softwarové riešenia. Efekty tak majú najčastejšie podobu pluginov, ktorý rozšíri nahrávací software a je jednoducho použiteľný. Čoraz viac sa podobný software presadzuje aj pri hraní živých vystúpení.

V tejto práci popisujeme viacero druhov efektov so zameraním na metódy, ktoré sú požité pri ich implementácii. Obsiahnuté sú nasledovné efekty:

- Volume, Boost, Tremolo – kapitola 2
- Delay,Flanger,Chorus,Vibrato – kapitola 3
- Ekvalizéry, Wah-Wah, Envelope filter – kapitola 4
- Pitchshifter – kapitola 5

Pri popisoch jednotlivých efektov budeme často používať diagramy znázorňujúce tok signálu efektom. V týchto diagramoch budeme používať značenie, ktoré popíšeme v tejto kapitole. Táto konvencia je inšpirovaná značením, ktoré používa J. O. Smith III [15]. Značky používané sú vždy vysvetlené pomocou jednoduchého príkladu, ktorý je väčšinou doplnený rovnicou.

Vstupný signál budeme označovať symbolom x (prípadne x_1, x_2 v prípade, že vstupných signálov je viacero). Tento signál budeme chápať ako funkciu $Z \to [-1,1]$, pre ktorú platí:

- $\forall n \geq 0$, je $x(n)$ rovné hodnote n-tej vzorky

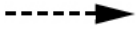
- $\forall n < 0$, definujeme $x(n) = 0$. Toto obmedzenie zodpovedá konštantnému signálu s nulovou amplitúdou (a teda „tichu“) pred začiatkom pred začiatkom súboru.

Rovnako budeme uvažovať aj výstupný signál, ktorý označíme symbolom y.

- Smer toku signálu v diagrame znázorňuje jednoduchá šípka.

Obr 1.1: Značenie smeru toku signálu

- Prerušovaná šípka znamená užívateľské nastavenie nejakého parametra efektu. Šípka vychádza od názvu parametra a smeruje k bloku, ktorý tento parameter ovplyvňuje.

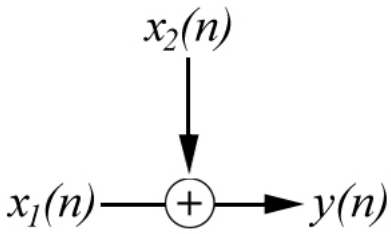


Obr 1.2: Značenie užívateľského parametra

- Hrubá šípka znázorňuje kontrolovanie nastavenia jedného konštrukčného bloku iným blokom. Napr. kontrolovanie činnosti nejakého bloku kmitaním oscilátora.

Obr 1.3: Kontrolovanie konštrukčného bloku iným blokom

- Sčítanie signálov x_1, x_2 podľa vzťahu $y(n) = x_1(n) + x_2(n)$.

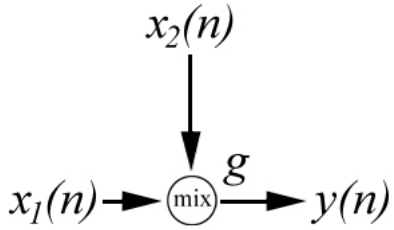


Obr 1.4: Sčítanie dvoch signálov

- Vynásobenie vstupného signálu x zadaným koeficientom g: $y(n) = g \cdot x(n)$

Obr 1.5 Násobenie vstupného signálu koeficientom g

- Miešanie signálov x_1, x_2 v pomere, ktorý určuje koeficient $g \in [0,1]$. Tento koeficient určuje aký veľký bude podiel vstupu x_1 vo výstupnom signály. V krajných nastaveniach ($g = 0$, resp. $g = 1$) je prepustený len jeden zo signálov. Výstupný signál y teda môžeme vyjadriť pomocou vzťahu: $y(n) = g \cdot x_1(n) + (1-g) \cdot x_2(n)$



Obr 1.6: Zmiešanie dvoch signálov

- Konštrukčný blok, ktorý vykonáva inú ako uvedenú operáciu budeme značiť obdĺžnikom. V ploche obdĺžnika sa nachádza stručný popis činnosti bloku. Konkrétny použitý blok bude podrobnejšie vysvetlený, pri jeho použití.

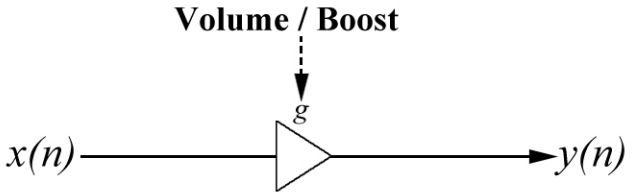
Obr 1.7: Všeobecný konštrukčný blok

Kapitola 2

Efekty pracujúce s úrovňou signálu

2.1 Volume

Volume a Boost su najpriamočiarejšie zo všetkých popisovaných efektov. Oba efekty ovplyvňujú úroveň hlasitosti vstupného signálu podľa užívateľského nastavenia. Jedinou operáciou, ktorú vykonávajú, je prenásobenie každej vzorky vstupného signálu koeficientom g udávajúcim úroveň zoslabenia, resp. zosilnenia.



Obr 2.1: Štruktúra efektov Volume a Boost

Efekty majú totožný princíp fungovania, ale mierne sa líšia vo svojej funkcie. Volume upravuje úroveň signálu iba pasívne, t. j. signál iba zoslabuje. Riadiaci koeficient g je preto reálne číslo z intervalu $[0,1]$. V prípade, že $g = 0$, každá vzorka na výstupe efektu bude tiež rovná nule: $\forall n : y(n) = 0$. Efekt teda úplne stlmí akýkoľvek signál. Na druhej strane pre $g = 1$ platí $\forall n : y(n) = x(n)$, čiže efekt nebude mať žiadny vplyv.

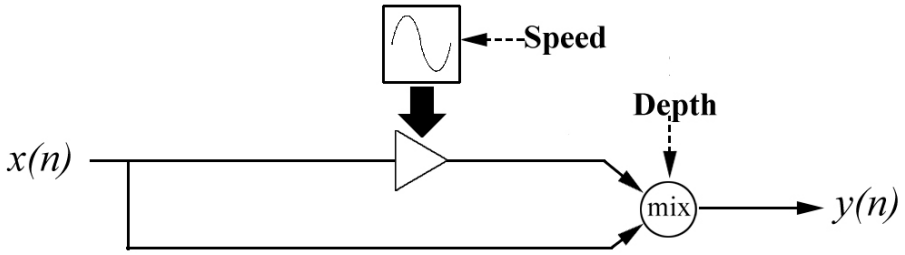
Boost oproti Volume ovplyvňuje svoj vstup aktívne, t. j vstupný signál zosilňuje. Koeficient g nadobúda hodnoty z intervalu $[1,10]$. V maximálnom nastavení teda efekt zosilňuje vstupný signál 10-násobne.

Oba efekty boli pôvodne určené na testovanie pluginového systému aplikácie. Predstavujú však užitočný nástroj ako regulovať hlasitosť signálu v priebehu spracovania, preto sme sa rozhodli efekty ponechať. Zdrojový kód efektu Volume navyše poskytuje ukážku implementácie efektu, ktorý môže byť ako plugin dodatočne pridaný do aplikácie.

2.2 Tremolo

Tremolo je jednoduchý efekt, ktorý kontroluje úroveň hlasitosti signálu na základe kmitania oscilátora. Takto modifikovaný signál potom mieša s čistým vstupným signálom. Užívateľ má možnosť kontrolovať pomer, v akom sa tieto signály miešajú a tým ovplyvňuje, do akej miery sa efekt prejaví (parameter označený Depth – hĺbka efektu). Druhým obvyklým parametrom efektu je ovládanie

frekvencie oscilátora (Speed – rýchlosť kmitania). Hodnotu parametra Speed môžeme tiež alternatívne vyjadriť ako dĺžku periódy oscilátora. V našej implementácii sme zvolili rozsah od 30 ms do 1000 ms, čo pokrýva prakticky využiteľný rozsah efektu.



Obr 2.2: Štruktúra efektu Tremolo

Posledným najčastejším parametrom je ovplyvnenie priebehu kmitania oscilátora. Vo všeobecnosti sú najpoužívanejšie sínusový, trojuholníkový alebo štvorcový priebeh. V našej implementácii sme použili sínusový a štvorcový oscilátor podľa vzoru efektu Electro Harmonix Pulsar (pôvodný model zo 70-tych rokov [19]). Sínusový oscilátor znie uhladenejšie a viac klasicky, zatiaľ čo pri použití štvorcového kmitania je výsledný zvuk agresívnejší a pri vhodnom nastavení rýchlosti kmitania nie nepodobný zvuku vrtuľníka.

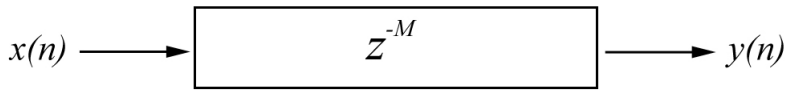
Digitálna implementácia nám umožňuje zostrojiť ideálny štvorcový oscilátor, ktorý nadobúda iba 2 hodnoty. Jeho použitie však produkuje výsledok, ktorý znie veľmi neprirodzene. Dôvodom je fakt, že v analógových implementáciách nie je priebeh oscilátora skutočne štvorcový, ale obsahuje nábehové hrany. Preto sme v našej implementácií museli ideálny štvorcový priebeh nahradiť deformovaným, ktorý kombinuje sínusový a obdĺžnikový priebeh.

Kapitola 3

Efekty využívajúce oneskorovacie linky

3.1 Jednoduchá oneskorovacia linka

Oneskorovacia linka v najjednoduchšej forme je znázornená na obrázku 3.1.



Obr 3.1: Jednoduchá oneskorovacia linka

Symbol z^{-M} znamená oneskorenie o M vzoriek. Ak označíme vstup ako postupnosť

$$x(n), n = 0,1,2,....$$

Potom výstupom jednoduchej oneskorovacej linky s dĺžkou M je postupnosť

$$y(n) = x(n - M), n = 0,1,2,... ,$$

pričom predpokladáme, že $x(n) = 0, n < 0$. Dostávame tak základný stavebný blok, pomocou ktorého je možné vytvoriť primitívnu ozvenu s pevnou dĺžkou oneskorenia signálu. Jeho jedinou vlastnosťou je čas oneskorenia, ktorý je daný dĺžkou linky (počtom vzoriek, o ktoré signál oneskoruje.)

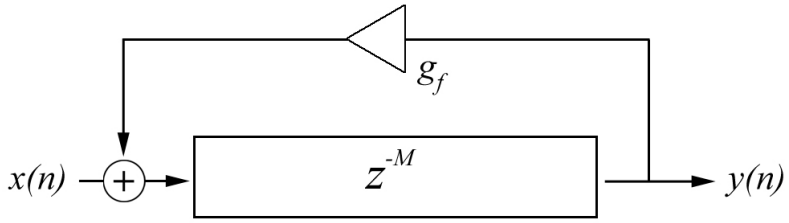
Nech f_s je vzorkovacia frekvencia vstupného signálu a $T = \frac{1}{f_s}$ je doba trvania jednej vzorky. Potom pre celkový čas oneskorenia platí $t = T \cdot M = \frac{M}{f_s}$. Naopak pre požadovaný čas oneskorenia t je dĺžka oneskorenia v počte vzoriek rovná $M = t \cdot f_s$. Pripomeňme, že čas oneskorenia aj vzorkovacia frekvencia sa musia vzťahovať k rovnakej časovej jednotke (najčastejšie k sekunde).

Získaná dĺžka oneskorenia v počte vzoriek nemusí byť prirodzené číslo. V takom prípade sa použije zaokrúhlená hodnota. Chyba, ktorá takto vznikne, sa tak pohybuje v intervale $\left[0, \frac{T}{2}\right]$. Pre typickú hodnotu vzorkovacej frekvencie vstupného signálu $44100Hz$ je teda maximálna možná chyba dĺžky oneskorenia $0.0113ms$. Táto chyba je natoľko malá, že je nemožné ju pri posluchu výsledku zaregistrovať. Ak by však predsa len bola žiadúca vyššia presnosť, je možné zvoliť $M = \lceil t \cdot f_s \rceil$, resp.

$M = \lceil t \cdot f_s \rceil + 1$ a výstup oneskorovacej linky získať interpoláciou z niekoľkých hodnôt.

3.2 Oneskorovacia linka so spätnou väbou

Miernou modifikáciou jednoduchej oneskorovacej linky dostaneme oneskorovaciu linku so spätnou väzbou. Odlišuje sa tým, že do vstupného signálu primiešava výstupný. Znázornenie tejto linky diagramom vyjadruje obrázok 3.2



Obr 3.2: Oneskorovacia linka so spätnou väzbou

K vstupnému signálu je primiešavaný výstup oneskorovacej linky, ktorý je vynásobený číslom g_f. To je parameter určujúci úroveň spätnej väzby. Pre vstupný signál

$$x(n), n = 0,1,2,....$$

dostávame výstup

$$y(n) = x(n-M) + g_f \cdot x(n-2M) + {g_f}^2 \cdot x(n-3M)..., \; n = 0,1,2,...$$

čo môžeme vyjadriť pomocou sumy

$$y(n) = \sum_{i=1}^{\infty} {g_f}^{i-1} \cdot x(n-iM).$$

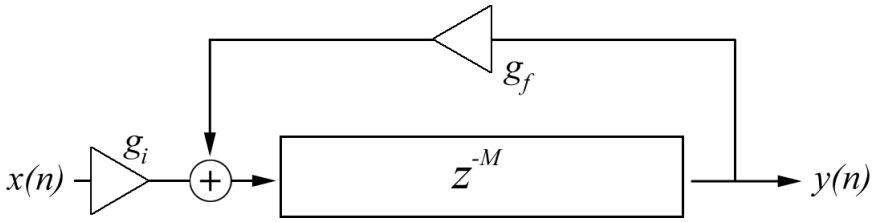
Úroveň vzorky sa teda mení exponenciálne vzhľadom k počtu absolvovaných oneskorení v linke.

Problémom popísanej linky so spätnou väzbou je situácia, pri ktorej platí:

$$|x(n) + y(n)| > 1$$

Úroveň signálov v súčte presiahne najvyššiu hodnotu, ktorú môže signál nadobúdať a dochádza k prebudeniu. Jednoduché riešenie popisuje obrázok 3.3 Upravená linka, ktorá okrem parametra g_f obsahuje aj parameter g_i kontrolujúci úroveň vstupného signálu. Ak je splnený invariant $g_i + g_f \leq 1$, potom žiadna vzorka nemôže prekročiť hodnotu 1, a teda nedôjde k prebudeniu výstupu. Jeden zo spôsobov, ako zaručiť splnenie invariantu, je vždy zvoliť $g_i = 1 - g_f$.

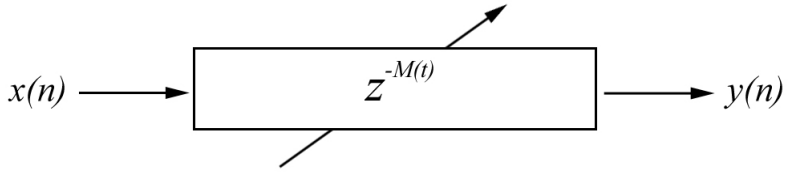
Pri tomto riešení môže nastať situácia, že $g_i = 0$ a do oneskorovacej linky neprichádza žiaden vstup. Niekomu sa to môže javiť nelogické, ale ak by táto situácia bola vyvolaná náhlou zmenou počas hry. Oneskorovacia linka by bola naplnená a na výstup by sa stále periodicky posielal jej obsah, čo môže byť niekedy žiadúce ako výrazový prostriedok.



Obr 3.3: Upravená oneskorovacia linka so spätnou väzbou

3.3 Oneskorovacia linka s premenlivou dĺžkou oneskorenia

V tomto prípade, na rozdiel od jednoduchej oneskorovacej linky, nie je dĺžka oneskorania konštantná. Môžeme ju teda chápať ako funkciu $M(t)$, ktorej riadiacou premennou je čas. Dĺžku oneskorenia vo väčšine prípadov určuje kmitanie nízkofrekvenčného oscilátora. Zmena doby oneskorenia sa teda predpokladá s každou vzorkou na vstupe, navyše doba oneskorenia vo väčšine prípadov nebude zodpovedať celočíselnému počtu vzoriek. V tomto prípade však už nestačí zaokrúhliť dĺžku linky na najbližšiu celočíselnú hodnotu. Pre eliminovanie rušivého šumu, ktorý by takto vznikol, je potrebné použiť interpoláciu medzi oneskorenými vzorkami ako uvádza J. O. Smith III v [15].



Obr 3.4: Oneskorovacia linka s premenlivou dĺžkou oneskorenia

3.4 Interpolácia v premenlivých oneskorovacích linkách

Pri realizovaní oneskorovacej linky s premenlivou dĺžkou oneskorenia sa najčastejšie používa lineárna interpolácia medzi dvoma vzorkami, ktoré sú najbližšie k požadovanej hodnote oneskorenia. Táto metóda je nenáročná na výpočtový výkon a jej kvalita je postačujúca, pokiaľ je frekvenčný rozsah vstupného signálu malý v porovnaní s polovicou vzorkovacej frekvencie [15]. Pri splnení tejto podmienky sú rozdiely medzi susednými vzorkami dostatočne malé a lineárna interpolácia poskytuje dostatočnú aproximáciu skutočného signálu.

Pri praktickom testovaní sa však ako kvalitnejšia ukázala kvadratická interpolácia využívajúca 4 najbližšie vzorky k požadovanej hodnote. Táto metóda nám poskytuje dobrú aproximáciu signálu v každom momente, ktorý oneskorovacia linka zaznamenáva. (Ide v podstate o jednoduchú formu oneskorovacej linky s náhodným prístupom.) To nám umožňuje použiť ju aj v prípadoch, kedy sú jednotlivé zmeny v požadovanej dĺžka oneskorenia veľké. Preto môže byť použitá aj pri vysokých hodnotách frekvencie oscilátora.

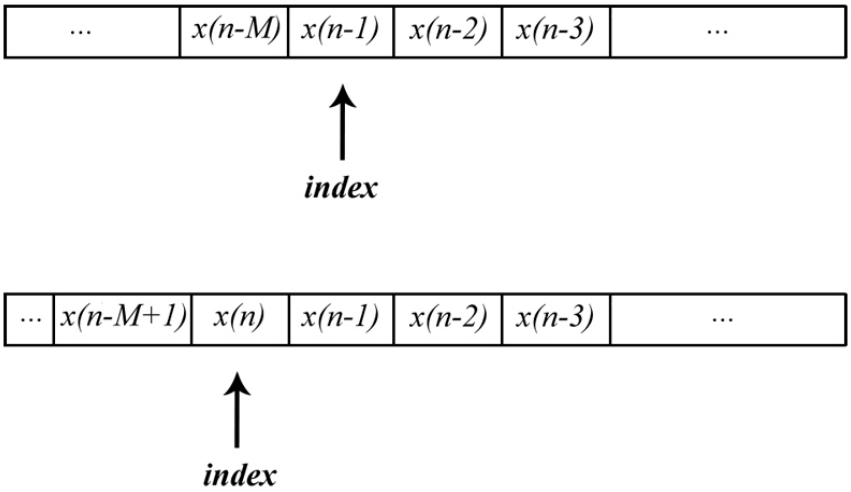
3.5 Implementácia

Rozhranie pre prácu s oneskorovacími linkami definuje abstraktná trieda *DelayLine*. Jej hlavným prostriedkom je metóda *Step*, ktorá ako vstupný parameter dostáva vzorku zo vstupného signálu a vracia vzorku výstupného signálu. Spoločná

nadtrieda môže byť užitočná pri viacúčelových zariadeniach, ktoré plnia viacero funkcií a podľa toho zvolia akú konkrétnu oneskorovaciu linku budú používať.

Na implementovanie oneskorovacej linky s dĺžkou M je potrebné uchovávať posledných M vzoriek vstupného signálu, pretože tieto vzorky budú návratové hodnoty nasledujúcich volaní metódy *Step* a *DelayLine* nemá iný prístup k vstupnému signálu ako vzorky predané do metódy *Step*.

Na prvý pohľad by sa ako ideálne riešenie mohla javiť fronta, keďže oneskorovacia linka naozaj spĺňa základnú požiadavku FIFO štruktúry, t. j. každá vzorka vždy opustí linku pred všetkými, ktoré vstúpili do linky až po nej. Treba si ale uvedomiť, že vieme dopredu povedať, aká bude veľkosť fronty. Navyše pridanie novej vzorky vytlačí tú najstaršiu na výstup. Stačí nám teda jednorozmerné pole veľkosti *M*, a *index* do tohto poľa ukazujúci na najnovšiu vzorku. Rovnako môžeme uchovávať hodnotu $index - 1$, ktorá ukazuje na najstaršiu vzorku v linke.



Obr 3.5: Vzorky uložene v oneskorovacej linke pred a po vykonaní jedného kroku

Situáciu pred a po volaní metódy *Step* môžeme pozorovať na obrázku 3.5. Ľahko nahliadneme, že každá zapísaná hodnota zo vstupu bude poslaná na výstup práve v momente, keď index opäť bude ukazovať na rovnaké miesto v poli. Keďže index pri každom volaní metódy step zmenšíme o 1, nastane presne po M volaniach metódy *Step*.

Oneskorovacia linka so spätnou väzbou funguje rovnako s jediným rozdielom. Do pomocného poľa nie je zapísaná hodnoty priamo zo vstupu, ale súčet $vstup + g_f \cdot pole[index - 1]$ (resp. $g_i \cdot vstup + g_f \cdot pole[index - 1]$). Dosiahneme tak presne požadovaného primiešania výstupu na vstup.

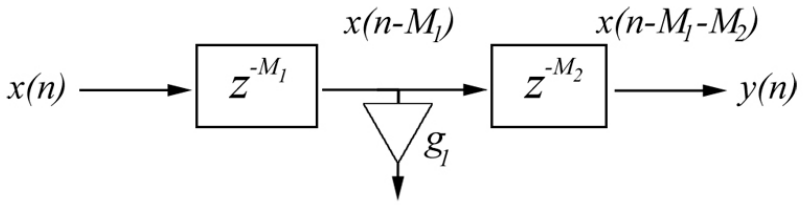
Oneskorovacia linka s premenlivou dĺžkou je implementovaná pomocou rovnakého princípu. Rozdiel spočíva vo výpočte indexu, ktorý ukazuje na oneskorenú vzorku. Index zodpovedajúci požadovanému oneskoreniu $M(t)$ vypočítame ako $index + M(t) \cdot M$, kde M je maximálna dĺžka linky (počet zaznamenaných vzoriek). Výsledné číslo nemusí (a najčastejšie ani nie je) prirodzené číslo. Výsledná hodnota sa preto musí interpolovať pomocou susedných vzoriek. Ich indexy sú rovné $\lfloor index + M(t) \cdot M \rfloor \bmod M$, resp. $\lceil index + M(t) \cdot M \rceil \bmod M$. Pri kubickej interpolácii použijeme aj vzorky $\lfloor index + M(t) \cdot M - 1 \rfloor \bmod M$ a $\lceil index + M(t) \cdot M + 1 \rceil \bmod M$.

3.6 Delay

Delay je anglické označenie efektu, ktorý simuluje ozvenu. J. O. Smith III v [15] opisuje simulovanie jednoduchej ozveny, pričom sa zameriava na vzťah dráhy, po ktorej zvuk putuje a času oneskorenia, ktorý táto dráha vyžaduje. Z hľadiska hudobného efektu je dráha zvuku irelevantná pokiaľ nesimuluje aj iné vlastnosti priestoru (frekvenčné vlastnosti odrazu). Preto sa pri delay-och ustálilo pohodlnejšie zadávanie času oneskorenia, ktoré pravdepodobne súvisí aj s historickým vývojom tohto typu efektu (pôvodne sa používali magnetofónové pásky rotujúce v nekonečnej slučke).

Najjednoduchší delay teda využíva jednoduchú oneskorovaciu linku, do ktorej posiela vstupný signál. Výstup oneskorovacej linky potom mieša so vstupným signálom v pomere, ktorý určuje užívateľ. Výsledkom je jedna ozvena vstupného signálu.

Viacero ozvien je možné dosiahnuť použitím tzv. tapped delay lines, čiže oneskorovacou linkou, ktorá obsahuje niekoľko odbočiek. Príklad takejto oneskorovacej linky s jednou odbočkou vidíme na obrázku 3.6.



Obr 3.6: Oneskorovacia linka s jednou odbočkou

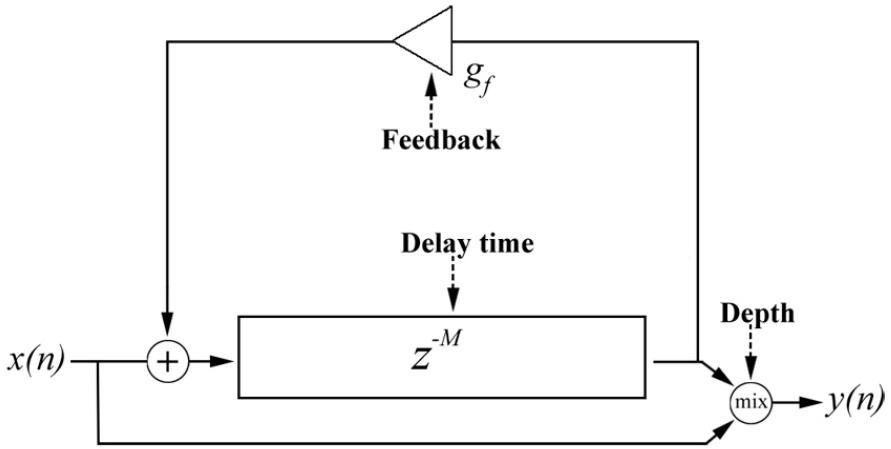
Použitie takejto oneskorovacej linky umožňuje kontrolovať čas oneskorenia pre každý výstup (a teda rozstupy medzi jednotlivými ozvenami) a úroveň pre každý výstup zvlášť. Veľkou nevýhodou však je pamäťová náročnosť. Keď chceme dosiahnuť N ozvien a každá bude od predchádzajúcej vzdialená M_i vzoriek. Celková dĺžka oneskorovacej linky je $\sum_{i=1}^{N} M_i$, čo je v prípade pravidelných rozstupov ($M_i = M, i = 1..N$) rovné $M \cdot N$.

Alternatívou je použitie oneskorovacej linky so spätnou väzbou. Oproti predošlémmu postupu kladie obmedzenia na rozstupy medzi jednotlivými ozvenami. Tie sú pravidelné a majú všetky dĺžku zodpovedajúcu dĺžke linky. Takisto úrovne jednotlivých výstupov sa nedajú nastavovať individuálne, ale riadiaci koeficient má exponenciálny vzťah k poradiu ozveny. Výhodou je, že spätnoväzbová linka potrebuje uchovávať iba M vzoriek, kde M je veľkosť rozstupu medzi ozvenami.

V prospech oneskorovacej linky so spätnou väzbou tiež hovorí jednoduchosť nastavenia pomocou jediného koeficientu. Hodnota koeficientu spätnej väzby sa pohybuje v intervale $[0,1]$. Tieto hodnoty prirodzene zodpovedajú správaniu, ktoré užívateľ intuitívne očakáva. Exponenciálna funkcia so základom z intervalu [0,1] je klesajúca a teda spôsobuje postupné zoslabovanie ozvien.

Obrázok 3.7 popisuje štruktúru implementácie jednoduchého delay-u. Užívateľ má možnosť nastaviť parametre Delay time, Feedback a Depth. Delay time kontroluje čas oneskorenia, resp. rozostup medzi jednotlivými ozvenami a jeho rozsah je 0 ms až 2000 ms. Feedback určuje úroveň spätnej väzby a teda počet ozvien. Je to reálne číslo z intervalu $[0,1]$. Rovnaké hodnoty nadobúda aj parameter

Depth, hĺbka efektu, ktorý nastavuje pomer medzi čistým a upraveným vstupným signálom.



Obr 3.7: Štruktúra efektu Delay

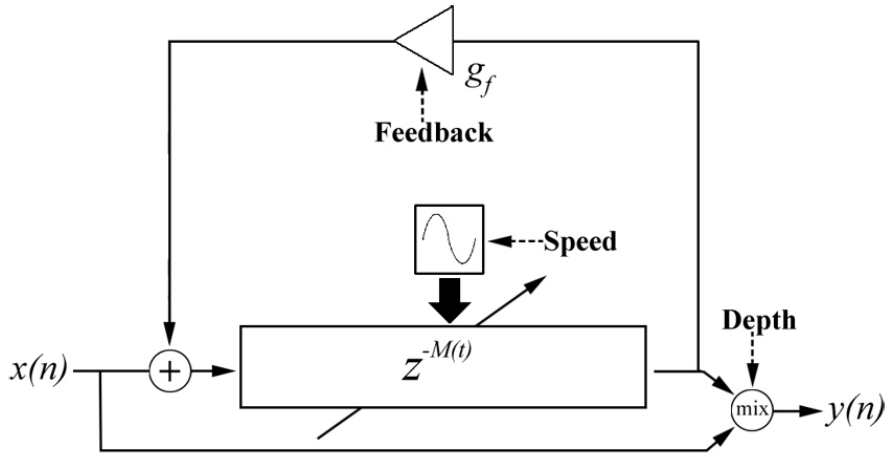
3.7 Flanger

Flanger je efekt pôvodne dosahovaný pomocou dvoch magnetofónov, prehrávajúcich rovnaký záznam. Ich výstup je zmiešaný v pomere 1:1. V priebehu prehrávania je jeden z magnetofónov mechanicky spomalený ľahkým dotykom. To spôsobí, že sa páska postupne spomalí, čo má za následok frekvenčný posun prehrávaného záznamu. Potom, čo operátor uvoľní pásku, tá nadobudne svoju pôvodnú rýchlosť. Bude však mierne oneskorená oproti druhej. Rovnaký postup sa zopakuje na druhom zariadení, čím sa tento rozdiel vyrovná a oba magnetofóny budú opäť hrať unisono.

Ideálnym prostriedkom na digitálnu implementáciu je oneskorujúca linka, ktorej dĺžka je riadená kmitaním oscilátora. Hodnota oscilátora spočiatku rastie. Tým sa zväčšuje dĺžka oneskorenia, čo simuluje oneskorovanie (a spomaľovanie) jednej pásky. V polovici periódy oscilácia dosiahne maximum, čo zodpovedá okamihu, keď majú obe pásky rovnakú rýchlosť, ale prvá je v porovnaní s druhou oneskorená o maximálnu dĺžku linky. Hodnota oscilátora potom začne klesať, dĺžka oneskorenia sa zmenšuje, čím dosiahneme spomalenie druhej pásky. V momente, keď hodnota oscilátora sa rovná 0, obidve pásky opäť prehrávajú záznam v rovnakom čase.

Na obrázku 3.8 vidíme blokovú štruktúru, ktorá zachycuje implementáciu efektu Flanger. Užívateľ má možnosť kontrolovať frekvenciu oscilácie, parameter Speed, a pomer čistého a modifikovaného signálu, parameter Depth. Parameter Speed zodpovedá dĺžke periódy použitého oscilátora a je nastaviteľný v rozsahu 1 ms až 5000 ms.

Najčastejšie používaným priebehom oscilácie je trojuholníkový oscilátor [6]. My sme rozhodli ponechať voľbu tvaru na užívateľa, ktorý ma možnosť zvoliť medzi trojuholníkovým a sínusovým priebehom. Rozdiely pri voľbe rôznych oscilátorov sa však na výsledku prejavujú len veľmi nepatrne.



Obr 3.8: Štruktúra efektu Flanger

V našom prípade sme túto linku, podľa publikácie [15], doplnili o možnosť kontrolovať spätnú väzbu. Úroveň spätnej väzby môže užívateľ nastaviť pomocou parametra Feedback. Jeho hodnoty sa pohybujú v intervale $[-1,1]$. Záporné hodnoty zodpovedajú otočeniu fázy signálu o 180°.

Podľa článkov [6] a [7] by sa dĺžka oneskorovacej linky mala meniť v intervale 1 ms až 10 ms. Hodnotu 10 ms sme za maximálnu dĺžku linky zvolili aj my pri našej implementácii. Konkrétny čas oneskorenia sa teda vypočíta pomocou vzorca

$$t_{delay} = t_{\max} \cdot Osc(t),$$

kde $Osc(t)$ predstavuje kmitanie oscilátora v čase a nadobúda hodnoty z intervalu $[0,1]$. Vyjadrené pomocou počtu vzoriek dostávame rovnicu:

$$M(t) = M_{\max} \cdot Osc(t)$$

Efekt Flanger bol dlho realizovaný pomocou upravenej varianty oneskorovacej linky so spätnou väzbou, ktorá predchádza prebudeniu. Neskôr sa však ukázalo, že použitím tejto varianty nie je možné dosiahnuť viacero zaujímavých výsledkov, ktorých je Flanger schopný. Rozhodli sme sa preto ponechať kontrolovanie prebudenia na užívateľovi.

3.8 Chorus

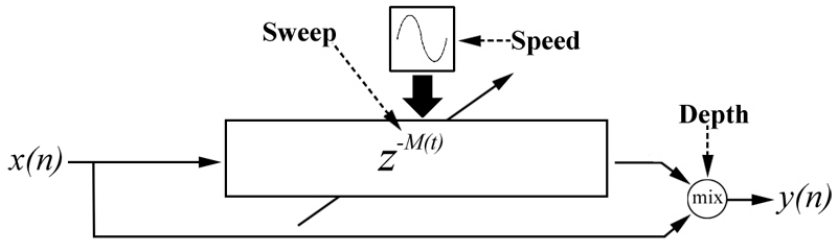
Efekt Chorus slúži na simulovanie druhého hlasu. Snaží sa teda napodobniť zdroj rovnakého signálu, ktorý je však hraný s malými odlišnosťami v tempe a výške tónu.

Princíp fungovania je totožný s efektom flanger. Líši sa však v zafixovaných časoch oneskorenia a tiež neobsahuje možnosť spätnej väzby. Štruktúru efektu zachytáva obrázok 3.9. Čas oneskorenia v použitej linke sa pohybuje od 20 ms do 30 ms [7]. Chorus teda využíva modifikovanú oneskorovaciu linku, ktorá nemení čas oneskorenia od 0 po maximum, ale v danom obmedzenom rozsahu. Ten je daný minimálnym časom oneskorenia $t_{\min}$ a dĺžkou variujúceho intervalu t_{var}. Celkový čas oneskorenia linky sa teda môžeme vyjadriť pomocou vzťahu:

$$t_{delay} = t_{\min} + Osc(t) \cdot t_{\text{var}},$$

kde $Osc(t)$ predstavuje kmitanie oscilátora v čase. Vyjadrenie oneskorenia pomocou počtu vzoriek teda vyzerá:

$$M(t) = M_{\min} + M_{\text{var}} \cdot Osc(t)$$



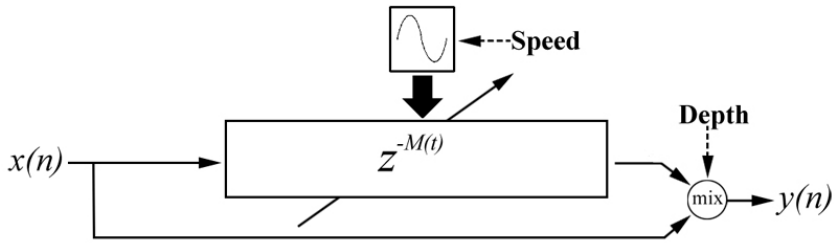
Obr 3.9: Štruktúra efektu Chorus

3.9 Vibrato

Vibrato efekt je pokusom o napodobenie techniky hry na hudobný nástroj, pri ktorej hráč, počas hrania noty, periodicky mení výšku tónu v malom frekvenčnom intervale. Táto technika hry je typická pre strunové a sláčikové nástroje. Podľa J. O. Smitha III [15] môže byť implementovaná pomocou oneskorovacej linky s premenlivou dĺžkou. Poznamenajme však, že pri moderných technikách hry na elektrickú gitaru sú často používané vibráta v rozsahu celého tónu.

Naša implementácia je podobná efektom Flanger, resp. Chorus. Vibrato však neobsahuje spätnú väzbu ani parameter Sweep. Dĺžka oneskorovacej linky sa tak podľa hodnoty oscilátora pohybuje v intervale $[0, t_{\max}]$. Pri testovaní efektu sa ako najvhodnejšia ukázala hodnota $t_{\max} = 30ms$. Väčšia maximálna dĺžka oneskorenia spôsobuje výraznejšiu odchýlku od frekvencie vstupného signálu. Takisto parameter Speed má rozsah hodnôt rozdielny ako pri efekte Flanger: 1 ms až 1000 ms. Štruktúru efektu popisuje obrázok 3.10.

Užívateľovi sme opäť ponechali možnosť voľby priebehu oscilácie. K trojuholníkovému a sínusovému oscilátoru, sme navyše pridali štvorcový oscilátor, ktorý umožňuje dosiahnuť agresívnejšiu moduláciu vstupného signálu.



Obr 3.10: Štruktúra efektu Vibrato

Kapitola 4

Efekty využívajúce digitálne filtre

Digitálne filtre sú používané na oddelenie signálov, ktoré boli zmiešané a tiež pre obnovu skreslených signálov. Analógové filtre môžu byť použité na rovnaké úlohy, avšak digitálne filtre poskytujú mnohé výhody. Prehľad ich vlastností, v ktorých majú oproti analógovým filtrom navrch, môžeme nájsť v článku [18]. Pri implementovaní efektov majú rozsiahle využitie a v našej aplikácii ich využijeme pri vytváraní ekvalizéra a tiež v efektoch wah-wah a envelope filter.

4.1 Digitálny filter vo všeobecnosti

Filter vo všeobecnosti je zariadenie, ktoré určitým spôsobom modifikuje vstupný signál x a produkuje výstupný signál y. Jednoduchú predstavu nám poskytuje obrázok 4.1. Digitálny filter je typ filtra, ktorý pracuje s digitálnou reprezentáciou signálu - jeho vzorkami.

Obr 4.1: Filter - základná predstava

Digitálny filter je najčastejšie vyjadrený pomocou diferenčnej rovnice, ktorá vyjadruje vzťah medzi vzorkami vstupného signálu x a výstupného signálu y. Najjednoduchší príklad je filter s jednotkovým ziskom. Tento filter môžeme vyjadriť pomocou rovnice:

$$y(n) = x(n).$$

Filter teda nemá žiadny účinok, každá vzorka na vstupe sa totiž prepustí na výstup bez akýchkoľvek úprav. Druhým jednoduchým príkladom je filter, ktorý ma zvolené zosilnenie, resp. útlm signálu:

$$y(n) = g \cdot x(n).$$

Každá vstupná vzorka je prenásobená koeficientom zisku g. Podľa aktuálnej hodnoty g je signál zosilnený alebo zoslabený.

- $g \in (0,1)$ - vstupný signál je zoslabený
- $g = 1$ - filter je totožný s filtrom s jednotkovým ziskom z predchádzajúceho príkladu
- $g \in (1, \infty)$ - vstupný signál je zosilnený

Tento filter vlastne zodpovedá efektom Volume a Boost popísaným v kapitole 2.

Vo všeobecnosti môžeme diferenčnú rovnicu napísať nasledovne:

$$b_0 \cdot y(n) + b_1 \cdot y(n-1) + b_2 \cdot y(n-2) + ... = a_0 \cdot x(n) + a_1 \cdot x(n-1) + a_2 \cdot x(n-2) + ...$$

prípade v tvare, ktorý priamo vyjadruje aktuálnu výstupnú hodnotu $y(n)$:

$$y(n) = b_1' \cdot y(n-1) + b_2' \cdot y(n-2) + ... + a_0' \cdot x(n) + a_1' \cdot x(n-1) + a_2' \cdot x(n-2) + ...$$

pričom platia vzťahy:

$$a_i' = \frac{a_i}{b_0} \text{ a } b_i' = \frac{-b_i}{b_0}$$

Rádom filtra nazveme

$$\max\{i | a_i \neq 0 \vee b_i \neq 0\}.$$

Tento údaj teda vypovedá o tom, koľko predchádzajúcich vstupných alebo výstupných vzoriek filter využije pri výpočte aktuálneho výstupu. Nepriamo teda vypovedá, koľko posledných vstupných a výstupných vzoriek si musíme v štruktúre filtra pamätať.

Iný obvyklý spôsob vyjadrenia digitálneho filtra je pomocou prevodovej funkcie (angl. transfer function), ktorá vyjadruje vzťah medzi vstupom a výstupom filtra. Prechodová funkcia filtra $H(z)$ je definovaná $H(z) = \frac{Y(z)}{X(z)}$, kde $X(z)$ je *z*-transformácia vstupného a $Y(z)$ je *z*-transformácia výstupného signálu (diskrétne vyjadrenie pomocou vzoriek). Detaily ohľadom z-transformácie môže čitateľ nájsť v knihe [13].

Článok [18] predstavuje intuitívny spôsob ako prevádzať medzi sebou diferenčnú rovnicu a prechodovú funkciu filtra. Definujme operátor z^{-1}, ktorý, rovnako ako pri oneskorujúcich linkách znamená oneskorenie o jednu vzorku. Teda platí:

$$y(n-1) = z^{-1} y(n)$$

$$y(n-2) = z^{-1} y(n-1) = z^{-2} y(n)$$

Po prevedení príslušných substitúcií do diferenčnej rovnice dostávame:

$$b_0 \cdot y(n) + b_1 \cdot z^{-1} y(n) + b_2 \cdot z^{-2} y(n) + ... = a_0 \cdot x(n) + a_1 \cdot z^{-1} x(n) + a_2 \cdot z^{-2} x(n) + ...$$

$$y(n) \cdot (b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2} + ...) = x(n) \cdot (a_0 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2} + ...)$$

odkiaľ dostaneme

$$\frac{y(n)}{x(n)} = \frac{(a_0 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2} + ...)}{(b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2} + ...)}$$

Ako vidíme prechodová funkcia obsahuje všetky potrebné informácie o filtri. Navyše je to užitočný prostriedok na zistenie vlastností filtra ako napr. frekvenčná odozva filtra [16].

4.2 Rozdelenie filtrov podľa impulzovej odozvy

Digitálne filtre môžeme rozdeliť podľa viacerých kritérií. Jedným z nich je to, či pri filtrácii využívajú predchádzajúce výstupné hodnoty alebo nie. Filtre, ktoré nevyužívajú predchádzajúce výstupné hodnoty, sa označujú ako nerekurzívne alebo FIR (skratka od Finite Impulse Response) – filtre s konečnou impulzovou odozvou. Pre tieto filtre platí:

$$b_i = 0, \forall i = 1,2,...$$

Rekurzívne filtre sa tiež označujú ako IIR (Infinite Impulse Response) – filtre s nekonečnou impulzovou odozvou. Tieto filtre využívajú aj predchádzajúce výstupné hodnoty, t. j. $\exists i; b_i \neq 0$

Pod impulzovou odozvou rozumieme výstup filtra, ktorého vstupom je tzv. jednotkový impulz. Jednotkový impulz je postupnosť vzoriek $\delta(t)$, pre ktorú platí:

- $\delta(t) = 1$ pre $t = 0$
- $\delta(t) = 0$ pre $t > 0$

Ide teda o postupnosť začínajúcu jednotkou, za ktorou nasledujú samé nuly [18].

Pre FIR filtre platí, že ich impulzová odozva skončí po konečnom počte vzoriek [18], t.j. od istej vzorky výstupu sa všetky nasledujúce budú rovnať 0. Ľahko nahliadneme, že nerekurzívne filtre sú FIR. Posledný nenulový výstup totiž bude $y(n)$, kde $n = \max\{i | a_i \neq 0\}$.

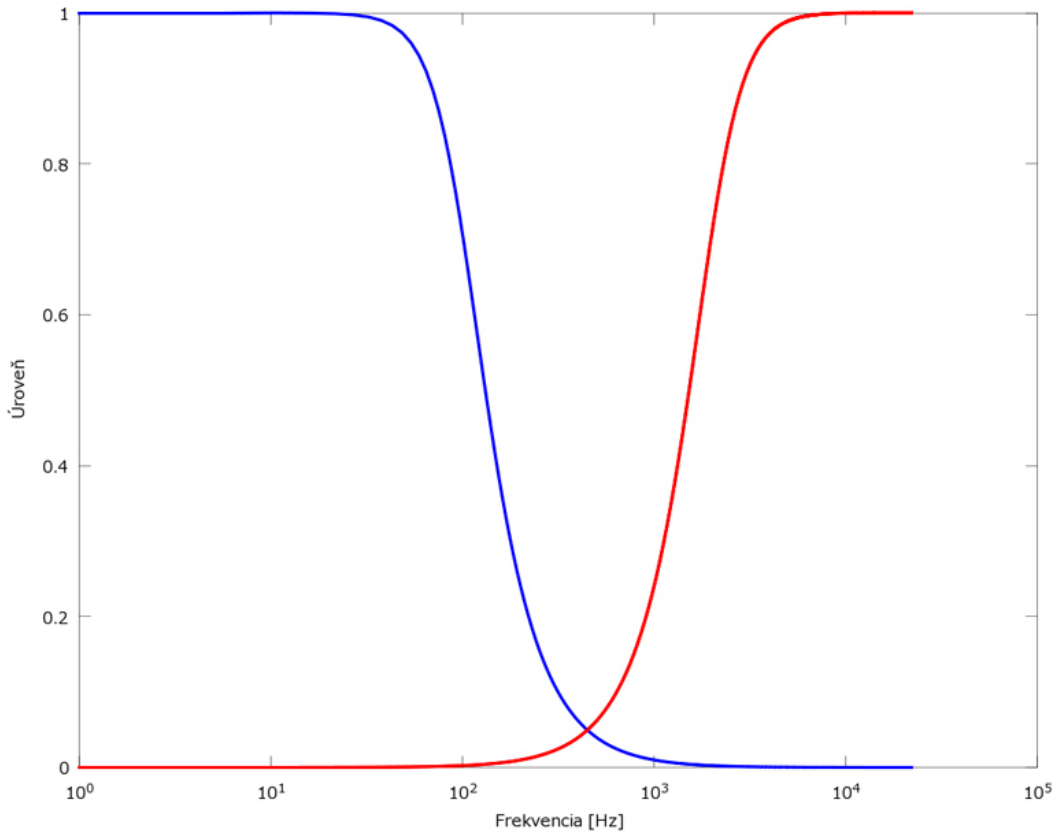
IIR filtre majú (teoreticky) nekonečnú impulzovú odozvu. V skutočnosti však impulzová odozva priblíži 0 v konečnom čase [18]. Tento jav je umocnený obmedzenou presnosťou digitálneho spracovania a chybami pri zaokrúhľovaní.

4.3 Rozdelenie filtrov podľa frekvenčnej odozvy

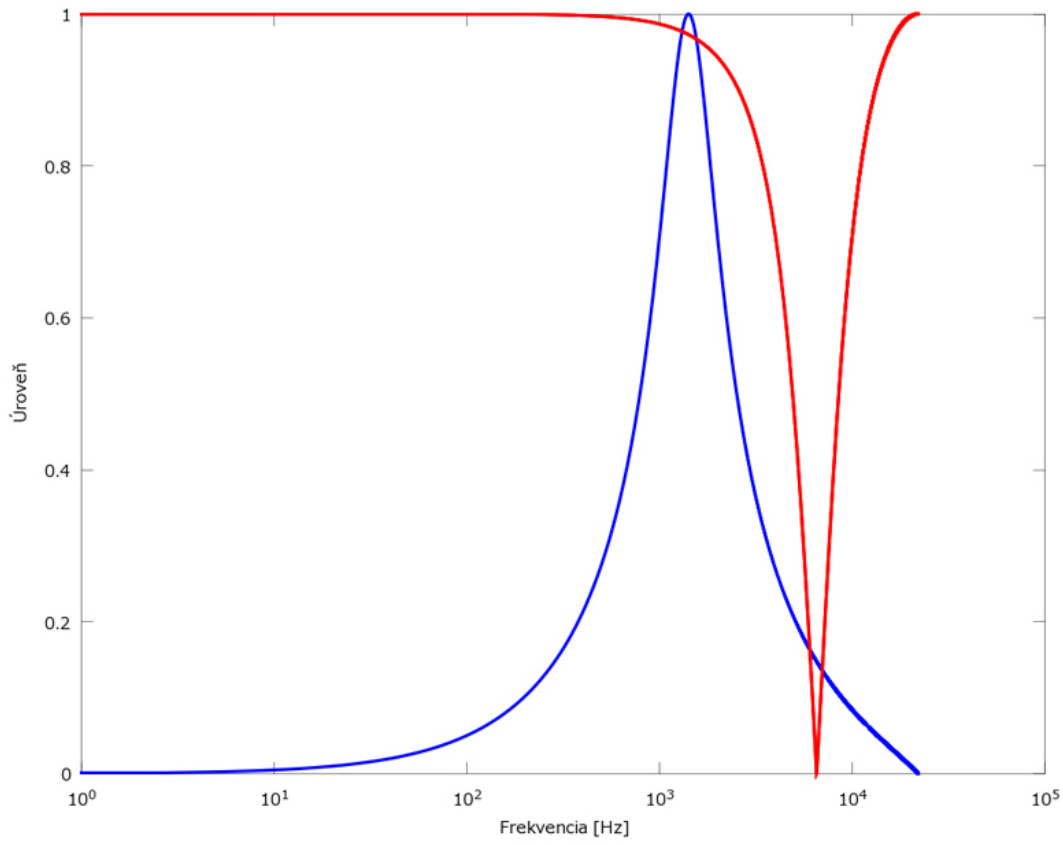
Druhou možnosťou delenia digitálnych filtrov je ich frekvenčná odozva. Frekvenčná odozva vypovedá o tom, ktoré zložky zo vstupného signálu filter prepustí (prípadne tiež zoslabí, zosilní, ...).

Základným filtrom je dolnopriepustný filter. Tento typ filtra prepustí všetky zložky s frekvenciami nižšími ako zvolená hraničná frekvencia a potlačí zložky vyšších frekvencií. Jeho opakom je hornopriepustný filter, ktorý naopak to, čo je pod zvolenou frekvenciou potlačí a zvyšok prepustí. Frekvenčnú odozvu dolnopriepustného filtra s hraničnou frekvenciou 100 Hz znázorňuje na obrázku 4.2 modrá krivka. Červená krivka na rovnakom obrázku znázorňuje frekvenčnú odozvu hornopriepustného filtra s hraničnou frekvenciou 2000 Hz.

Pásmová priepusť je typ filtra, ktorý prepúšťa vybrané frekvenčné pásmo. Toto frekvenčné pásmo je dané dvojicou hraničných frekvencií. Opakom pásmovej priepusti je pásmová zádrž, čiže filter, ktorý prepustí všetky frekvencie okrem vyznačeného pásma. Modrá krivka na obrázku 4.3 znázorňuje frekvenčnú odozvu pásmovej priepusti s hraničnými frekvenciami 1000 Hz a 2000 Hz. Červenou farbou, na rovnakom obrázku, je zachytená frekvenčná odozva pásmovej zádrže, ktorej hranične frekvencie sú 4000 Hz a 10000 Hz.



Obrázok 4.2: Frekvenčná odozva dolnpriepustného a hornopriepustného filtra



Obrázok 4.3: Frekvenčná odozva pásmovej priepusti a pásmovej zádrže

4.4 Návrh digitálnych filtrov

Existuje množstvo softwarových prostriedkov, ktoré automaticky navrhnú či už analógový, alebo digitálny filter. Výsledkom je však filter, ktorý má pevne zafixované parametre ako hraničná frekvencia alebo vzorkovacia frekvencia signálu, s ktorým filter pracuje. Naša aplikácia však umožňuje pracovať so vstupnými signálmi s rôznymi vzorkovacími frekvenciami. Preto potrebujeme spôsob ako automaticky navrhnúť filter podľa aktuálnych požiadaviek.

Existuje viacero metód, pomocou ktorých je možné spočítať koeficienty digitálneho filtra s požadovanými vlastnosťami. My sme použili metódu, ktorú popisuje Elena Punskaya v materiáloch [11]. Táto metóda využíva bilineárnu transformáciu aplikovanú na analógový filter. (Ide o konverziu analógového filtra na digitálny). Pomocou substitúcie

$$s = \psi(z) = \frac{1 - z^{-1}}{1 + z^{-1}},$$

transformujeme prechodovú funkciu analógového filtra $H_A(s)$ na prechodovú funkciu digitálneho filtra $H_D(z)$.

Táto transformácia mapuje frekvencie Ω z intervalu $(-\infty, \infty)$ na frekvencie ω intervalu $(-\pi, \pi)$. Toto mapovanie je monotónne, hodnota $\Omega = 0$ sa mapuje na hodnotu $\omega = 0$ a $\Omega = \infty$ na $\omega = \pi$ (polovica vzorkovacej frekvencie). A vzťah medzi frekvenciami Ω a ω je nasledovný: $\Omega = \tan\left(\frac{\omega}{2}\right) \Leftrightarrow \omega = 2 \cdot \arctan(\Omega)$ [11]. Teoretické detaily tejto metódy môže čitateľ nájsť v knihe [13].

Pripomeňme že frekvencie používané v digitálnej prechodovej funkcii sú tzv. normalizované frekvencie. To znamená sú to frekvencie namapované v intervale $(-\pi, \pi)$. Normalizovanú frekvenciu Ω získame z frekvencie f zadanej v Hz pomocou vzťahu:

$$\Omega = \frac{f}{f_s} \cdot 2\pi,$$

kde f_s je vzorkovacia frekvencia vstupného signálu [11].

Postup pre vytvorenie digitálneho filtra je teda nasledovný:

- Z požadovaných normalizovaných frekvencií ω_i vypočítame analógové hodnoty $\Omega_i = \tan\left(\frac{\omega_i}{2}\right)$.
- Navrhneme analógový filter spĺňajúci naše požiadavky.
- Aplikujeme bilineárnu transformáciu.

Analógové prototypy filtrov zvyčajne bývajú definované ako dolnopriepustné filtre. Materiál [11] poskytuje návod ako z takéhoto prototypu odvodiť hornopriepustný filter, resp pásmovú priepusť či zádrž. Použijeme substitúcie do rovnice analógového prototypu, ktoré sú uvedené v tabuľke 4.1 (Tabuľka 4.1 je prevzatá z materiálu [11]). Ω_c, Ω_u, Ω_l sú požadované hraničné frekvencie (jednostranná, vrchná a spodná).

Konverzia na ...	Použitá substitúcia	Nová hraničná frekvencia
Dolnopriepustný filter	$s \leftarrow s/\Omega_c$	Ω_c
Hornopriepustný filter	$s \leftarrow \Omega_c / s$	Ω_c
Pásmová priepusť	$s \leftarrow \left(s^2 + \Omega_l \Omega_u\right)/\left(s \cdot \left(\Omega_u - \Omega_l\right)\right)$	Ω_l, Ω_u
Pásmová zádrž	$s \leftarrow s \cdot \left(\Omega_u - \Omega_l\right)/\left(s^2 + \Omega_l \Omega_u\right)$	Ω_l, Ω_u

Tab 4.1: Substitúcie používane pri konverziách dolnopriepustných filtrov, prevzaté z [11]

4.5 Implementácia

V našej aplikácii sme sa rozhodli implementovať niekoľko IIR filtrov predovšetkým, kvôli tomu, že na dosiahnutie požadovanej frekvenčnej odozvy stačia filtre nižšieho rádu (oproti FIR filtrom) a teda potrebujú menej operácii na výpočet výsledku [18].

Spoločnou nadtriedou pre všetky filtre v aplikácií je trieda *Filter*. Pre naše potreby sme implementovali viacero tried reprezentujúce IIR filtre. Sú to hlavne triedy *SecondOrderFilter*, *FourthOrderFilter* a *EighthOrderFilter*. Tieto triedy implementujú všeobecný IIR filter druhého, štvrtého a ôsmeho rádu. Na výpočet výstupnej hodnoty využívajú všeobecnú diferenčnú rovnicu pre filter daného rádu. Funkcia filtra teda závisí od hodnôt koeficientov a_i a b_i v diferenčnej rovnici

Pomocou týchto tried je možné implementovať aj FIR filtre daného rádu, a to tak, že koeficienty, ktorými sa násobia predošlé výstupy nastavíme na 0. Poznamenajme však, že imeplementácia špecializovaná na nerekuzívny (FIR) filter bude rýchlejšia a menej náročná.

Všimnime si, že substitúcie, ktoré prevádzajú prototyp dolnopriepustného filtra na pásmovú priepusť, resp. zádrž, zdvojnásobujú maximálnu mocninu s v prechodovej funkcii. Rovnako tak zdvojnásobujú aj rád výsledného digitálneho filtra. Preto sme sa rozhodli implementovať filtre párnych rádov. Filtre rádov 2 a 4 patria v praxi medzi najpoužívanejšie [16]. Filter rádu 8 sme pridali kvôli použitiu v prípadoch, že je požadovaná presnejšia frekvenčná odozva (vzťah medzi rádom a frekvenčnou odozvou popisujeme ďalej v tejto kapitole).

Tieto triedy disponujú statickými metódami, ktoré pripravia koeficienty a_i a b_i podľa požadovaných vlastností filtra. Potrebnými hodnotami sú hraničná frekvencia (resp. spodná a vrchná hraničná frekvencia v prípade pásmovej priepusti), vzorkovacia frekvencia vstupného signálu.

Metódy vypočítajú koeficienty, ktoré implementujú tzv. Butterworthove filtre. Výhodou Butterworthových filtrov je predovšetkým hladká frekvenčná odozva v prepustenom pásme [12], t. j. úroveň frekvencií v tomto pásme je 1. Analógové prototypy Butterworthových filtrov nájdeme v tabuľke 4.2, ktorá je prevzatá z článku [17]. Ako analógové prototypy filtrov využívajú prechodové funkcie príslušného rádu z tabuľky. Pri výpočte koeficientov filtrov sú použité vzťahy odvodené podľa metódy uvedenej v kapitole 4.4.

Filtre vyššieho rádu využívajú na výpočet výstupnej hodnoty väčší počet predchádzajúcich vstupných a výstupných hodnôt. To znamená, že potrebujú vykonať viac operácii v každom kroku. Ich výhodou je však presnejšie dosiahnutie požadovanej frekvenčnej odozvy [16], [12], [17]. Ako môžeme pozorovať na

obrázku 4.4, krivka, ktorá ohraničuje prepustené pásmo, je s rastúcim stupňom strmšia. Obrázok znázorňuje frekvenčné odozvy troch Butterworthových dolnopriepustných fiiltrov. Každý z nich má hraničnú frekvenciu 1000 Hz. Modrá krivka popisuje frekvenčnú odozvu filtra rádu 2, červená rádu 4 a zelená rádu 8.

Rád filtra	Analógová prechodová funkcia
1	$\frac{1}{s+1}$
2	$\frac{1}{s^2+\sqrt{2}\cdot s+1}$
3	$\frac{1}{(s+1)\cdot(s^2+s+1)}$
4	$\frac{1}{(s^2+0.7654\cdot s+1)\cdot(s^2+1.8478\cdot s+1)}$
5	$\frac{1}{(s+1)\cdot(s^2+0.618\cdot s+1)\cdot(s^2+0.618\cdot s+1)}$
6	$\frac{1}{(s^2+0.5176\cdot s+1)\cdot(s^2+\sqrt{2}\cdot s+1)\cdot(s^2+1.9319\cdot s+1)}$
7	$\frac{1}{(s+1)\cdot(s^2+0.445\cdot s+1)\cdot(s^2+1.247\cdot s+1)\cdot(s^2+1.8019\cdot s+1)}$
8	$\frac{1}{(s^2+0.3902\cdot s+1)\cdot(s^2+1.1111\cdot s+1)\cdot(s^2+1.6629\cdot s+1)\cdot(s^2+1.9616\cdot s+1)}$

Tab 4.2: Prototypy Butterworthových filtrov, prevzaté zo [17]

4.6 Alternatívne odvodenie hornopriepustného filtra

Substitúcie uvedené v tabuľke 4.1 boli použité pri implementovaní dolnopriepustného filtra a pásmovej priepusti. Pre získanie koeficientov hornopriepustného filtra sme použili metódu ktorá je uvedená v materiáloch [11]. Ak máme dolnopriepustný filter daný diferenčnou rovnicou:

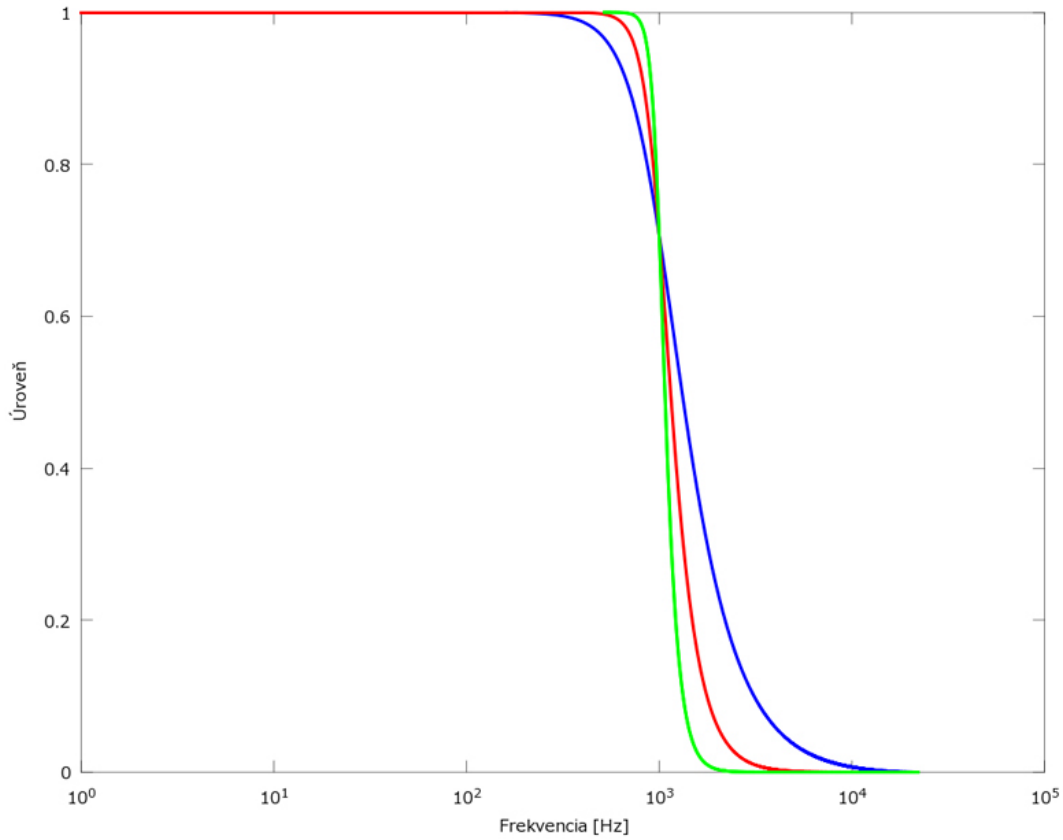
$$y(n) = b_1 \cdot y(n-1) + b_2 \cdot y(n-2) + \ldots + a_0 \cdot x(n) + a_1 \cdot x(n-1) + a_2 \cdot x(n-2) + \ldots =$$

$$\sum_{k=1}^{N} b_k \cdot y(n-k) + \sum_{k=0}^{M} a_k \cdot x(n-k),$$

potom hornopriepustný filter môžeme vyjadriť ako:

$$y(n) = \sum_{k=1}^{N} (-1)^k \cdot b_k \cdot y(n-k) + \sum_{k=0}^{M} (-1)^k \cdot a_k \cdot x(n-k)$$

Veľkou výhodou tejto metódy je fakt, že využívame už vypočítané koeficienty digitálneho filtra. Ušetríme tak pracné odvodzovanie týchto koeficientov pomocou substitúcie z tabuľky 4.1 a následnej bilineárnej transformácie.



Obr 4.4: Frekvenčná odozva dolnopriepustného filtra 2., 4. a 8. rádu

4.7 Ekvalizér

Trojpásmový ekvalizér sa skladá z troch nezávislých filtrov, takže umožňuje nastaviť úroveň v troch rôznych frekvenčných pásmach. Tieto pásma sú nastavené nasledovne:

- basové pásmo pokrýva interval 20 Hz – 200 Hz
- pásmo stredov pokrýva interval 200 Hz – 2000 Hz
- pásmo výšok pokrýva interval 2000 Hz – 20480 Hz

Jednotlivé pásma sú kontrolované pomocou tzv. peak filtra (niekedy tiež označovaný ako peaking filter). Ide o typ filtra, ktorý zosilňuje, resp. zoslabuje zložky vo vybranom frekvenčnom pásme, zatiaľ čo ostatné necháva neovplyvnené. Tento filter je podľa U. Zölzera [16] možné realizovať pomocou paralelného zapojenia priamej cesty a pásmovej priepusti. Prechodovú funkciu teda môžeme napísať ako:

$$H(s) = 1 + H_{BP}(s)$$

kde, $H_{BP}(s)$ je prechodová funkcia pásmovej priepusti. Zoslabenie, alebo zosilnenie je možné kontrolovať pomocou koeficientu H_0, ktorým sa prenásobí výstup pásmovej priepusti.

$$H(s) = 1 + H_0 \cdot H_{BP}(s)$$

Ekvalizér teda využíva 3 rôzne pásmové priepuste a priamu linku, tak ako znázorňuje obrázok 4.5. Užívateľ môže nastaviť úroveň každého pásma nezávisle

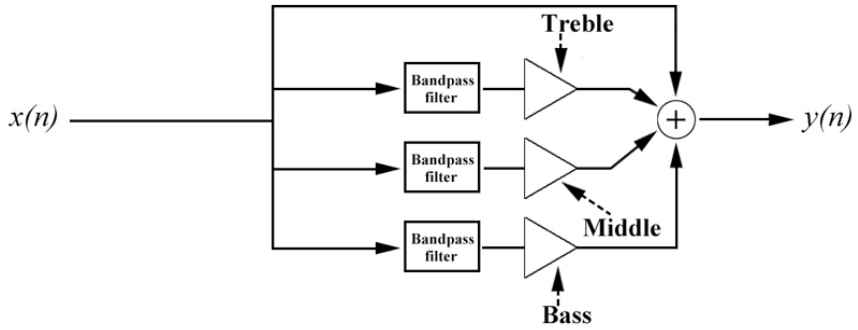
pomocou parametrov Bass, Middle a Treble. Hodnoty parametrov patria do intervalu $[-6dB,6dB]$. Koeficient, ktorým prenásobíme výstup filtra získame z tejto hodnoty pomocou vzťahu:

$$H_0 = 10^{\frac{g_{dB}}{20}} - 1$$

kde $g_{dB} \in [-6dB,6dB]$, je užívateľom zvolená hodnota zisku. Tento vzťah je odvodený zo vzťahov:

- $V_0 = 10^{\frac{g_{dB}}{20}}$, v prípade zosilnenia
- $V_0 = 10^{\frac{-g_{dB}}{20}}$, v prípade zoslabenia.

V našom prípade nadobúda g_{db} záporné hodnoty v prípade zoslabenie, preto nie je nutné prenásobenie hodnotou –1. Naopak odpočítanie 1 bolo odvodené zo vzťahu $V_0 = 1 + H_0$

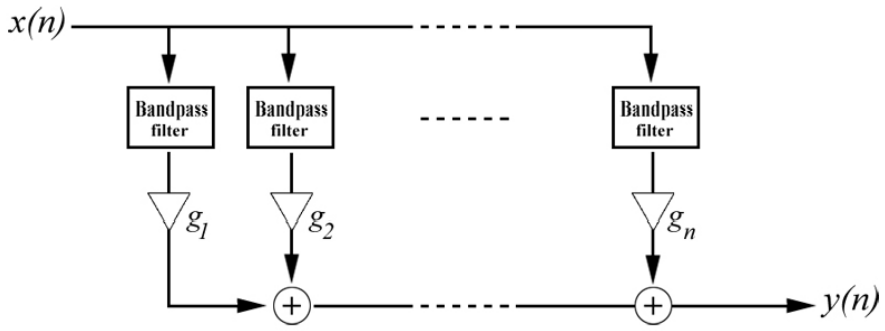


Obrázok 4.5: Štruktúra trojpásmového ekvalizéru

4.8 Grafický ekvalizér

Grafický ekvalizér je druh viacpásmového ekvalizéra, ktorý dostal svoj názov vďaka spôsobu ovládania. Úroveň každého pásma sa nastavuje pomocou posuvného potenciometra vo vertikálnom smere. Pohľad na výsledné nastavenia teda približne zodpovedá frekvenčnej odozve celého ekvalizéra.

Grafický ekvalizér môžeme implementovať paralelným zapojením viacerých pásmových priepustí, pričom výstup každého filtra bude mať úroveň kontrolovanú užívateľským nastavením (koeficienty g_i pri jednotlivých pásmach) [8]. Takéto zapojenie nám umožňuje jednotlivé pásma iba zoslabovať. V prípade, že by sme do zapojenia paralelne pridali čistý vstupný signál, pásmové priepuste by pracovali ako peak filtre a dostali by sme možnosť aktívne zosilňovať jednotlivé pásma.



Obrázok 4.6: Štruktúra grafického ekvalizéru

Typický počet pásiem grafického ekvalizéra sa pohybuje od 6 do 30. Voľba počtu pásiem závisí od určenia efektu. V našej implementácii sme sa rozhodli použiť často používaných 10 pásiem, z ktorých každé má šírku jednej oktávy. Rozloženie pásiem v počuteľnom spektre vidíme v tabuľke 4.3.

Pásmo	Stredná frekvencia - f_c	Hraničné frekvencie pásma - f_l, f_u
1	32 Hz	22 Hz, 45 Hz
2	63 Hz	45 Hz, 89 Hz
3	125 Hz	89 Hz, 176 Hz
4	250 Hz	176 Hz, 353 Hz
5	500 Hz	353 Hz, 707 Hz
6	1000 Hz	707 Hz, 1414 Hz
7	2000 Hz	1414 Hz, 2828 Hz,
8	4000 Hz	2828 Hz, 5656 Hz
9	8000 Hz	5656 Hz, 11313 Hz
10	16000 Hz	11313 Hz, 22627 Hz

Tab 4.3: Hraničné frekvencie pásiem v 10-pásmovom ekvalizéri

Každé pásmo má šírku jednej oktávy. Teda jeho vrchná hraničná f_u je dvojnásobkom dolnej hraničnej frekvencie f_l: $f_u = 2 \cdot f_l$. Stredná frekvencia f_c je v polovici intervalu ohraničeného f_l, f_u. Od oboch hraničných frekvencií je teda vzdialená pol oktávy. Platí teda $f_l \cdot \sqrt{2} = f_c = f_u / \sqrt{2}$. Hodnoty uvedené v tabuľke sú približné a zaokrúhlené, aby korešpondovali s typicky uvádzanými frekvenciami.

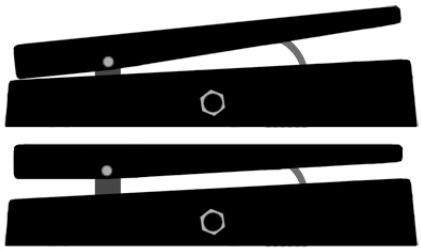
4.9 Grafický ekvalizér – poznámka k implementácii

V implementácii grafického ekvalizéra sme použili pásmové priepuste druhého rádu. Pôvodne sme chceli použiť filtre ôsmeho rádu pomocou, ktorých by sme mohli dosiahnuť menšie presahy medzi pásmami filtrov a teda umožniť užívateľovi presnejšie nastavenia. Ukázalo sa však, že použitie 10-tich IIR filtrov ôsmeho rádu, je príliš náročné na výpočtový výkon. Na každú vzorku zo vstupného signálu v tomto prípade pripadá 17 násobení a 16 sčítaní v každom filtri. Spolu je to 330 aritmetických operácií, ku ktorým musíme pripočítať následné sčítanie výstupov z filtrov (19 operácií). Pri vstupnom signále so vzorkovacou frekvenciou 44100 Hz, by to znamenalo uskutočniť 15,390,000 operácií za sekundu. Pritom sme ešte zanedbali presúvanie predošlých vstupov a výstupov v rámci filtra.

4.10 Wah-Wah

Pomocou efektu zvaného Wah-wah, alebo kvákadlo môžeme zvuk podobný ľudskému hlasu, ktorý hovorí: „Kvák, kvák!“ Najčastejšie je tento efekt využívaný hráčmi na elektrickú gitaru. Za svoju masovú popularitu vďačí predovšetkým Jimimu Hendrixovi. Výbornú ukážku tohto typu efektu predstavuje úvod Hendrixovej skladby Voodo Chile.

Základom efektu je frekvenčný filter, ktorý zvýrazňuje úzky rozsah frekvencií okolo strednej frekvencie a ostatné potláča. Stredná frekvencia je nastaviteľná a práve rýchle zmeny v nastavení tohto parametra produkujú charakteristický zvuk efektu. Ovládanie tohto parametru je takmer vždy možné uskutočniť pomocou pohybu chodidla, ktoré posunie pedál a nastaví tak strednú frekvenciu filtra. Tento systém zaručuje, že ruky hráča ostanú voľné. Obrázok 4.7 zachytáva pohyblivú časť efektu, v dvoch krajných polohách. V polohe, keď je pedál stlačený úplne nadol (hráč pritlačí špičku chodidla), je stredná frekvencia filtra nastavená na najvyššiu možnú. Naopak, keď hráč pritlačí pätou a zdvihne pedál do najvyššej polohy, je stredná frekvencia filtra nastavená na najnižšiu hodnotu.



Obr 4.7: Wah pedál v krajných polohách nastavenia

R. G. Keen v článku [5] na diagrame zachycuje typickú frekvenčnú odozvu wah pedálu. Ako vidíme (obrázok 4.8) podobný efekt je možné realizovať pomocou pásmovopriepustného filtra, resp. dolnopriepustného filtra.

Pre implementovanie filtra sme sa rozhodli použiť štruktúru filtra, ktorú prezentuje S. K. Mitra v [9]. Ide o digitálne filtre s nekonečnou impulznou odozvou, ktorým je možné nastaviť hraničnú frekvenciu, prípadne aj šírku pásma pri pásmovo priepustnom filtri. Jeho výhodou je, to že pri zmene centrálnej frekvencie, resp. šírky priepustného pásma, nie je potreba počítať všetky koeficienty diferenčnej rovnice.

Filter má riadiace koeficienty α a β. Koeficient α riadi šírku pásma a získame ho pomocou vzťahu:

$$\cos(\omega_c) = \frac{2\alpha}{1+\alpha^2},$$

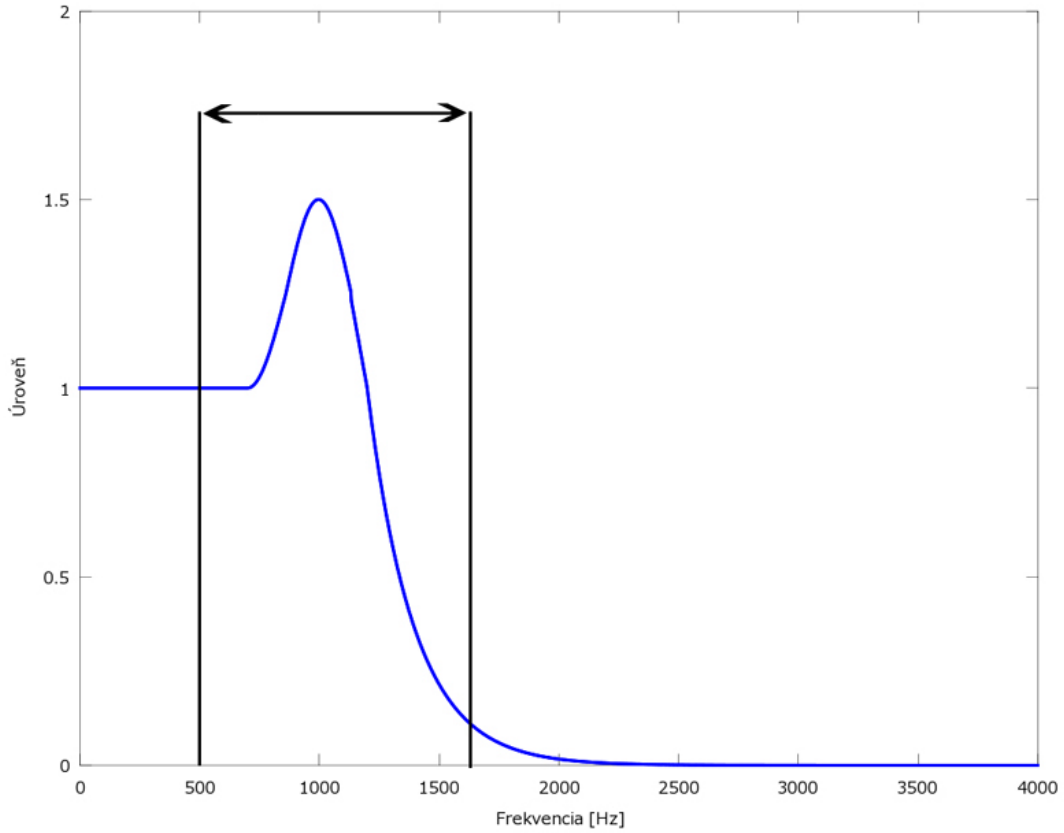
po úprave dostávame stabilné riešenie:

$$\alpha = \frac{1-\sin(\omega_c)}{\cos(\omega_c)}.$$

Koeficient β riadi strednú frekvenciu a jeho hodnotu vypočítame pomocou vzťahu:

$$\beta = \cos(\omega_c),$$

kde ω_c označuje normalizovanú strednú frekvenciu.



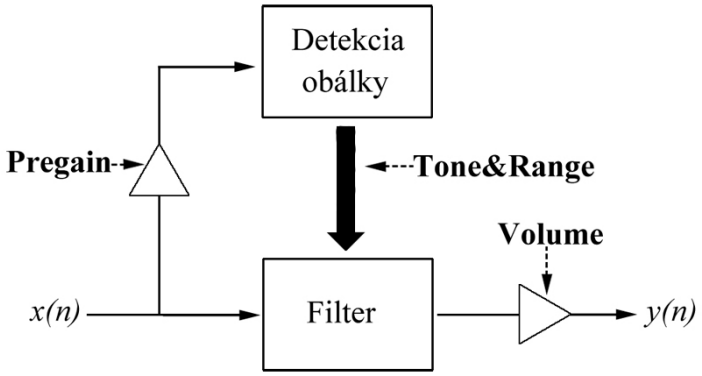
Obr 4.8: Frekvenčná odozva efektu wah, prevzaté z [5]

J. O. Smith III v knihe [15] prezentuje výsledky meraní amplitúdovej odozvy wah pedálu Dunlop Cry Baby, v troch rôznych polohách. Z diagramov môže získať približnú predstavu o nastavení použitého filtra. V polohe, keď je pedál posunutý úplne dozadu, t.j. stredná frekvencia je nastavená na najnižšiu hodnotu vo svojom rozsahu, sú krajné body pásma ($-3db$) približne $2500rad \cdot s^{-1}$ a $3500rad \cdot s^{-1}$. Tieto hodnoty zodpovedajú frekvenciám $398Hz$ a $557Hz$. V strednej polohe sú krajné body približne $4500rad \cdot s^{-1}$ a $6000rad \cdot s^{-1}$ ($716Hz$ a $955Hz$). A nakoniec v opačnej krajnej polohe sú krajné body zhruba $11000rad \cdot s^{-1}$ a $17000rad \cdot s^{-1}$ ($1750Hz$ a $2706Hz$). Z týchto výsledkov vidíme, že šírka pásma sa s posunom polohy pedálu smerom dopredu zväčšuje. Preto sme v našej implementáciu použili lineárnu závislosť medzi strednou frekvenciou a šírkou pásma.

Pri vývoji efektu sa ako najvýhodnejšie ukázali nasledujúce nastavenia. Stredná frekvencia filtra sa pohybuje v rozmedzí $200Hz$ až $4200Hz$. Šírka prepusteného pásma v najnižšej polohe je $200Hz$ a v najvyššej $3200Hz$. Výsledne parametre sa tak mierne odlišujú od efektu Dunlop Cry Baby. Je to spôsobené tým, že sme sa ani nepokúšali vytvoriť digitálnu podobu tohto efektu a tiež faktom, že Cry Baby je efekt určený špeciálne pre elektrickú gitaru. Preto má nastavený taký rozsah, ktorý zodpovedá frekvenčnému rozsahu elektrickej gitary.

4.11 Envelope filter

Envelope filter je efekt využívajúci filter, ktorého nastavenia sú ovládané obálkou vstupného signálu. Typickým príkladom efektu tohto typu je autowah, čiže wah efekt, ktorý mení svoju strednú frekvenciu automaticky na základe vstupného signálu.



Obr 4.9: Štruktúra efektu Envelope filter

Základná štruktúra efektu je zobrazená na obrázku 4.9. Vstupný signál prechádza filtrom a opúšťa efekt. Ešte pre vstupom do filtra je tiež privádzaný do časti, ktorá určuje obálku signálu a na jej základe ovláda nastavenia filtra. Táto štruktúra je univerzálna môžu v nej byť použité rozličné spôsoby vyhodnocovania signálu, ktoré riadia filter a rovnako aj rozličné typy filtrov.

Pre náš envelope filter sme sa rozhodli implementovať efekt, ktorý bude využívať rovnaký filter, aký sme využili aj pri implementácii efektu wah. Výsledný efekt by sa teda do istej miery mal správať ako autowah. Typické „kváknutie“ sa dosahuje tým, že hráč počas hrania tónu nastaví frekvenciu filtra z nižšej na vyššiu a naspäť. Aby sme podobný jav dosiahli automaticky, za zariadenie ovládajúce filter, zvolíme merač sily signálu, tzv. envelope detector. Teda zariadenie, ktoré zisťuje amplitúdovú obálku vstupného signálu. Tento blok podľa aktuálnej sily signálu nastaví strednú frekvenciu filtra (a tiež aj šírku prepusteného pásma, podobne ako pri efekte wah).

Konkrétne implementujeme štruktúru popísanú U. Zölzerom v knihe [16]. Ide o jednoduchý digitálny filter, ktorého nám umožňuje jednoducho určiť RMS hodnotu posledných N vzoriek vstupného signálu $x(n)$.

$$x_{RMS}(n) = \sqrt{\frac{1}{N}\sum_{i=0}^{N} x^2(n-i)}$$

Zavedieme priemerovací koeficient:

$$TAV = 1 - \exp\left(\frac{-2.2 \cdot T_s}{t_M / 1000}\right),$$

kde T_s je vzorkovací interval a t_m je dĺžka časového intervalu, cez ktorý prebieha meranie. Diferenčná rovnica filtra podľa publikácie [9] je

$$x_{RMS}^2(n) = (1 - TAV) \cdot x_{RMS}^2(n-1) + TAV \cdot x^2(n).$$

Výhodou tohto riešenia, je fakt, že pracuje v lineárnom čase.

Zjednodušene by sa dalo povedať, že medzi silou signálu a strednou frekvenciou filtra je lineárna závislosť. Teda čím je sila signálu silnejšia tým vyššia je stredná frekvencia. Túto závislosť môžeme zachytiť vzťahom:

$$f_c = f_{\min} + x_{RMS} \cdot f_{range}.$$

Toto nastavenie sa však uplatní iba v prípade, že sila signálu presiahne prahovú hodnotu (angl. threshold) $x_{RMS} \geq threshold$. Ak však bude platiť $x_{RMS} < threshold$ filter sa ponechá bezo zmien.

Ako vidíme na obrázku 4.9 Užívateľ má možnosť ovplyvniť efekt viacerými parametrami. Pregain kontroluje zosilnenie vstupného signálu predtým ako bude meraná jeho aktuálna úroveň a umožňuje tak aj slabším signálom presiahnuť prahovú hodnotu. Tone nastavuje strednú frekvenciu ktorú má filter, ak sila signálu nedosahuje prahovú hodnotu. Range ovláda rozsah frekvencií, ktorými filter prejde pri zmene úrovne vstupného signálu. Volume ovláda výstupnú úroveň signálu.

Kapitola 5

Efekty využívajúce Fourierovu transformáciu

5.1 Rýchla Fourierova transformácia

Diskrétna Fourierova transformácia je užitočný nástroj, ktorý umožňuje reprezentovať signál pomocou jeho spektra. Spektrum môžeme chápať ako postupnosť čísel, ktoré udávajú veľkosť príslušných frekvenčných zložiek v skúmanom signále. Diskrétna Fourierova transformácia signálu x je definovaná nasledovne:

$$X(\omega_k) = \sum_{n=0}^{N-1} x(t_n) e^{-i\omega_k t_n}\ ,\ k = 0,1,.....,N-1$$

kde:

- $t_n = n \cdot T$, pričom $T = \frac{1}{f_s}$ je časový interval medzi dvoma nasledujúcimi vzorkami pri vzorkovacej frekvencii f_s
- $\omega_k = k \cdot \Omega$ je k-ta frekvenčná zložka
- $\Omega = \frac{2\pi}{NT}$ je interval medzi frekvenčnými zložkami

Keďže v praxi pracujeme výlučne so vzorkovaným signálom, preto budeme používať značenie vstupného signálu x aj výstupných frekvenčných zložiek X ako postupnosti:

$$x_n = x(t_n),\ n = 0,1,...,N-1$$
$$X_k = X(\omega_k),\ k = 0,1,...,-1$$

Uvedenú definíciu môžeme upraviť dosadením t_n, ω_k a Ω. Dostávame tak obvyklý tvar definície:

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-2\pi i}{N} kn}\ ,k = 0,1,.....,N-1$$

Inverznú transformáciu definujeme ako:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn}\ ,\ n = 0,1,.....,N-1$$

Pre N vzoriek reálneho vstupného signálu x, dostaneme N komplexných čísel X. Na tieto čísla sa môžeme pozerať ako na koeficienty, ktoré udávajú veľkosti jednotlivých frekvenčných zložiek [14].

Použitím Eulerovho vzťahu: $e^{i\theta} = \cos(\theta) + i \cdot \sin(\theta)$, môžeme definíciu DFT prepísať ako:

$$X_k = \sum_{n=0}^{N-1} x_n \left(\cos\left(\frac{-2\pi}{N} kn \right) + i \cdot \sin\left(\frac{-2\pi}{N} kn \right) \right)$$

Využitím toho, že funkcia kosínus je párna a sínus nepárna dostaneme:

$$X_k = \sum_{n=0}^{N-1} x_n \left(\cos\left(\frac{2\pi}{N} kn \right) - i \cdot \sin\left(\frac{2\pi}{N} kn \right) \right)$$

Dostávame sa tak k definícii, ktorú nepriamo používa S. M. Bernsee v článku [1], kde predstavuje intuitívny prístup k výsledkom DFT. Reálnu zložku čísla X_k môžeme teda považovať za veľkosť kosínusovej zložky s príslušnou frekvenciou ω_k a imaginárnu časť za veľkosť sínusovej zložky s rovnakou frekvenciou.

5.2 Windowing

Windowing je technika, ktorá umožňuje vylepšiť výsledky diskrétnej Fourierovej transformácie. Predpokladajme, že vstupom pre Fourierovu transformáciu je krátky úsek vstupného signálu pozostávajúci z N vzoriek $x_0,...,x_{N-1}$.

Nevýhodou Fourierovej transformácie aplikovanej na krátky úsek vstupného signálu sú zafixované frekvenčné zložky (tzv. referenčné frekvencie), na ktoré rozkladá vstupný signál. Referenčné frekvencie sú celočíselné násobky základnej frekvencie:

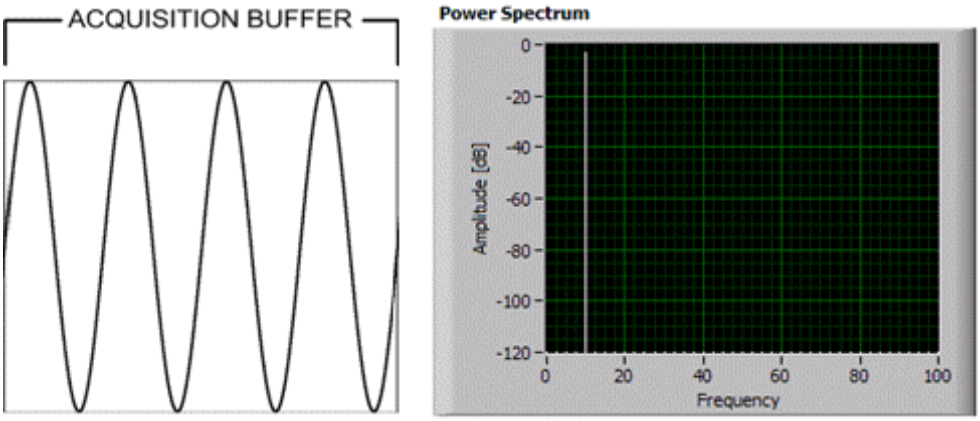
$$f_k = k \frac{2\pi}{N},\ k = 0,1,...N-1.$$

Základná frekvencia

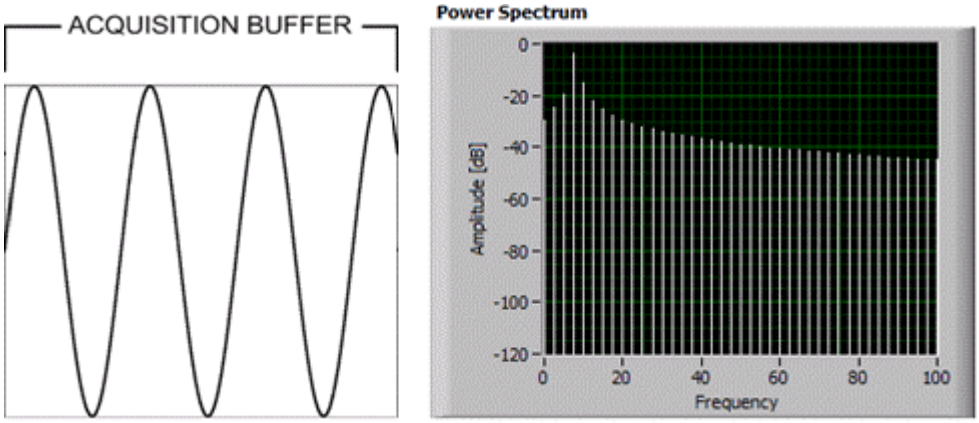
$$f_1 = \frac{2\pi}{N}$$

má dĺžku periódy N vzoriek. Ľahko nahliadneme, že aj vyššie referenčné frekvencie majú v intervale N vzoriek celočíselný počet períód. Fourierova transformácia predpokladá, že postupnosť vzoriek na vstupe je jedna perióda periodického signálu a teda krajné vzorky sú spojené (inak povedané, že po poslednej vzorke x_{N-1} opäť nasleduje prvá x_0) [21]. V praxi však táto podmienka nie je zaručená a krajné vzorky môžu byť ľubovoľne vzdialené.

Dopad vstupného signálu, ktorého frekvencia leží medzi referenčnými frekvenciami je popísaný v článkoch [2], [21]. Z obrázkov 5.1 a 5.2, prevzatého z článku [21], vidíme že vstupný signál sa prejaví príspevkom nielen na najbližších referenčných frekvenciách, ale je rozmazaná v celom pásme. Najhoršie sa pritom prejavia signály s najväčšou nespojitosťou medzi krajnými vzorkami.



Obr 5.1: Výsledok FT aplikovanej na signál s frekvenciou zodpovedajúcuo referenčnej frekvencii, prevzaté z [21]



Obr 5.2: Výsledok FT aplikovanej na signál, ktorého frekvenciu nezodpovedá žiadnej referenčnej frekvencii, prevzaté z [21]

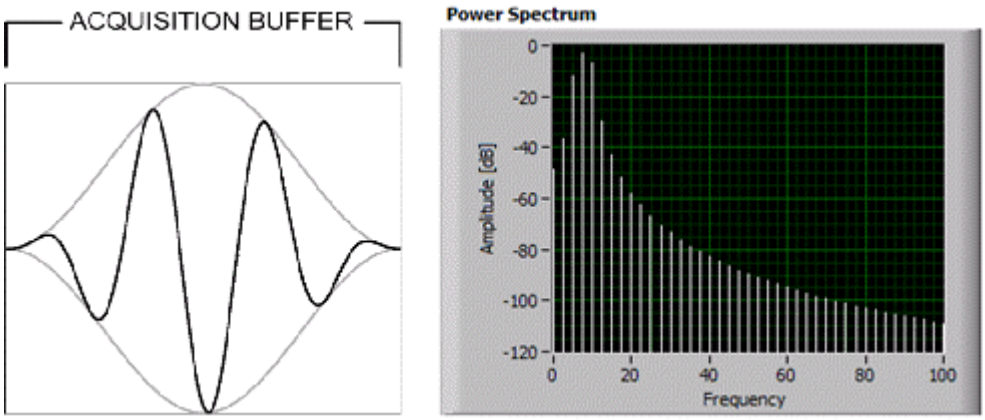
Čiastočným riešením je práve aplikovanie oknovej funkcie (angl. window function) na vzorky vstupného signálu predtým ako je prevedená Fourierova transformácia. Oknová funkcia násobí každú vzorku váhou, pričom táto váha závisí na poradí vzorky v úseku signálu. Opäť budeme uvažovať úsek signálu tvorený N vzorkami $x_0,...,x_{N-1}$. Váhy pre jednotlivé vzorky označíme $w_0,....,w_{N-1}$.

Najjednoduchšou oknovou funkciou je štvorcové (niekedy tiež pravouhlé) okno. Táto funkcia všetkým vzorkám priraďuje konštantnú váhu rovnú 1.

$$w_i = 1,\ i = 0,1,...,N-1$$

Toto okno nijako vzorky neovplyvní a teda ani nezmení výsledky Fourierovej transformácie.

Typické oknové funkcie majú váhy na krajoch intervalu menšie a tým k sebe približujú koncové vzorky, čím zmenšujú problém popisovaný v predchádzajúcej kapitole. V článku [21] môžeme nájsť prehľad najčastejšie používaných oknových funkcií spolu s druhmi signálov, na ktoré sa najviac hodia. Na obrázku 5.3, môžeme pozorovať jednak vstupný signál po aplikovaní oknovej funkcie a tiež výsledok, akým sa prejaví v spektre. Efekt rozmazania sa prejavil oveľa menej a vzdialenejšie referenčné frekvencie majú nižšiu úroveň.



Obr 5.3: Výsledok FT po aplikovaní oknovej funkcie, prevzaté z [21]

5.3 Pitchshifter

Pitchshifter je anglické označenie pre efekt, ktorý frekvenčne posúva vstupný signál, bez toho aby menil jeho dĺžku [3]. Tento typ efektu je implementovaný digitálne, preto prvé zariadenia začali vznikať až v priebehu osemdesiatych rokov minulého storočia.

Pri realizovaní efektu sme postupovali podľa metódy, ktorú popisuje S. M. Bernsee v tutoriáli [2]. Využívame pritom spektrálnu reprezentáciu signálu, ktorá je výsledkom DFT.

Základná schéma efektu je veľmi jednoduchá. Pomocou DFT zistíme frekvenčné zložky signálu. Tieto upravíme príslušným posunutím a pomocou inverznej transformácie prevedieme signál späť do časovej domény. Samozrejme konkrétny postup je o niečo komplikovanejší.

Nemôžeme ako vstup použiť celú dĺžku vstupného signálu. Jednotlivé zmeny v spektre signálu by boli spriemerované a ťažko pozorovateľné [2]. Ďalším dôvodom je tiež snaha implementovať efekt použiteľný v reálnom čase, ktorý musí čo najrýchlejšie reagovať na zmeny v nastavení. Preto budeme vstupný signál spracovávať po menších častiach, tzv. rámcoch (angl. frames), počas ktorých budeme predpokladať, že signál je frekvenčne konštantný. Využívame tak koncept krátkodobej Fourierovej transformácie (Short Time Fourier Transform, STFT).

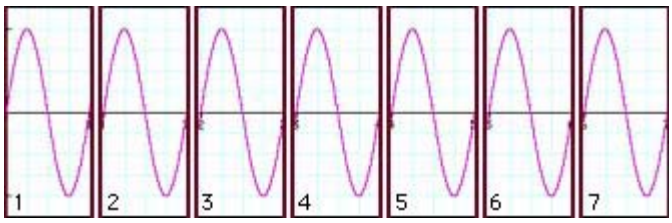
Ako sme uviedli v kapitole 5.2 na vylepšenie výsledkov DFT aplikujeme na spracúvaný rámec signálu oknovú funkciu. Rozhodli sme sa použiť Hanningovo okno, ktorého váhy sú definované ako:

$$w_i = 0.5 - 0.5 \cdot \cos\left(\frac{2\pi i}{N}\right),\ i = 0,...,N$$

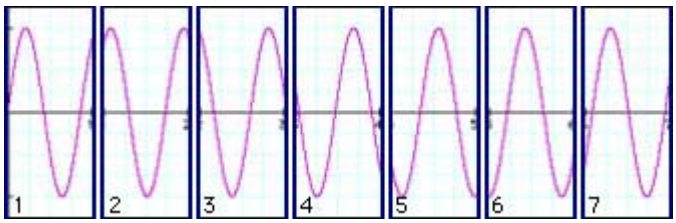
Túto oknovú funkciu používa aj S. M. Bernsee v článku [2] a podľa prehľadu v článku [21] je to najlepšia voľba v prípadoch, keď o vstupnom signále nemáme žiadne informácie, resp. aplikácia je určená pre všeobecné použitie.

S. M. Bernsee v článku [2] predstavuje postup ako kosínusovú a sínusovú zložku každej referenčnej frekvencie môžeme vyjadriť pomocou jednej sínusoidy, ktorá je definovaná príslušnou referenčnou frekvenciou, amplitúdou a fázou. Ďalej z fázy sínusoidy zistíme skutočnú frekvenciu zložky. Na obrázkoch 5.4 a 5.5 (prevzaté z článku [2]) vidíme dva príklady vstupného signálu. Prvý z nich má frekvenciu zodpovedajúcu niektorej referenčnej frekvencii a teda v dĺžke jedného rámca sa

nachádza celočíselný počet periód tohto signálu. Delenie na rámce spôsobí, že signál začína každý rámec v rovnakom bode svojej periódy. Inak povedané s rovnakou fázou. Na druhom obrázku vidíme signál, ktorý sa líši od všetkých referenčných frekvencií. V tomto prípade v každom z rámcov začína v inom bode svojej periódy a teda s iným fázovým posunom. Čím väčší bude tento posun, tým viac sa signál líši od najbližšej referenčnej frekvencie. Takto môžeme získať presnú frekvenciu danej zložky.



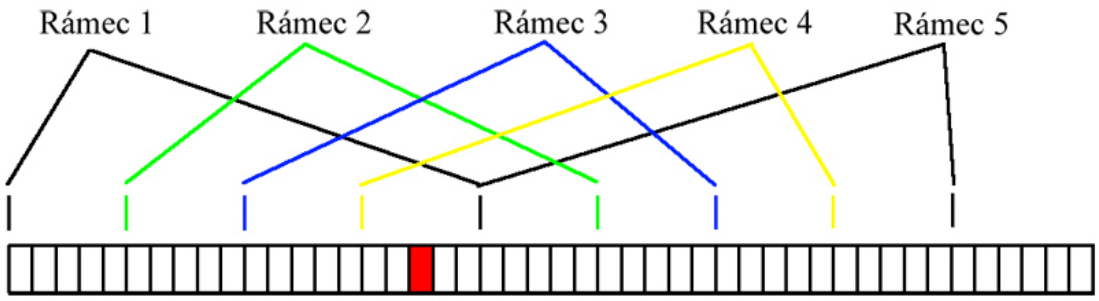
Obr 5.4: Rozdelenie signálu na rámce, prevzaté z [2]



Obr 5.5: Rozdelenie signálu na rámce - fázový posun medzi rámcami, prevzaté z [2]

Celý postup navyše využíva postupné prekrývanie rámcov signálu. To znamená, že úseky, ktoré spracúvame v jednotlivých krokoch, nesusedia, ale zdieľajú niektoré časti vstupného signálu. Dôvodom k tomuto kroku je presnejšie vyjadrenie skutočnej frekvencie zložiek signálu a tiež použitie oknovej funkcie [2]. To, ako je vzorka ovplyvnená oknovou funkciou, závisí na umiestnení vzorky v spracúvanom rámci. Presnejšie výsledky dosiahneme, keď každú vzorku spracujeme niekoľkokrát vo viacerých rámcoch. V každom bude mať totiž iné umiestnenie. Za konečný výsledok potom vezmeme priemer týchto čiastkových spracovaní. Detaily výpočtu presnej frekvencie a výsledky dosiahnuté pomocou prekrývania rámcov nájdeme v článku [2].

Znázornenie prekrývania rámcov vidíme na obrázku 5.6. Dĺžka rámca je 20 vzoriek a *overlap* parameter má hodnotu 4. To znamená, že pri spracovaní začiatok každého rámca je od toho predchádzajúceho vzdialený o $\frac{20}{4} = 5$ vzoriek. Zároveň vidíme, že každá vzorka bude spracovaná v štyroch rôznych rámcoch. Viac prekrývajúce rámce nám umožňujú dosiahnuť kvalitnejšie výsledky, ale tiež predstavujú vyššie nároky na výpočtový výkon. Parameter *overlap*, riadiaci prekrývanie, sme ponechali ako užívateľsky nastaviteľný.



Obr 5.6: Prekrývanie sa spracovávaných rámcov

Samotné posunutie výšky tónu je veľmi jednoduché. Zložky vstupného signálu máme teraz vyjadrené sínusoidami, ktoré sú definované ich skutočnou frekvenciou a amplitúdou. Každú zložku teda frekvenčne posunieme pomocou užívateľsky ovládaného parametra *Pitchshift*. Veľkosti zložiek, ktorým po posunutí pripadne rovnaká referenčná frekvencia, sčítame a poznačíme si skutočnú frekvenciu.

Parameter *Pitchshift* nadobúda hodnoty z intervalu $[-0.5,2]$. Jeho hodnota predstavuje koeficient, ktorým sa vynásobia frekvenčné zložky vstupného signálu. Ak má vstupný signál frekvenciu

$$f_i,$$

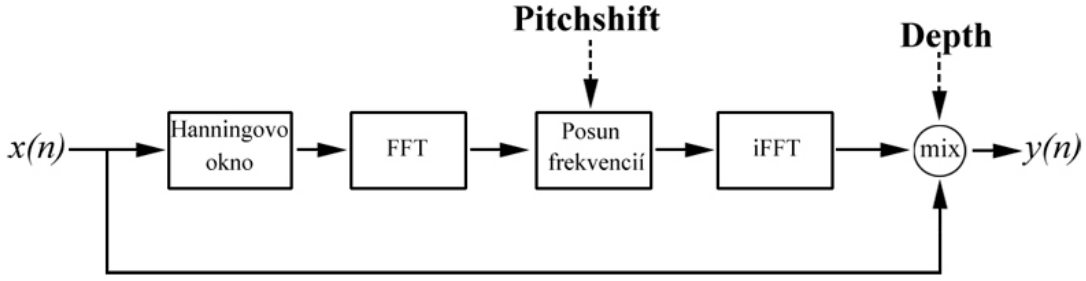
frekvencia výstupného signálu bude

$$f_o = f_i \cdot Pitchshift\ .$$

Interval $[-0.5,2]$ zodpovedá posunom v rozsahu oktávy nadol až oktávy nahor. Nakoľko vzťah medzi frekvenciami f_l, f_u, ktoré ohraničujú jednu oktávu je

$$f_u = 2 \cdot f_l\ .$$

Následné prevedenie signálu do časovej domény je zopakovanie krokov v opačnom poradí a prevedenie inverznej DFT. Výstupný signál je nakoniec získaný ako priemer zo všetkých spracovaní príslušných vstupných vzoriek. Výslednú štruktúre efektu zachytáva obrázok 5.7.



Obr 5.7: Štruktúra efektu pitchshifter

Kapitola 6

Implementácia

6.1 Popis aplikácie

Našim cieľom je vytvoriť aplikáciu, ktorá umožňuje prácu so súbormi formátu Wave. Dokáže tieto súbory prehrávať a aplikovať na ne zvolené zvukové efekty. Pre jednoduchosť sa obmedzíme na nekomprimovanú variantu formátu Wave, ktorá zvukový signál reprezentuje priamo hodnotami vzoriek v čase. Poznamenajme, že podpora iných zvukových formátov, by bola jednoducho dosiahnuteľná pridaním mechanizmu, ktorý by prevádzal zvolený súbor do vnútornej reprezentácie signálu. Túto reprezentáciu popisujeme v kapitole 6.3.

V súčasnom stave je implementovaná plná podpora pre jednokanálový a dvojkanálový vstup (mono a stereo). Rozšírenie na podporu viackanálové vstupu je priamočiare a jednoducho realizovateľné. Rozhodli sme sa však obmedziť iba na spomenuté počty kanálov, pretože viackanálové Wave súbory sú používané iba zriedkavo. Navyše väčšina efektov spracováva iba jednokanálový vstupný signál, preto na oboznámenie sa s efektmi a ich testovanie nám postačia mono a stereo súbory.

Implemntácia aplikácie je realizovaná v jazyku C#.

6.2 Reprezentácia signálu

Aplikácia pracuje so signálmi uloženými v súboroch formátu Wave. Musíme sa vyrovnať s faktom, že vzorky v rôznych súboroch môžu byť uložené s rôznymi bitovými hĺbkami a vzorkovacími frekvenciami.

Bitová hĺbka udáva koľko bitov pripadá na reprezentáciu hodnoty jednej vzorky. Každá vzorka je v súbore uložená ako celé číslo. Rozsah hodnôt ktoré môže nadobúdať je daný použitou bitovou hĺbkou. Najčastejšie hodnoty sú 8,16 a 24 bitov. Problémom je tiež fakt, že kým pri hĺbke 8 bitov vzorka nadobúda iba kladné hodnoty z intervalu $[0,255]$, pri ostatných nadobúda rovnako záporné aj kladné hodnoty. Napríklad pri hĺbke 16 bitov je interval možných hodnôt $[-32768,32767]$ [20].

Tento fakt spôsobuje pri realizovaní efektov značné problémy. Napríklad zistenie, či hodnota vzorky presahuje určitú hraničnú hodnotu, by sa muselo prispôsobovať bitovej hĺbke signálu. Problémom je tiež nesymetrické rozloženie kladných a záporných hodnôt pri 8-bitovej hĺbke. Preto sme sa rozhodli použiť jednotnú reprezentáciu hodnôt vzoriek, ktoré nadobúdajú hodnoty z intervalu $[-1,1]$. Pri načítaní zo súboru sú teda hodnoty vzoriek mapované do tohto intervalu a pri

zápise do súboru naopak prevádzané naspäť. Vďaka tomu môžu efekty pracovať s jednotnou reprezentáciou vzoriek vstupného signálu.

Vzorkovacia frekvencia nám hovorí, koľko vzoriek reprezentuje 1 sekundu uloženého záznamu. Nemôžeme ju však zjednotiť podobným spôsobom ako bitovú hĺbku. Podľa Nyquistovho teorému môže byť jednoznačne rekonštruovaný signál, ktorý bol vzorkovaný frekvenciou, ktorá je aspoň dvojnásobnou oproti najvyššej frekvencii prítomnej v signále [14]. To znamená, že keby sme chceli zaviesť jednotnú vzorkovaciu frekvenciu v celej aplikácii, musela by byť aspoň taká vysoká ako najvyššia vzorkovacia frekvencia, ktorú bude aplikácia spracúvať. V prípade, že by sme zvolili nižšiu vzorkovaciu frekvenciu, môžeme prevodom stratiť časť užitočného signálu. Ostáva nám teda zvoliť dostatočne vysokú vzorkovaciu frekvenciu, ktorá by postačovala všetkým potenciálnym signálom. Táto frekvencia však bude pre väčšinu vstupných signálov zbytočne vysoká a bude potrebné vykonanie priveľa zbytočných operácií.

Ako logická pripadá námietka, že ak bude aplikácia spracovávať zvukové záznamy, stačí predsa taká vysoká frekvencia, aby bolo znovu zrekonštruovateľné počuteľné pásmo frekvencií, t. j. interval 20 Hz až 20 kHz. Za univerzálnu vzorkovaciu frekvenciu by teda stačilo zvoliť napr. 44100 Hz. To je naozaj pravdivé tvrdenie, ale tým by sme aplikáciu obmedzili na spracovávanie zvukových signálov. Aplikácia rozšírená o nové efekty však môže slúžiť aj pre spracovanie iných ako zvukových dát. Druhým dôvodom je fakt, že vyššia vzorkovacia frekvencia môže zlepšiť výsledok pri aplikovania efektov. (Napr. pri vyššej vzorkovacej frekvencii môže postačovať lineárna interpolácia na dostatočné presnú aproximáciu signálu medzi hodnotami vzoriek.)

Vzorkovaciu frekvenciu teda necháme závislú na vstupnom súbore a budeme jej hodnotu predávať jednotlivým efektom. Pre väčšinu efektov je znalosť hodnoty vzorkovacej frekvencie podmienkou fungovania.

6.3 Štruktúra aplikácie

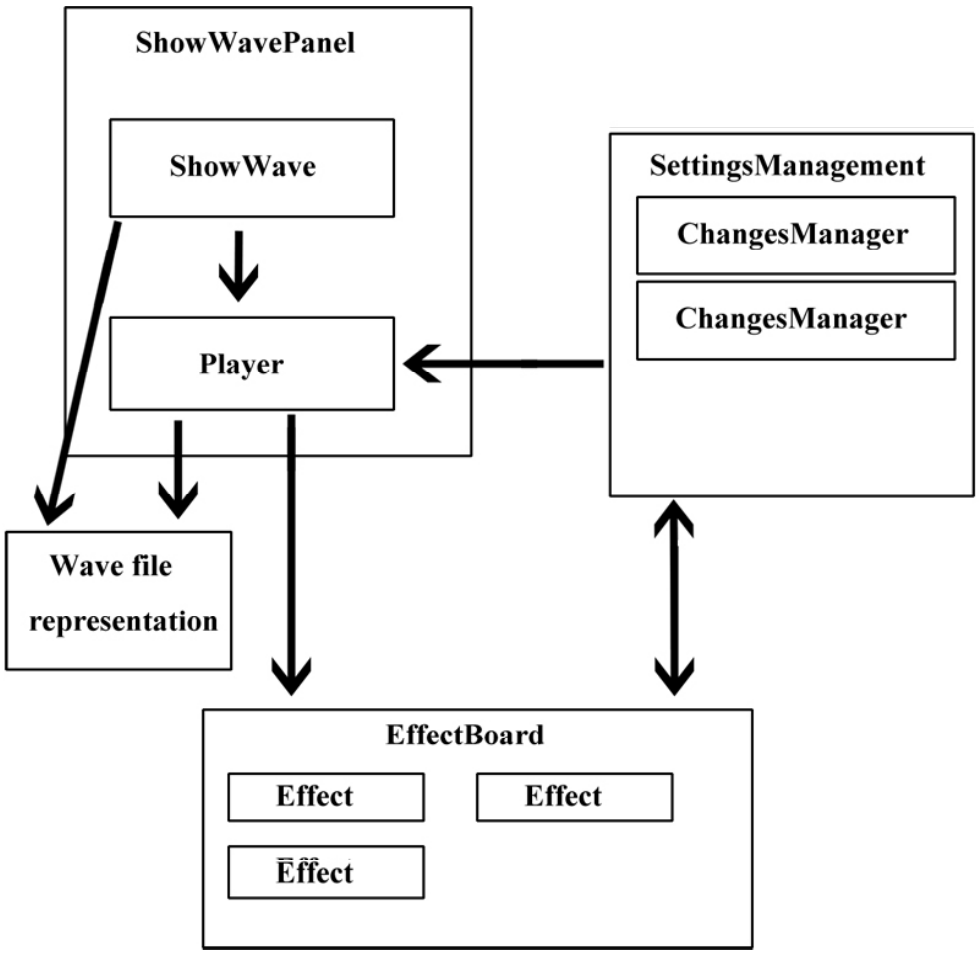
Obrázok 6.1 popisuje štruktúru celej aplikácie. Šípky znázorňujú smer volania metód. Otvorený súbor je prehrávaný komponentou Player. Táto komponenta počas prehrávania aplikuje efekty, ktoré sú združované komponentou EffectBoard. Táto komponenta organizuje efekty graficky (zobrazuje ich) aj logicky (spravuje poradie efektov).

Zobrazovanie prehrávaného obsahu zabezpečuje ShowWave. Táto komponenta sa udržuje synchronizovaná s prehrávačom pomocou mechanizmu, ktorí opisujeme v kapitole 6.2. Zároveň nastavuje aktuálnu pozíciu prehrávania, podľa užívateľským pokynov. Komponenty ShowWave a Player sú súčasťou komponenty ShowWavePanel, ktorá slúži na ich zobrazenie a tiež pridáva tlačidla, ktorými sa Player ovláda.

Komponenta SettingsManagement je zodpovedná za ukladanie zmien v nastavení efektov, ktoré vykoná užívateľ. SettingsManagement reaguje na pridanie efektu a vytvorí preň nový ChangesManager, ktorý bude spravovať daný efekt. Zároveň využíva Player, aby zistil čas, v ktorom zmena nastala. Na druhej strane, počas prehrávania záznamu, mení nastavenia efektu podľa zaznamenaných zmien.

Zosumarizovaním všetkých požiadaviek sme vytvorili rozhranie, ktoré musia spĺňať všetky efekty v aplikácii. Toto rozhranie obsahuje metódu, ktorá slúži na modifikovanie rámca signálu a niekoľko metód nastavujúcich technické parametre efektu (napr. vzorkovacia frekvencia signálu). Posledná skupina metód súvisí so

systémom zaznamenávania zmien v parametroch efektov, ktoré vykoná užívateľ. Záujemcom o podrobnejší prehľad o rozhraní odporúčame prehliadnuť zdrojový kód triedy *Effects.GUI.Effect*.



Obrázok 6.1: Štruktúra aplikácie

6.4 Prehrávanie záznamu

Program počas prehrávania súboru musí aplikovať na vstupný signál užívateľom zvolené efekty. Efekty majú vo väčšine prípadov viacero nastaviteľných parametrov. Tieto parametre slúžia na dosiahnutie rôzneho výsledku pri použití efektu. Užívateľ počas prehrávanie, bude samozrejme očakávať, že zmena v nastavení parametrov sa okamžite prejaví na výslednom zvuku.

Proces prehrávania sa tak do veľkej miery podobá postupu pri spracovaní signálu v reálnom čase. Rozdelíme vstupný signál na kratšie rámce, ktoré budeme spracovávať izolovane. Každý rámec predáme reťazcu efektov na spracovanie a výsledok prehráme užívateľovi. Takto sa zmeny v nastaveniach efektu, ktoré užívateľ vykoná, prejavia už pri spracovaní najbližšieho rámca.

Poznamenajme, že rozhranie komponenty Player (viď. obrázok 6.1) definuje interface Effects.GUI.AudioPlayer. Konkrétnu implementáciu, ktorá aplikácia používa, predstavuje trieda *Effects.GUI.EffectStreamPlayer*. Táto trieda na prehrávanie využíva knižnicu *DirectSound*. Postupujeme pritom podľa metódy, ktorá predstavil I. Munoz v článku [10]. Ku knižnici *DirectSound* pristupujeme prostredníctvom projektu Managed DirectX, ktorý sprístupňuje volania *DirectX* pre

tzv. managed kód (čiže štandardný zdrojový kód určený pre platformu .NET) Ako sa ukázalo počas testovania celej aplikácie spoločnosť Microsoft ukončila podporu tohto projektu skôr ako bola pripravená podpora 64-bitových operačných systémov. Z tohto dôvodu je aplikácia spustiteľná iba na 32-bitovom systéme.

Realizovanie iného mechanizmu prehrávania, by bolo možné uskutočniť novou implemntáciu rozhrania *AudioPlayer*, ktorá by mohla využívať odlišnú knižnicu.

6.5 Používanie timerov

Na viacerých miestach v aplikácii potrebujeme udržiavať nejakú komponentu synchronizovanú s inou, ktorá vykonáva svoju činnosť. Najčastejšie potrebujeme, takpovediac, „udržiavať krok“ s prehrávačom záznamu. (Napr. vykresľovať kurzor na príslušnú pozíciu.)

Na tento účel používame štandardnú triedu *Timer* z namespacu *System.Timers*. Komponenta, ktorá sa potrebuje synchronizovať s prehrávačom obsahuje inštanciu triedy *Timer*. Táto inštancia v nastavenom časovom intervale spustí zvolenú metódu, ktorá vykoná potrebné operácia. Telo tejto metódy sa pritom vykonáva v separátnom vlákne. Ako príklad uveďme vykreslovaciu komponentu *ShowWave*, ktorá ma časový interval synchronizácie nastavený na 100 ms. Pri každom zavolaní zistí aktuálnu pozíciu prehrávača v súbore a upraví podľa toho polohu kurzora.

Používanie timerov však prinieslo jeden neočakávaný problém. Popísaná synchronizácia často vyžaduje grafickú úpravu dotyčnej komponenty a teda jej následné prekreslenie. .Net framework nepovoľuje zavolanie metódy *Refresh* na triede z vlákna, ktoré nevytvorilo túto triedu. Pre riešenie tohto problému je potrebné vytvoriť novú metódu, ktorá využije metódu *BeginInvoke,* pomocou ktorej môže vyvolať metódu *Refresh* vo vlákne, ktoré triedu vytvorilo.

6.6 Pluginový systém a balík efektov Effectspack

Ako sme už uviedli v kapitole 1.4 existencia jednotného rozhrania pre efekty umožňuje jednoduché rozširovanie aplikácie o ďalšie efekty. Je potrebné vytvoriť novú triedu, ktorá implementuje triedu *Effect* a preloženú assembly s touto triedou umiestniť do adresára Effects v hlavnom adresári aplikácie. Program pri štarte načíta všetky assemblies z tohto adresára a triedy v nich, ktoré sú potomkami triedy *Effect*, pridá do systému. Pri načítaní tried sa používa mechanizmus Reflection.

Na testovanie funkčnosti systému bolo implementovaných niekoľko efektov, ktoré sa nachádzajú v rozširujúcom balíku Effectspack. Pre predstavu o tom, ako vyzerá implementácia takéhoto efektu, odporúčame prezrieť zdrojový kód efektu Volume z balíka Effectspack. Kód obsahuje podrobné komentáre, ktoré vysvetľujú základné princípy implementácie efektu pre našu aplikáciu.

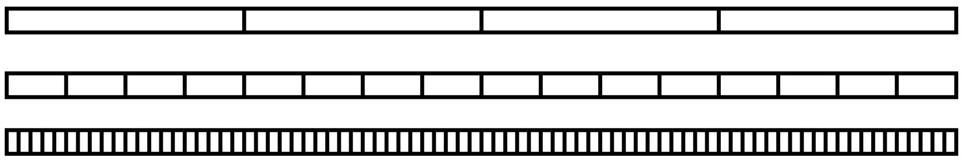
6.7 Vykresľovanie signálu

O vykresľovanie krivky signálu sa stará trieda *Effects.GUI.ShowWave*. Znázornenie signálu je v podstate vykreslenie grafu funkcie, ktorej hodnotami sú práve hodnoty vzoriek. Pri priamom vykreslení však narazíme na problém, že na každú vzorku pripadá veľmi malá časť šírky komponenty (oveľa menej ako 1 pixel). Použijeme teda metódu, ktorá je popísaná v práci [4]. Pre každý pixel šírky komponenty spočítame minimálnu a maximálnu hodnotu so vzoriek, ktoré by na

tento pixel pripadli. Body zodpovedajúce týmto hodnotám pri vykresľovaní spojíme úsečkou.

Uvedený postup, je však pomalý pre súbory normálnej dĺžky (rádovo minúty). Pri každom prekreslení totiž musí znovu spočítať potrebné minimálne a maximálne hodnoty vzoriek pre každý pixel. Na zrýchlenie tohto výpočtu rozdelíme vzorky na intervaly. Z každého intervalu zistíme hodnotu maximálnej a minimálnej vzorky. Pri samotnom hľadaní minima a maxima pre konkrétny pixel budeme prechádzať tieto predpripravené hodnoty. Na obrázku 6.2 vidíme znázornený príklad, keď každý pomocný interval (stredný pás) zastupuje 5 vzoriek (spodný pás). Vidíme, že na zistenie požadovaných hodnôt pre jeden pixel, nemusíme prejsť hodnoty 20 vzoriek, ale iba 3 predpripravené hodnoty minima a maxima.

V prípade, že užívateľ použije funkciu zoom a priblíži sa tak blízko, že na jeden pixel by pripadlo menej vzoriek, ako ich obsahujú predpripravené intervaly, komponenta automaticky začne zisťovať minimálnu a maximálnu hodnotu priamo zo vzoriek. V takom prípade totiž zobrazuje dostatočne malú časť súboru, čiže nie je potrebné prechádzať veľký počet relevantných vzoriek.



Obrázok 6.2: Znázornie vzťahu medzi vzorkami, pomocnými intervalmi a pixelmi na obrazovke

Kapitola 7

Záver

Podarilo sa nám realizovať aplikáciu, ktorá prehráva zvukové súbory a v reálnom čase aplikuje na vstupný signál efekty. Na dosiahnutie sme museli navrhnúť rozhranie pre komunikáciu s efektmi. Toto rozhranie sa ukázalo ako dostatočne flexibilné pri rozširovaní aplikácie o systém zaznamenávania zmien v nastaveniach efektov.

Osobne za najväčšie pozitívum považujem nazbierané vedomosti o fungovaní efektov, ktoré už dlhodobo patria medzi moje veľké záľuby vo voľnom čase a tiež získané skúsenosti s prácou na rozsiahlejšom projekte.

Hoci potenciálnych možností pre budúci vývoj aplikácie je hneď niekoľko, smer vývoja aplikácie ešte nie je určený a vysoko pravdepodobne sa súčasný stav aplikácie ako takej už nebude nijako výrazne meniť.

Jednou z možností je implementovať niektoré efekty, aby boli kompatibilné so formátom VST. VST je v súčasnosti najrozšírenejší formát digitálnych efektov. Výsledné efekty by tak boli kompatibilné s väčšinou nahrávacieho softwaru.

Ďalšou možnosťou je pokúsiť sa o realizáciu niektorých efektov na nižšej úrovni pomocou integrovaného obvodu FV-1 od spoločnosti Spin Semiinductor. Táto voľba zahŕňa aj výrobu potrebného elektronického obvodu na ktorom bude FV-1 osadený. Z programátorského hľadiska by išlo o malú výzvu, keďže programovanie obvodu FV-1 prebieha priamo v assembleri a na každú vzorku vstupného signálu je obmedzení počet inštrukcií.

Poslednou možnosťou je rozšíriť aplikáciu tak, aby umožňovala exportovať uložené nastavenia do súboru formátu MIDI, prípadne takéto súbory aj načítať. MIDI je najpoužívanejší formát komunikácie medzi hudobnými zariadeniami. Aplikácia by sa tak stala o niečo hodnotnejšou a použiteľnejšou v praxi.

Literatúra

[1] Bernsee S. M.: The DFT "a Pied": Mastering The Fourier Transform in One Day [online], [cit. 2010-05-07]. Dostupné na internte: http://www.dspdimension.com/admin/dft-a-pied/

[2] Bernsee S. M.: Pitch Shifting Using The Fourier Transform [online], [cit. 2010-05-07]. Dostupné na internte: http://www.dspdimension.com/admin/pitch-shifting-using-the-ft/

[3] Bernsee S. M.: Time Stretching And Pitch Shifting of Audio Signals – An Overview, [cit. 2010-05-07] Dostupné na internete: http://www.dspdimension.com/admin/time-pitch-overview/

[4] Cihelková T.: Hudební editor s anlýzou zvukového vstupu, 2008, bakalárska práca. Praha: MFF UK

[5] Keen R. G.: Human Voices and the Wah Pedal [online], [cit. 2010-04-26]. Dostupné na internete: http://www.geofex.com/article_folders/wahpedl/voicewah.htm

[6] Lehman S.: Flanging [online]: [cit. 2010-04-30]. Dostupné na internete: http://www.harmonyclicks.com/Effects/Articles/Flanging/

[7] Lehman S.: Chorus [online]: [cit. 2010-04-30]. Dostupné na internete: http://www.harmonyclicks.com/Effects/Articles/Chorus/

[8] Lehman S.: Equalization [online], [cit. 2010-04-26]. Dostupné na internete: http://www.harmonyclicks.com/Effects/Articles/Chorus/

[9] Mitra S. K.: Digital Signal Processing: A Computer-Based Approach, McGraw-Hill, 2001

[10] Munoz I.: Building a Drum Machine with DirectSound [online], [cit. 2010-05-12]. Dostupné na internete: http://msdn.microsoft.com/en-us/library/ms973091.aspx

[11] Punskaya E.: Materiály k prednáške Digital Signal Processing, Department of Engineering, University of Cambridge, [cit. 2010-04-26]. Dostupné na: http://www-sigproc.eng.cam.ac.uk/%7Eop205/

[12] Smith S. W.: The Scientist and Engineer's Guide to Digital Signal Processing, online verzia, [cit. 2010-04-29]. Dostupné na internete: http://www.dspguide.com/

[13] Smith III J. O.: Introduction to Digital Filters with Audio Aplications, online verzia, [cit. 2010-04-29]. Dostupné na internete: http://www.dsprelated.com/dspbooks/filters/

[14] Smith III J. O.: Mathematics of the Discrete Fourier Transform, online verzia, [cit. 2010-05-08]. Dostupné na internete: http://www.dsprelated.com/dspbooks/mdft/

[15] Smith III J. O.: Physical Audio Signal Processing for Virtual Musical Instruments and Audio Effects, online verzia, [cit. 2010-04-25]. Dostupné na internete: http://www.dsprelated.com/dspbooks/pasp

[16] Zölzer U.: Digital Audio Signal Processing, John Wiley & Sons Ltd, 2008

[17] Butterworth Filter [online], [cit. 2010-04-25]. Dostupné na internete: http://en.wikipedia.org/wiki/Butterworth_filter

[18] Digital Filter: An Introduction [online] [cit. 2010-05-01]. Dostupné na internte: http://www.dsptutor.freeuk.com/dfilt1.htm

[19] Electro-Harmonix Pulsar – DIY projekt [online], [cit. 2010-04-20]. Dostupné na internete: http://tonepad.com/getFile.asp?id=29

[20] Wave – popis formátu [online], [cit. 2010-05-13]. Dostupné na internete: http://www.lightlink.com/tjweber/StripWav/WAVE.html

[21] Windowing: Optimizing FFTs Using Window Functions [online], [cit. 2010-05-07]. Dostupné na internete: http://zone.ni.com/devzone/cda/tut/p/id/4844

Dodatok A

Obsah disku CD-ROM

V nasledujúcich adresároch na priloženom CD možno nájsť

- Bin – preložený program, vyžadujúci platformu .NET a knižnicu DirectX
- Doc – stručný užívateľský manuál a automaticky generovaná programátorská dokumentácia
- Samples – niekoľko zvukových súborov WAVE a pripravených ukážok nastavení efektov
- Source – Visual Studio projekt obsahujúci zdrojový kód aplikácie
- Thesis – text práce vo formáte PDF