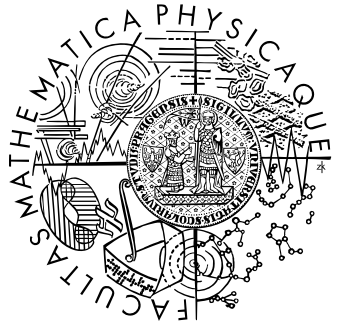
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Branislav Fábry

Kytarové efekty s automatickým nastavováním parametrů

Katedra softvérového inženýrství

Vedoucí bakalářské práce: RNDr. Tomáš Poch

Studijní program: Informatika, obecná informatika

2010

Ďakujem pánovi RNDr. Tomášovi Pochovi za poskytnuté vedenie, pripomienky a čas pri vypracovávaní tejto práce.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce a jej zverejňovaním.

V Prahe dňa 17.5.2010 Branislav Fábry

Obsah

Názov práce : Kytarové efekty s automatickým nastavováním parametrů

Autor : Branislav Fábry

Katedra : Katedra softvérového inženýrství

Vedúci práce : RNDr. Tomáš Poch

E-mail vedúceho : poch@d3s.mff.cuni.cz

Abstrakt : Cieľom tejto práce implementovať sadu zvukových efektov použiteľných pri gitarovom hraní a aplikáciu ReSounder, ktorá bude na základe analýzy pôvodného a požadovaného zvukového signálu schopná nastaviť týmto efektom parametre tak, aby sa dosiahol požadovaný zvuk. Úlohou je navrhnúť vhodné algoritmy a metriky na porovnávanie signálov. Efekty sú implementované na základe voľne použiteľnej platformy VST od firmy Steinberg, ktorá je najrozšírenejšou platformou na PC. Úvod práce je venovaný bližšiemu oboznámeniu čitateľa s problematikou digitálneho audia, hlavnú časť práce tvorí popis návrhu a implementácie programu ReSounder. V závere je uvedené zhodnotenie praktických výsledok.

Title : Guitar Effects With Automatic Parameters Setting

Author : Branislav Fábry

Department : Department of Software Engineering

Supervisor : RNDr. Tomáš Poch

E-mail Adress of Supervisor : poch@d3s.mff.cuni.cz

Abstract : The subject of this work is to implement a set od sound effects applicable during guitar playing and application entitled ReSounder, which will be able, on the groud of analysis of original and wanted sound signal, to set the parameters of these effects in order to achieve wanted sound. The task is to develop appropriate algorithms and metrics for signal comparing. The effects are implemented with using VST software development kit by Steinberg, which is the most expanded platform for PC. The beginning of this work is dedicated to closely acquaintance the reader with problematique of digital audio, the main part contains a describtion of implementation of ReSounder program. The practical results are mention in the end.

1 Úvod

Rapídny nástup digitalizácie počas posledných rokov sa nevyhol ani oblasti gitarového hrania. Aj napriek tomu, že gitaristi majú všeobencne konzervatívne názory držiace sa toho, že analógovému zvuku sa nič nevyrovná, vzniká mnoho digitálnych systémov a produktov určených pre vytváranie gitarového zvuku. Už od počiatkov elektrickej gitary datujúcich sa do 50.-tych rokov sa gitaristi snažia obohacovať prirodzený zvuk gitary a rozširovať jej zvukové možnosti.

Súčasný štandard gitarového zvuku je všetkým dobre známy kvílivý tón v sólach alebo drsný zvuk rockových doprovodov, spôsobený skreslením prirodzeného gitarového zvuku. Okrem tohto základného efektu sa stalo obľúbené množstvo iných efektov, hlavne z kategórií modulačných a oneskorovacích. Dnes má gitarista na výber z obrovského množstva zariadení upravujúcich gitarový zvuk. Od klasických analógových zariadení v podobe gitarových "krabičiek" zahrňujúcich prevažne drahé výrobky až po voľne dostupné digitálne simulácie, ktoré si môže pustiť na domácom PC. Každý efekt alebo zariadenie má väčšinou niekoľko parametrov, ktorými je možné obmeniť zvuk efektu. V analógovom prípade počet týchto parametrov zvyčajne nepresiahne päť, no v digitálnom svete, kde neexistujú žiadne fyzické prekážky, to môže byť niekoľkonásobne viac. Pre neskúseného používateľa je nastavovanie týchto parametrov v snahe "podľa sluchu" dosiahnuť konkrétny zvuk veľmi zložité.

Cieľom bakalárskej práce je uľahčiť riešenie práve tohoto problému. Pomocou sady efektov, vzorky signálu pred a vzorky signálu po spracovaní efektom nájsť čo najbližšie možné nastavenie, s ktorým sa dá docieliť efektová replikácia. Riešenie zahŕňa implementáciu hľadania hodnôt parametrov v programe ReSounder a implementáciu sady základných efektov ktoré budú pri replikácií používané.

V teoretickej časti, ktorú tvorí kapitola 2, sú rozobrané potrebné základné znalosti a možnosti súčasných zvukových technológii. Je popísaný spôsob vytvárania digitálneho signálu z analógového s možnosti jeho úprav teoreticky aj prakticky. V kapitole 3 je popísaný návrh a implementácia programu ReSounder, jeho štruktúra a použité algoritmy. Kapitola 4 je venovaná zhodnoteniu dosiahnutých výsledkov. V prílohe sa nachádza obsah priloženého CD a používateľská dokumentácia programu ReSounder.

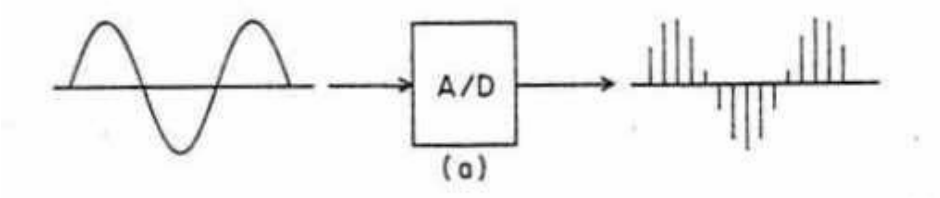
2 Digitálne spracovanie zvuku

2.1 Digitálne audio

Zvuk je pozdĺže mechanické vlnenie v látkovom prostredí, ktoré je schopné vyvolávať sluchový vnem. Z matematického hľadiska to je spojitá funkcia v čase. Digitálne zariadenia však nie sú schopné pracovať so spojitými signálmi a je nutné pred spracovaním zvukového signálu previesť túto spojitú funkciu do diskrétnej reprezentácie (AD – analog to digital prevod) a zase naopak, po spracovaní pre dosiahnutie počuteľného výstup, z diskrétnej do analógovej (DA – digital to analog prevod).

2.1.1 AD prevod

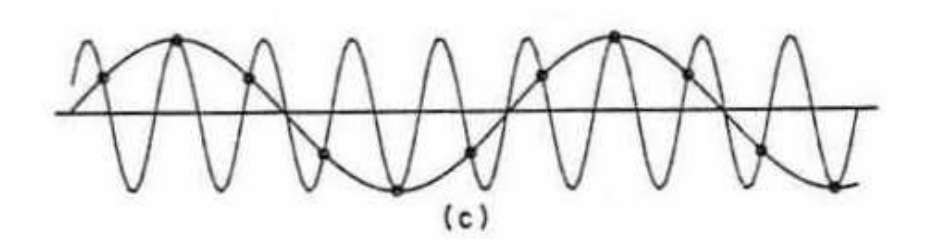
Základným pojmom spojeným s prevodom zvuku zo spojitej do diskrétnej reprezentácie je vzorkovanie (sampling), čo je samotná redukcia spojitého signálu na signál diskrétny. S vopred určenou frekvenciou v čase (vzorkovacia frekvencia) zariadenie na AD prevod (sampler) zaznamenáva hodnoty spojitej funkcie. Tieto hodnoty za nazývajú vzorky (sample). Teoretický ideálny sampler vytvára sample ekvivalentné hodnotám spojitej funkcie pre ľubovoľný časový bod.



Obrázok 1: Prevod spojitého signálu na diskrétny pomocou AD prevodníku. Zdroj: [6]

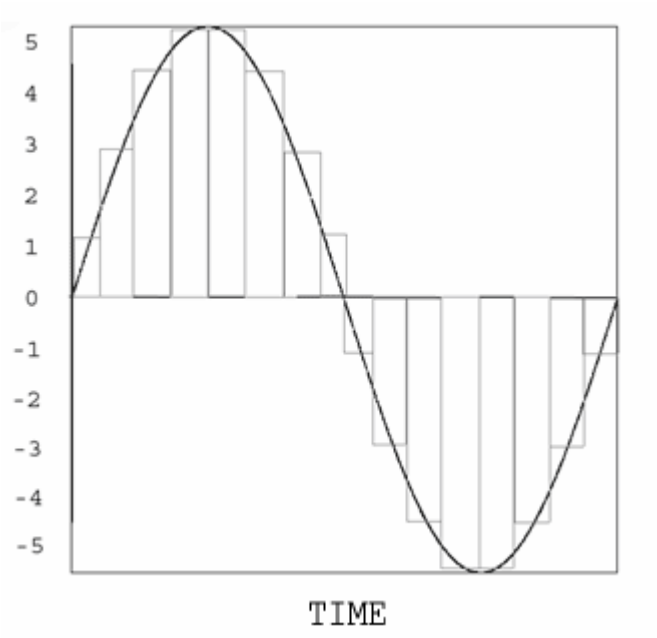
V praxi sa používa Nyquist-Shannonova veta, ktorá hovorí, že pre zachytenie všetkých frekvencií je potrebné použiť vzorkovaciu frekvenciu rovnú minimálne 2-násobku najvyššej frekvencie v signále. Na obrázku 2 je znázornené vzorkovanie za použitia príliš nízkej vzorkovacej frekvencie. Najvyššie frekvencia počuteľná ľuďmi

je 20 kHz, takže podľa tejto vety je nutné použiť vzorkovaciu frekvenciu minimálne 44 kHz. V praxi sa najčastejšie používa frekvencia 44,1 kHz (Audio CD), no pri náročnejších aplikáciách, napr. v nahrávacích štúdiách, sa používajú aj frekvencie 48 kHz, 96 kHz alebo dokonca 192 kHz.



Obrázok 2: Vzorkovanie za použitia príliš nízkej vzorkovacej frekvencie. Signál s vyššou frekvenciou je pôvodný. Signál s nižšou frekvenciou je výsledný signál po vzorkovaní. Zdroj: [6]

Rozsah hodnôt, ktoré môžu sample nadobudnúť, je určený bitovou hĺbkou. Pri 8-bitovej hĺbke tak môže jedna vzorka nadobúdať hodnoty 0 – 255. Pri vzorkovaní je potrebné aktuálne hodnoty v spojitom priestore kvantizovať na najbližšiu hodnotu, ktorá je zaznamenateľná pomocou daného počtu bitov (obrázok 3). Tu vzniká kvantizačná odchýlka, ktorá sa prejavuje kvantizačným šumom. Je zrejmé, že čím vyššia bitová hĺbka, tým väčší počet možných hodnôt jednej vzorky a tým pádom menší kvantizačný šum. Bitová hĺbka zvukového CD je 16 bitov, v nahrávacích štúdiách sa používa aj 24- alebo 32-bitová hĺbka.



Obrázok 3: Kvantizácia spojitého signálu. Zdroj: [4]

Pri oboch týchto faktoroch – vzorkovacia frekvencia a bitová hĺbka - platí, že čím vyššia hodnota, tým vyššia kvalita prevodu, no aj vyšší objem výsledných dát.

2.1.2 Formát WAVE

Skratka WAVE alebo WAV znamená Waveform Audio File Format. Je to formát vyvinutý firmami IBM a Microsoft slúžiaci na ukladanie zvukových dát na platforme PC. Napriek tomu, že WAVE formát dokáže uchovávať aj komprimované dáta, používa sa takmer výhradne na ukladanie neskomprimovaného zvuk. Je kompatibilný s platformami Windows, Macintosh a Linux.

Neskomprimovaný WAVE súbor je pomerne veľký, preto sa tento formát používa iba tam, kde nie je problém s objemnými dátami a je požadovaná najvyššia kvalita zvuku, napr. v nahrávacích štúdiách.

2.2 DSP

Skratka DSP znamené Digital Signal Processing (spracovanie digitálneho signálu). Je to súhrnný názov pre skúmanie a spracovanie akéhokoľvek digitálneho signálu. DSP zahŕňa napríklad spracovanie sonarových a radarových signálov, štatistických signálov, biomedicínskych signálov, spracovanie seizmických dát, spracovanie obrazu a samozrejme spracovanie zvukových signálov.

2.2.1 Filter

Základným komponentom DSP je filter. Funkciou filtru je odstrániť nechcené časti signálu, extrahovať užitočné časti signálu alebo vo všeobecnosti upraviť vstupný signál a vytvoriť signál výstupný.
Podľa spôsobu spracovania zvuku môžme filtre rozdeliť na dva typy - FIR (Finite Impulse Response – s konečnou odozvou) a IIR (Infinite Impulse Response – s nekonečnou odozvou). Filtre typu FIR pracujú tak, že práve spracovávaný vstupný signál nijako neovplyvňuje nasledujúce signály, tzn. vo vnútri filtru nenastáva niejaká spätná väzba. Napríklad pri vstupnom signále, ktorý začína s jednou vzorkou s hodnotou „1“ nasledovanou samými vzorkami s hodnotou „0“ bude výstupný signál vyzerať tak, že na začiatku bude vzorka so spracovanou hodnotou „1“ nasledovaná samými vzorkami s hodnotou „0“.

Filtre typu IIR pracujú s vnútornou spätnou väzbou. Výstupný signál z predchádzajúceho príkladu by v prípade IR bol zložený z (teoreticky) nekonečného radu vzoriek s nenulovými hodnotami.

Podľa fyzického hľadiska môžeme filtre rozdeliť na analógové a digitálne. Analógove filtre používajú elektronické obvody zložené z komponentov ako rezistory, kondenzátory a pod. Filtrovaný signál je elektrický prúd.

Digitálne filtre používajú digitálne procesory na prepočítavanie samplov vzorkovaného signálu. Analógový signál musí byť najpr prevedený do digitálnej formy AD prevodníkom, potom je prepočítaný procesorom a prevedený naspäť z digitálnej podoby do analógovej DA prevodníkom. Výhodou digitálnych filtrov

oproti analógovým je to, že môžu byť presne naprogramované na nejaký filtrovací úkon, sú precízne a stále za ľubovoľných vonkajších okolností. V hudobnom svete však ide o muzikalitu zvuku a v tomto ohľade vedú filtre analógové, pretože signál „nekúskujú“ samplovaním a paradoxne aj preto, že nedosahujú takej úrovne precíznosti ako digitálne filtre.

2.2.2 Zvukové efekty

Zvukovým efektom sa nazýva filter, ktorý slúži na úpravu zvukového signálu. Nasledujúca kapitola popisuje princípy a funkcie základných typov efektov.

2.2.2.1 Dynamické procesory

Ako názov napovedá, ide o filtre, ktoré majú za úlohy upravovať dynamiku signálu. Najbežnejšie parametre týchto filtrov sú :

attack – čas trvania nábehu dynamickej úpravy (rádovo jednotky až desiatky milisekúnd)

release – čas do uvoľnenia dynamickej úpravy

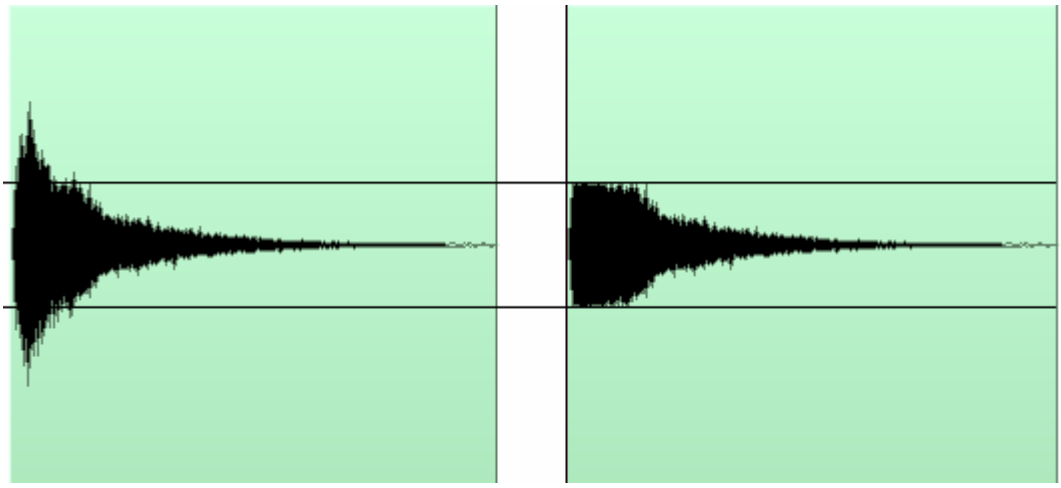
treshold – hranica, pri ktorej prekročení sa má spustiť dynamická úprava

ratio – kompresný pomer, pomer hlasitosti pôvodného a upraveného signálu

Najpoužívanejším typom dynamického procesoru je kompresor. Kompresor funguje tak, že po prekročení hranice určenej urovňou treshold sa signál po uplnynutí času určenom hodnotou attack stíši na úroveň určenú pomerom ratio. Po návrate signálu späť pod úroveň treshold sa nastaví na pôvodnú úroveň až po uplynutí času, ktorý je určený hodnotou release. Praktický jednoduchý príklad použitia predstavuje nahrávka úderu na bubon. Signál začína veľmi silnou intenzitou pri udretí, no pokračuje slabou intenzitou doznievania zvuku. Pri použití kompresoru s nastaveným ratio 1:4 sa počiatočná vysoká intenzita stíši na jednu štvrtinu, no zvuk doznievania ostane nestíšený. Vo výsledku môžeme celý zvuk zosilniť, pretože nám v tom už nebráni silný signál na začiatku.

Dalším používaným typom dynamického procesoru je limiter. Ide vlastne o kompresor s veľmi veľkým kompresným pomerom. Jednoducho povedané – limiter neprepustí nad uroveň treshold nič.

Opakom limiteru je gate. Funguje tak, že prepúšťa iba signál, ktorý je silnejší ako hodnota treshold. Ako príklad znovu uvediem nahrávku úderov na bubon – gate sa použije vtedy, keď chceme počuť iba samotné údery a žiaden ruch pomedzi údery.



Obrázok 4: Príklad signálu pred a po úprave limiterom s názornenou úrovňou treshold. V miestach, kde signál neprekročil túto úroveň zostal nedotknutý.

2.2.2.2 Modulačné efekty

Tieto typy efektov zvyčajne pracujú tak, že mierne oneskorujú vstupný signál a pomocou nizkofrekvenčného oscilátoru (LFO – low frequency oscilator) tento oneskorený signál modulujú a primiešavajú do čistého vstpného signálu. V niektorých efektoch sa LFO používa aj na moduláciu oneskorovacieho času. Frekvencie generované LFO sú príliš nízke, takže sú nepočuteľné.

Parametre modulačných efektov zvyčajne zahŕňajú rýchlosť (frekvenciu LFO), intenzitu (amplitúdu LFO) a pomer medzi čistým a zmodulovaným signálom na výstupe.

Najznámajšie modulačé efekty sú chorus, flanger a phaser. Chorus oneskoruje vstupný signál, tento oneskorovací čas nie je konštantný ale je modulovaný pomocou LFO a pohybuje sa niekde medzi 20 – 30 ms. Následne oneskorený signál primiešava do pôvodného vstupného signálu.

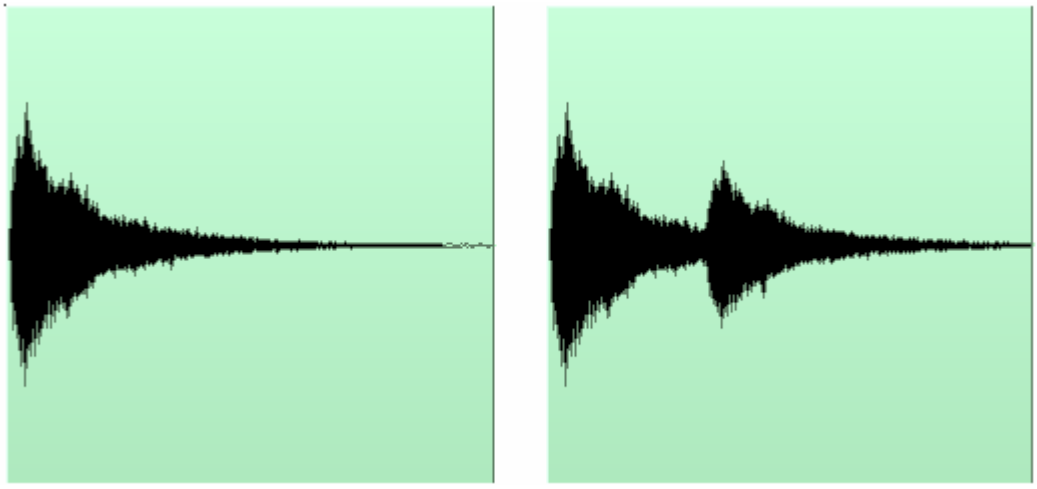
Flanger pracuje veľmi podobne ako chorus, no oneskorovací čas je menší – zvyčajne sa pohybuje medzi 1 – 10 ms. Naviac, výstupný signál môže byť poslaný aj naspäť na vstup a tým sa vytvorý spätná väzba. Úroveň tohto spätného signálu sa dá kontrolovať parametrom.

Predchádzajúce dva efekty sú zjednodušenými verziami phaseru. Na rozdiel od nich, phaser (tiež nazývaný phase shifter) neoneskoruje každú zložkovú frekvenciu vstupného signálu o rovnaký čas, ale každú frekvenciu o iný čas. Cieľom je vytvoriť fázový posun každej zložkovej frekvenciu vstupného signálu. Veľkosť tohto fázového posunu riadi LFO. Na výstupe sa pôvodný vstupný signál zmieša s fázovo posunutým sigálom.

2.2.2.3 Efekty pracujúce v časovej doméne

Dva hlavné efekty spadajúce do tejto skupiny sú delay a reverb. Delay znamené oneskorenie. Vo svojej najjednoduchšej podobe tento efekt oneskorí vstupný signál a zmieša ho s priamym signálom. Existuje množstvo rozšírení tohto efektu, napríklad, oneskorený signál sa neprimieša iba jedenkrát, ale primiešava sa viacej krát s časovými rozostupmi, tieto rozostupy môžu byť konštantné alebo sa môžu v meniť, môže sa meniť hlasitosť oneskorených signálov a podobne.

Reverb simuluje odrazy zvuku od prekážok. Zvyčajne sa používa na simuláciu odrazov v malej miestnosti (v tomto prípade nazývaný tiež room) alebo vo väčšej hale (nazývaný tiež hall).



Obrázok 5: Príklad signálu pred a po úprave efektom delay.

2.3 Možnosti spracovania zvuku v počítači

V minulosti boli pojmy nahrávanie a úprava zvuku spojené výhradne s nahrávacími štúdiami, ktoré mali k dispozícií drahé zariadenia. S rozmachom výpočtovej techniky sa však tieto možnosti dostali ku každému a v súčasnosti je z technickej stránky veľmi jednoduché vyprodukovať takmer profesionálne znejúcu nahrávku v domácich podmienkach.

2.3.1 Hardware

Základom pre kvalitný záznam zvuku je hardware. V súčasnosti už väčšina počítačov obsahuje aspoň integrovanú zvukovú kartu s jedným zvukovým výstupom a jedným vstupom. Kvalita týchto zariadení je však na veľmi nízkej úrovni a s mierne lepšou reprosústavou je šum evidentný. Neporovnateľne lepšia kvalita sa dá dosiahnuť so zvukovými kartami určenými výhradne na poloprofesionálny a profesionálny záznam zvuku. Možnosti výberu siahajú od lacných ultrakompaktných 1-kanálových rozhraní až po ohromné profesionálne skriňové systémy so stovkami vstupov a výstupov.

2.3.2 Software

Možnosti výberu softwaru sú tiež veľké. Najjednoduchšie systémy umožňujú aplikovať jeden efekt na vstupný zvuk a posielať ho real-time priamo na výstup. Niektoré vedia zvuk nahrať do jednej zvukovej stopy, na ňu aplikovať krátky reťazec efektov a následne výsledný zvuk prehrať alebo uložiť.

Pokročilejšie systémy majú súhrnné označenie DAW (digital audio workstation). Najdôležitejšou časťou DAW je okno so zobrazenými stopami, ich počet je voliteľný. Stopy zobrazuje pod seba, do každej sa dá nahrávať zvuk zo vstupu zvukovej karty. Je možné súčasne, v závislosti na hardware, nahrávať desiatky stôp, prehrávať je možné stovky stôp. Nezávisle na každú stopu je možné aplikovať reťazec efektov, ktoré stopu spracovávajú real-time pri prehrávaní, spracovaný zvuk je možné uložiť. Je možné prehrávať iba zvolené stopy, vykonávať

komplexnú analýzu zvuku na ktorejkoľvek úrovni spracovania efektami, už samotné aplikácie obsahujú množstvo efektov, naviac majú podporu pre rôzne formáty externých digitálnych efektov (najrozšírenejší je formát VST). Príkladom DAW aplikácií sú systémy ako ProTools, Cubase alebo Samplitude.

2.3.3 Rozhranie VST

VST (Virtual Studio Technology) je voľne dostupné rozhranie vyvinuté firmou Steinberg, ktoré slúži na integráciu virtuálnych zvukových modulov a virtuálnych nástrojov simulujúcich reálne štúdiové vybavenie. Základnou jednotkou je VST plugin, existujú dve skupiny týchto pluginov - VST efekty a VST inštrumenty (označované tiež VST-i). VST efekt predstavuje digitálny zvukový filter, takže prijíma zvukové dáta, tie upravuje a posiela na výstup. VST-i zvukové dáta prijímať nemusí, jeho hlavnou úlohou je zvuk vytvárať na základe prijímaných ovládacích dát. Pre chod pluginov je potrebná hostiteľská aplikácia, ktorá zabezpečuje interakciu medzi zvukovými, resp. ovládacími dátami a VST efektom, resp. VST inštrumentom.

Súbor VST pluginu je vo formáte dll, kento je možné načítať v hodtiteľskej aplikácii. Ako hostiteľská aplikácia byť použitá väčšina z DAW aplikácií, pretože VST sa stalo štandardom. Niekedy je však zbytočné zamestnávať zložitú DAW, v ktorej VST podpora tvorí veľmi malú časť a zbytok ostáva nevyužitý. Či už v tomto prípade potrebujeme spracovávať zvuk real-time alebo ho nahrávať, je lepšie použiť jednoduchšiu aplikáciu, ktorá zvláda iba zreťazenie niekoľkých efektov a vstupný zvuk s týmito efektami upraviť real-time, bez toho aby niečo ukladala na disk, prípadne aplikovať efekty na nahraný zvuk.

Kedže VST rozhranie je voľne prístupné, stala sa táto platforma veľmi obľúbenou na všetkých úrovniach profesionality. Všetky zvukové efekty vyrábané ako komerčné produkty, ktorých je obrovské množstvo, sa predávajú aj ako VST pluginy. Na internete existuje ešte oveľa viac voľne stiahnuteľných VST efektov napísaných našincami. Niektoré z nich sú na kvalitatívne nízkej úrovni, niektoré sú však porovnateľné s platenými alternatívami.

2.3.4 Súbory FXP

Takmer všetky VST efekty obsahujú nastaviteľné parametre. Pri práci s nimi sa stretneme s problémom, ako si zapamätať nastavenie efektu, tzv. VST program, keďže samotný efekt neobsahuje žiadne možnosti na uloženie nastavení. Toto sa môže hodiť napríklad v prípade, že používame niekoľko hostiteľských aplikácií a potrebujme preniesť nastavenie efektu medzi nimi. Tento problém riešia súbory typu fxp, do ktorých je možné uložiť nastavenia všetkých parametrov efektu. Po uložení nastavení do fxp súboru v jednej hostiteľskej aplikácii stačí tento súbor načítať v aplikácii inej. Do jedného fxp súboru je možné uložiť práve jeden VST program, tzn. pre každé nastavenie efektu potrebujeme samostatný súbor. Fxp súbor je pri svojom vytvorení viazaný na efekt pre ktorý bol vytvorený, takže nie je možné používať jeden fxp súbor s niekoľkými rôznymi efektami. Podpora fxp súborov musí byť implementovaná vo VST hostiteľskej aplikácii.

2.4 Súvisiace problematiky

2.4.1 Audio Fingerprints

Technológia audio fingerprintov (zvukových odtlačkov prstov) slúži na rozpoznávanie zvukových súborov na základe určitých charakteristík obsiahnutej zvukovej informácie. Používa sa napríklad pri identifikácii piesní, reklám, zvukových efektov v databázach alebo pri štatistických úlohách.

Audio fingerprint predstavuje jedinečný popis vygenerovaný zo zvukového signálu. Ak sa dva zvukové súbory javia ľudskému poslucháčovi rovnaké, mali by mať rovnaký fingerprint. Teda jeho generovanie musí byť nezávislé na forme súboru. Väčšina kompresných systémov robí veľké zmeny v spôsobe uloženia zvukového súboru bez zmeny jeho znenia. Systém audio fingerprintov by mal správne identifikovať zvukovú informáciu aj v takýchto súboroch.

Existuje niekoľko implementácii tejto technológie, napríklad komerčný *AudioID* od Fraunhofer Institute alebo open-source *fdmf*. [11]

3 Návrh a implementácia

Nasledujúca kapitola popisuje návrh a implementáciu programu ReSounder a sady efektov ním používaných. Samotný program je uložený ako príloha na priloženom CD nosiči.

Základnou požiadavkou bolo vypracovať aplikáciu, ktorá podľa zvoleného čistého a zefektovaného zvukového súboru pomocou sady vlastných implementovaných efektov vypočíta nastavenie efektu, s ktorým bol vytvorený zefektovaný signál. Cieľom programu je implementovať rôzne algoritmy a metriky na porovnávanie dvoch signálov a vo VST efektoch niektoré z najpoužívanejších zvukových filtrov.

Aplikácia je naprogramovaná v jazyku C++, vo verzii native aj managed. Dôvod k programovaniu vo verziii managed bola práca s knižnicou Windows Forms použitou pre grafické rozhranie. Pre komunikáciu s VST efektom je použitá knižnica Steinberg VST SDK.

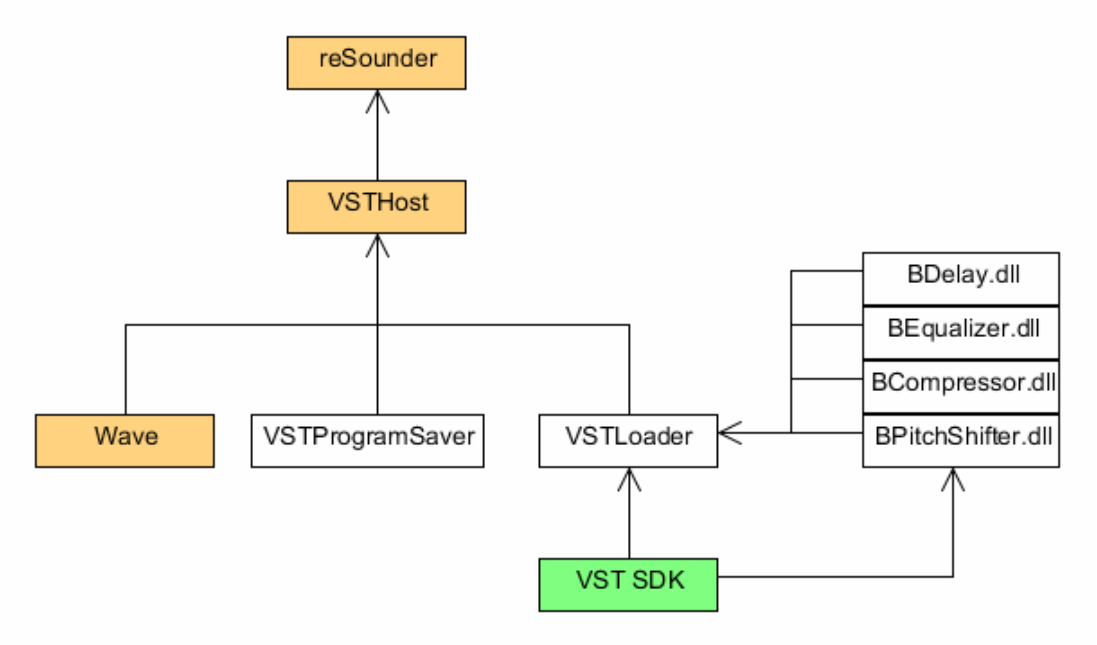
3.1 Program ReSounder

Program ReSounder pracuje s dvomi zvukovými súbormi, ktorých cestu zadá používateľ. Prvý zvukový súbor je pôvodný signál nazývaný „čistý“. Druhý zvukový súbor je čistý signál po úprave efektom, tento je nazývaný „hľadaný“. Úlohou programu je nájsť nastavenia parametrov efektu, ktoré boli použité pri vytváraní hľadaného signálu a uložiť ich do fxp súboru.

3.1.1 Štruktúra programu

Celý program sa skladá z niekoľkých častí. Ovládanie a grafické rozhranie zabezpečuje trieda *ReSounder*. Najdôležitejšou triedou je trieda *VstHost*, ktorá zabezpečuje všetky potrebné výpočty. Na svoju prácu využíva niekoľko pomocných tried. Čítanie zvukových dát prebieha pomocou tiedy *Wave*, komunikáciu s VST

efektom zabezpečuje trieda *VstLoader* a služby pre ukladanie výsledkov do fxp súborov poskytuje trieda *VstProgramSaver*.



Obrázok 6: Schéma štruktúry aplikácie. Šípky znázorňujú smer poskytovaných služieb. Oranžovou farbou sú znázornené triedy naprogramované v managed C++. Zelenou farbou je zobrazená VST knižnica od firmy Steinberg.

3.1.2 Knižnica VST

Program ReSounder aj implementované efekty využívajú knižnicu Steinberg VST SDK verzie 2.4. V súčasnosti je dostupná aj novšia verzia 3.0, no jej podpora medzi hostiteľskými aplikáciami nie je ešte veľmi rozšírená. Knižnicu je možné voľne stiahnuť zo stránky firmy Steinberg (1]. Je distribuovaná vo forme zdrojových kódov v C++.

3.1.3 Knižnica FFT

Na prácu s rýchlou fourierovej transformáciou sa využíva knižnica *fft*. Je postavená na zdrojovom kóde popísanom v [2], umožňuje spočítať priamu a spätnú transformáciu. Obsahuje jedinú metódu, ktorá má na vstupe pole čísel a parameter. Parameter udáva, či sa má previesť priama alebo spätná transformácia.

V prípade prevedenia priamej transformácie predstavuje pole na vstupe originálny signál. Po vykonaní metódy sa toto pole upraví tak, že sú v ňom uložené

sínusové a kosínusové zložky všetkých frekvencií vypočítané zo vstupného signálu. Frekvencie sú pevne dané počtom prvkov v poli.

V prípade, že vykonávame spätnú transformáciu, pole na vstupe predstavuje sínusové a kosínusové zložky a po vykonaní metódy sa upraví tak, že reprezentuje signál.

3.1.4 Grafické rozhranie

Ovládanie programu ReSounder a komunikáciu s používateľom pomocou grafického rozhrania zabezpečuje trieda *ReSounder* naprogramovaná v managed verzii C++. Pre grafické rozhranie používa knižnicu Windows Forms. Dôležitou časťou grafickéo rozhrania je konzola, ktorá zobrazuje aktuálne informácie o stave programu a výsledky výpočtov.

3.1.5 Manipulácia s dátami

Program podporuje čítanie zvukových súborov vo formáte WAVE. Podporovaná je 16-bitová verzia s jedným alebo dvoma kanálmi (mono alebo stereo). Ďalej program podporuje zápis do súborov typu fxp, ktoré slúžia na ukladanie nastavení VST efektu s cieľom načítať ich v akomkoľvek inom programe podporujúcom fxp súbory.

3.1.5.1 Čítanie zvukových dát

Čítanie zvukového súboru zabezpečuje trieda *Wave* naprogramovaná v managed verzii C++. V konštruktore načíta celý zvukový súbor do pamäte a z jeho hlavičky zístí potrebné údaje pre jeho dalšie využívanie ako je počet kanálov, vzorkovacia frekvencia, dĺžka zvukových dát, bitová hĺbka. Na samotné čítanie zvukových dát slúži metóda *getCurrentSample*, ktorá vracia hodnotu vzorky určenej ukazovateľom. Keďže táto hodnota je vo WAV súbore uložená ako Integer, z dôvodu dalšieho spracovania signálu je nutné previesť ju na typ Float tak, aby hodnoty spadali do intervalu (-1, 1). Ukazovateľ sa štandarde pohybuje od začiatku až po koniec zvukových dát. Posun ukazovateľa nie je automatický, ale je potrebné volať metódu *step*. Tiež je k dispozícii metóda *resetPosition*, ktorá nastaví ukazovateľ na začiatok zvukových dát.

3.1.5.2 Zápis fxp súboru

Na zápis výsledných hodnôt parametrov efektu do fxp súboru slúži trieda *VstProgramSaver* naprogramovaná v native C++. Pri volaní tejto triedy nie je potrebné vytvárať jej inštanciu ale stačí zavolať jej konštruktor s príslušnými argumentami. Na začiatku fxp súboru je hlavičky o veľkosti 52 Bytov. V nej je okrem riadiacich informácií dát uložené tiež identifikačné číslo efektu, na ktorý je fxp súbor viazaný a názov ukladaného VST programu. Tento názov je zhodný s názvom súboru, ktorý zvolil používateľ. Po hlavičke nasledujú samotné hodnoty parametrov, každá je uložená ako 4-bytový Float. Ich poradie je súhlasné s poradím parametrov vo VST efekte.

3.1.6 Komunikácia s VST efektom

Komunikáciu medzi programom a VST efektom zabezpečuje trieda *VstLoader* naprogramovaná v native C++. Táto trieda načíta zadaný dll súbor efektu a následne umožňuje získavať informácie o efekte (názov, počet parametrov), o aktuálnom nastavení hodnôt parametrov, nastavovať hodnoty parametrov a používať efekt na spracovanie signálu. Táto trieda je vytvorená podľa špecifikácie VST rozhrania a využíva triedu *aeffectx*, ktorá je súčasťou VST SDK.

3.2 Implementované efekty

Softvérové riešenie programu ReSounder zahŕňa aj vlastnú implementáciu VST efektov pre každý analizovateľný typ. Ich použitie je v „rekonštrukčnej“ fáze, keď sme už programom našli vhodné nastavenia parametrov efektu a uložili sme ich ako fxp súbor. Práve tento fxp súbor je viazaný na konkrétny implementovaný VST efekt a je možné ho načítať v ktorejkoľvej VST hostiteľskej aplikácii. Niektoré typy efektov sú používané aj vo výpočtovej fáze na vytváranie zefektovaných signálov.

Štruktúra VST efektu je daná špecifikáciou firmy Steinberg. Súbor efektu je vo formáte dll. VST efekt musí obsahovať pevne dané verejné metódy, ktoré hostiteľskej aplikácii dovoľujú nastavovať a získavať informácie o parametroch a vlastnostiach efektu. Hodnoty parametrov efektu sa pohybujú v rozsahu 0.0 až +1.0 a sú typu Float. Samotné spracovanie zvuku má na starosti metóda *processReplacing*.

Tá ako argument dostáva dva polia čísel typu Float, kde prvé predstavuje vstupný signál a druhé signál výstupný, a dĺžku jedného poľa. Hodnoty vzoriek sa pohybujú v rozsahu -1.0 až +1.0.

3.2.1 BDelay

Efekt BDelay implementuje efekt typu delay. Tento typ funguje tak, že ku priamemu signálu primiešava signál oneskorený. Efekt BDelay obsahuje tri parametre:

delay : veľkosť oneskorenia (0 až 1000 milisekúnd)
wet : zmena hlasitosti oneskoreného signálu (-oo až 0 dB)
dry : zmena hlasitosti neoneskoreného signálu (-oo až 0 dB)

Algoritmus efektu pracuje s oneskorovacím bufferom, ktorého veľkosť závisí na oneskorení nastavenom parametrom *Delay* a vzorkovacej frekvencii vstupného signálu.

$$velkostBufferu = vzorkovaciaFrekvencia * Delay$$

Pozíciu zápisu a čítania v bufferi určuje kruhový ukazovateľ, ktorý sa pohybuje v rozmedzí 0 až veľkosť bufferu. Po prečítaní hodnoty z bufferu sa táto prepíše hodnotou vstupného signálu a ukazovateľ sa zväčší o 1, prípadne sa nastaví na 0. Výstupný signál efektu je súčet aktuálneho vstupného signálu vynásobeného hodnotou parametru *Dry* a signálu prečítaného z bufferu vynásobeného hodnotou parametru *Wet*.

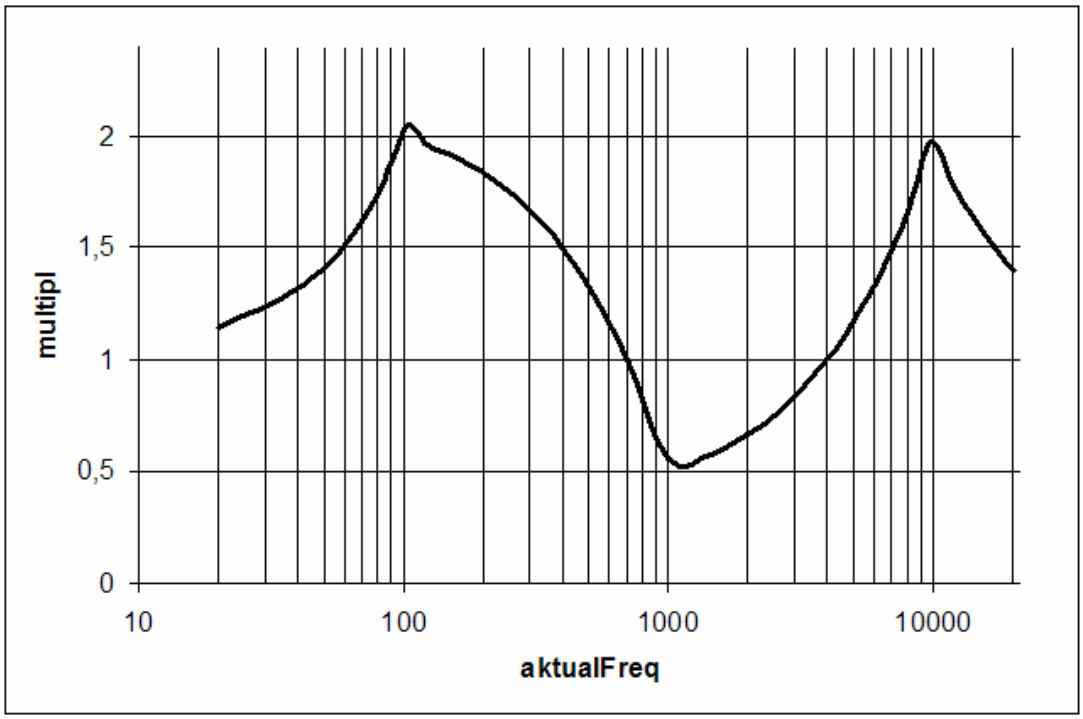
$$vystup = vstup * Dry + buffer[i] * Wet$$

3.2.2 BEqualizer

Efekt BEqualizer implementuje efekt typu equalizer. Tento typ funguje tak, že mení hlasitosť frekvenčných pásiem nezávisle na sebe. Efekt BEqualizer je 3-pásmový, tzn. je možné nastaviť hlasitosť troch frekvenčných pásiem. Navyše je možné upraviť aj celkovú hlasitosť výstupného signálu. Parametre efektu:

Bass : hlasitosť spodného frekvenčného pásma (-12 až +12 dB)
Middle: hlasitosť stredného frekvenčného pásma (-12 až +12 dB)
Treble : hlasitosť vysokého frekvenčného pásma (-12 až +12 dB)
Output : celková hlasitosť (-12 až +12 dB)

Pri výpočte výsledného signálu sa najprv vstupný signál rozloží na malé segmenty a každý segment je spracovaný nasledovne. Pomocou rýchlej fourierovej transformácie (FFT) sa spočíta rozklad signálu na sínusové a kosínusové frekvenčné zložky. Následne sa pre každú frekvenciu spočíta jej multiplikátor pomocou hodnôt parametrov *Bass*, *Middle* a *Treble*. Frekvenčné spektrum sa podľa frekvencií týchto parametrov, ktoré sú 100 Hz pre *Bass,* 1 kHz pre *Middle* a 10 kHz pre *Treble* rozdelí na štyri frekvenčné pásma.



Obrázok 7: Znázornené hodnoty multiplikátoru pre nastavené hodnoty Bass=2, Mid=0,5, Treb=2. Os X zobrazuje frekvencie v logaritmickom meradle.

Pre frekvencie v spodnom okrajovom pásme sa hodnota multiplikátoru spočíta nasledovne:

$$multipl = Bass^{aktualFreq/100\,Hz}$$

Podobne pre frekvencie vo vrchnom okrajovom pásme, s tým rozdielom, že delenec a deliteľ v exponente sú prehodené:

$$multipl = Treble^{10000\,Hz\,/\,aktualFreq}$$

Pre frekvencie vo vnútorných pásmach sa multiplikátor vypočíta ako hodnota lineárnej funkcie prechádzajúcej hodnotami *Bass* a *Middle* pre nižšie pásmo, resp. *Middle* a *Treble* pre vyššie pásmo. Pre nižšie pásmo platí:

$$dy = Middle - Bass$$
$$dx = 1000Hz - 100Hz = 900$$
$$multipl = Bass + (aktualFreq - 100Hz) * \frac{dy}{dx}$$

Podobne pre vyššie pásmo:

$$dy = Treble - Middle$$
$$dx = 10000Hz - 1000Hz = 9000$$
$$multipl = Middle + (aktualFreq - 1000Hz) * \frac{dy}{dx}$$

Následne sa s vypočítaným multiplikátorom vynásobí sínusová aj kosínusová zložka danej frekvencie. Po spracovaní všetkých frekvencií sa prevedie inverzná FFT a jej výsledok sa vynásobí hodnotou parametru *Gain*. Po spracovaní všetkých segmentov a ich spojení dostávame výstupný zekvalizovaný signál.

3.2.3 BCompressor

Efekt BCompressor implemetuje dynamické efekty kompresor a limiter. Typ efektu kompresor funguje tak, že ak vstupný signál prekročí určenú hranicu treshold, po uplynutí doby attack time tento signál zoslabí v kompresnom pomere ratio. Zoslabenie trvá dobu release time. Limiter funguje podobne ako kompresor, rozdiel je v tom, že kompresný pomer je pevne nastavený na 0 a časy attack time a release

time nezohrávajú úlohu, resp. sú pevne nastavené na najmenšiu možnú hodnotu. Inými slovami, nad hranicu treshold neprepustí nič. Efekt BCompressor je v jednom momente možné použiť len ako kompresor, resp. len ako limiter. Obsahuje parametre:

Input :	úroveň vstupného signálu (-13 až +13 dB). Tento parameter môže používateľ použiť v prípade, keď signál je príliš silný, resp. príliš slabý na to, aby mohol byť účinne spracovaný
Comp/Lim :	prepína funkciu efektu medzi kompresorom a limiterom
Attack :	čas attack pri použití kompresoru (0,1 až 1000 milisekúnd)
Release :	čas release pri použití kompresoru (0,1 až 1000 milisekúnd)
Ratio :	kompresný pomer pri použití kompresoru (1:1 až 1:10)
Treshold :	úroveň treshold pri použití kompresoru aj limiteru(-oo až 0 dB)
MakeUp :	vypínač automatického nastavovania výstupnej úrovne signál
Output :	manuálne nastavenie výstupnej úrovne signálu pokiaľ je funkcia MakeUp vypnutá

Spracovanie signálu prebieha v krokoch po vzorkách, signál sa nijako neukladá, ale na základe hodnôt určených predchádzajúcim signálom sa aktuálna vzorka hneď spracuje a hneď sa pošle na výstup.

V prvom kroku sa vstupný signál vynásobí hodnotou parametru *Input*. Následne sa podľa parametru *Comp/Lim* rozhodne, či sa zavolá metóda funkcie kompresor alebo funkcie limiter.

V prípade, že efekt funguje ako kompresor, skontroluje sa aktuálna hodnota signálu. Ak je väčšia ako hodnota treshold nastavená parametrom *Treshold*, spustí sa počítadlo *attack_counter*, čo znamená, že sa nastaví na hodnotu danú parametrom *Attack*. Toto počítadlo meria čas do začatia komprimácie signálu. V každom kroku sa o 1 zníži a pokiaľ nie je rovné 0, signál sa nekomprimuje, tede na výstup sa posielajú tie isté hodnoty, ktoré sú na vstupe. Ako náhle *attack_counter* dosiahne 0, spustí sa počítadlo *release_counter* tak, že sa nastaví na hodnotu určenú parametrom *Release*. Toto počítadlo počíta čas do konca komprimácie signálu a tiež sa v každom kroku znižuje o 1. Ak je *release_counter* väčší ako 0 a zároveň je hodnota signálu na vstupe väčša ako *Treshold*, táto hodnota sa skomprimuje hodnotou určenou parametrom *Release* a pošle na výstup metódy:

$$vystupMetody = Treshold + (vstup - Treshold) * Ratio$$

Ak nastane prípad, že vstupný signál prekročí treshold, *attack_counter* je rovný 0 a zároveň je *release_counter* väčší ako 0, nespúšťa sa *attack_counter*, ale reštartuje sa *release_counter*. To znamená, že *release_counter* začne klesať až v momente, keď signál opustí hodnoty vyššie ako treshold a ak sa na tieto hodnoty vráti ešte počas klesania *release_counter*-u, začnú sa komprimovať okamžite, bez toho, aby bolo nutné čakať na *attack_counter*.

Pokiaľ efekt funguje ako limiter, skontroluje sa hodnota vstupného signálu a ak je väčšia ako hodnota treshold, na výstup sa pošle hodnota treshold, inak sa na výstup metódy pošle hodnota signálu.

$$vystupMetody = \begin{cases} vstup & ; vstup < paramTresh \\ paramTresh & ; vstup \geq paramTresh \end{cases}$$

Po spracovaní vzorky kompresorom alebo limiterom sa ešte upraví jej konečná hodnota. V prípade, že používateľ zvolil funkciu *MakeUp*, výstup metódy sa vynásobí číslom 1/treshold. V opačnom prípade sa výstup metódy vynásobí číslom určeným parametrom *Output*.

$$vystup = \begin{cases} vystupMetody * (1/Treshold) & ; MakeUp \\ vystupMetody * Output & ; inak \end{cases}$$

Výsledná hodnota sa pošle na výstup efektu.

3.2.4 BPitchShifter

Efekt BPitchShifter implementuje typ efektu pitch shifter. Funkciou tohto typu efektu je frekvenčný posun signálu, tzn., že mení celkovú výšku zvuku (z hudobného hľadiska). Efekt BPitchShifter obsahuje jeden parameter :

Shift : nastavenie veľkosti posunu (-12 až +12 poltónov)

Pre implementáciu tohto efektu bol použitý postup popísaný v (3]. Algoritmus na začiatku rozdelí vstupný signál na malé segmenty a každý segment spracuje nasledovne: Najpr sa spočíta rozklad na frekvenčné zložky pomocou rýchlej fourierovej transformácie (FFT) a pre každú zložkovú frekvenciu sa spočíta jej magnitúda a fáza.

$$magn[freq] = 2 * \sqrt{fftRozklad[freq][i_{\sin}] + ffrRozklad[freq][i_{\cos}]}$$

$$faza[freq] = a\tan 2(fftRozklad[freq][i_{\sin}], fftRozklad[freq][i_{\cos}]$$

Následne sa vykoná samotný posun určený parametrom *Shift* a to tak, že magnitúda z frekvencie *freq* sa priradí frekvencii *freq* vynásobenej parametrom *Shift*.

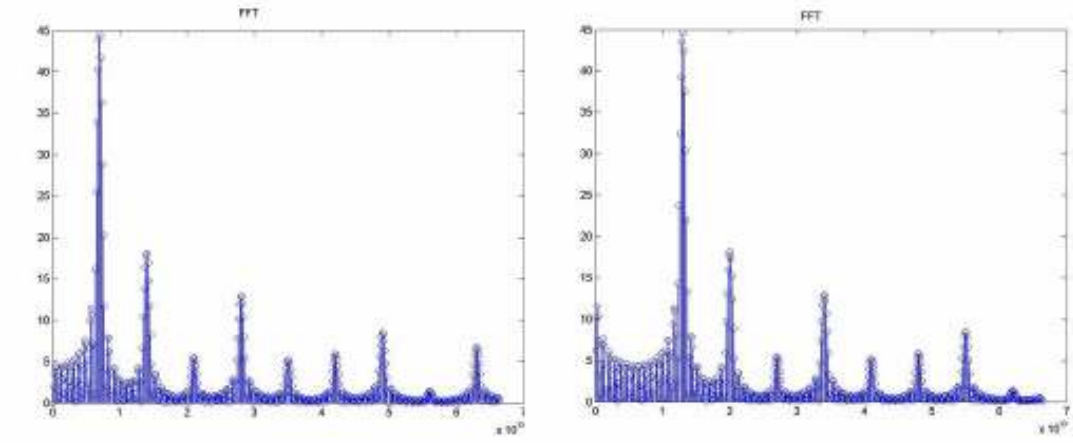
$$magn[freq] \rightarrow magnOut[freq * Shift]$$

V ďalšom kroku sa pre každú frekvenciu posunutých magnitúd spočíta ich sínusová a kosínusová zložka.

$$fftRozkladPosunuty[freq][i_{\sin}] = magnOut[freq] * \sin(faza[freq / Shift])$$

$$fftRozkladPosunuty[freq][i_{\cos}] = magnOut[freq] * \cos(faza[freq / Shift])$$

Po následnej spätnej FFT sa signál pošle na výstup efektu.



Obrázok 8: Príklad frekvenčného spektra pred (vľavo) a po (vpravo) spracovaní pitch shifterom. Posun je smerom k vyšším frekvenciám. Zdroj: [10]

3.3 Výpočet parametrov

Cieľom programu ReSounder je nájsť nastavenie parametrov efektu tak, aby sa pri spracovaní čistého signálu s efektom s nastavenými nájdenými parametrami dosiahol signál totožný s hľadaným signálom. Na vstupe používateľ zadá dva signály – čistý a hľadaným, ďalej typ efektu, zvolí parametre, ktoré sa majú spracovávať a zvolí, či sa má vykonávať orezávanie alebo nie.

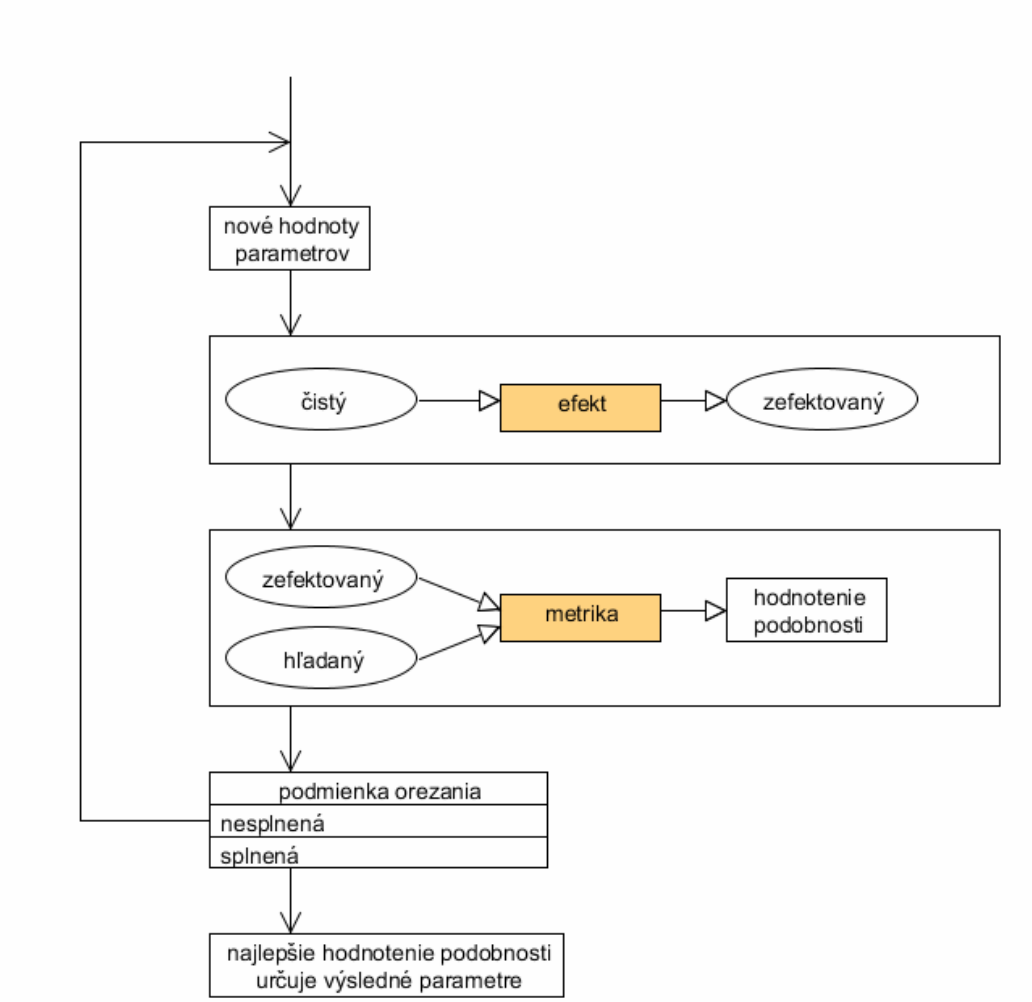
Výpočty prebiehajú dvoma spôsobmi v závislosti na zvolenom type efektu. Prvým spôsobom je výpočet hodnôt parametrov priamou analýzou čistého a hľadaného signálu. Druhým spôsobom je systematicky prechádzať všetkými kombináciami nastavení parametrov, spracovávať s týmito nastaveniami čistý signál a tento zefektovaný signál porovnávať s hľadaným signálom. Rôznych metrikami odmerať ich podobnosť a podľa výsledkov určiť hodnoty parametrov, s ktorými boli signály najpodobnejšie. V snahe minimalizovať výpočtový čas orezávať prehľadávaný priestor. Pretože typy úprav signálu sa pre rôzne typy efektov líšia, je potrebné používať rôzne metriky. Postup je ilustrovaný na obrázku 9.

Prechádzanie všetkými kombináciami je riešené rekurzívne, kde každá úroveň rekurzie nastavuje hodoty práve jedného parametru a kombinuje ich s hodnotami parametrov získaných z nižších úrovní rekurzie. Parameter, ktorý sa bude spracovávať v aktuálnej úrovni rekurzie sa vyberá podľa poradového čísla v efekte, začína sa od najvyššieho. Hĺbka rekurzie je tak rovná počtu parametrov efektu, čo je tri.

Výber spôsobu výpočtu pre každý typ efektu bol zvolený po uvážení jeho vhodnosti vzhľadom k tomu, aké zmeny vytvára efekt v signále a na čom tieto zmeny závisia.

Pre úspešné spustenie výpočtu je nutné, aby používateľ pomocou grafického ovládacieho prostredia programu zadal cestu k súboru s čistým signálom, k súboru s hľadaným signálom a aby zvolil typ efektu, ktorý bol použitý pri vytváraní hľadaného signálu. Pri každej z týchto troch akcií sa zavolá príslušná metóda triedy *VstHost*, ktorá pripravý zvukový súbor na čítanie, resp. inicializuje príslušný efektový plugin. Používateľ má tiež možnosť zvoliť si konkrétne parametre efektu, ktorých hodnoty sa budú hľadať. Táto možnosť je užitočná hlavne pri časovo náročných spôsoboch výpočtu. Pri typoch efektov, ktoré používajú druhý spôsob

výpočtu má používateľ možnosť povoliť alebo zakázať použitie orezávania zaškrtnutím checkboxu *Fast method*.



Obrázok 9: Postup hľadania parametrov druhým spôsobom – pomocou prechádzania všetkými kombináciami nastavení parametrov. V prvom kroku získame nové doteraz nepoužité nastavenia parametrov, následne s efektom s nastavenými týmito parametrami vygenerujeme zefektovaný signál, ďalej metrikou porovnáme zefektovaný a hľadaný signál. V ďalšom kroku sa rozhodneme, či skončíme hľadanie alebo pokračujeme odznova. Parametre s ktorými sa dosiahla najväčšia podobnosť zefektovaného a hľadaného signálu sú výsledné.

Po stlačení tlačidla *Process* sa zavolá metóda *process* triedy *VstHost*. Tá podľa zvoleného typu efektu zavolá metódu s implementovaným algoritmom pre konkrétny typ a výsledné hodnoty parametrov vypíše do konzoly. Po výpočte môže používateľ uložiť výsledné hodnoty do fxp súboru pomocou triedy *VstProgramSaver*.

Okrem 5 konkrétnych typov efektov vzťahujúcich sa k implementovaným VST efektom program obsahuje aj možnosť výpočtu parametrov ľubovoľného efektu, ktorého cestu zadá používateľ. Táto možnosť je však iba v testovacom štádiu.

3.3.1 Delay

Výpočet prebieha tak, že sa systémom prechádzania všetkými kombináciami hodnôt parametrov v efekte BDelay.

Algoritmus výpočtu parametrov pre tento typ efektu je implementovaný na základe prechádzania všetkými kombináciami nastavení parametrov. Využíva sa VST efekt BDelay.

Efekt BDelay obsahuje dva typy parametrov, ku ktorým je treba pristupovať rozdielne. Prvým typom je parameter *Delay*, ktorým sa nastavuje veľkosť oneskorenia a tým sa mení celkový tvar signálu. Druhým typom sú parametre *Dry* a *Wet*, ktoré tvar signálu nemenia, ale menia iba jeho úroveň – násobia signál. Dôvodom k odlišnému prístupu k týmto dvom typom parametrov je to, že algoritmus použitý pre parameter *Delay* hľadá odlišnosti v priebehu dvoch signálov, kde nezáleží na ich intenzite. V prípade použitia tohto algoritmu pre parameter druhého typu by bol výsledok ekvivalentný výsledku pri porovnávaní dvoch rovnakých signálov s rovnakou intenzitou.

3.3.1.1 Výpočet

Hlavnou časťou metódy je cyklus, ktorý iteruje cez všetky hodnoty aktuálne nastavovaného parametru. Tieto hodnoty sú určené podľa veľkosti posunu parametru, ktorý je nastavený pre parameter delay na 1% a pre ostatné parametre na 10 %. V prvom kroku sa nastaví efektu BDelay tento parameter na aktuálnu hodnotu a zavolá sa rekurzia. Toto rekurzívne volanie vráti hodnoty všetkých parametrov s nižším poradovým číslom. Následne efektom BDelay vytvoríme zefektovaný signál tak, že parametre z nižších úrovní rekurzie nastavíme na nájdené hodnoty, jeden aktuálne nastavovaný parameter nastavíme na hodnotu podľa aktuálnej iterácie cyklu a ostatné parametre nenastavujeme, tzn. ostávajú na hodnotách, ktoré im nastavili vyššie úrovne rekurzie.

Vytvorený zefektovaný signál sa porovná s hľadaným signálom a ratingom sa numericky vyjadrí ich podobnosť. Platí, že čím menší rating, tým je signál

podobnejší. Podľa typu aktuálne nastavovaného parametru sa vyberie konkrétna metrika.

Nasleduje orezávanie rekurzívneho stromu tak, že sa v kladnom prípade zastavia ďalšie iterácie cyklu. Aj v tomto kroku záleží výber rozhodovacieho algoritmu na type aktuálne nastavovaného parametru.

3.3.1.2 Metriky

V type efektu *Delay* sa používajú dve rôzne metriky. Ich výber záleží na type parametru nastavovaného v aktuálnej úrovni rekurzie. To znamená, že pre parameter *Delay,* ktorý mení celkový priebeh signálu sa zvolí prvá metrika a pre parametre *Dry* a *Wet* ktoré menia úroveň signálu sa zvolí metrika druhá.

Prvá metrika funguje takým spôsobom, že sa dva signály porovnávajú tak, že sa tieto spolu vynásobia a následne sa sčítajú záporné hodnoty výsledného vynásobeného signálu. To znamená, že ak je signál rovnaký (má rovnaký priebeh), násobí sa buď kladná hodnota s kladnou alebo záporná so zápornou - v oboch prípadoch dostaneme kladný súčin. Ak majú dva signály rôzne priebehy, násobíme kladné číslo so záporným, čím dostávame záporný súčin. Súčet záporných hodnôt číselne určuje odlišnosť týchto dvoch signálov a absolútna hodnota tohto čísla je použitá ako rating.

$$vynasobeny[i] = signal1[i] * signal2[i]$$

$$rating = \left| \sum_i \left(vynasobeny[i]; vynasobeny[i] < 0 \right) \right|;$$

Pri type parametru, ktorý nastavuje hlasitosť, sa rozdiel dvoch signálov vypočíta ako rozdiel ich RMS hodnôt (root mean square). Z oboch signálov sa vypočítajú RMS hodnoty s okom veľkosti n (defaultne 1024). To znamená, že z každých n hodnôt (bez prekrývania) sa vypočíta jedna RMS hodnota. Následne sa nájdu rozdiely medzi príslušnými RMS hodnotami čistého a zefektovaného signálu. Výsledný rating predstavuje súčet týchto rozdielov vydelený ich počtom.

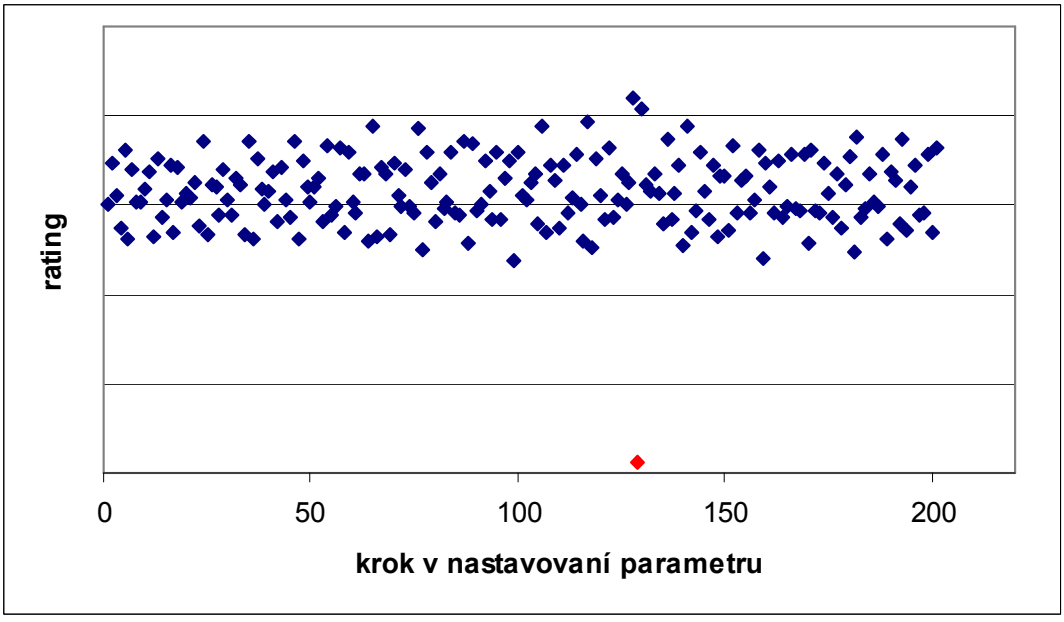
$$m = \frac{|x|}{n}; x = signal$$

$$\underset{i\in[0..m)}{rms[i]} = \sqrt{\frac{x[n*i]^2 + x[n*i+1]^2 + \ldots + x[n*i+(n-1)]}{n}}$$

$$rating = \frac{\sum_{i=0}^{m-1} |rms1[i] - rms2[i]|}{m}$$

3.3.1.3 Orezávanie

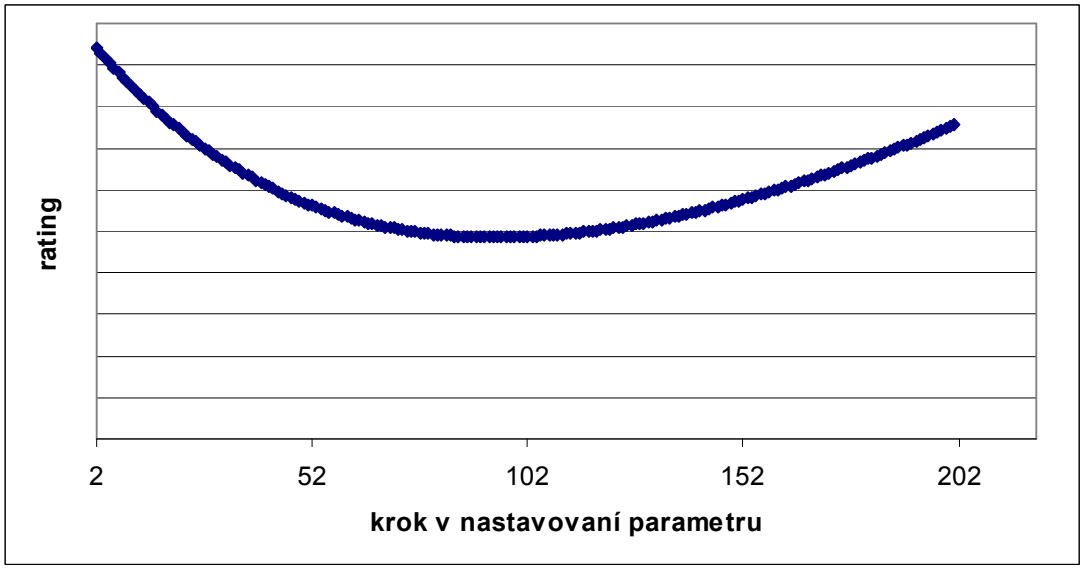
Rozhodovací spôsob pri orezávaní sa tiež vyberá podľa typu aktuálne nastavovaného parametru. V prípade bežného typu vzniká kladná situácia vtedy, keď je práve nájdený rating „oveľa menší“ ako predchádzajúci rating *lastRating*, tzn. že nepatrí do malého okolia určeného doteraz nájdenými ratingami. Na obrázku 6 sú znázornené hodnoty ratingov pri nastavovaní parametru *Delay*. Najnižšia hodnota signalizuje nastavenie parametru najbližšie k hľadanému signálu. Z porozovania vyplynulo, že stačí, aby práve nájdený rating bol menší ako 6/10 predchádzajúceho ratingu.



Obrázok 10: Príklad priebehu ratingov pri výpočte parametru *Delay*. Červená hodnota označuje rating výslednej hodnoty parametru.

V prípade typu parametru, ktorý nastavuje hlasitosť, vzniká kladná situácia vtedy, keď sme sa aktuálnym ratingom dostali na minimum postupnosti ratingov (predpokladáme, že existuje iba jedno minumum), čiže práve nájdený rating je väčší ako predchádzajúci rating *lastRating* a ten je menší ako rating pred ním *lastLastRating* (obrázok 7).

$$lastRating < lastLastRating$$
$$lastRating < rating$$



Obrázok 11: Príklad priebehu hodnôt ratingov pri výpočte parametru *Dry* alebo *Wet*.

Počet orezaných nastavení závisí od úrovne rekurzie, v ktorej k orezaniu došlo. Číslo nastavovaného parametru so zväčšujúcou sa hĺbkou rekurzie klesá, teda z usporiadania parametrov v efekte BDelay vyplýva, že postupnosť parametrov vzhľadom na zväčšujúcu sa hĺbku rekurzie je *Dry*-*Wet*-*Delay*. To znamená, že pri orezaní na spodnej úrovni rekurzie sa preskakujú iba zvyšné nastavenia parametru *Delay* a pri orezaní na najvrchnejšej úrovni sa preskakujú všetky ostatné nastavenia, teda výpočet končí.

3.3.2 Equalizer

Algoritmus pre tento typ efektu nevyužíva prechádzanie všetkých kombináciami nastavenia efektu, ale hodnoty parametrov vypočíta priamo z porovnania čistého a hľadaného signálu. Nastavuje VST efekt BEqualizer.

Na začiatku sa pre čistý aj hľadaný signál sa vytvorí frekvenčný obraz *obrazCisty* a *obrazHladany*. A to tak, že pre každých n (defaultne 1024) vzoriek signálu (bez prekrývania) sa pomocou rýchlej fourierovej transformácie (FFT) vypočíta rozklad na sinusové a kosínusové signály.

$$\underset{i\in\left\langle 0..\frac{|x|}{n}\right\rangle}{obrazCisty[i]} = FFT\left(x[n*i....n*i+(n-1)]\right)$$

Parametre efektu BEqualizer pracujú s tromi frekvenčnými pásmami, a to 100 Hz, 1 kHz a 10 kHz. Vo frekvenčných obrazov oboch signálov nájdeme hodnoty prisluchajúce malému okoliu týchto frekvencií. Po spočítaní pomeru medzi hľadaným a čistým signálom pre každé okolie zvlášť dostaneme výsledné hodnoty parametrov. Pre prvý parameter sa jeho hodnota spočíta nasledovne, pre ostatné parametre je postup obdobný.

$$parameter[100Hz] = \frac{\sum_{f=100Hz-\varepsilon}^{100Hz+\varepsilon} \frac{obrazHladany[f]}{obrazCisty[f]}}{2*\varepsilon}$$

3.3.3 Limiter

V prípade efektu typu limiter spočítame výsledok taktisto priamo z porovnania čistého a hľadaného signálu. Je využívaný VST efekt BCompressor, ktorý obsahuje aj funkciu limiteru. Je potrebné nájsť hodnotu iba jedného parametru – *Treshold*. V prvom kroku spočítame pomer hľadaného signálu(y) s čistým signálom (x).

$$z[i] = \frac{y[i]}{x[i]}; \forall i$$

Následne nájdeme takú hodnotu t v čistom signále, že vypočítaný pomer na pozícii tejto hodnoty je rovný 1 a absolútna hodnota t je maximálna.

$$t = x[i]; z[i] = 1 \wedge \neg\exists\left(j \neq i; \left|x[j]\right| > \left|x[i]\right| \wedge z[j] = 1;\right)$$

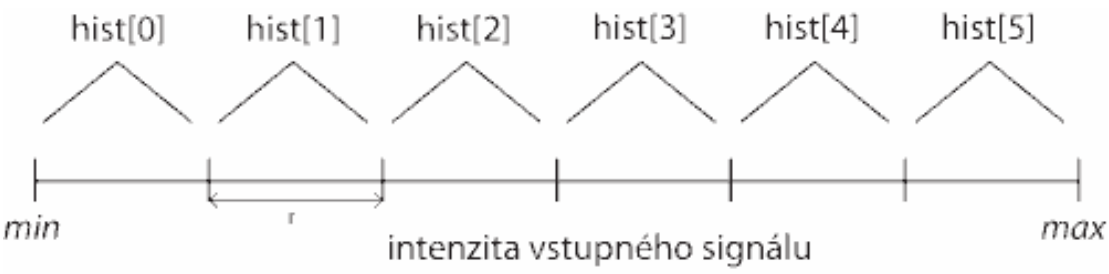
Tým sme získali hodnotu tresholdu pre limiter.

3.3.4 Compressor

Aj v prípade tohto typu sa nepostupuje testovaním všetkých kombnácií hodnôt parametrov, ale výsledné hodnoty sa spočítajú priamo. Výpočet hodnôt parametrov kompresoru ma niekoľko fází. V každej fáze používame namiesto samotného signálu vypočítané hodnoty RMS (s oknom 1/1000 sekundy). To znamená, že zo čistého signálu vypočítame pole *rmsCisty* a z hľadaného signálu pole *rmsHladany*. Na začiatku si tiež spočítame pole *rmsPomer*.

$$rmsPomer[i] = \frac{rmsHladany[i]}{rmsCisty[i]}$$

Prvá fáza vypočíta kompresný pomer *Ratio*. Vytvorí sa pole množín *hist*, kde každá množina reprezentuje určitý interval hodnôt vstupného signálu. Tieto intervaly sú rozdelené lineárne – jeden interval obsahuje rozsah *r* (obrázok 12).



Obrázok 12: Príklad rozdelenia vstupného signálu do poľa *hist*.

$$r = \frac{1}{\left|hist\right|}$$

Iterujeme cez *rmsCisty* a pre každú hodnotu *rmsCisty*[i] nájdeme množinu *j* v *hist*, do ktorej táto hodnota spadá.

$$j = \frac{rmsCisty[i]}{|hist|}$$

Do tejto množiny pripojíme hodnotu *rmsPomer*[i].

$$hist[j].append(rmsPomer[i])$$

Toto opakujeme až kým nedôjdeme na koniec *rmsCisty* alebo kým nemáme zaplnenú predom danú kapacitu. Tá je nastavená na 1000 prvkov *hist* a 10000 prvkom pre každú *hist*[j]. Následne pre každú množinu *hist*[j] spočítame jej priemernú hodnotu a najmenšia takáto hodnota určuje kompresný pomer.

$$Ratio = \min(priemer(hist[j]); \forall j)$$

V ďalšej fáze sa nájde hodnota *Treshold*. Je to cyklus, ktorý iteruje cez všetky hodnoty *rmsCisty* a hľadá také pozície *i*, že hodnota *rmsPomer* je na aktuálnej pozícii rovná, prípadne menšia ako kompresný pomer a na predchádzajúcej pozícii je väčšia ako kompresný pomer. V prípade, že aktuálna pozícia túto podmienku splňuje, k celkovej sume sa pripočíta hodnota *rmsCisty* na aktuálnej pozícií.

$$suma = \sum rmsCisty[i];\ rmsPomer[i-1] > Ratio \wedge rmsPomer[i] \leq Ratio$$

Po skončení cyklu sa celková suma vydelí počtom takýchto nájdených hodnôt a toto číslo určuje hodnotu *Treshold*.

Nasledujúca fáza nájde hodnotu parametru *Attack*. V cykle, ktorý iteruje cez všetky hodnoty *rmsCisty* sa hľadajú pozície *i* , ktoré v *rmsCisty* určujú konce nekomprimovaných úsekov, tzn. konce takých úsekov, ktorých hodnota *rmsRatio* je

rovná 1. Po nájdení takejto pozície sa vraciame v signále späť pokiaľ je *rmsPomer* rovný 1 (na začiatok nekomprimovaného signálu) alebo pokiaľ sa nevrátime o *maxAttack*. Hodnota *maxAttack* určuje maximálny attack time, ktorý môžme nájsť a je nastavená na hodnotu 100 milisekúnd. Počas tohto spätného behu hľadáme vzdialenosť najvzdialenejšej hodnoty signálu väčšej ako *Treshold* od pozície *i*. Zo všetkých takýchto nájdených vzdialeností priemerom spočítame hodnotu *Attack*.

Posledná fáza má za úlohu nájsť hodnotu parametru *Release*. Funguje podobne ako fáza predchádzajúca. Na rozdiel od nej sa hľadajú pozície v *rmsCisty* určujúce začiatky nekomprimovaných úsekov. V spätnom behu hľadáme vzdialenosť prvej takej hodnoty, ktorá je väčšia ako *Treshold*. Dĺžka behu je maximálne *maxRelease*, čo je hodnota určujúca maximálny release time, ktorý môžme nájsť a je nastavená na 500 milisekúnd. Po prejdení celým poľom *rmsCisty* spočítame priemer všetkých takýchto nájdených vzdialeností a ten určuje hodnotu *Release*.

3.3.5 Pitch shifter

Algoritmus pre tento typ efektu je podobne ako pre typ Delay implementovaný na základe prechádzania všetkými kombináciami nastavení parametrov. Využíva sa VST efekt BPitchShifter.

Efekt BPitchShifter má však na rozdiel od efektu BDelay iba jeden parameter, ktorým sa nastavuje veľkosť frekvenčného posunu signálu. Na tento parameter je nutné použiť odlišnú metriku ako na parametre v efekte BDelay.

3.3.5.1 Výpočet

Výpočet prebieha obdobne ako v prípade typy Delay, s tým rozdielom, že nie je nutné vytvárať rekurziu, keďže nastavujeme iba jeden parameter. Hlavnou časťou je cyklus iterujúci cez všetky hodnoty parametru. Na začiatku každej iterácie sa efektu BPitchShifter nastaví aktuálna hodnota parametru a vyvorí sa zefektovaný signál. Ten sa potom porovná s hľadaným signálom a podľa vypočítaného ratingu sa určí najlepšia hodnota prametru.

3.3.5.2 Metrika

V prípade efektu tohto typu sa na porovnávanie dvoch signálov používa metrika založená na rozklade signálu pomocou rýchlej fourierovej transformácie (FFT).

Pre každý signál sa vytvorí jeho frekvenčný obraz., obdobne ako pri type efektu Equalizer.

$$\underset{i\in\left[0..\frac{|x|}{n}\right)}{fftRozklad[i] = FFT\left(x[n*i....n*i+(n-1)]\right)}$$

Ďalej sa každú frekvenciu vypočíta magnitúda signálu.

$$magn[i][freq] = \sqrt{fftRozklad[i][freq][\sin]^2 + fftRozklad[i][freq][\cos]^2}$$

Frekvenčný obraz vypočitame ako sumu magnitúd celej dĺžky signálu pre každú frekvenciu zvlášť.

$$obraz[freq] = \sum_i magn[i]$$

Výsledný rating je súčet rozdielov hodnôt frekvencií medzi frekvenčnými obrazmi dvoch signálov.

$$rating = \sum_{freq} \left| obrazA[freq] - obrazB[freq] \right|$$

3.3.5.3 Orezávanie

Orezávanie funguje obdobne ako pri type delay, tzn. pri pozitívnej situácii sa zastavia ďalšie iterácie cyklu. Pozitívna situácia nastáva vtedy, keď sme sa dostali na minimum v postupnosti ratingov (predpokladáme, že existuje iba jedno minimum). To znamená, že rating *lastRating* nájdený v predchádzajúcom kroku je menší ako rating *lastLastRating* nájdený v kroku pred a zároveň je *lastRating* menší ako aktuálny *rating*.

$$lastRating < lastLastRating$$
$$lastRating < rating$$

3.3.6 General

Zvolením tohto typu má používateľ možnosť vypočítať parametre ľubovoľného efektu. Táto možnosť je však iba v testovacom štádiu.

Po vybraní tohto typu sa v okne aplikácie zobrazie nové ovládacie prvky, ktorými je potrebné zvoliť dll súbor s VST efektom a označiť parametre, ktoré nemenia signál ako taký, ale menia iba jeho úroveň (sú typu „vol“ = volume). Ďalej je možné nastaviť veľkosť kroku nastavovania parametru, defaultné nastavenie je 10%.

Výpočet a metrika pre tento typ sú zobecnené verzie typu Delay. Pri výpočte zefektovaného signálu sa používa efekt zadaný používateľom. Metrika pre konkrétny parameter sa vyberá na základe jeho označenia typom „vol“. Pre parametre, ktoré nie sú typu „vol“ sa použije prvá metrika a pre parametre, ktoré sú typu „vol“ sa použije druhá metrika.

4 Zhodnotenie

V nasledujúcej kapitole je uvedené zhodnotenie praktických výsledkov pri používaní programu ReSounder.

4.1 Delay

Výsledky pre tento typ efektu sú dobré. Hlavný dôvod je ten, že signál po úprave efektom delay do veľkej miery zmení svoj priebeh a použitým algoritmom je jednoduché nájsť dva najpodobnejšie signály (viď. obrázok 5).

Nevýhodou použitého postupu je časová náročnosť. Hlavné činiteľe ovplyvňujúce potrebný čas sú dĺžka čistého, resp. hľadaného signálu, počet parametrov, ktorých hodnoty sa nastavujú a samotný hľadaný signál.

Dôvod závislosti na dĺžke signálu je zrejmý – pri každej zmene nastavení parametrov je potrebné zakaždým celý signál spracovať efektom. Táto závislosť je lineárna.

Závislosť na počte hľadaných parametrov vychádza z dôvodu použitia rekurzie. Vytvára sa rekurzívny strom, ktorý má počet úrovní rovný počtu parametrov. Z toho vyplýva, že náročnosť rastie exponenciálne vzhľadom na počet hľadaných parametrov.

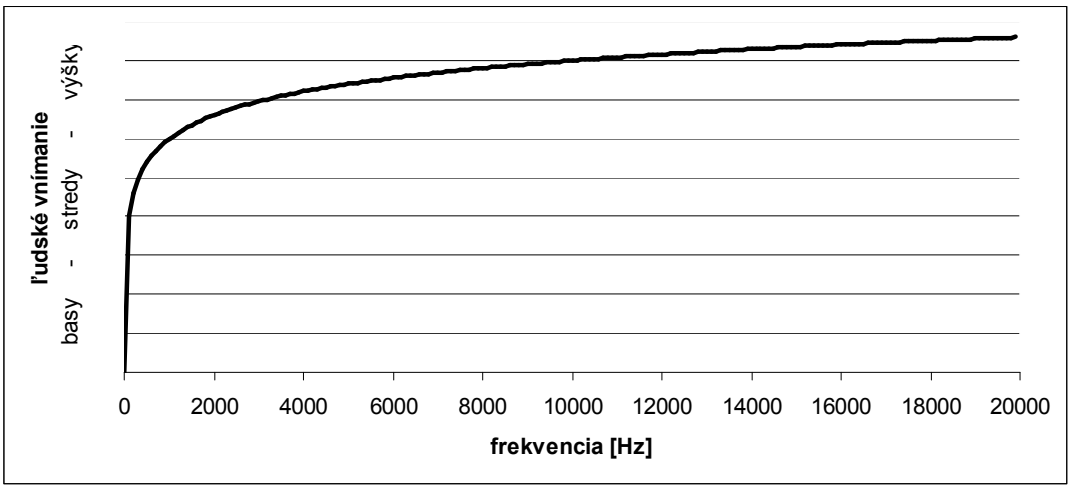
To, že čas výpočtu ovplyvňuje aj hľadaný signál, je z dôvodu použitia orezávania. V tomto prípade je to však vplyv pozitívny. Tento čas sa rôzni podľa toho, či je hľadaná hodnota parametru nízka, v tomto prípade sa odreže väčšina stromu, alebo je vysoká, v tomto prípade sa odreže sa iba malá časť.

Problémom pri tomto algoritme je nutnosť toho, aby parameter delay počas priebehu výpočtu nadobudol hodnotu veľmi blízku hľadanej hodnote. V prípade, že je veľkosť kroku nastavovania parametru príliš veľká a parameter počas výpočtu nadobude iba hodnotu menšiu a väčšiu ako je hľadaná hodnota, výpočet bude neúspešný. Preto je nutné zvoliť veľkosť kroku čo najmenšiu.

4.2 Equalizer

Výsledky pre tento typ efektu sú uspokojivé. Nájdené hodnoty parametrov sa väčšinou líšia o niekoľko decibelov, čo však na signále zefektovanom s týmito parametrami nie je príliš badateľné. Platí to však iba pre hľadaný signál, ktorý bol vytvorený úpravou hlasitosti širokých frekvenčných plôch. V prípade, že hľadaný signál obsahuje upravenú hlasitosť iba malého okolia nejakej frekvencie, do nájdených výsledných hodnôt sa to nepremietne. Riešenie spočíva v použití väčšieho počtu pásiem vo VST efekte a vo vyhľadávacom algoritme.

Tu sa dostávame k ďalšiemu problému, a tým je to, že fourierova transformácia pracuje s frekvenciami lineárne, kďežto človek ich vníma logaritmicky. Znamená to nasledovné : v počuteľnom pásme 20 Hz až 20 kHz človek vníma „stred“ tohto pásma v okolí frekvencie 1 kHz. V ideálnom prípade by to malo znamenať, že vo fourierovom rozklade je počet prvkov, ktoré reprezentujú frekvencie nižšie od tohto pásma v pomere 1:1 k počtu prvkov, ktoré reprezentujú frekvencie vyššie od tohto pásma. Keďže však fourierova transformácia pracuje s frekvenciami lineárne (teda počet prvkov reprezentujúcich frekvencie od 1 do 2 kHz je rovný počtu prvkov reprezentujúcich frekvencie od 3 do 4 kHz), je tento pomer 1:19. Stred postupnosti prvkov, ktoré reprezentujú frekvencie spadá na frekvenciu 10 kHz, čo je frekvencia dosť blízko hornej hranici počuteľnosti (viď. obrázok 10). Z toho vyplýva, že počet prvkov reprezetujúcich spodné frekvenčné pásmo je nedostatočný a, naopak, počet prvkov reprezentujúcich horné frekvenčné pásmo je prebytočne veľký. Kôli presnému spracovaniu nízkych frekvencií je preto nútné obrovsky navyšovať celkový počet prvkov.

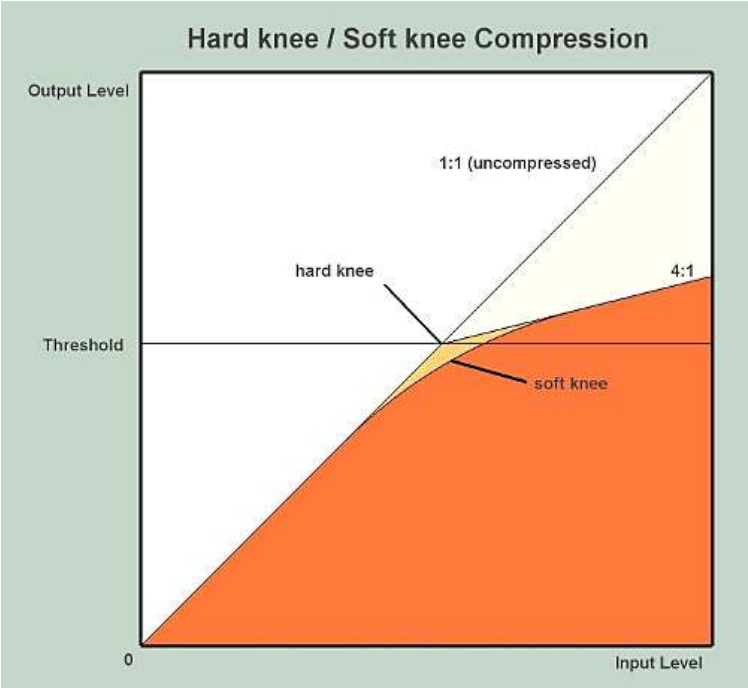


Obrázok 13: Znázornené ľudské vnímanie zvukového spektra.

4.3 Compressor

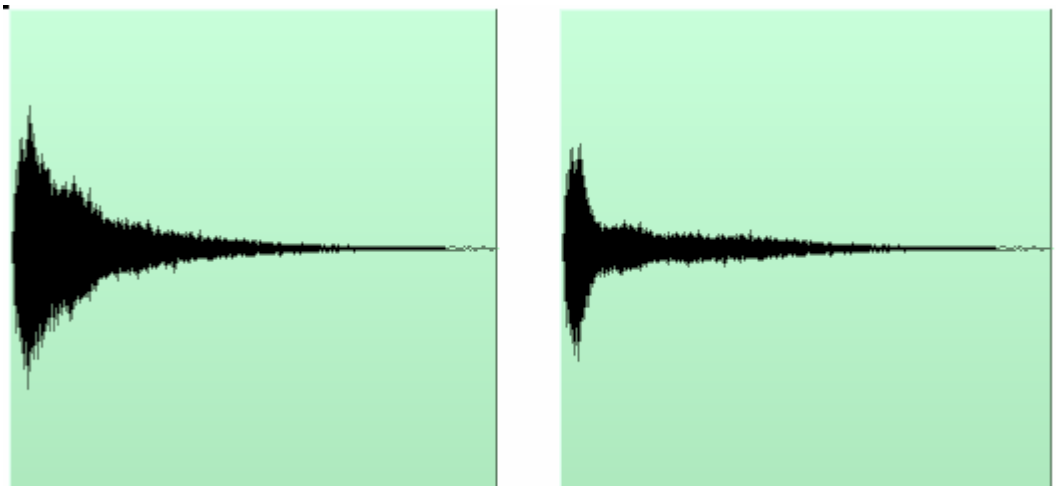
Výsledky pre typ efektu compressor sú neuspokojivé. Úspešnosť tohto výpočtu je silno závislá na type kompresoru použitého pri vytváraní hľadaného signálu. Je to z toho dôvodu, že každý plugin má vlastný algoritmus na výpočet komprimácie, ktorý často obsahuje rôzne funkcie pre vylepšenie výsledného zvuku.

Je to napríklad funkcia „soft knee“, ktorá problematizuje výpočet kompresného pomeru. Soft knee znamená, že hodnota treshold, nad ktorou je celý signál komprimovaný hodnotou kompresného pomeru nie je presne určená, ale tento kompresný pomer narastá postupne v závislosti na intenzite vstupného signálu a hodnota treshold určuje iba orientačné umiestnenie hranice.



Obrázok 14: Porovnanie kompresných kriviek soft knee a bežného (hard knee) kompresoru pri kompresnom pomere 4:1. Os X znázorňuje úroveň signálu na vstupe, os Y úroveň signálu na výstupe. Zdroj: [5]

Ďalší problém je v tom, že bežne používané hodnoty attack time sú rádovo v milisekundách a teda po prekročení hranice treshold chvíľu trvá, kým sa spustí komprimácia. Napríklad, ak je nastavený vysoký kompresný pomer, určitý treshold, ale vysoký attack time, všetky hodnoty vyššie ako treshold ostanú neskomprimované pokiaľ neuplynie attack time (viď. obrázok 11). Tým sa komplikuje výpočet hodnoty treshold - hranice, ktorej prekročenie spustilo kompresor.



Obrázok 15: Znázornený priebeh signálu pred a po spracovaní kompresorom s nastaveným dlhým attack časom.

Vo všeobecnosti platí, že čím väčší kompresný pomer bol použitý pri vytváraní hľadaného signálu, tým je výpočet parametrov úspešnejší.

Riešením tohto problému by mohlo byť použitie algoritmu prechádzania všetkými kombináciami nastavenia efektu s použitím špeciálnej metriky, ktorá by zohľadňovala úpravy na signále vykonené kompresorom.

4.4 Limiter

V prípade tohto typu efektu sú výsledky uspokojivé. Dôvody úspešnosti tohto výpočtu sú jasné vtedy, keď si uvedomíme, že limiter je vpodstate kompresor s kompresným pomerom nastaveným na „nekonečno“ a s časmi attack a release čo najnižšími. Z textu venovanému zhodnoteniu kompresoru vyplýva, že negatívne vplyvy na úspešnosť výpočtu sú týmto minimalizované.

4.5 Pitch Shifter

Výsledky pre tento typ efektu sú dobré. Problémami sú časová náročnosť, ktorá vyplýva z časovej náročnosti výpočtov v efekte BPitchShifter, a občasné predčasné orezanie vyhľadávacieho stromu skôr ako mohla byť nájdené najlepšia hodnota, a tým pádom je ako výsledok výpočtu interpretovaná chybá hodnota. Toto sa deje hlavne pri signáloch, ktoré sú „frekvenčne plné“, tzn. celé frekvenčné spektrum majú na približne rovnakej úrovni (extrémnym prípadom je biely šum). Ale nie je to pravidlo a vždy to záleží na konkrétnom signále. Riešenie spočíva v použití výpočtu bez orezávania.

5 Záver

Cieľom bakalárskej práce bolo implementovať riešenie problému replikácie nastavenia efektu podľa zadaného čistého a hľadaného signálu. Bola implementovaná sada samotných efektov, algoritmy pre hľadanie parametrov týchto efektov a rôzne metriky slúžiace na porovnávanie dvoch signálov.

Možné zlepšenie programu vidím hlavne v celkovom zlepšení práce s externým VST efektom zadaným používateľom. Ďalšie možnosti sú v zlepšení časovej náročnosti niektorých algoritmov výpočtov parametrov, v ďalšom vylepšovaní porovnávacích metrík a v pridávaní typov efektov, pre ktoré je možné aplikáciu použiť.

Ako záverečné zistenie môžem uviesť fakt, že počítačovo sa priblížiť k spôsobu ľudského porovnávania dvoch signálov je veľmi náročná úloha. Dva signály, ktoré by sa človeku javili veľmi podobné, ba priam také isté, môžu byť z matematického hľadiska veľmi odlišné. Ako príklad môžem uviesť bežné komerčné nahrávky populárnej hudby. Na konci produkčnej fázy je vždy úprava limiterom z dôvodu zvýšenia celkovej úrovne signálu. Poslucháč, ktorý nie je skúsený majster zvuku rozdiel medzi signálom pred a po úprave limiterom nespozná, no pri bližšom skúmaní signálu môžme zistiť, že jeho priebeh sa veľmi zmenil.

6 Referencie

[1] http://www.steinberg.net/en/company/3rd_party_developer.html

[2] http://www.dspdimension.com/admin/dft-a-pied/

[3] http://www.dspdimension.com/admin/pitch-shifting-using-the-ft/

[4] http://www.behardware.com

[5] http://www.jiscdigitalmedia.ac.uk/

[6] http://www.dspguide.com/

[7] http://www.harmonycentral.com

[8] http://en.wikipedia.org/wiki/Audio_filter

[9] http://www.dsptutor.freeuk.com

[10] http://cnx.org/content/m11757/latest/

[11] http://en.wikipedia.org/wiki/Acoustic_fingerprint

7 Príloha

7.1 Obsah CD

K práci je priložené CD s jej digitálnou verziou. CD ďalej obsahuje projekt aplikácie vypracovaný v Microsoft Visual C++ 2008 Express Edition, skompilovanú verziu programu, vygenerovanú dokumentáciu a testovacie vstupné zvukové súbory.

/bakalarska_praca	: text bakalarskej práce vo formáte pdf
/doc	: vygenerovaná dokumentácia
/projekt	: projekt pre Microsoft Visual C++ 2008 Express Edition
/skompilovany	: skompilovana verzia programu ReSounder a VST efektov
/test_subory	: testovacie zvukove subory

7.1.1 Testovacie zvukové súbory

Ako testovacie zvuky sú použité tri signály. Prvý s názvom *„cymbal“* je jeden úder na činel, ďalší s názvom *„guitar“* je pár sekúnd vybrnkávanej gitary a tretí s názvom *„song“* je pár sekúnd plne sprodukovanej piesne.

Kým prvé dva signály sú nijak nespracované priame nahrávky jedného nástroja, tretí signál je pieseň s plným frekvenčným zastúpením a limitovanou dynamikou.

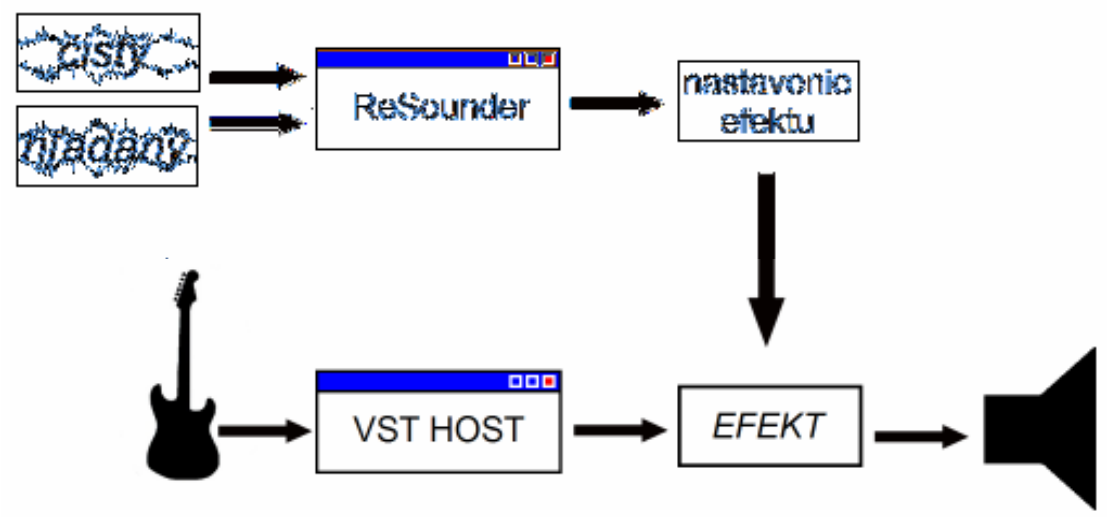
V názve každého súboru je uvedený typ efektu a hodnoty parametrov, s ktorými bol upravený.

7.2 Používateľská dokumentácia

V kapitole je uvedená používateľská dokumentácia programu ReSounder.

7.2.1 K čomu aplikácia slúži

Aplikácia ReSounder slúži na nájdenie nastavení efektu podľa používateľom zadaných dvoch zvukových súborov. Prvý súbor je originálny čistý zvuk pred aplikovaním efektu a druhý súbor je hľadaný zvuk po spracovaní efektom. Typické použitie aplikácie je v prípade, že používateľ potrebuje dosiahnuť nastavenie efektu zhodné s nastavením, ktoré bolo použité pri vytváraní hľadaného zvuku, no sám nie je schopný toto nastavenie dosiahnuť. Kôli samotnej práci s efektom je nutné paralelne používať VST hostiteľskú aplikáciu. VST efekty používané programom ReSounder sa nachádzajú v jeho domovskom adresári ako dll súbory. Spôsob ich použitia vo VST hostiteľskej aplikácii by mal byť uvedený v jej dokumentácii.



Obrázok 16: Graficky znázornené typické použitie aplikácie ReSounder.

7.2.2 Rozvrhnutie okna

Okno aplikácie ReSounder pozostáva z troch hlavných častí. Prvú časť tvoria ovládacie prvky pre výber zvukových súborov („*Select clean*“, „ *Select wanted*“), typu efektu („*select effect type*“) a spustenie výpočtu („*Proces*“*s*). Druhú časť tvorí skupina s názvom *Parameters to process*, ktorá obsahuje zoznam parametrov efektu podľa zvoleného typu efektu. Tretiu sekciu tvorí konzola, ktorá vypisuje informácie o stave programu a výsledky výpočtov.

7.2.3 Postup práce

V prvok kroku je potrebné pomocou tlačidla „*Select clean“* zvoliť zvukový súbor s čistým signálom a pomocou tlačidla „*Select wanted“* zvoliť zvukový súbor s hľadaným signálom. Po ich zvolení sa plná cesta k súborom vypíše veďla príslušných tlačidiel. Program podporuje formát .wav so 16-bitovou hĺbkou. Je nutné, aby oba súbory mali rovnakú vzorkovaciu frekvenciu.

Ďalej je potrebné pomocou kombo boxu s textom „*select effect type“* vybrať typ efektu, ktorým bol hľadaný signál upravený. Po zvolení typu efektu sa v sekcii „*Parameters to process“* zobrazia dostupné parametre efektu. Je možné zvoliť len určité parametre, ktorých hodnoty sa budú programom vyhladávať. Nezvolené parametre ostanú na základných hodnotách. Pri niektorých typoch efektu je možnosť použiť rýchlejšiu metódu výpočtu zaškrtnutím checkboxu „*Fast method“*, ktorý povolí použitie orezávania pri výpočtoch.

Pri zvolení typu *„General“* sa zobrazia nové ovládacie prvky. Pomocou tohto typu je možné hľadať nastavenia ľubovoľného efektu, táto možnosť je však iba v testovacom štádiu. Cestu k efektu je potrebné zadať pomocou tlačitla „*Select effect“*. Ďalej je potrebné v sekcii *„Parameters to process“* zaškrtnúť checkboxy v stĺpci „*Vol“* pri tých parametroch, ktoré menia hlasitosť signálu a prípadne nastaviť požadovanú percentuálnu veľkosť kroku pri nastavovaní týchto parametrov v stĺpci *„Step“*. (defaultne je nastavená na 10%).

Ďalší krok je spustenie samotného výpočtu stlačením tlačidla „*Process“*. Na progress bare je možné sledovať stav výpočtu. Po skončení výpočtu sa nájdené hodnoty vypíšu do konzoly a objaví sa tlačidlo „*Save“*, pomocou ktorého je možné hodnoty nájdených parametrov uložiť do .fxp súboru.

Formát výpisu výsledku do konzoly je nasledovný :

-názov parametru- = -hodnota- -jednotky- (-pozícia posuvníka-);

Pozíciu posuvíka predstavujú desatinné čisla z intervalu 0 až 1, kde 0 znamená pozíciu úplne vľavo a 1 pozíciu úplne vpravo.

V prípade voľby zápisu do fxp súboru stlačením tlačidla „*Save“* sa zobrazí dialógové okno požadujúce zvolenie umiestnenia a názvu fxp súboru.

Po skončení výpočtu je možné pozmeniť nastavenia programu a spustiť výpočet znovu.

7.2.4 VST efekty

Aplikácia ReSounder využíva sadu vlastných VST efektov, ktoré sa nachádzajú ako dll súbory v jej domovskom adresári.

7.2.4.1 BDelay

Efekt BDelay implementuje efekt typu delay. Tento typ funguje tak, že ku priamemu signálu primiešava signál oneskorený. Efekt BDelay obsahuje tri parametre:

delay : veľkosť oneskorenia (0 až 1000 milisekúnd)
wet : zmena hlasitosti oneskoreného signálu (-oo až 0 dB)
dry : zmena hlasitosti neoneskoreného signálu (-oo až 0 dB)

7.2.4.2 BEqualizer

Efekt BEqualizer implementuje efekt typu equalizer. Tento typ funguje tak, že mení hlasitosť frekvenčných pásiem nezávisle na sebe. Efekt BEqualizer je 3-pásmový, tzn. je možné nastaviť hlasitosť troch frekvenčných pásiem. Navyše je možné upraviť aj celkovú hlasitosť výstupného signálu. Parametre efektu:

Bass : hlasitosť spodného frekvenčného pásma (-12 až +12 dB)
Middle: hlasitosť stredného frekvenčného pásma (-12 až +12 dB)
Treble : hlasitosť vysokého frekvenčného pásma (-12 až +12 dB)
Output : celková hlasitosť (-12 až +12 dB)

7.2.4.3 BCompressor

Efekt BCompressor implemetuje dynamické efekty kompresor a limiter. Typ efektu kompresor funguje tak, že ak vstupný signál prekročí určenú hranicu treshold, po uplynutí doby attack time tento signál zoslabí v kompresnom pomere ratio. Zoslabenie trvá dobu release time. Limiter funguje podobne ako kompresor, rozdiel je v tom, že kompresný pomer je pevne nastavený na 0 a časy attack time a release time nezohrávajú úlohu, resp. sú pevne nastavené na najmenšiu možnú hodnotu.

Inými slovami, nad hranicu treshold neprepustí nič. Efekt BCompressor je v jednom momente možné použiť len ako kompresor, resp. len ako limiter. Obsahuje parametre:

Input :	úroveň vstupného signálu (-13 až +13 dB). Tento parameter môže používateľ použiť v prípade, keď signál je príliš silný, resp. príliš slabý na to, aby mohol byť účinne spracovaný
Comp/Lim :	prepína funkciu efektu medzi kompresorom a limiterom
Attack :	čas attack pri použití kompresoru (0,1 až 1000 milisekúnd)
Release :	čas release pri použití kompresoru (0,1 až 1000 milisekúnd)
Ratio :	kompresný pomer pri použití kompresoru (1:1 až 1:10)
Treshold :	úroveň treshold pri použití kompresoru aj limiteru(-oo až 0 dB)
MakeUp :	vypínač automatického nastavovania výstupnej úrovne signál
Output :	manuálne nastavenie výstupnej úrovne signálu pokiaľ je funkcia MakeUp vypnutá

7.2.4.4 BPitchShifter

Efekt BPitchShifter implementuje typ efektu pitch shifter. Funkciou tohto typu efektu je frekvenčný posun signálu, tzn., že mení celkovú výšku zvuku (z hudobného hľadiska). Efekt BPitchShifter obsahuje jeden parameter :

Shift :	nastavenie veľkosti posunu (-12 až +12 poltónov)