České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Jakub Berka**

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: **Realizace polyfonního syntezátoru ve standardu VST**

Pokyny pro vypracování:

Seznamte se základy technologie VST a základními principy modulární zvukové syntézy. Navrhněte jednoduchou architekturu polyfonního syntezátoru a syntezátor realizujte v softwarovém standardu Virtual Studio Technology (VST). Syntezátor nechť má obálku hlasitosti a obálku filtrů. Syntezátor bude pracovat i s informací o tempu, poskytovanou hostujícím programem. Syntezátor bude plně integrovatelný do běžných nástrojů pro tvorbu hudby, např. REAPER, včetně grafického uživatelského rozhraní. Rozsah zadání průběžně konzultujte se zadavatelem. Realizovaný syntezátor podrobte testu použiteslnosti a vyjednejte s vybraným hudebníkem, že syntezátor použije v produkci některé ze svých skladeb.

Seznam odborné literatury:

Rick Snoman: Dance Music Manual, Focal Press, 2008.

Vedoucí: Ing. Adam Sporka, Ph.D.

Platnost zadání: do konce zimního semestru 2017/2018

L.S.

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

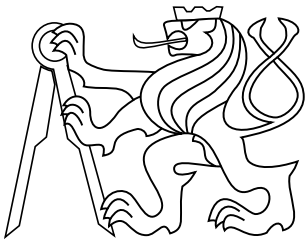
prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 29. 2. 2016

Bakalářská práce

Realizace polyfonního syntezátoru ve standardu VST

Jakub Berka



2016

Vedoucí práce: Ing. Adam Sporka, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra počítačové grafiky a interakce

Poděkování

Děkuji vedoucímu práce Ing. Adamu Sporkovi za odbornou pomoc a vstřícnost. V neposlední řadě také děkuji své rodině a blízkým přátelům za podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 24. května 2016 .....

Abstrakt

Předmětem bakalářské práce je návrh a následná realizace polyfonního syntezátoru pomocí Virtual Studio Technology (VST). V úvodní části je provedena rešerše nutného teoretického základu, jsou nastíněny jednotlivé metody zvukové syntézy a podrobněji rozebrána subtraktivní metoda, dále je vysvětleno číslicové komunikační rozhraní pro ovládání syntezátorů (MIDI) a princip technologie VST. V návrhové části práce jsou aplikovány tyto principy na navržení architektury polyfonního syntezátoru, je dbáno na možnost vytvoření široké škály zvuků. Syntezátor je implementován pomocí jazyka C++, je zajištěna jeho integrovatelnost do běžně používaných nástrojů pro tvorbu hudby. Součástí implementace je použitelné grafické uživatelské rozhraní. Výsledek práce je otestován oslovenými hudebníky, kteří implementovaný syntezátor použili při produkci vlastní hudební skladby.

Klíčová slova

VST, modulární syntéza, polyfonie, syntezátor, DAW

Abstract

This thesis focuses on design and implementation of polyphonic synthesizer using Virtual Studio Technology (VST). Research of necessary theoretical basics was made in first part of this document. Methods of sound synthesis are mentioned and the subtractive method is described in detail. Also Musical Instruments Digital Interface and Virtual Studio Technology are explained. These principles are applied for developing the architecture of polyphonic synthesizer. Synthesizer is implemented using C++ programming language, it can be integrated into most common digital audio worksations. Part of the implementation is useable grafical user interface. The result will be tested by selected musician, who will use it to produce own musical composition.

Keywords

VST, modular synthesis, polyphony, synthesizer, DAW

Obsah

Zkratky

Zkratky použité v tomto dokumentu:

VST	Virtual Studio Technology
DAW	Digital Audio Workstation
ADSR	Atack Decay Sustain Release neboli náběh, útlum, podržení, uvolnění
OSC	Oscilátor
LFO	Low Frequency Oscillator, nízkofrekvenční oscilátor
MIDI	Musical Instruments Digital Interface neboli česky číslicové rozhraní hudebních nástrojů
GUI	Grafical User Interface, grafické uživatelské rozhraní
SDK	Software Development Kit, sada vývojových nástrojů
API	Application Programming Interface, rozhraní pro programování aplikací

1 Úvod

Technologie VST, a hlavně zásuvné moduly (plugins) vyvinuté pomocí tohoto standardu, jsou v současném světě hudební produkce hojně používané. Na trhu jich je možné najít obrovské množství, od volně dostupných pluginů až po profesionální nástroje a efekty, které jsou používány v nahrávacích studiích. Nutno poznamenat, že i některé freeware zásuvné moduly jsou velmi kvalitní a použitelné v profesionální produkci, zejména vzhledem k ceně jejich analogových předloh. Např. společnost MOOG má ve svém katalogu řadu fyzických modelů analogových syntezátorů s pořizovací cenou v řádu stovek tisíců Kč.

VST plugin nabízí zajímavou cenovou i tvůrčí alternativu, neposkytuje nám skutečný kus nástroje, ale software, který může jeho zvukové vlastnosti věrně simulovat, např. firma Arturia se zabývá vývojem simulací syntezátorů právě od firmy MOOG, pořizovací cena takového pluginu je v jednotkách tisíců korun.

Jinou motivací, proč vytvářet takovýto plugin, může být vývoj unikátních hudebních nástrojů založených na standardu MIDI, které nevyžadují žádné úsilí při vývoji hardware.

1.1 Cíl práce

Cíl této bakalářské práce se skládá z více částí:

- Seznámení se základy technologií VST a MIDI a základními principy modulární zvukové syntézy.
- Návrh polyfonního syntezátoru s využitím principů z první části.
- Implementace syntezátoru v programovacím jazyku C++ s využitím frameworku WDL-OL/IPlug
- Výsledek z implementace podrobit testování za pomoci hudebníků, kteří syntezátor použijí při produkci některé svojí skladby.

Implementační část by měla splňovat, aby byl syntezátor plně integrovatelný do běžně používaných nástrojů pro tvorbu hudby (např. REAPER), součástí implementace je také vytvoření srozumitelného grafického uživatelského rozhraní viz. Obrázek 8.

2 Zvuková syntéza

K vytváření zvuků využívají syntezátory více metod, které se liší náročností a kvalitou výsledného zvuku. Předem je nutné říci, že existuje více kritérií a rovin, přes které se jednotlivé metody zvukové syntézy dají členit. Pro zjednodušení jsou metody syntézy následovně rozčleněny, bylo čerpáno z [1].

2.1 Metody zvukové syntézy

1. **Aditivní syntéza** - „*tato metoda vychází ze součtu signálů v časové i frekvenční oblasti. Jestliže funkce F(t) a G(t) reprezentují dva zvukové signály, tak F(t) + G(t) je novým signálem kvantitativně i kvalitativně odlišným.*“
2. **Subtraktivní syntéza** - „*tato metoda vychází z rozdílu frekvenčních charakteristik komplexního signálu a filtru. Jedná s v podstatě o analytický postup potlačování či zdůrazňování frekvenčních složek daného signálu filtrací.*“
3. **Modulační syntéza** - tato metoda vychází z ovlivňování daného parametru zvukového signálu jiným signálem. Máme více typů modulací signálu, k základním typům patří modulace amplitudová, frekvenční a pulzně šířková. „*V případě dvou funkcí F(t) a G(t) odpovídá výsledný signál funkci F[G(t)].*“
4. **Tvarová syntéza** nebo také *waveshaping synthesis* - „*tato metoda vychází buď z přímého působení na signál v jeho časové oblasti nebo z realizace požadovaného tvaru jeho časového průběhu.*“

2.1.1 Subtraktivní syntéza

Subtraktivní syntéza, nebo také rozdílová je založená na **řízené filtraci komplexního signálu**, tato metoda patří pro svoji jednoduchost a zvukovou účinnost k velmi rozšířeným způsobům syntézy. Oproti aditivní syntéze je tento typ syntézy zvuku přímočařejší. Zjednodušeně řečeno, protože výsledný signál je dán omezením harmonických složek nějakým modifikátorem, nejčasteji filtrem. Do tohoto postavení se dostala díky důsledným využitím napěťového řízení funkce elektrických obvodů, které jsou spojovány převážně se jménem R.A.Moog, autora prvního komerčně využitelného syntezátoru. [1]

Syntezátory využívající tuto metodu syntézy, mají svojí architekturu navrženou dle myšlenky zdroje a modifikátorů.

Zdroj

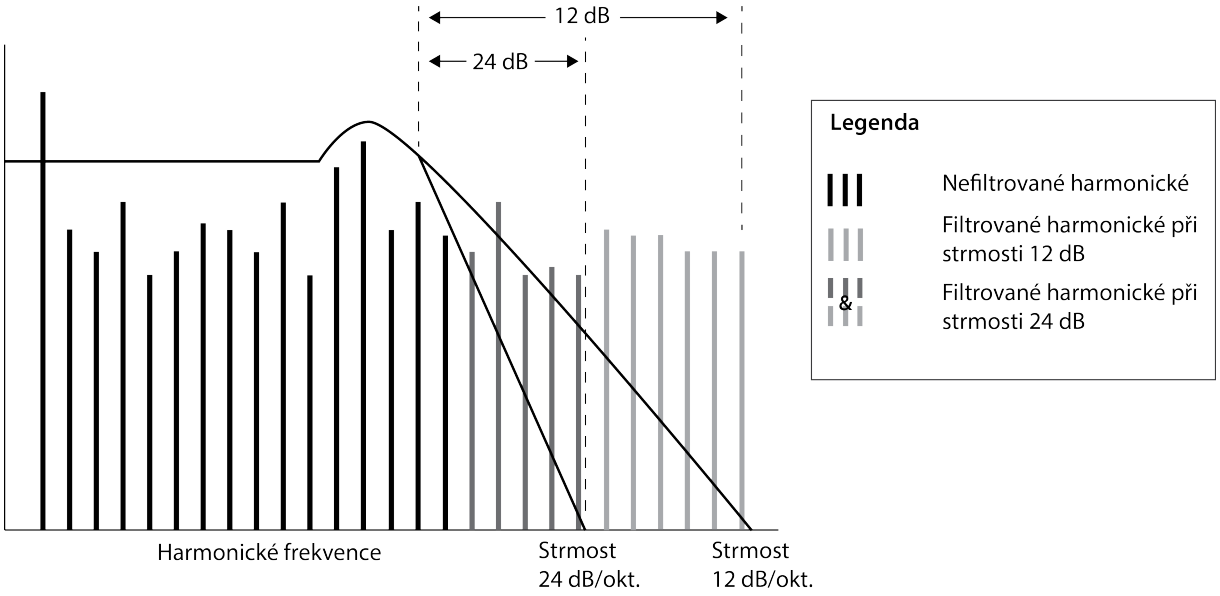
Na začátku je napětím řízený **oscilátor**, což je modul, který generuje signál pokud možno bohatý na strukturu harmonických. Především to jsou signály typu obdélník, pila a trojúhelník, které mají různé harmonické spektrum.

Oscilátory bývají také vybaveny ovladači pro ladění výšky výsledného tónu, jedním pro nastavování o půltóny (semitones) a druhým pro jemnější doladění o tzv. centy.

Častokrát mají analogové syntezátory dva a někdy i více oscilátorů a jsou tak schopné generovat komplexnější typy signálů. Běžně se také v syntéze používá generátor bílého šumu, který vytváří náhodný signál s rovnoměrnou energií skrz celé spektrum. Generátor šumu je užitečný při generování perkusivních zvuků nebo pro simulování zvuků větru, dechu apod.

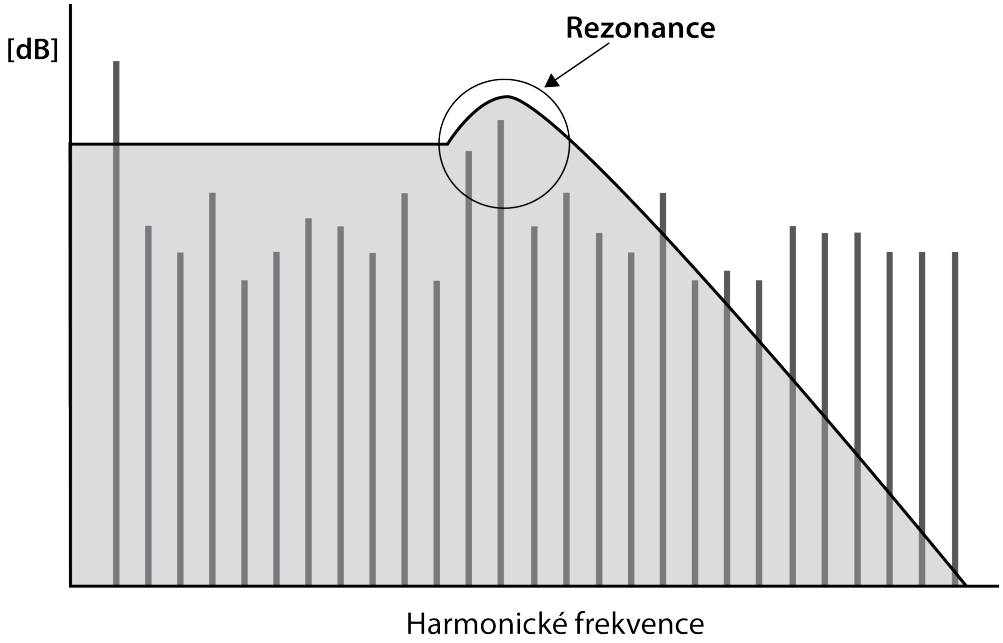
Modifikátory

Z oscilátoru nebo šumového generátoru je signál veden do napěťově řízeného filtru. **Filtr** je velmi důležitým prvkem subtraktivní syntézy, je to jediný modul, který dokáže změnit harmonické spektrum signálu vygenerovaného oscilátorem. Většinou funguje jako dolní, horní nebo pásmová propust, propouštějící frekvenční složky do určité zvolené mezní frekvence, v případě pásmové propusti frekvenčního rozsahu. Filtr může být kontrolován také obálkou viz. Obrázek 3. Někdy se můžeme na syntezátorech setkat i s nastavením strmosti filtru, která se udává v utlumení o decibely na jednu oktávu, rozdíl viz. Obrázek 1.



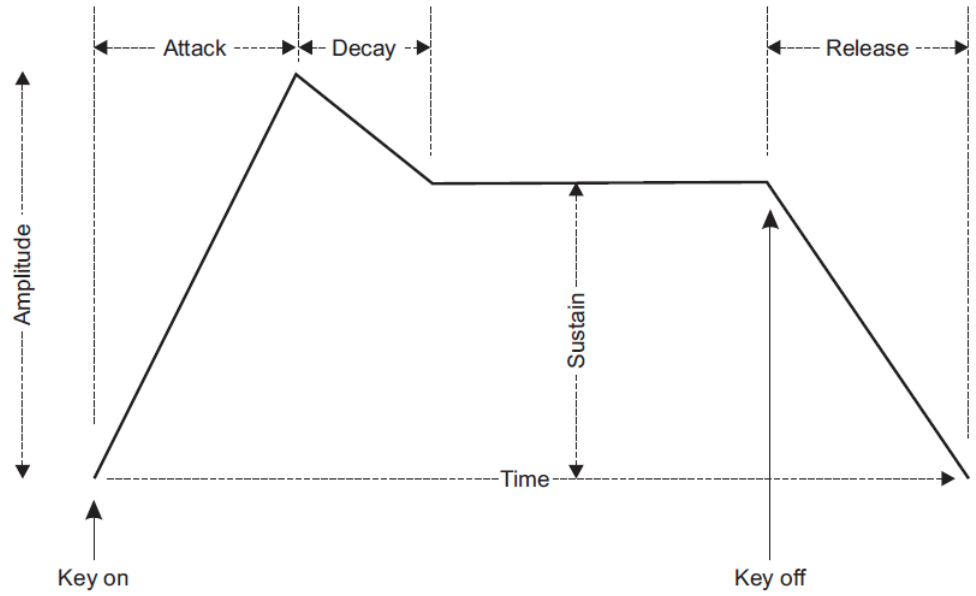
Obrázek 1 Rozdíl v nastavení strmosti filtru, dle: [2]

Častým využitím filtru je metoda resonanční syntézy. V této metodě se využívá náchylnosti filtru k rozkmitání na naladěné mezní frekvenci, poté stačí vnější impulz ke generaci více či méně tlumených kmitů. Na syntezátorech je u filtru umístěn ovladač rezonance, ten určuje, jak velké množství výstupu z filtrů se pošle zpětnou vazbou do jeho vstupu. Tím je možno ovládat zesílení kolem mezní frekvence (označováno též jako *peak*), viz. Obrázek 2. Pokud je ovladač nastaven na vysokou hodnotu, může být dosaženo zmíněného rozkmitání self-oscilace) filtru, tím je vyprodukována sinusová vlna o stejné frekvenci, jako je mezní frekvence filtru. Tuto vlastnost používají hudebníci k vytvoření hlubokých a silných basových zvuků, převážně vhodných pro styl *drum 'n' bass*. [2]



Obrázek 2 Jak se projevuje resonance u filtru, dle: [2]

Zvuk, vygenerovaný oscilátorem a ořízntutý filtrem, nemusí po celou dobu znít stejně hlasitě (tak, jak je to i u reálných nástrojů). Používá se tedy **amplitudová obálka**, která je zodpovědná za generování průběhu amplitudy signálu v čase. Většinou má 4 parametry – Attack, Decay, Sustain, Release, viz. Obrázek 3, parametrů může být samozřejmě více či méně v závislosti na architektuře syntezátoru.



Obrázek 3 Obálka se stavy ADSR, dle: [2]

- Časový úsek od stisknutí klávesy, které vyvolá MIDI zprávu Note On, po dosáhnutí nejvyšší úrovně obálky, se nazývá *Attack*(náběh).

- Čas mezi nejvyšší úrovní a hodnotou dalšího stavu obálky, na kterou úroveň poklesne, se nazývá *Decay*(pokles).
- Stálá úroveň obálky, která je udržovaná po dobu stisknutí klávesy, se nazývá *Sustain*(podržení).
- Po puštění klávesy se obálka dostane do stavu, kdy signál klesá na nulovou hodnotu, čas za který se to stane, se nazývá *Release*(uvolnění).

Nízkofrekvenční oscilátor (LFO) patří nezbytně k modulačnímu konceptu syntézy zvuku, používá se ale i v subtraktivní syntéze jako modulátor, uplatnění má i v jiných metodách syntézy.

Podobá se základnímu oscilátoru, ale kmitá pouze na frekvencích v rozsahu přibližně od 0,1 do 30 Hz. Pomocí jeho výstupu je možné modulovat jiné parametry syntezátoru, např. mezní frekvenci filtru, nebo výšku zvuku atd.

Oblíbeným hudebním efektem, který poskytuje právě LFO je vibrato, což je pravidelná změna frekvence signálu. Pro ilustraci, LFO se nastaví na relativně vysokou frekvenci, např. 5 Hz a jeho výstupem se moduluje výška oscilátoru (neboli také *pitch-modulation*). Výška tónu oscilátoru poté klesá a stoupá dle frekvence a tvaru vlny LFO, výsledný zvukový efekt je podobný policejní siréně. [2]

2.2 Digitální syntéza

Digitální syntéza znamená zpracovávat a vytvářet audio signál výhradně digitální cestou. Znamená to, že signál není spojitý jako u analogové syntézy, ale je nahrazen posloupností diskrétních vzorků.

2.2.1 Vzorkování

Vzorkování je proces, kterým je ze spojitého signálu získána posloupnost diskrétních vzorků. Tyto vzorky jsou odebírány s určitou periodou vzorkování T, její převrácená hodnota je vzorkovací frekvence (*sample rate*) f_s.

Shannonův teorém

Shannonův teorém se zabývá problematikou při zvolení vzorkovací frekvence. Maximální frekvence signálu, kterou je ještě možné zaznamenat při použití dané vzorkovací frekvence, někdy též označovaná jako Nyquistova frekvence je:

$$f_{max} = f_s/2 \tag{1}$$

Při vzorkovací frekvenci blízké f_{max} nebude již však věrně zaznamenáno množství energie na této frekvenci.

Frekvenční spektrum při vzorkování signálu se opakuje periodicky, s frekvencí f_s. Pokud neplatí, že:

$$f_s >= 2f_{max} \tag{2}$$

Poté dojde k překrytí spekter a jejich deformaci a není možné z něj získat původní signál, tento jev nazýváme *aliasing*.

3 MIDI

Jelikož realizovaný syntezátor bude využívat MIDI zprávy, je nutné seznámit se s jeho základními principy.

MIDI je zkratka pro Musical Instruments Digital Interface neboli česky číslicové rozhraní hudebních nástrojů.

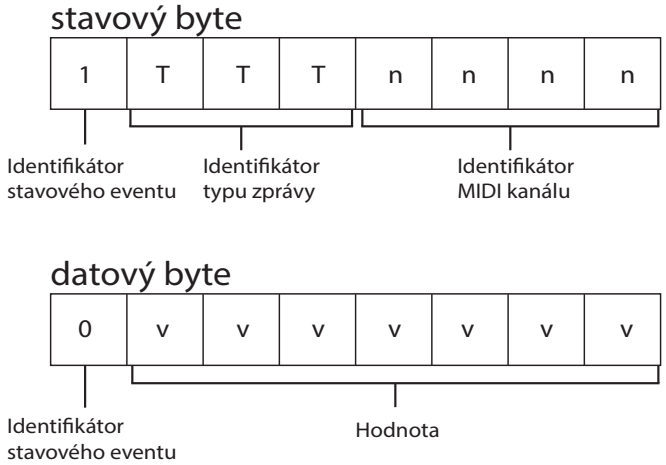
3.1 Historie MIDI

„V roce 1981 se na NAMM Show setkali Tom Oberheim, Ikutaro Kakehashim (president firmy Roland), Dave Smith a Chet Wood a vymysleli první návrh možného univerzálního rozhraní, které se mělo jmenovat USI (Universal Synthesizer Interface). Při představení tohoto návrhu v Japonsku byli přítomny také firmy Yamaha, Korg a Kawai. Následně se ale ukázalo, že americké firmy by raději daly přednost rychlejším systémům. Avšak v říjnu 1982 byla překvapivě dokončena specifikace protokolu MIDI a o měsíc později byl vybaven první syntezátor Prophet 600 protokolem MIDI. V lednu 1983 byly poprvé propojeny dva MIDI nástroje různých výrobců Prophet 600 a Roland JP-6.“ [3]

3.2 Princip MIDI

MIDI je systém fungující na principu posílání zpráv, vždy je nějaký odesílatel a příjemce zprávy. MIDI zprávy jsou navrženy tak, aby obsahovaly nejvíce informací v co možná nejkratší zprávě.

MIDI zpráva se skládá z jednoho stavového a dvou datových MIDI bytů.



Obrázek 4 struktura stavového a datového bytu MIDI zprávy

MIDI specifikace obsahuje několik základních typů zpráv (nebo také příkazů), těch je několik typů, nás budou nejvíce zajímat následující.

3.2.1 Note On

Zprávu Note On vyvolá stisknutí klávesy na MIDI kontroleru. Její identifikátor typu zprávy je 1 a dále přenáší dva datové byty, jeden nese informaci o čísle noty, která byla stisknuta a druhý byte informaci o rychlosti s jakou byla klávesa stisknuta (velocity).

Status Byte		**Data Byte 1**	**Data Byte 2**
1001	0000	0011 1100	0111 1111
Note	Channel	Note Number	Note Velocity
On	0	60 (C4)	127

Tabulka 1 Příklady MIDI zprávy Note On

3.2.2 Note Off

Její identifikátor typu zprávy je 0 a dále přenáší, stejně jako Note On zpráva, informaci o číslu uvolněné klávesy a intenzitu uvolnění klávesy (release-velocity). Zprávu Note Off lze zapsat také jako Note On, akorát s nastavenou velocity na 0. V MIDI standardu je 128 not, jsou číslované od 0 do 127. Nota C4 má číslo 60. Velocity neboli rychlost úderu má v MIDI také rozsah od 0 do 127, kde 0 znamená vypnuto a 127 fortissimo forte (hudební výraz znamenající nejsilněji, značí se fff).

Status Byte		**Data Byte 1**	**Data Byte 2**
1000	0000	0011 1100	0111 1111
Note	Channel	Note Number	Release Velocity
Off	0	60 (C4)	127

Tabulka 2 Příklady MIDI zprávy Note Off

3.2.3 Pitch Bending

MIDI zprávu pitch bend ovládá kolečko, které je standardně umístěno na MIDI ovladači (*controller*), potažmo syntezátoru, na levé straně od klávesnice. Má fixovanou středovou polohu, do které se po puštění samočinně vrátí. Jeho běžná funkce je modulace výšky tónu.

Podle MIDI specifikace tato zpráva obsahuje dva datové byty pro vyšší rozlišení přenášené hodnoty, z důvodu, že jemné změny výšky tónu jsou v horních polohách velmi dobře slyšet, bohužel ne všechny syntezátory a pluginy oba datové byty opravdu využívají.

3.2.4 Modulation Wheel

Modulation Wheel neboli modulační kolečko bývá umístěno vedle kolečka pro pitch bend, někdy se lze setkat i se spojením obou ovladačů do jednoho v podobě joysticku, jehož vertikální pohyb ovládá modulaci a horizontální výšku tónu.

Modulační kolečko nemá fixaci polohy, každý pohyb tímto kolečkem vyvolá MIDI zprávu jejíž hodnota je v rozmezí 0 až 127, klasické použití kolečka je ovládání amplitudy nízko frekvenčního oscilátoru, někdy označovaného také jako vibrato. Záleží ale

pouze na implementaci, modulovat lze např. mezní frekvenci filtru nebo jiné parametry syntezátoru.

3.2.5 Sustain Pedal

Někdy také označovaný jako damper pedal nebo také sustain switch, je v podstatě spínač který má dvě polohy - On a Off. Data byte se přenáší ve zprávě jen jeden, pro polohu Off jsou to hodnoty 0-63, 64-127 pro polohu On.

Jeho funkcí je zábránit zpracovat zprávu Note Off pro ty noty, které byly spuštěny až po sešlápnutí pedálu. Poté co je pedál vyšlápnut, přeruší se tyto noty, ale pouze v případě, že na ně již byla zavolána zpráva Note Off, noty které jsou stále fyzicky drženy, budou znít po vyšlápnutí pedálu dále. Podobně jako u modulačního kolečka i sustain pedal je možné naprogramovat na ovládání jiných parametrů syntezátoru. [3]

4 Technologie VST

Virtual Studio Technology neboli VST byla vyvinuta firmou Steinberg v roce 1996 jako otevřený standard, který by,l a i v nových verzích je, dostupný vývojářům, ze stránek firmy Steinberg je možné zadarmo stáhnou vývojářský balíček, tzv. SDK. Ten obsahuje soubor tříd napsaných v jazyce C++.

VST je softwarové rozhraní pro komunikaci mezi hostitelským programem označovaným jako *VST host*, často nějaký software pro tvorbu hudby neboli *digital audio workstation* - DAW (např.: Ableton Live, Cubase, REAPER), a zásuvnými moduly - *VST plugins*. [4]

VST plugins slouží buďto ke generování digitálního audio signálu, poté se nazývají *VST instruments* (VSTi), k úpravě již existujícího audio signálu, což jsou *VST effects*, typicky delay, reverb apod., třetí kategorií jsou moduly, které řídí činnost jiných VST pluginů nebo jiných fyzických nástrojů - *VST controllers*. Často také dochází ke kombinaci těchto kategorií, to znamená, že VSTi může obsahovat nějaké VST efekty.

4.1 Verze VST

- VST 1.0 rok 1996
- VST 2.0 rok 1999, přidaná podpora pro zpracování MIDI
- VST 2.4 rok 2006, podpora 64bit softwarů, podpora Mac s procesory Intel
- VST 3.0 rok 2008, dynamicky se měnící počet vstupů a výstupů
- VST 3.5 rok 2011, technologie „Note expression“ – umožňuje nastavování parametrů pro každou jednotlivou notu v polyfonním souzvuku
- VST 3.6 rok 2013 – podpora pro operační systém iOS [4]

4.2 Princip VST pluginu

VST plugin sám o sobě není aplikace, potřebuje již zmíněnou hostující aplikaci, která se stará o stream audio dat a využívá procesy s těmito daty, které poskytují právě VST pluginy. Z pohledu hostitelské aplikace je VST plugin tzv. black box s libovolným počtem vstupů, výstupů a parametrů.

Obecně řečeno, si VST plugin převezme stream audio dat (v případe VSTi si stream audio dat vytváří přímo plugin, nemá tedy žádný vstup, pouze výstup), aplikuje na ně algoritmus a pošle výsledek v podobě streamu dat zpět hostující aplikaci.

Stream dat je rozdělen na sérii bloků, hostující aplikace poskytuje bloky v sekvenci a také kontroluje velikost bloků. Hostující aplikace si neuchovává žádnou informaci o tom, co VST plugin s blokem dat udělal. [5]

4.3 Vývoj VST pluginu

V současné době je několik možností vývoje, lze použít tzv. high-level přístup a některý z programů podporující vizuální programování (např. Synthedit, Synthmaker - oba pro Windows) a výstupem bude také VST plugin. Nebo se při vývoji využije low-level přístupu a tedy je nutné napsat zdrojový kód VST pluginu. Výhod tohoto přístupu je více, ale mezi největší výhody patří jistá volnost při vývoji. Je možné vyvinout plugin, který bude podporovat různé platformy, a který zkompiluje vytvořený kód do všech běžných formátů (VST2 , VST3, AudioUnit, Real-Time AudioSuite). Možnost je také využívat podpory většiny API.

Dalším způsobem, jak vyvinout VST plugin, a to je také způsob, který byl zvolen v této práci, použití aplikačního rámce *framework*, který v mnoha věcech usnadnuje práci. *Frameworks* je k dispozici více (JUCE, VST.NET, jVSTwRapper, IPlug). Pro implementaci syntezátoru byl zvolen multiplatformní open-source C++ framework WDL-OL/IPlug, což je rozšíření frameworku WDL/IPlug, který je zaměřen na vývoj audio zásuvných modulů a jejich GUI ve formě VST2, VST3, AudioUnit RTAS a AAX API. Také dokáže vytvořit samostatné audio/MIDI aplikace pro Windows/OSX a pro IOS zařízení od firmy Apple. To vše z jednoho kódu. Samotné WDL obsahuje další knihovny třetích stran (např. JNetLib, LibPNG, GIFLib, JPEGLib, zlib). [6]

5 Návrh

Následující kapitola se zabývá návrhem syntezátoru jako celku a podrobněji se věnuje jednotlivým blokům navržené architektury.

5.1 Specifikace vlastností syntezátoru

Realizovaný syntezátor není simulací konkrétního hardware analogového syntezátoru, ale velkou měrou na ně odkazuje. Jsou to například syntezátory od firem MOOG, Korg a Arturia, konkrétně *MOOG MINITAUR, MINIMOOG VOYAGER, Korg MS-20 a Arturia Microbrute*.

Syntezátor by měl mít následující parametry:

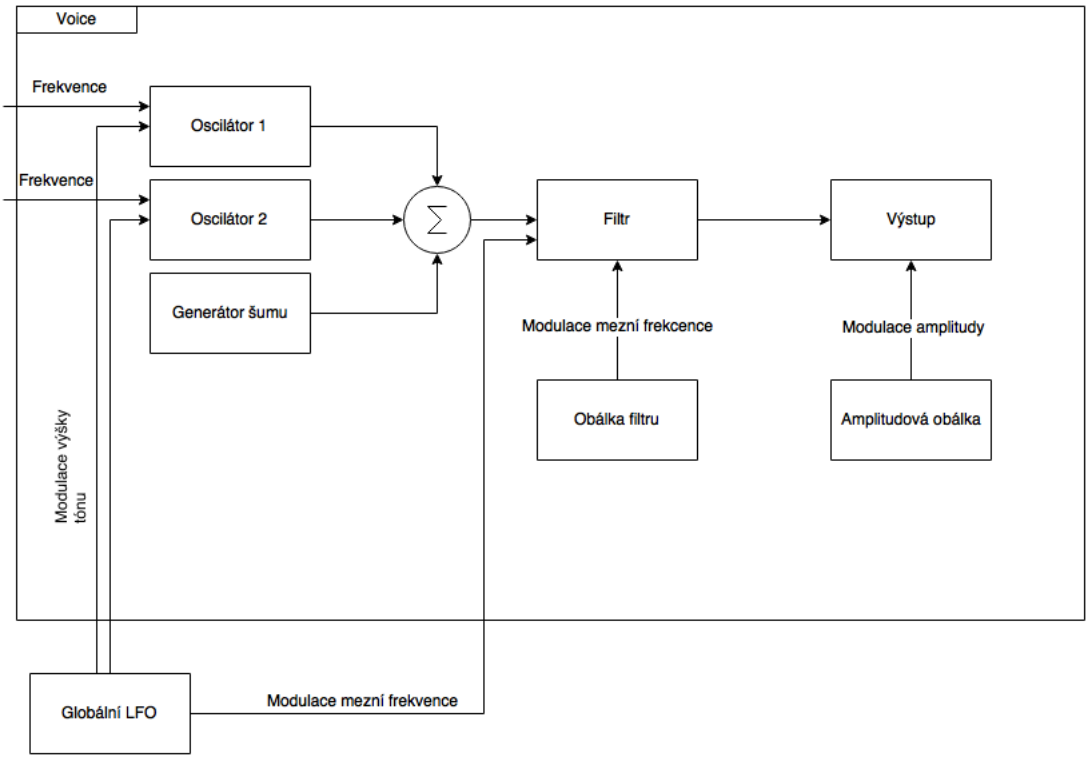
- 2 oscilátory s možností nastavení 4 typů vln (sinus, pila, čtverec, trojúhelník) a hrubým nebo jemným laděním (půltóny, centy)
- 1 generátor šumu
- Možnost mixu těchto tří komponent
- 3 typy filtrů (horní, dolní a pásmová propust) s útlumem 24 dB na oktávu
- 2 generátory obálky typu ADSR – jedna pro modulaci amplitudy a druhá pro modulaci filtru
- 1 nízkofrekvenční oscilátor s možností nastavení 4 typů vln(sinus, pila, čtverec, trojúhelník), kterým je možné modulovat výšku tónu oscilátorů a filtr.
- Nízkofrekvenční oscilátor bude možné synchronizovat s tempem hostujícího programu.
- 32-hlasá polyfonie, možnost rozladění všech hlasů
- Podpora MIDI komunikace, stisk/uvolnění klávesy, modulační kolečko, pitch-bend, sustain pedál.
- Digitální efekt delay(zpoždění) se zpětnou vazbou.

5.2 Návrh architektury syntezátoru

Realizovaný syntezátor je založený na modulárním principu konstrukce analogových syntezátorů. Používá se z modulů požívaných v subtraktivní resp. modulační syntéze. Tato architektura má velikou výhodu v tom, že je jednoduše rozšiřitelná. Jednotlivé moduly tohoto pluginu by měly také být znovupoužitelné.

5.2.1 Polyfonie

Jednotlivé moduly budou analogické k součástkám reálného syntezátoru. Realizovaný syntezátor má být dle zadání polyfonní. Pro slovo polyfonie je uveden ve slovníku význam vícehlas. V případě realizovaného syntezátoru to znamená, že bude schopný zahrát více tónů naráz. Základním blokem architektury je tedy hlas (*Voice*), který obsahuje další moduly potřebné pro vytvoření komplexního zvuku a průběhu tónu. Blokové schéma hlasu viz. Obrázek 5.



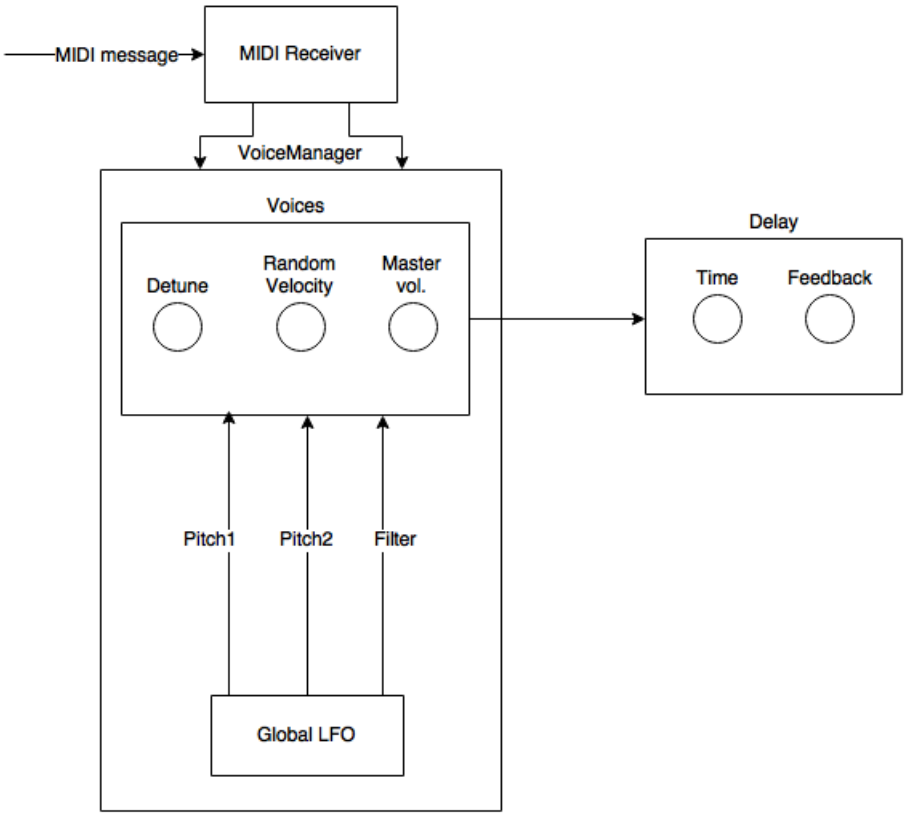
Obrázek 5 Blokové schéma jednoho hlasu syntezátoru

Jak je vidět ze schématu, viz. Obrázek 5, základními moduly jsou dva oscilátory a jeden generátor šumu. Hlavním vstupem pro oscilátory je frekvence a druhým vstupem je její modulace. Modulátor LFO (což je v podstatě stejný modul jako oscilátor) bude pro všechny hlasy syntezátoru jeden, je tedy na schématu označen jako globální. V grafickém rozhraní bude možné zvolit u oscilátoru tvar vlny (sinus, pila, čtverec, trojúhelník), výstup z oscilátorů a generátoru šumu směřuje do filtru. Filtru lze pomocí GUI nastavit jeho typ (horní, dolní nebo pásmová propust), mezní frekvenci, která je také modulovaná pomocí globálního LFO, nebo obálkou a resonanci. Možná není úplně jasné, proč není filtr také globální, když z GUI je nastavena jeho mezní frekvence a rezonance pro všechny hlasy stejně, důvod je takový, že mezní frekvenci je možné modulovat pomocí obálky, pro každý hlas bude tedy mezní frekvence ve stejný okamžik jiná. Výstup z filtru je následně modulován amplitudovou obálkou. Všechny parametry obálek jsou také nastavitelné pomocí GUI.

5.3 Správa hlasů

O zpracování MIDI zpráv se stará objekt nazvaný *MIDI Receiver*, viz. Obrázek 6, ten zachycuje zprávy, které nás zajímají - *Note On/Off, Pitch Bend, Modulation Wheel, Sustain Pedal*, každá tato odchycená zpráva je předána správci hlasů(*Voice Manageru*), to je velmi důležitý prvek celé architektury polyfonního syntezátoru. Voice Manager v sobě bude obsahovat pole 32 hlasů, to znamená, že je možné docílit 32-hlasové polyfonie a globální LFO.

Úkolem Voice Manageru je zpracovávat všechny signály získané od MIDI Receiveru, když přijme MIDI zprávu o tom, že byla stisknuta klávesa, bude pro ni muset najít



Obrázek 6 Blokové schéma syntezátoru

volný hlas a pokud již žádný volný nebude (všech 32 hlasů z nějakého důvod zní) k dispozici, bude se muset zahájit tzv. *voice-stealling* algoritmus a najít nějaký vhodný hlas, který bude moci úplně a rychle ztišit a hlas by se měl následně rozeznít s tou správnou frekvencí pro danou notu a amplitudou, obdobně když obdržíme zprávu, že byla uvolněna klávesa, musí se Voice Manager postarat o to, aby přestal hrát hlas se správným číslem noty. O tom více v implementační části, viz. kapitola 6.7.

Pomocí Voice Manageru je možné z GUI nastavit rozladění hlasů, tímto parametrem získá syntezátor určitý charakter, který je důležitý k tomu, aby se za prvé podobal svým reálným protějškům a za druhé byl pokud možno unikátní. Druhým možným parametrem je randomizace rychlosti úderu na klávesu, jeho nastavení si hudebník určí, jak moc se bude lišit změněná rychlost úderu od té skutečné (přičtením/odečtením náhodného čísla), tím je dodáno syntezátoru více či méně nepředvídatelné chování, které může být hudebníkem vyžadováno, tuto vlastnost je samozřejmě možné úplně vypnout. Posledním parametrem je nastavení celkové hlasitosti - *Master vol.*.

5.4 Delay

Výstup syntezátoru ještě prochází efektem zpoždění - *Delay*, který zpozdí signál o určitý časový úsek. Tento efekt lze úplně vypnout a procházející signál se nijak nezpozdí. Z GUI je možné nastavit čas zpoždění a úroveň zpětné vazby, která vrací již zpožděný signál zpět do vstupu efektu a zpoždění se zopakuje.

6 Implementace

Pro implementaci VST syntezátoru je použit multiplatformní open-source C++ framework WDL-OL/IPlug, což je rozšíření frameworku WDL/IPlug, který je zaměřen na vývoj audio pluginů a jejich GUI ve formě VST2, VST3, AudioUnit RTAS a AAX API. Také dokáže vytvořit samostatné audio/midi aplikace pro Windows/OSX a pro IOS zařízení od firmy Apple. A to vše z jednoho kódu. Samotné WDL obsahuje další knihovny třetích stran (JNetLib, LibPNG, GIFLib, JPEGLib, zlib), které usnadňují práci např. s implementací grafického uživatelského rozhraní.

Všechny moduly, které byly popsány v kapitole 5.2, bude reprezentovat jedna třída. Třídy oscilátoru, filtru a obálky budou pokud možno co nejvíce nezávislé na ostatních a tudíž, co možná nejlépe znovupoužitelné.

6.1 Struktura projektu

V hlavní třídě pluginu, která se jmenuje *MicroBery.cpp* je implementováno GUI a základní funkčnosti pluginu. Důležité pro ní jsou hlavičkové soubory *resource.h* a *MicroBery.h*. V hlavičkovém souboru *resource.h* se definují důležité konstanty pro plugin – jméno, verze, typ pluginu, audio vstup, mono/stereo výstup, latence pluginu, podpora MIDI atd. Poté také konstanty pro GUI - velikost okna, cesty k obrázkům pro GUI a jejich unikátní ID. Všeobecně VST pluginy běží na dvou vláknech, jedno je určené pro grafické rozhraní a druhé pro digitální zpracování signálu. Použitý framework se o takzvanou *vláknovou bezpečnost* postará za programátora.

6.1.1 Hlavní třída projektu

Hlavičkový soubor *MicroBery.h* definuje hlavní třídu pluginu *MicroBery.cpp*, ta dědí třídu *IPlug*, která je součástí frameworku, na kterém je tento plugin realizován. Jsou zde deklarovány důležité metody pluginu: *Reset, OnParamChange, ProcessDoubleReplacing, ProcessMidiMsg*, které jsou popsány níže a přepisují metody definované v *IPlugBase* třídě. Dále jsou vytvořeny instance *VoiceManagement* a *MIDIReceiver*. V konstruktoru třídy je implementace GUI, detailněji popsáno v kapitole o implementaci GUI 6.9.

6.1.2 ProcessDoubleReplacing

Nejdůležitější metoda pluginu. Je volaná hostujícím programem pluginu (VST host, DAW, atd.) pokaždé, když je potřeba zpracovat blok vzorků. Předávají se jí ukazatelé na buffery s double hodnotami vzorků, pro vstup i výstup a počet vzorků. V tomto případě se plní dva buffery pro výstup, pravý a levý kanál (je možné mít i více kanálů).

V metodě probíhá for-cyklus přes počet vzorků, v každém kroku for-cyklu, to znamená pro každý vzorek, zachytává *MIDIReceiver* nové MIDI zprávy, zpracovává je

a předává hlasům pomocí třídy *VoiceManagement*, poté se zavolá hlavní metoda VoiceManagementu - *nextSample* pro vygenerování hodnoty vzorku všech hlasů a také efektu delay, tento vzorek je ještě vynásoben hodnotou pro výslednou hlasitost - *Master volume*. A předán do obou bufferů pro levý i pravý kanál.

6.1.3 OnParamChange

Funkce *OnParamChange* je volána, je-li editovaný nějaký parametr pomocí GUI (otočení potenciometru), nebo také například pomocí automatizace z VST hostující aplikace. Zavolá se také, když se změní přednastavený program (preset) – těch je implementováno celkem pět, viz. kapitola 6.9.

6.1.4 Reset

Funkce *Reset* je volána pokaždé, když se změní vzorkovací frekvence. Pro všechny instance objektů v pluginu se pomocí setteru nastaví nová vzorkovací frekvence. Vzorkovací frekvence se může změnit v nastavení VST hostu. Pomocí metody *GetSampleRate* získáme její aktuální hodnotu přímo z hostovací aplikace. V metodě je *switch*, který pozná podle *id*, u kterého potenciometru byla změněna hodnota a je nastavena jeho nová hodnota.

6.1.5 ProcessMidiMsg

Funkce *ProcessMidiMsg* je volána pokaždé, když je obdržena nějaká MIDI zpráva. V metodě se následně zavolá na instanci MIDIReceiveru metoda *midiMessageReceived*. MIDI zprávy jsou *přesné na vzorek* a jsou označeny časovým razítkem, které udává, jak moc se skutečný čas vytvoření MIDI zprávy liší od času vzorku ve kterém byla zpráva zachycena (tzv. *sample offset*). [7]

6.2 Implementace MIDI receiveru

Zpracování MIDI zpráv je realizováno v souborech *MIDIReceiver.h* a *MIDIReceiver.cpp*. Inspirace k MIDIRecieveru je čerpána z ukázkových příkladů, které jsou součástí použitého frameworku.

Syntezátor bude reagovat na MIDI zprávy *Note On, Note Off, Pitch Bend, Mod Wheel a Sustain Pedal*, které jsou popsány v kapitole 3.

Framework IPlug poskytuje připravené třídy, které ulehčují práci s MIDI zprávami. MIDIReceiver využívá třídu IMIDIQueue. Což je vlastně naimplementovaná speciální fronta pro MIDI zprávy. V třídě je tedy instance IMIDIQueue, která nám ukládá MIDI zprávy.

6.2.1 Zpracování MIDI zprávy

Kdykoli plugin syntezátoru zachytí nějakou MIDI zprávu, zavolá se metoda z hlavní třídy pluginu *ProcessMidiMsg*, ve které se následně na instanci MIDIReceiveru zavolá metoda *midiMessageReceived*, ta zachytí zprávy, které nás zajímají a následně se uloží do fronty IMIDIQueue.

V hlavní metodě pluginu – ProcessDoubleReplacing, se ve for-cyklu pro každý vzorek zavolá metoda MIDIReceiveru *updateValues*, která nastaví hodnoty podle právě stisknuté klávesy, nebo pozici modulačního kolečka nebo kolečka ohýbání tónu.

V MIDIReceiveru je použitá knihovna Signals [8], která implementuje návrhový vzor Observer. Pokaždé, když se odchytí požadovaná zpráva, vyšle se signál s konkrétními hodnotami (v případě sustain peadálu bez hodnoty) instanci VoiceManagementu, která zprávy dále zpracuje.

6.3 Implementace oscilátoru

Oscilátor je implementovaný v souborech Oscillator.cpp a Oscillator.h. Umí generovat čtyři základní typy vln, sinusoidu, pilu, čtverec a trojúhelník. Oscilátor si ukládá informaci o aktuální frekvenci, vzorkovací frekvenci a času, což je vlastně aktuální poloha vzorku v periodě vlny, v rozsahu 0 až 2π.

Parametry které mohou být pro oscilátor nastaveny z grafického rozhraní, viz. tabulka 3.

Nejdůležitější metodou třídy Oscillator je *nextSample*, kde se počítá hodnota aktuálního vzorku v čase, vzorek nabývá hodnoty od -1 do +1.

Parametr	**Rozsah**
Typ vlny	sinus, pila, čtverec, trojúhelník
Půltóny	-24, +24
Centy	-100, +100

Tabulka 3 Parametry oscilátoru

6.3.1 Algoritmus

Algoritmy použité pro výpočet všech typů vln jsou naivní, matematicky jsou správně a průběhy jsou vypočítány přesně, to je ale u digitální syntézy nežádoucí, protože například ostré skoky u čtvercové vlny z hodnoty +1.0 na -1.0 a naopak, způsobují tzv. *aliasing* ve vysokých frekvencí. Pokročilejší algoritmy, které se zabývají odstraňováním tohoto jevu, nebudou implementovány, každopádně je to jedna z oblastí, ve které by se dal syntezátor rozšířit.

K vypočítání frekvence je zapotřebí znát číslo noty, které je získáno z MIDI zprávy *Note On*. Rozsah definuje standard MIDI na celkem 128 hodnot (0-127), to jsou noty od C-2 až po G8. Klaviatury mohou mít různý rozsah (např. jenom dvě oktávy). Jedna oktáva obsahuje 12 kláves. Frekvenci vypočítáme od klávesy A4 (440Hz - komorní A) ta má MIDI číslo 69. Rovnice pro výpočet frekvence vypadá následovně, n udává číslo noty.

$$f_n = 440 * 2^{(n-69)/12} \tag{3}$$

Implementovaný oscilátor má možnost rozladit svojí frekvenci pomocí parametrů z GUI. Hudebník může oscilátor rozladit výšku tónu v půltónech, nebo jemněji v takzvaných centech(v kladných i záporných hodnotách). Rovnice pro výpočet frekvence bude tedy složitější. Na jednu oktávu připadá celkem 12 půltónů(*semitones*) a na jeden půltón 100 centů(*cents*). Výpočet frekvence: [9]

$$f = (f_n) * (2^{(semitones)/12}) * (2^{(cents)/1200}) \tag{4}$$

6.3.2 Nízkofrekvenční oscilátor - LFO

Implementace Low Frequency Oscilátoru neboli LFO využívá již naimplementovaný Oscilátor, takže LFO umí také generovat čtyři typy vln. Pouze je omezen jeho frekvenční rozsah, což je realizováno v GUI rozsahem potenciometru *Rate*.

Parametr	**Rozsah**
Typ vlny	sinus, pila, čtverec, trojúhelník
Frekvence	0,01 - 35 Hz
Synchronizace s tempem	On/Off
Synchronizovaná frekvence	1/16, 1/8, 1/4, 1/2, 2/1 (délka noty)
Key retrigger	On/Off
Modulace filtru	-1.0 - +1.0
Modulace oscilátoru 1	-1.0 - +1.0
Modulace oscilátoru 2	-1.0 - +1.0
Amplituda	0.0 - 1.0 (modulační kolečko)

Tabulka 4 Parametry LFO

LFO slouží jako modulátor a v této implementaci moduluje výšku tónu obou oscilátorů a mezní frekvenci filtru. Všechny modulační parametry mají nulovou polohu, tím lze modulační účinek LFO úplně vypnout, druhou možností je modulace všech tří parametrů zároveň, každý s libovolnou úrovní modulace.

Synchronizace s tempem

Je velmi vyžadovanou vlastností, aby nízkofrekvenční oscilátor byl synchronizovaný s tempem, které získá od hostitelské aplikace. Je dán ve formě BPM, neboli *Beats Per Minute*, pomocí příkazu *GetTempo()*.

Z grafického rozhraní lze nastavit pomocí *sync rate* paramteru frekvenci LFO pomocí délky not, viz. tabulka 4. Pro názornost, když hudebník nastaví délku noty na 1/16 (šestnáctiny) a bude mít zapnutý přepínač *sync*, frekvence LFO a tedy perioda jeho vlny, bude synchronizovaná s daným tempem a o délce šestnáctinové noty. Výpočet

frekcence je následující:

$$\begin{aligned} f &= BPM/60.0 * ((délkanoty)^{-1}/4) \\ f_{1/16} &= BPM/60.0 * 16/4 \\ f_{1/8} &= BPM/60.0 * 8/4 \\ f_{1/4} &= BPM/60.0 * 4/4 \\ f_{1/2} &= BPM/60.0 * 2/4 \\ f_{1/1} &= BPM/60.0 * 1/4 \\ f_{2/1} &= BPM/60.0 * 0.5/4 \end{aligned} \tag{5}$$

Key retrigger

Dalším parametrem, který jde nastavit LFO je tzv. *Key retrigger*, při jeho zapnutí se při každém stisknutí klávesy vynuluje fáze nízkofrekvenčního ocilátoru. Tímto je možné vytvářet tzv. synkopy (přízvuk na nepřízvučné době, nebo předražení doby).

6.3.3 Generátor šumu

Generátor šumu je v podstatě generátor náhodných čísel od -1.0 do +1.0. Využívající funkci *rand()* ze standardní knihovny C++ (*stdlib.h*).

$$\begin{aligned} noiseOutput &= ((double)rand()/(double)RANDMAX) \\ noiseOutput &= (noiseOutput * 2) - 1 \end{aligned} \tag{6}$$

6.4 Implementace filtru

Filtr je implementovaný v souborech Filter.h a Filter.cpp. Je to resonanční IIR filtr neboli filtr s nekonečnou impulzní odezvou. Implementovány jsou tři typy filtru:

Parametr	**Rozsah**
Typ filtru	dolní, horní a pásmová propust
Mezní frekvence	0.01, 0.99
Resonance	0.01, 1.00
Množství modulace obálkou	-1.0, +1.0

Tabulka 5 Parametry filtru

6.4.1 Algoritmus

Filtr je implementován podle algoritmu od Paula Kelletta[10]. Oproti jeho implementaci je ale tento filtr složen ze čtyř low-pass filtrů prvního řádu „zapojených“ v sérii. Jelikož jeden filtr prvního řádu nám podle tohoto algoritmu dává pokles signálu o 6dB na oktávu nad mezním kmitočtem. Čili pokles tohoto filtru je 24dB/okt.

Aby byl filtr IIR musí mít alespoň jednu zpětnou vazbu. V tomto případě závisí výpočet *b1, b2, b3* na svojí předchozí hodnotě.

```
b0 += modulatedCutoff * (inputSample - b0 + feedback * (b0 -
    b1));
b1 += modulatedCutoff * (b0 - b1);
b2 += modulatedCutoff * (b1 - b2);
b3 += modulatedCutoff * (b2 - b3);
switch(type){
        case LOWPASS:
                return b3;
        case HIGHPASS:
                return inputSample - b3;
        case BANDPASS:
                return b0 - b3;
        default:
                return 0.0;
}
```

Tento algoritmus je volaný na každý zpracovávaný vzorek (*inputSample*). *Modulated cutoff* je již hodnota modulované mezní frekvence buďto obálkou filtru nebo nízkofrekvenčním oscilátorem, potažmo oběma modulátory najednou.

Implementování resonance, což je v podstatě zesílení u mezní frekvence, je realizováno na prvním řádku kódu, vynásobením hodnoty pásmové propusti jistou hodnotou (*feedback*) a přičteme jí k signálu. *Feedback* je vypočten právě z resonance následujícím vztahem.

$$feedback = resonance + [resonance/(1.0 - mezníFrekvence)] \tag{7}$$

Algoritmus je původně navržen pouze jako low-pass filtr, ale během výpočtu jsou získány hodnoty i pro high-pass a band-pass. Pro horní propust jsou od sebe jednoduše odečteny vstupní hodnoty vzorku a nízké frekvence, tím získáme pouze ty vysoké. V případě pásmové propusti odečteme hodnotu low-passu od nejnižšího řádu filtru.

6.5 Implementace obálky

Obálka je implementovaná v souborech Envelope.h a Envelope.cpp. Syntezátor využívá exponenciální typ obálky ADSR, která určuje průběh signálu (zvuku) v čase. Zkratka ADSR znamená, že se obálka skládá ze 4 částí, nebo také stavů - *Attack, Decay, Sustain, Release*.

Obálka je implementovaná jako konečný stavový automat, ten v tomto případě bude mít 6 stavů (OFF, ATTACK, DECAY, SUSTAIN, RELEASE, FAST-FADEOUT) tím je zaručeno, že v jakémkoliv momentě bude probíhat jeden z těchto stavů. FAST-FADEOUT je speciální stav, který nastává pouze v případě, že byly vyčerpány všechny hlasy a jeden musí být umlčen, v tom případě se obálka přepne do tohoto stavu, který není nic jiného než velmi krátký stav RELEASE, který trvá 5 ms.

6.5.1 Přechody mezi stavy

Ze stavů ATTACK, DECAY a RELEASE se signál dostane sám po uplynutí nastaveného času pomocí grafického rozhraní.

Ve stavech OFF a SUSTAIN zůstává nekonečně dlouho, do té doby než se zavolá funkce *setState* (funkce pro změnu stavu), která je vyvolána buďto stisknutím/uvolněním klávesy nebo uvolněním sustain pedálu (pro notu, která již měla uvolněnou klávesu). To znamená, že obálka se může do stavu RELEASE dostat ze tří možných stavů (ATTACK, DECAY, SUSTAIN). Podobně je to u speciálního stavu FAST-FADEOUT, do tohoto stavu se může obálka dostat ze všech, kromě stavu OFF.

6.5.2 Algoritmus

Syntezátor využívá rychlý algoritmus pro exponenciální typ obálky od Christiana Schoenebecka[11], který nepoužívá naivní způsob počítání exponenciály pomocí exp(), pro každý vzorek, ale využívá numerickou metodu, pomocí log().

Nejdůležitější metoda v implementaci obálky je stejně jako u oscilátoru *nextSample*, která vrátí double hodnotu aktuálního vzorku. V případě, že je obálka ve stavu A, D nebo R a index aktuálního vzorku je stejný jako index vzorku z následujícího stavu, obálka metodou *setState* přejde do následujícího stavu a poté se spočítá hodnota vzorku.

6.5.3 Obálka filtru

Pro obálku filtru můžeme použít stejnou implementaci obálky, zmíněnou výše. Obálka filtru bude modulovat mezní frekvenci filtru(*cutoff*).

6.6 Implementace hlasu

Hlas je implementován v souborech *Voice.h* a *Voice.cpp*. Jak již bylo napsáno v návrhové části viz. kapitola 5.2.1, jeden hlas se skládá z instancí dvou oscilátorů, šumového generátoru, filtru a dvou obálek (jedna pro hlasitost a druhá pro filtr). Hlas má proměnnou *isPlaying*, která v sobě udržuje informaci, jestli je daný hlas aktivní - tedy hraje. Do nehrajícího stavu se hlas dostane pokaždé, když se jeho amplitudová obálka dostane do fáze OFF. Implementováno pomocí knihovny *Signals*, což je v podstatě návrhový vzor observer. Změnu z nehrajícího stavu na hrající zajišťuje správce hlasů, popsaný v kapitole 6.7.

Vše podstatné se děje v metodě *nextSample()*, jak název napovídá vše se znova odehrává na jednom vzorku. Pokud hlas není aktivní vrací tato metoda nulovou hodnotu, aniž by něco počítala.

Nejdříve musí být sečtena dohromady hodnota vzorků z dvou oscilátorů a šumového generátoru (*mix*). Z grafického rozhraní lze nastavit úroveň pro jednotlivé moduly, každý otočný potenciometr má rozsah 0-100%. Nastaví-li se pro každý modul hodnota 100%, musí být zajištěno, aby výsledná úroveň vzorku ze všech tří modulů nebyla součtem - tedy 300%, ale právě 100% (hodnota jednoho modulu bude 33,3%). Úroveň tedy musí být znormalizovaná - všechny jednotlivé úrovně jsou vyděleny součtem všech úrovní.

Než bude sečtený vzorek filtrován, musí být pro filtr vypočítána modulovaná mezní frekvence, k tomu je nutné určit součet aktuální hodnoty obálky filtru a hodnoty nízkofrekvenčního oscilátoru. Tato modulace je poté jednoduše předána instanci filtru pomocí setteru a ještě navíc hodnotu modulace výšky tónu (modulováno aktuální hodnotou LFO) pro oba oscilátory, podobným způsobem. Následně je filtrován vzorek vynásobený hodnotou amplitudové obálky a rychlostí úderu. Vyfiltrovaný vzorek je návratová hodnota této funkce.

6.7 Implementace správce hlasů

Správce hlasů, implementován v souborech *VoiceManagement.h* a *VoiceManagement.cpp*, spravuje pole 32 hlasů (viz. kapitola 6.6) a také obsahuje jednu instanci nízkofrekvenčního oscilátoru. Tato třída se stará o podstatnou část logiky polyfonního syntezátoru. Pomocí signálů získává od *MIDI receiveru* hodnoty z MIDI zpráv a dál s nimi nakládá. Stará se o rozladění hlasů a randomizaci rychlosti úderu, obsahuje *settery* pro nastavování parametrů všem modulům v syntezátoru, které se dají nastavovat pomocí GUI. Další podstatnou částí je algoritmus, který se spustí, když nejsou k dispozici žádné nehrající hlasy a hledá nejvhodnější hlas k umlčení.

6.7.1 Metoda whenNoteOn

Jak napovídá název, tato metoda se zavolá pokaždé, když plugin obdrží MIDI zprávu Note On. Jejími parametry jsou číslo noty a rychlost úderu - *velocity*.

Nejdříve je snaha o nalezení jednoho hlasu v poli hlasů, který není aktivní. Když se tak stane, přiřadí se mu parametry, pokud je potřeba, tak se rychlost úderu randomizuje. Obě jeho obálky (amplitudy a filtru) se přepnou do stavu *Attack* a je nastaveno, že tento hlas je již aktivní (a dále bude, dokud se jeho obálky nedostanou do stavu OFF). Každému hlasu se v tento okamžik začne počítat čas jeho života, to bude důležitý údaj který je používán při krádeži hlasu(*voice-stealing*).

Pokud se nenajde žádný neaktivní hlas, bude pomocí *voice-stealing algorimtu* nalezen ten nejvhodnější aktivní hlas pro použití na novou notu, kterému se nejdříve přepnou obě obálky do stavu FAST-FADOUT a následně přiřadí parametry nové noty a dále se postupuje stejně jakoby hlas aktivní nebyl.

6.7.2 Metoda whenNoteOff

Zavolá se pokaždé, když plugin obdrží MIDI zprávu Note Off. Parametry jsou číslo noty, která byla uvolněna a rychlost s jakou byla uvolněna. Metoda musí najít podle čísla noty hlas, který tuto konkrétní notu právě hraje a poté mu nastavit obě obálky do stavu RELEASE.

Pokaždé to ovšem není tak jednoduché, protože plugin implementuje také sustain pedál, který zabraňuje vypínání hlasů, když je sešlápnutý. I když je sešlápnutý sustain pedál a hlasy se nevypínají, je potřeba si zapamatovat že na konkrétním hlase byla zavolána metoda Note Off. Tyto hlasy poté vypíná vyšlápnutí sustain pedálu, jiné ne.

6.7.3 Voice-stealing algoritmus

Algoritmus se spustí v případě, že je stisknuta klávesa, ale nemáme k dispozici žádný nehrající hlas.

Nejdříve jsou všechny hrající(aktivní) hlasy umístěny do datové struktury *vector*, která se následně seřadí podle času, ten udává, jak dlouhou dobu hlas hraje. Tato struktura se následně prochází for-cyklem a hledá se požadovaný hlas. Strategie algoritmu vypadá následovně, seřazeno podle priority nejlepšího kandidáta na krádež hlasu sestupně.

- Krádež nejstaršího hlasu, který hraje stejnou notu, jako nově stisknutá klávesa
- Krádež nejstaršího hlasu, který má obálky ve stavu RELEASE
- Krádež nejstaršího hlasu

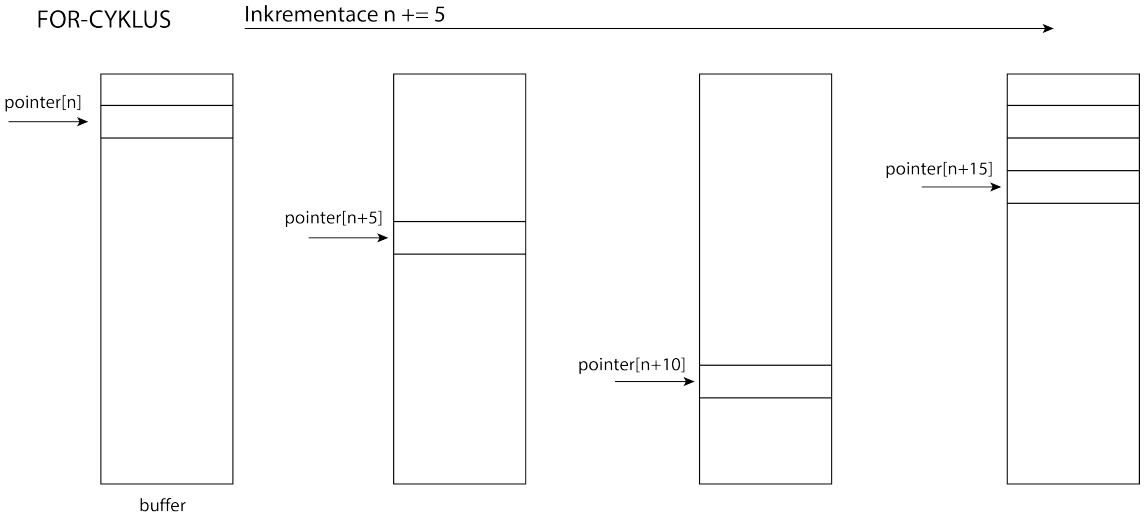
6.8 Implementace efektu Delay

Delay efekt je v implementaci tohoto pluginu tzv. *master effect*, protože pracuje s master výstupem syntezátoru, neboli výstupem ze všech hlasů, které právě zní.

Parametr	**Rozsah**
Čas zpoždění	0.0 - 1.0 s
Feedback	0.0 - 1.0

Tabulka 6 Parametry efektu delay

Implementace efektu zpoždění(*Delay*) je založena na principu kruhového bufferu, ten je velmi vhodný při zpracovávání audio signálů. Mějme buffer o délce l a ukazatel na index n, for-cyklus bude procházet buffer a v každém cyklu zvyšovat $n+=5$. Co se stane, když se narazí na konec a ukazatel ukazuje mimo buffer? U kruhového bufferu, na rozdíl od lineárního, je ukazatel automaticky přemístěn na začátek a se správným offsetem, viz. Obrázek 7.[12]



Obrázek 7 Kruhový buffer

Jednoduchý princip delay pracuje s dlouhým kruhovým bufferem, který ukládá vzorky. Vzorky vstupují do bufferu jedním koncem a druhým koncem vystupují ven zpožděny o délku bufferu. Delay implementuje také zpětnou vazbu(*feedback*), která umožňuje zopakovat, i vícekrát zpoždění. Když je nastavena na nulovou hodnotu, zazní pouze jedenkrát ozvěna(zpoždění), čím větší hodnota zpětné vazby, tím více opakování je možné slyšet.[12] Vztah pro zpoždění vzorku, kde D je délka bufferu:

$$y(n) = x(n - D) + feedback * y(n - D) \tag{8}$$

6.9 Implementace GUI

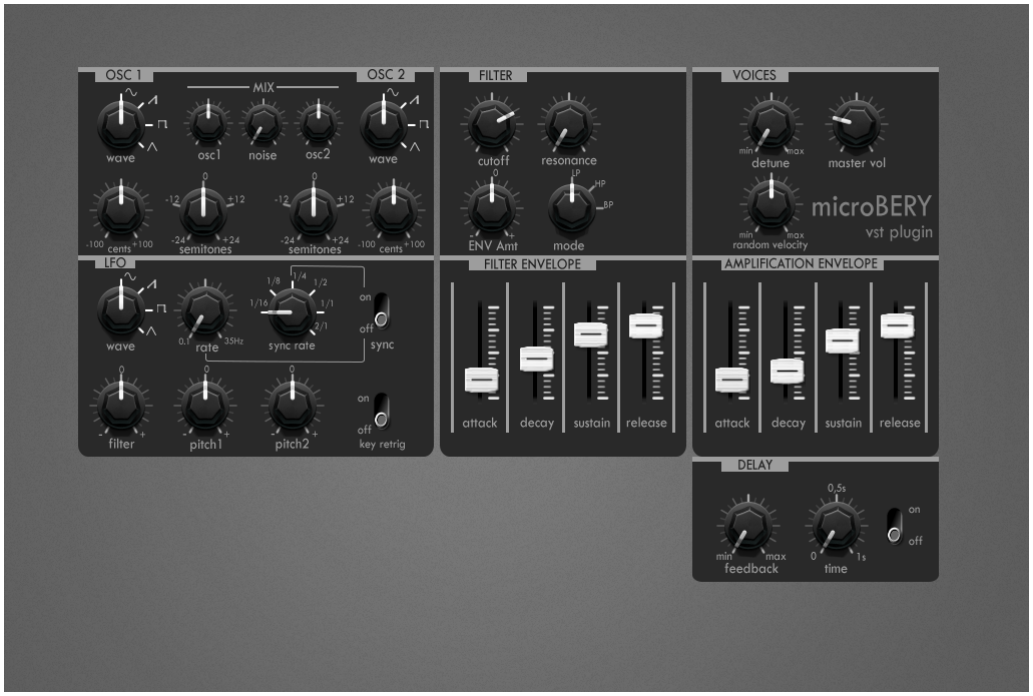
GUI je implementováno v hlavní třídě pluginu *MicroBery.cpp*. Konkrétně v konstruktoru a v metodě *OnParamChange*, kde se zachytává změna v GUI (otočení potenciometru apod.). Jako inspirace pro podobu GUI posloužil vzhled analogových syntezátorů Arturia MiniBrute a MicroBrute. Nejedná se ale o jejich úplnou kopii. Náhled grafického rozhraní viz Obrázek 8.

V hlavičkovém souboru *resource.h* se definují rozměry okna, cesty a ID pro zdrojové obrázky. V konstruktoru třídy MicroBery se vytvoří pozadí pomocí metody *AttachBackground* a následně se umisťují ovládací prvky na pozadí, přiřadí se jím jejich funkcionalita pomocí *AttachControl* – může se jednat o otočný/posuvný potenciometr, přepínač atd. Jednotlivým ovládacím prvkům se nastaví jejich jméno, počáteční hodnota, min. a max. hodnota a velikost kroku, o který se pootočí.

6.9.1 Přednastavené programy

Součástí realizovaného syntezátoru jsou přednastavené programy, neboli zvuky (*presets*).

- Deep bass - hluboký basový zvuk.
- Space bass - basový zvuk ve stylu *80s*.
- Atmo pad - atmosférická plocha.
- Hammond organ - napodobení typického zvuku varhan značky Hammond



Obrázek 8 Grafické rozhraní VST pluginu

6.9.2 Vytváření bitmap

K vytvoření bitmap otočných/posuvných potenciometrů a přepínačů, byl použit softwarový nástroj *JKnobMan ver 1.3.2*[1], který je převážně určený pro vytváření těchto grafických prvků pro VST zásuvné moduly.

Tento program umožňuje vytvořit sekvenci PNG obrázků, kde každý Obrázek v sekvenci představuje jeden krok otočení/posunutí potenciometru.

[1]K dispozici na: `http://www.g200kg.com/en/software/knobman.html`

7 Testování

Testování proběhlo tak, že realizovaný syntezátor byl zaslán vybraným hudebníkům, kteří měli syntezátor použít při produkci své vlastní skladby.

Způsob použití nebyl nijak omezen, syntezátor mohl hudebník použít do jakékoliv části skladby a pro vytvoření jakéhokoliv typu zvuku. Podle navržené architektury by měl syntezátor umět vytvářet širokou škálu zvuků, od perkusivních zvuků pro rytmickou linku skladby, přes basovou linku až po melodickou. Je tedy teoreticky možné vyprodukovat kompletní skladbu pouze pomocí několika instancí stejného syntezátoru s jinak nastavenými parametry.

Cílem testování je ověření praktické použitelnosti realizovaného syntezátoru, pokud možno také jeho flexibility a kvality jako softwarového hudebního nástroje.

7.1 Hudebník 1

unTIL BEN je producent elektronické hudby, vlastním jménem Ben Until, pocházející z francouzského města Bayonne. Jeho hudba je na pomezí žánrů *new wave* a moderní elektroniky, k tvorbě využívá hardwarové i softwarové syntezátory.

Má za sebou vydání debutového EP *Spacetronic music*, které obsahuje 8 skladeb. Více informací k dispozici na internetových stránkách hudebníka: `http://untilben.com/`.

Producent *unTIL BEN* vytvořil v rámci testování skladbu *Blurried*, všechny zvuky ve skladbě jsou vytvořené pomocí realizovaného syntezátoru, tedy i včetně bicích. Tím se ukázaly široké zvukové schopnosti syntezátoru.

Dvě tabulky 7 a 8 obsahují hodnoty nastavených parametrů pro všech 12 zvuků, použitých ve skladbě *Blurried*, skladba se nachází na přiloženém DVD.

7.1.1 Komentář hudebníka

Hudebník zhodnotil grafické rozhraní jako přehledné a jednoduché, plugin poskytuje široké možnosti nastavení zvuku. Setkal se se zvláštními zvukovými projevy při spuštění více instancí syntezátoru najednou a zároveň zapnutým efektem *Delay*, po jeho vypnutí vše v pořádku.

7.2 Hudebník 2

Vojtěch Nedvěd, vystupující také pod přezdívkou *Nooly*, v současné době pracuje jako zvukový designer na počítačové a konzolové hře Kingdom Come: Deliverence ve společnosti Warhorse Studios. V minulosti pracoval jako zvukový designér reklamních projektů, krátkých filmů a animací a vytvořil hudbu pro několik počítačových her.

Hudebník v rámci testování vytvořil krátkou skladbu obsahující čtyři typy zvuků, z nichž všechny byly vygenerovány za pomoci testovaného syntezátoru. Parametry jednotlivých zvuků této skladby jsou podrobně popsány v tabulce 9. Skladba se nachází na přiloženém DVD pod názvem *microbery_test_nooly.wav*.

7.2.1 Komentář hudebníka

Hudebník zhodnotil grafické rozhraní jako přehledné, překvapivé mu přišlo pouze řešení přepínacích tlačítek (volba tvaru vlnu u oscilátorů), kde se musí tlačítkem myši klikat a přepínač se otáčí, je zvyklý spíše držet tlačítko myši a hýbat s myší nahoru/dolu nebo doprava/doleva (tak jako fungují všechny ostatní otočné/posuvné potenciometry v GUI).

7.3 Hudebník 3

Adam Sporka se podílí na vývoji adaptivní hudby do připravované hry Kingdom Come: Deliverence od studia Warhorse. Je zakládající člen kapely *The Wasteland Wailers* a produkuje elektronickou hudbu napříč žánry. Jeho hudební portfolio je možné poslechnout na `https://soundcloud.com/adam_sporka`.

V rámci testování vytvořil skladbu, kde všechny použité zvuky jsou vytvořeny testovaným syntezátorem. Podrobně popsány jsou v tabulce 10. Skladba se nachází na přiloženém DVD pod názvem *goofy.ogg*.

7.4 Hudebník 4

Cebit je producent elektronické hudby a také tvůrce remixů. Pochází z řeckého města Larisa. Jeho tvorba jde napříč žánry (*techno, synthpop, italo-disco, hip-hop, drum'n'bass*). Jeho hudební portfolio je možné si poslechnout na: `https://soundcloud.com/cebit`.

Producent *Cebit* v rámci testování předělal svojí již existující skladbu *Ravenous Female*, testovaný syntezátor použil na tzv. *lead sound* v této skladbě. Skladba se nachází na přiloženém DVD pod názvem *Cebit - Ravenous Female (MicroBery Lead Version).wav*.

Tabulka 11 ukazuje nastavení parametrů syntezátoru, pro vytvoření tohoto zvuku.

7.4.1 Komentář hudebníka

Hudebník zaslal společně s vytvořenou skladbou také krátkou recenzi na realizovaný plugin:

"Microberry is a multipurpose virtual analog instrument for the VST Platform by Steinberg, created by Jakub Berka. It features two oscillators (plus a noise generator) with 4 types of waveforms each, 2 envelope generators for the amplifier and the filter section each, and a versatile LFO section. Moreover, it employs a minimal delay effect processor for swift experimentations.

The interface is monochromatic, and follows an intuitive design approach which lets the user dial patches effortlessly. All parameters of Microbery can be controlled via automation inside the DAW. Considering the digital nature of its implementation, the sound of Microbery is rich and clean, ideally suited for bass, lead, pad and SFX patches.

Pros: Good Sound, on par with hardware equivalents, quick and intuitive interface Cons: incomplete MIDI support"

7.5 Tabulky parametrů

Parametr/ Charakter zvuku	**Basový zvuk**	**Varhany**	**"Woosh"**	**"Bleep"**	**"Synth arp"**	**Plocha (pad)**
OSC1	50%	100%	0%	50%	100%	50%
OSC2	50%	90%	0%	75%	90%	50%
Noise	0%	0%	100%	0%	0%	0%
Vlna OSC1	Pila	Čtverec	Sinus	Pila	Čtverec	Čtverec
Vlna OSC2	Čtverec	Trojúhel.	Sinus	Čtverec	Trojúhel.	Čtverec
OSC1 centy	0	+10	0	+4	+10	0
OSC2 centy	0	-10	0	0	-10	0
OSC1 půltóny	-12	-12	0	0	-12	-12
OSC2 půltóny	+12	0	0	+12	0	0
OSC1 modulace	0,094	0,0	0,0	0,0	0,0	0,0
OSC2 modulace	0,0	0,08	0,0	0,0	0,08	0,0
Attack [s]	0,05	0,1	0,05	0,15	0,01	0,01
Decay [s]	0,3	11,5	0,55	0,95	11,83	3,15
Sustain	0,1	0,9	0,5	075	0.78	0,45
Release [s]	0,01	3,5	0,95	0,35	2.76	1,90
Typ Filtru	LP	HP	LP	LP	HP	LP
Filtr mezní frek.	0,25	0,38	0,65	0,42	0.51	0,51
Rezonance	0,01	0,459	0,74	0,76	0.01	0,44
Filter Attack [s]	3,53	0,01	0,01	0,01	0.01	0,47
Filter Decay [s]	1,53	1,53	1,53	5,9	0.8	0,72
Filter Sustain	0,75	0,75	0,75	0,45	0.40	0,27
Filter Release [s]	2,0	2,0	2,0	0,25	1.28	0,10
Obálka filtru	+0,063	+0,26	+1,0	+1,0	+0.20	-0,27
Modulace filtru LFO	0,0	0,16	-1,0	0,328	0,15	-0,10
Vlna LFO	Sinus	Sinus	Pila	Sinus	Sinus	Sinus
LFO frekvence [Hz]	–	–	–	–	–	1,24
LFO synnchronizace	ON	ON	ON	ON	ON	OFF
LFO délka noty	1/8	1/1	2/1	1/2	1/1	–
LFO rettriger	OFF	OFF	ON	OFF	OFF	ON
Delay On/Off	OFF	OFF	OFF	ON	OFF	OFF
Delay čas	0,0	0,0	0,0	0,12	0,0	0,0
Delay zpětná vazba	0,0	0,0	0,0	0,414	0,0	0,0
Rozladění hlasů	0%	0%	0%	0%	0%	0%
Randomizace úderu	50%	0%	50%	0%	0%	0%
Hlasitost [dB]	-10	-10	-10	-10	-7	-10

Tabulka 7 Parametry syntezátoru pro jednotlivé zvuky použité ve skladbě Blurried

Parametr/ Charakter zvuku	**Basový buben**	**Nízký tom**	**Výsoký Tom**	**Zavřená Hi-hat**	**Otevřená Hi-hat**	**Virbl**
OSC1	50%	50%	50%	0%	0%	5%
OSC2	50%	50%	50%	0%	0%	5%
Noise	0%	0%	0%	50%	50%	75%
Vlna OSC1	Sinus	Sinus	Sinus	Sinus	Sinus	Trojúhelník
Vlna OSC2	Sinus	Sinus	Sinus	Sinus	Sinus	Sinus
OSC1 centy	0	+5	+5	0	0	0
OSC2 centy	0	+2	+2	0	0	0
OSC1 půltóny	-24	-16	-4	0	0	-12
OSC2 půltóny	-24	-16	-4	0	0	0
OSC1 modulace	0,0	0,0	0,0	0,0	0,0	0,0
OSC2 modulace	0,0	0,0	0,0	0,0	0,0	0,0
Attack [s]	0,01	0,01	0,01	0,01	0,01	0,01
Decay [s]	3,15	3,15	3,15	0,12	0,25	0,31
Sustain	1,0	1,0	1,0	0,15	0,14	0,074
Release [s]	0,35	0,35	0,35	0,01	0,45	0,93
Typ Filtru	LP	LP	LP	HP	HP	LP
Filtr mezní frek.	0,99	0,16	0,16	0,99	0,88	0,55
Rezonance	0,98	0,38	0,38	0,87	0,87	0,49
Filter Attack [s]	0,01	0,01	0,01	0,01	0,01	0,01
Filter Decay [s]	0,76	0,76	0,76	0,76	0,76	0,76
Filter Sustain	0,48	0,48	0,48	0,45	0,45	0,45
Filter Release [s]	1,88	1,88	1,88	1,88	1,88	1,88
Obálka filtru	+0,63	+0,61	+0,61	0,0	0,0	0,0
Modulace filtru LFO	0,0	0,0	0,0	0,0	0,0	0,0
Vlna LFO	Pila	Pila	Pila	Pila	Pila	Pila
LFO frekvence [Hz]	0,5	0,1	0,1	0,1	0,1	0,1
LFO synnchronizace	OFF	OFF	OFF	OFF	OFF	OFF
LFO délka noty	–	–	–	–	–	–
LFO rettriger	OFF	OFF	OFF	OFF	OFF	OFF
Delay On/Off	OFF	OFF	OFF	OFF	OFF	OFF
Delay čas	0,0	0,0	0,0	0,0	0,0	0,0
Delay zpětná vazba	0,0	0,0	0,0	0,0	0,0	0,0
Rozladění hlasů	0%	0%	0%	0%	0%	0%
Randomizace úderu	50%	50%	50%	50%	50%	50%
Hlasitost [dB]	-3	-3	-3	-17	-17	-15

Tabulka 8 Parametry syntezátoru pro jednotlivé zvuky použité ve skladbě Blurried - perkusivní zvuky

Parametr/ Charakter zvuku	**Basový zvuk**	**Klávesový zvuk**	**"Lead"**	**Pling**
OSC1	50%	50%	50%	50%
OSC2	0%	50%	50%	50%
Noise	0%	0%	0%	0%
Vlna OSC1	Čtverec	Sinus	Čtverec	Pila
Vlna OSC2	Čtverec	Trojúhel.	Trojúhel.	Sinus
OSC1 centy	0	0	0	0
OSC2 centy	0	0	0	0
OSC1 půltóny	0	0	0	0
OSC2 půltóny	0	0	0	0
OSC1 modulace	0,0	0,0	-0,04	0,0
OSC2 modulace	0,0	0,0	0,04	0,0
Attack [s]	0,01	0,01	0,01	0,01
Decay [s]	0,4	5,25	0,33	0,33
Sustain	0,4	0,609	0,415	0,415
Release [s]	0,255	1,96	1,96	1,96
Typ Filtru	LP	LP	LP	LP
Filtr mezní frek.	0,80	0,56	0,51	0,51
Rezonance	0,22	0,05	0,01	0,01
Filter Attack [s]	0,1	0,87	0,01	0,01
Filter Decay [s]	0,74	1,24	1,12	1,12
Filter Sustain	0,36	0,48	0,48	0,48
Filter Release [s]	0,84	1,89	1,89	1,89
Obálka filtru	0,297	0,48	0,0	0,0
Modulace filtru LFO	-0,039	0,0	0,0	0,0
Vlna LFO	Sinus	Sinus	Sinus	Sinus
LFO frekvence [Hz]	–	0,1	–	0,1
LFO synnchronizace	ON	OFF	ON	OFF
LFO délka noty	1/16	–	1/8	–
LFO rettriger	OFF	OFF	OFF	OFF
Delay On/Off	OFF	OFF	ON	OFF
Delay čas	0,0	0,0	0,297	0,0
Delay zpětná vazba	0,0	0,0	0,29	0,0
Rozladění hlasů	0%	0%	0%	0%
Randomizace úderu	50%	50%	50%	50%
Hlasitost [dB]	-10	-10	-10	-10

Tabulka 9 Parametry syntezátoru pro jednotlivé zvuky použité ve skladbě Vojtěcha Nedvěda

Parametr/ Charakter zvuku	Basový zvuk	Činel	Basový buben	El. piano	El. piano 2	Harpis-chord
OSC1	50%	0%	50%	50%	50%	50%
OSC2	75%	0%	0%	50%	10%	11%
Noise	0%	13%	0%	0%	0%	0%
Vlna OSC1	Trojúhel.	Sinus	Trojúhel.	Sinus	Trojúhel.	Pila
Vlna OSC2	Sinus	Sinus	Sinus	Čtverec	Pila	Pila
OSC1 centy	0	0	0	0	0	0
OSC2 centy	0	0	0	0	0	3
OSC1 půltóny	0	0	0	0	-12	0
OSC2 půltóny	+12	0	0	+12	0	+12
OSC1 modulace	0,0	0,0	0,516	-0,078	0,063	0,008
OSC2 modulace	0,0	0,0	0,0	0,227	0,055	0,0
Attack [s]	0,01	0,01	0,01	0,01	0,01	0,018
Decay [s]	0,4	1,9	0,4	0,4	0,4	15,0
Sustain	0,4	0,01	0,01	0,4	0,4	1,0
Release [s]	0,019	0,03	0,14	0,019	0,4	0,017
Typ Filtru	LP	HP	LP	LP	LP	LP
Filtr mezní frek.	0,5	0,28	0,066	0,365	0,5	0,452
Rezonance	0,01	0,38	0,01	0,149	0,01	0,080
Filter Attack [s]	0,1	0,1	0,01	0,1	0,1	0,1
Filter Decay [s]	0,8	0,8	0,8	0,8	0,8	0,8
Filter Sustain	0,5	0,5	0,5	0,5	0,5	0,5
Filter Release [s]	2,0	2,0	2,0	2,0	2,0	2,0
Obálka filtru	0,0	0,0	0,0	-0,086	0,0	0,0
Modulace filtru LFO	0,0	0,0	0,0	0,0	0,117	0,063
Vlna LFO	Sinus	Sinus	Pila	Trojúhel.	Trojúhel.	Sinus
LFO frekvence [Hz]	–	0,1	2,9	–	–	–
LFO synnchronizace	ON	OFF	OFF	ON	ON	ON
LFO délka noty	1/8	–	–	1/8	1/8	1/4
LFO rettriger	OFF	OFF	ON	ON	OFF	ON
Delay On/Off	OFF	OFF	OFF	OFF	OFF	OFF
Delay čas	0,0	0,0	0,0	0,0	0,0	0,0
Delay zpětná vazba	0,0	0,0	0,0	0,0	0,0	0,0
Rozladění hlasů	0%	0%	0%	0%	0%	0%
Randomizace úderu	50%	50%	50%	50%	50%	50%
Hlasitost [dB]	-10	-10	-10	-10	-10	-10

Tabulka 10 Parametry syntezátoru pro jednotlivé zvuky použité ve skladbě Adama Sporky

Parametr/ Charakter zvuku	**"Lead"**
OSC1	56%
OSC2	25%
Noise	0%
Vlna OSC1	Troúhel.
Vlna OSC2	Pila
OSC1 centy	0
OSC2 centy	0
OSC1 půltóny	0
OSC2 půltóny	0
OSC1 modulace	0,0
OSC2 modulace	0,0
Attack [s]	0,01
Decay [s]	2,534
Sustain	0,030
Release [s]	0,615
Typ Filtru	LP
Filtr mezní frek.	0,567
Rezonance	0,169
Filter Attack [s]	0,013
Filter Decay [s]	0,144
Filter Sustain	0,304
Filter Release [s]	0,012
Obálka filtru	0,508
Modulace filtru LFO	0,0
Vlna LFO	Sinus
LFO frekvence [Hz]	0,1
LFO synnchronizace	OFF
LFO délka noty	–
LFO rettriger	OFF
Delay On/Off	OFF
Delay čas	0,0
Delay zpětná vazba	0,0
Rozladění hlasů	0%
Randomizace úderu	50%
Hlasitost [dB]	-10

Tabulka 11 Parametry syntezátoru pro jednotlivé zvuky použité ve skladbě Ravenous Female od Cebit

8 Závěr

Cílem této bakalářské práce byla realizace polyfonního syntezátoru za pomoci technologie VST.

V první části této bakalářské práce byly popsány a vysvětleny důležité teoretické informace pro pochopení principu funkčnosti hudebních syntezátoru a jejich softwarových podob. Popsány byly nejčastěji používané metody zvukové syntézy a podrobněji rozebrána subtraktivní metoda. Vysvětlen byl princip číslicového komunikačního rozhraní pro ovládání syntezátorů (*Musical Instruments Digital Interface*) a principy technologie *Virtual Studio Technology*.

Při návrhu architektury syntezátoru a následné implementaci, bylo využito těchto teoretických informací a principů. Syntezátor byl implementován pokud možno tak, aby byl co nejlépe rozšiřitelný o další moduly a funkce. Výsledný syntezátor lze používat v běžně dostupných programech pro tvorbu hudby, konkrétně byly otestovány tyto ve 32bit verzích: REAPER, Ableton Live 9, FL Studio 10 a 11, MultitrackStudio.

Realizovaný syntezátor byl otestován vybranými hudebníky, kteří syntezátor použili k produkci vlastní hudební skladby. Tento způsob testování potvrdil, že syntezátor je jednoduše použitelný v praxi, hlavně díky srozumitelnému grafickému rozhraní a dostatečně flexibilní na to, aby hudebníkovi posloužil jako zdroj široké škály zvuků.

Syntezátor by mohl být dále rozšiřován a vylepšován, nabízí se přidání druhého nízkofrekvenčního oscilátoru, nebo sub-oscilátorů. Implementace tzv. modulační matice, kdy si uživatel může vybrat, který parametr bude modulován, kterým modulátorem, by byla rozhodně také zajímavým rozšířením. V současném stavu je syntezátor použitelný jako plnohodnotný softwarový nástroj, což bylo ověřeno za pomoci oslovených hudebníků.

A Obsah přiloženého DVD

Na přiloženém DVD se nachází:

- `MicroBery/` ..... hlavní adresář projektu
  - `DelayEffect.h`
  - `DelayEffect.cpp`
  - `Envelope.h`
  - `Envelope.cpp`
  - `Filter.h`
  - `Filter.cpp`
  - `MicroBery.h`
  - `MicroBery.cpp`
  - `MIDIReceiver.h`
  - `MIDIReceiver.cpp`
  - `NoiseGenerator.h`
  - `NoiseGenerator.cpp`
  - `Oscillator.h`
  - `Oscillator.cpp`
  - `Voice.h`
  - `Voice.cpp`
  - `VoiceManagement.h`
  - `VoiceManagement.cpp`
  - `resource.h`
  - `resources/`
    - `img/` ..... adresář s bitmapami pozadí a potenciometrů
- `Skladby/` ..... adresář skladeb vytvořených v rámci testování
- `MicroBery.dll` ..... zkompilovaný plugin

B Kompilace projektu

Prerekvizity nutné ke kompilaci projektu na Windows

1. Visual Studio C++ 2010, nebo 2012
2. Git for Windows – `https://git-scm.com/downloads`
3. VST3 SDK – `http://www.steinberg.net/en/company/developer.html`
4. RtAudio obsahuje ASIO SDK – `http://www.music.mcgill.ca/~gary/rtaudio/`

Návod na kompilaci

Instalace frameworku WDL-OL/IPlug

Po instalaci programu *Git for Windows* spusťte terminál *Git Bash* a v něm spusťte následující příkazy:

```
cd c:
git clone https://github.com/olilarkin/wdl-ol
cd wdl-ol
git checkout 0a360c90b3460717210eeaee7464bc7009c9a5ba
```

Tímto by se vám měl na disku C: vytvořit adresář *wdl-ol*.

- Do podadresáře *wdl-ol/ASIO_SDK* je nutné zkopírovat všechny soubory s příponou .cpp a .h ze složky *include* ze staženého *ASIO SDK*.
- Rozbalte stažený archiv *VST3 SDK* a zkopírujte soubory *aeffect.h* a *aeffectx.h* z adresáře *pluginterfaces* do podadresáře *wdl-ol/VST_SDK*.
- Nyní ze staženého *VST3 SDK* zkopírujte složky *base/source, pluginterfaces a public.sdk* do adresáře *wdl-ol/VST3_SDK*.

Instalace a kompilace projektu

- Celou složku *MicroBery*, která obsahuje realizovaný VST plugin zkopírujte do adresáře *wdl-ol/IPlugExamples*.
- Otevřete *MicroBery.sln* ve Visual Studio. V *Solution Explorer* klikněte pravým tlačítkem na *MicroBery-vst2* a vyberte *Set as StartUp Project*.
- Nyní zmáčkněte klávesu F5 a projekt by se měl zkompilovat.

C Uživatelská příručka

Instalace

Instalace se liší dle používaného DAW, obecně je potřeba knihovnu *MicroBery.dll* vložit do složky s ostatními *VST plugins*, nejčastěji *C:/Program Files/VstPlugins*. Poté postupujte dle návodu k vašemu DAW.

Použití

Syntezátor MicroBery je virtuálně analogový nástroj navržený dle standardu VST, je integrovatelný do běžně používaných nástrojů pro tvorbu hudby, které podporují VST minimálně ve verzi 2.

Všechny parametry syntezátoru mohou být ovládány pomocí automatizace uvnitř hostující aplikace. Syntezátor obsahuje 4 již připravené zvukové programy.

Základní funkce

Syntezátor nabízí polyfonii se 32 hlasy, každý hlas se stává ze dvou oscilátorů (každý oscilátor má 4 typy vlnových průběhů) a generátoru šumu. Dva generátory obálek typu ADSR modulují hlasitost a filtr. Syntezátor dále poskytuje nízkofrekvenční oscilátor pro další modulace filtru a také oscilátorů. Minimalistický delay efekt se zpětnou vazbou nabízí další možnosti při experimentování s výsledným zvukem.

Oscilátory

Každý oscilátor nabízí 4 typy vln - *sinus, pila, čtverec, trojúhelník.*Oscilátory je možné dolaďovat v rozmezí (-100;+100) centů a nebo (-24;+24) půltónů. Je možné nastavit úroveň oscilátorů společně s generátorem šumu v sekci *mix*.

LFO

Nízkofrekvenční oscilátor je možné synchronizovat s tempem, které je poskytováno DAW. Synchronizace se zapíná přepínačem *sync* v sekci LFO. Pokud je přepínač v poloze ON nastavuje se perioda vlny LFO přepínačem *sync rate* udávaná v délce noty. V případě, že je přepínač v poloze OFF, nastavuje se frekvence LFO pomocí potenciometru *rate*. Přepínač *key retrig* zajistí, že při každém stisknutí klávesy se vynuluje fáze LFO.

Filtr

Je možné zvolit 3 typy filtru dolní, horní nebo pásmovou propust, filtr má útlum 24 dB/okt. Filtr lze modulovat nízkofrekvenčním oscilátorem a obálkou, úroveň modulace lze nastavit z grafického rozhraní.

Hlasy

Hlasy je možné pseudo-náhodně rozlaďovat, dle zvolené úrovně. Zvuk syntezátoru poté získává zajímavý analogový charakter. Dále je možné nastavit randomizaci rychlosti úderu a celkovou hlasitost syntezátoru, to vše z grafického rozhraní v sekci *Voices*.

Přednastavené programy

- *Default* - základní program
- *Deep bass* - podladěný basový zvuk
- *Space bass* - basový zvuk ve stylu *80s*
- *Atmo pad* - atmosférická plocha
- *Hammond organ* - simulace varhan značky *Hammond*

Podporované MIDI zprávy

- Note On
- Note Off
- Modulation Wheel - moduluje amplitudu LFO
- Pitch bend
- Sustain pedal

Uživatelské rozhraní

Součástí uživatelského rozhraní jsou *otočné potenciometry*, *posuvné potenciometry* (obálka filtru a amplitudy), které se ovládají stisknutím levého tlačítka myši a tažením směrem nahoru/dolu.

Přepínače, kterými se nastavuje tvar vlny, typ filtru nebo délka noty u LFO se ovládají kliknutím levého tlačítka myši, to samé platí pro spínače (delay, LFO sync, LFO rettrig).

Zdroje

[1] Václav Syrový. *Hudební signál a jeho syntéza*. [online]. [cit. 05. 05. 2016]. 1986. URL: http://ziva-hudba.info/article.php?id=215.

[2] Rick Snoman. *Dance Music Manual - second edition*. Focal Press, 2009. ISBN: 978-0-2405-2107-7.

[3] Vanda Teocharisová. *Sound Design - Zvuková syntéza a tvůrčí programování zvuků v praxi*. Muzikus, 2009. ISBN: 80-86253-53-4.

[4] Steinberg. *VST: The integrative standard for virtual instruments and effects*. [online]. [cit. 28. 04. 2016]. 2016. URL: http://www.steinberg.net/en/company/technologies.html.

[5] Steinberg. *Virtual Studio Technology - Plug-In Specification 2.0 – Software Development Kit*. [online]. [cit. 29. 04. 2016]. 1999. URL: http://jvstwrapper.sourceforge.net/vst20spec.pdf.

[6] Oli Larkin. *WDL / IPlug - Oli Larkin Edition*. [online]. [cit. 24. 04. 2016]. URL: https://github.com/olilarkin/wdl-ol.

[7] Oli Larkin. *Life after Pluggo - Getting started programming plugins in C++ using IPlug*. [online]. [cit. 05. 05. 2016]. 2012. URL: http://olilarkin.blogspot.cz/2012/01/m4u-convention-iplug-workshop-slides.html.

[8] Patrick Hogan. *Signals*. [online]. [cit. 24. 04. 2016]. URL: https://github.com/pbhogan/Signals.

[9] Joe Wolfe. *Note names, MIDI numbers and frequencies*. [online]. [cit. 05. 05. 2016]. URL: https://newt.phys.unsw.edu.au/jw/notes.html.

[10] Paul Kellett. *Resonant filter*. [online]. [cit. 05. 05. 2016]. URL: http://www.musicdsp.org/showone.php?id=29.

[11] Christian Schoenebeck. *Fast Exponential Envelope Generator*. [online]. [cit. 05. 05. 2016]. URL: http://www.musicdsp.org/showone.php?id=189.

[12] Will Pirkle. *Designing Audio Effect Plug-Ins in C++*. Focal Press, 2013. ISBN: 978-0-123-97882-0.