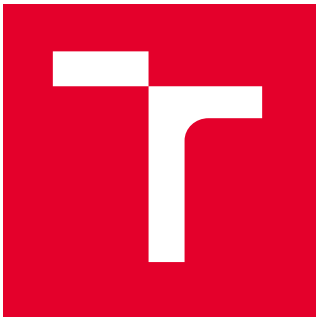


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

VST PLUGIN PRO GRANULÁRNÍ SYNTÉZU S EXPERIMENTÁLNÍMI FORMAMI NASTAVENÍ VSTUPU I VÝSTUPU

VST GRANULAR PLUGIN WITH EXPERIMENTAL INPUT AND OUTPUT SETTINGS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Jan Kalinič**

AUTHOR

VEDOUCÍ PRÁCE — **MgA. Michal Indrák, Ph.D.**

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Audio inženýrství**
specializace Zvuková produkce a nahrávání
Ústav telekomunikací

Student: Jan Kalinič ***ID:*** 221471
Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

VST plugin pro granulární syntézu s experimentálními formami nastavení vstupu i výstupu

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je funkční VST plugin, který bude umět realizovat granulární syntézu experimentálními metodami zadávání (např. kreslením, samotným zvukovým vzorkem atd.) nad různými druhy syntézy umělého zvuku a jeho kombinací, které budou součástí tohoto pluginu. Granulární syntézu bude možné realizovat i nad vloženým samplem nebo živým vstupem. Integrovaná bude podpora MIDI protokolu.
V rámci semestrální práce student nástroj kompletně navrhne a vytvoří ukázky programového řešení jednotlivých částí, v rámci navazující bakalářské práce pak realizuje plně funkční program.

DOPORUČENÁ LITERATURA:

[1] RUSS, Martin. Sound synthesis and sampling. Oxford: Focal Press, 1996. Music technology series. ISBN 0-240-51429-7.
[2] The Csound book: perspectives in software synthesis, sound design, signal processing, and programming. Editor Richard BOULANGER. Cambridge: MIT Press, c2000. ISBN 0262522616.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: MgA. Michal Indrák, Ph.D.

doc. Ing. Jiří Schimmel, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato práce pojednává o vývoji zásuvného modulu, který je schopen realizovat granulární syntézu za pomoci experimentálních způsobů nastavování hodnot parametrů. Grafické návrhy a plány byly vytvořeny v editoru vektorové grafiky Inkscape. Samostatný zásuvný modul byl naprogramován v jazyce C++ s využitím frameworku JUCE a technologie VST3. Pomocí tohoto modulu lze granulární syntézu provádět nad předem určeným vzorkem, ale i nad tokem zvukového kanálu digitální audio stanice. Součástí modulu jsou i bloky s ostatními druhy zvukové syntézy, které lze taktéž modifikovat experimentálními způsoby. Plug-in je díky svým vlastnostem možné v praxi aplikovat jako efekt nad jedním zvukovým kanálem v mixovací části audio stanice.

KLÍČOVÁ SLOVA

VST3, plug-in, DAW, granulární syntéza, syntéza zvuku, JUCE, C++

ABSTRACT

This thesis discusses the development of a plug-in module that is able to implement granular synthesis with help of experimental methods of parameter value settings. Graphic designs and schemes were created in the vector graphics editor Inkscape. The separate plug-in was programmed in C++ language using the JUCE framework and VST3 technology. With a use of this module, granular synthesis can be performed over a predetermined sample, but also over the stream of an audio channel of a digital audio station. The module also includes blocks with other types of sound synthesis, which can also be modified in experimental ways. Thanks to its properties, the plug-in can be applied as an effect over one sound channel in the mixing section of an audio station.

KEYWORDS

VST3, plug-in, DAW, granular synthesis, sound synthesis, JUCE, C++

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Jan Kalinič
VUT ID autora:	221471
Typ práce:	Bakalářská práce
Akademický rok:	2022/23
Téma závěrečné práce:	VST plugin pro granulární syntézu s experimentálními formami vstupu i výstupu

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu MgA. Michalovi Indrákovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

Všeobecně lze syntézu definovat jako skládání uceleného tvaru z menších částí a dílků. Zvuková syntéza potom specificky označuje souhrn činností vedoucí k umělému vytvoření zvuků. Syntetizování zvuků po umělecké stránce však mnohdy znamená více než jen náhodné skládání částí k sobě. Celý proces vyžaduje kreativní pohled, ať už se jedná o zavedené techniky, nebo o vytvoření technik nových. Syntéza tak dává vzniknout zvukům zcela novým, dokáže však i modifikovat zvuky již existující.

Tvorba takovýchto zvuků je uskutečněna za použití syntezátorů. Ty existují v podobě analogové nebo digitální. Tato práce se konkrétněji zaměří na tvorbu syntezátorů digitálních. Běžně se skládají ze dvou funkčních prvků. Jedním z nich je řídící prostředí dovolující uživateli nastavit parametry syntézy. Druhým je syntetizační jádro, které za pomoci algoritmů a zmíněných parametrů vytvoří zvukový výstup.

Přestože syntezátory mají ve své podstatě velmi široké možnosti výstupů, každý z nich nemusí být sluchově přijatelný. Konečný výsledek tudíž nezáleží pouze na funkcích syntezátoru, ale i na uživatelském úsudku a následném zásahu do vstupního nastavení. Obě zmíněné části syntezátorů jsou velmi důležité. Někdy se však větší váha přikládá řídícímu prostředí, které dává uživateli větší možnost interakce se vstupními parametry a do jisté míry tak přenechává svobodu lidské tvořivosti.

Měnit vstupní parametry algoritmu lze i nepřímým zásahem s dopomocí nízkofrekvenčních oscilátorů, na což se zaměří i tato práce. Klade si otázku, jak by mohlo vypadat netradiční nastavování parametrů podobné těmto oscilátorům a jak jej může uživatel ovlivnit i jinak než fixním otočením knoflíku. Velké pozornosti se v práci dostane také granulární syntéze, ke které pro zvukové doplnění přibydou jednodušší moduly aditivní a wavetable syntézy.

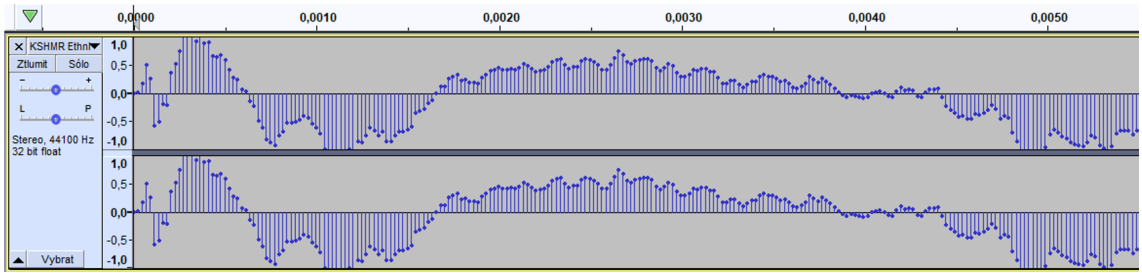
Jelikož se práce zabývá kompletním návrhem a vývojem softwaru zásuvného modulu VST3, je nutné si předem stanovit, pro jaké digitální audio stanice bude software určen a na kterých audio stanicích bude zásuvný modul testován z důvodu zajištění funkčnosti. Je taktéž žádoucí si porovnat a vybrat jednu z dostupných technologií pro vývoj zásuvných modulů. Práce zahrnuje nejen algoritmy zmíněných syntéz a experimentálních způsobů nastavení, ale také vzhled plug-in modulu po uživatelské stránce. Po ukončení vývoje bude výsledek dostupný a volně šiřitelný pro širokou veřejnost.

1 Teoretická část

1.1 Převod analogového signálu na digitální

Převod reálného zvuku do digitálního probíhá v ADC převodníku tzv. vzorkováním nebo-li samplingem, zde je důležitým parametrem vzorkovací frekvence, která udává kolikrát za jednu sekundu se odebere diskrétní vzorek z původního spojitého signálu. Dalším důležitým parametrem je bitová hloubka, ta udává počet kroků, přesněji jde o mocniny čísla dvě, kterým se hodnota amplitudy může rovnat. Protože se signál nemusí vždy přesně rovnat hodnotě kroku, používá se hodnota nejbližšího kroku. Toto zaokrouhlování však výsledný signál lehce zkresluje. Pro představu 24 bit depth je schopno reprezentovat až 144 dB dynamického rozsahu, 16 bit depth pouze 96 dB[29].

Obr. 1.1: Zjednodušené schéma převodu signálu.



Obr. 1.2: Ukázka diskrétních vzorků zvukového souboru v programu Audacity.

1.2 Audio Buffer

Buffer je velmi důležitou součásti digitálního zpracování zvuku. Jedná se o zásobník, ve kterém jsou uloženy hodnoty diskrétních zvukových vzorků. Tyto hodnoty na sebe berou formu desetinného čísla s rozsahem -1 až 1.

Jádro zásuvných modulů tvoří algoritmus, ve kterém se datový tok zvuku zpracovává po blocích určité délky. Tato délka je známá jako buffer size. Označuje počet vzorků, které se zpracují v jednom cyklu. Čím větší je buffer, tím déle trvá zpracování vzorků. Tento čas je při živém poslechu vnímán jako zvuková latence. Větší buffer se hodí například pro zpracování vícero náročnějších plug-inů. Naopak čím menší buffer, tím je proces zpracování náročnější na systémové prostředky, zejména na procesor a paměť. V tomto procesu může dojít k přehlcení těchto prostředků, to následně vede k nepříjemnému zvuku lupání a praskání[26].

Situace tedy velí ke kompromisu při volbě velikosti bufferu vzhledem k přípustnému zpoždění zvuku. Vhodné velikosti bufferu pro nahrávání a živý poslech s co nejmenším zpožděním bývají 128 a 256 vzorků. Pokud je zvuk již nahrán a účelem je spíše mixovat, masterovat, nebo vytvářet zvuk pomocí plug-inů, je lepší zvolit velikost bufferu 512 a vyšší[25].

1.3 DSP

Obr. 1.3: Zjednodušené schéma průchodu zvuku end-to-end s blokem DSP.

DSP je zkratkou pro Digital Signal Processing známé jako digitální zpracování signálů. Obor DSP zahrnuje algoritmy, matematické rovnice a postupy. Řeší, jak nakládat se zvukovým signálem, poté co byl převeden z analogového tvaru do digitálního. Tento obor je velmi důležitý a rozvíjí ostatní vědní obory již řadu let. Příkladem může být ultrazvuk, či radar. Ve zvukové oblasti se využívá při kompresi dat, nebo i praktické simulaci vln a spektrální analýze. Mnohdy digitální uchování dat eliminuje problémy, které s sebou nesou analogová média, u nichž často dochází k fyzickému

poškození. DSP umožňuje komplexní úkony, jako například mixování mnoha tracků do jednoho finálního. Dává možnost současně využít ekvalizace, zpoždění zvukových vzorků, nebo jiných zvuk obohacujících efektů[28].

1.4 DAW

Digital Audio Workstation zkráceně DAW je technicky jakékoliv digitální zařízení určené k nahrávání a produkci zvuku. Tato práce se však na DAW dívá ze softwarového pohledu.

Zatím co dříve se klasické hudební studio neobešlo bez značného množství zařízení potřebných k nahrávání, mixování a zpracování zvuku, dnes již většinu úloh těchto zařízení zastane jeden průměrně výkonný osobní počítač s nainstalovaným DAW programem. Z počátku skrze limitace výpočetní síly muselo DAW být realizováno na externích DSP zařízeních. To jsou zařízení dodávající počítači výpočetní zdroje na zpracování digitalizovaného zvuku. Nyní jsme s výpočetní sílou na takové úrovni, že tato zařízení nejsou potřeba. I bez nich je DAW schopné na velmi dobré úrovni nahrávat, přehrávat a editovat několik stop najednou[23].

Kromě již vyjmenovaných dovedností je možné v DAW používat takzvaných plug-in nebo-li zásuvných modulů a tím si tak rozšířit nebo doplnit audio stanici o efekty nebo zvukové generátory.

Výběr DAW je do jisté míry osobní záležitostí. Chce-li producent svou práci sdílet jako rozpracovaný projekt, a nikoliv jako zvukové soubory, je nucen přiklonit se k jednomu DAW, které je považováno za průmyslový standard[23].

Průmyslem uznávaných DAW není mnoho. Každá firma se snaží dostat co nejvýše na pomyslné příčce této kategorie. To vytváří soutěživost, která vede ke zlepšování poskytovaných služeb a zkvalitňování DAW softwaru.

Nejčastěji používanými DAW jsou:

- Image-Line - FL Studio
- Ableton - Ableton Live
- Avid - Pro Tools
- Apple - Logic Pro
- Steinberg - Cubase
- Cockos - REAPER
- Presonus - Studio One
- Garage Band

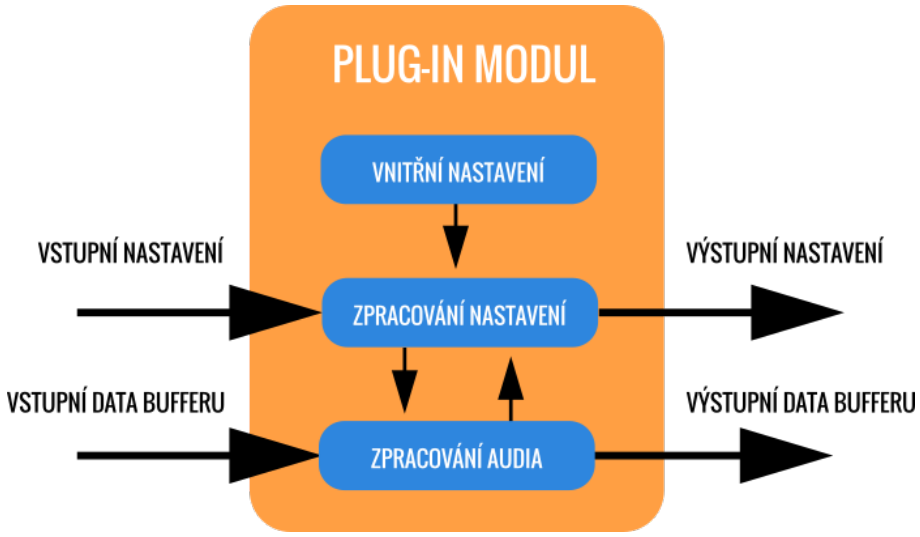
Ne každý program tohoto standardu je však přijímán celým spektrem hudebníků. Některé DAW jako například Garage Band jsou především používány na nenáročné

domácí produkování. Naopak jiné jako třeba Pro Tools a Cubase svými pokročilejšími vlastnostmi zaujaly svoje místo po boku producentů pohybujících se v profesionální sféře.

1.5 Zásuvné moduly

Plug-iny nebo-li zásuvné moduly jsou dnes již neodmyslitelnou součástí každého DAW. Plug-inem označujeme software, který narozdíl od samostatně spustitelných aplikací, nedokáže fungovat bez hostitelského prostředí. Neobsahuje totiž žádnou infrastrukturu potřebnou ke komunikaci se zvukovou kartou. Vývojáři plug-inů se tak mohou zaměřit na samotné zpracování zvuku, což eliminuje možné komplikace vzniklé při přímém manipulování se zvukovými zařízeními[15].

Základní model plug-inu představuje obecný způsob komunikace s hostitelem. Plug-in může přijímat na vstupu zvukový signál a nastavení. Uvnitř za pomocí svého algoritmu a interního nastavení zpracuje zvuk. Na výstupu lze očekávat zpracovaný signál a nastavení. V některých případech mohou být data a nastavení přijímány i vráceny bez jakýchkoliv hodnot[27].

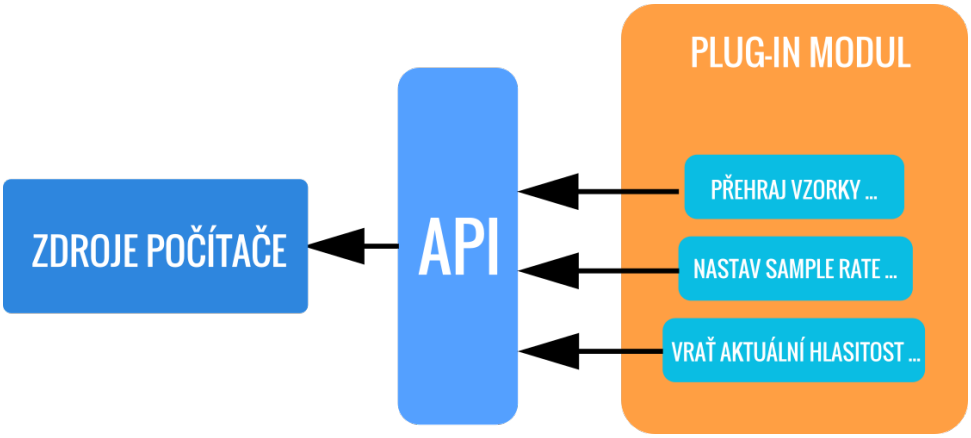


Obr. 1.4: Základní plug-in model.

Na plug-iny se můžeme také dívat jako na dynamické moduly či knihovny, které si DAW pro používání načítá za běhu. Jejich síla také tkví v principu, že pouze rozšiřují již uživatelem známý systém. Díky tomu se není třeba bát o zbytek dosavadní práce po přidání nového plug-inu. Dají se kdykoliv jednoduše přidat či odebrat z příslušného projektu. Jednoduchost není však společným jmenovatelem všech plug-inů, některé totiž dokáží být až přehnaně složité svojí instalací, nebo užíváním.

1.5.1 Plug-in API

Plug-in API je základem všech plug-inů. API je zkratkou pro Application Programming Interface. Jeho účelem je poskytnout rozhraní zaštiťující funkce, protokoly a procedury knihovny zpracování zvuku. Umožňuje tedy programátorovi vytvářet bloky kódu, které plug-in host očekává při svém spuštění a se kterými následně komunikuje. Jedná se většinou o třídy a jejich metody. Jelikož i samotný plug-in je třída, jejíž instanci si plug-in host vytvoří na začátku svého chodu. Následně využívá metody k nastavení vzorkovací frekvence, velikosti bufferu, nebo předává a čte MIDI události[15].



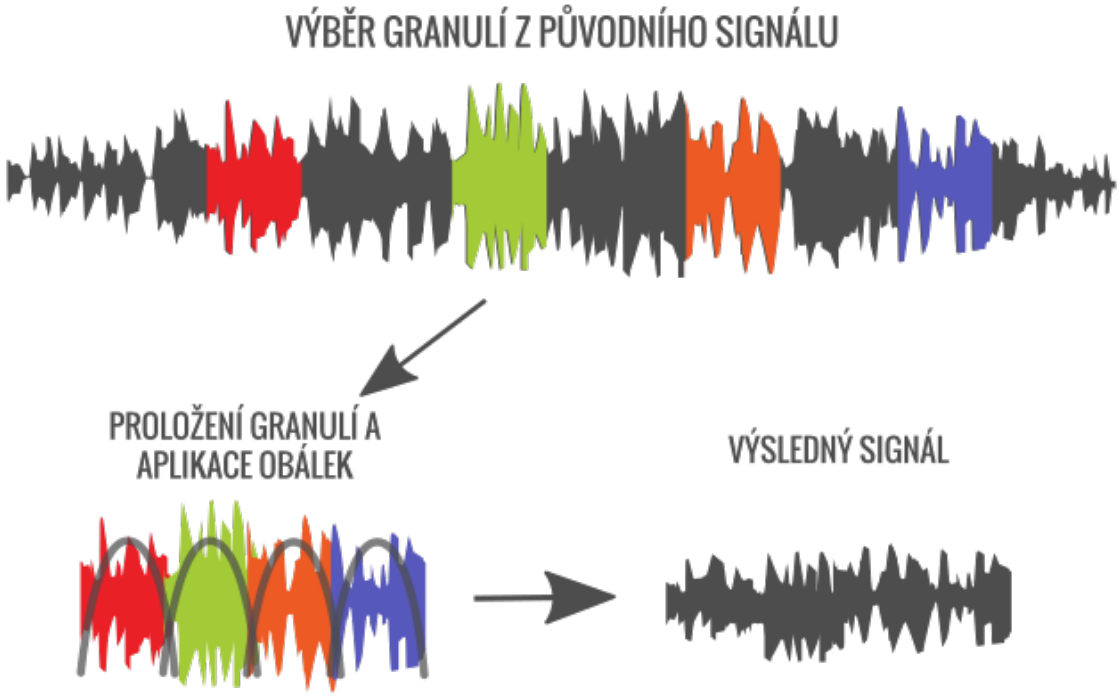
Obr. 1.5: Plug-in API model.

Velké množství plug-in modulů je psaných v jazyce C++, který je na tyto úkony díky své rychlosti, robustnosti a dynamičnosti velmi vhodný[14].

1.6 Granulární syntéza

Granulární syntéza je zvláštním druhem zvukové syntézy. Není to něco, s čím se dá běžně setkat v hudebním průmyslu. Své uplatnění nalézá spíše při tvorbě ambientních zvuků, či speciálních zvukových efektů. Velký zájem o ni jeví také v oblastech akademických a výzkumných. Boří totiž základní pohled na zpracování a vnímání zvuku[3].

Princip granulární syntézy spočívá ve skládání celků z krátkých zvukových segmentů zvaných zrna nebo také granule. Tato zrna mohou mít různou délku. Většinou se jedná o útržky o délce 10 až 100 milisekund. V tomto rozmezí se nachází hranice možností sluchového vnímání. Jde přibližně o 50 milisekund, při nichž lidské vnímání nedokáže oddělit takto spojený zvuk a kde jinak dvě rozdílné části, které spolu nemusejí souviset, do sebe splývají[3].



Obr. 1.6: Princip granulární syntézy.

Granulární syntézu je možné nastavit několika jednoduchými parametry. Nejčastěji se lze setkat s parametrem délky jedné granule, výchozí pozice a parametrem počtu granulí, které se mají přehrát. Pokročilejší nastavení pak zahrnuje třeba náhodný rozptyl granulí, nebo směr kterým se má přehrávání uskutečnit. Dále také rychlost přehrávání, jenž se mnohdy doplňuje i takzvaným pitch shiftingem. Pitch shifting je uskutečnění změny tónové výšky zvuku. Mnohdy bývá obohacen i systémem převzorkování, což zajistí, že se délka přehrávaného úseku nezmění a pouze se posune v kmitočtové oblasti[4].

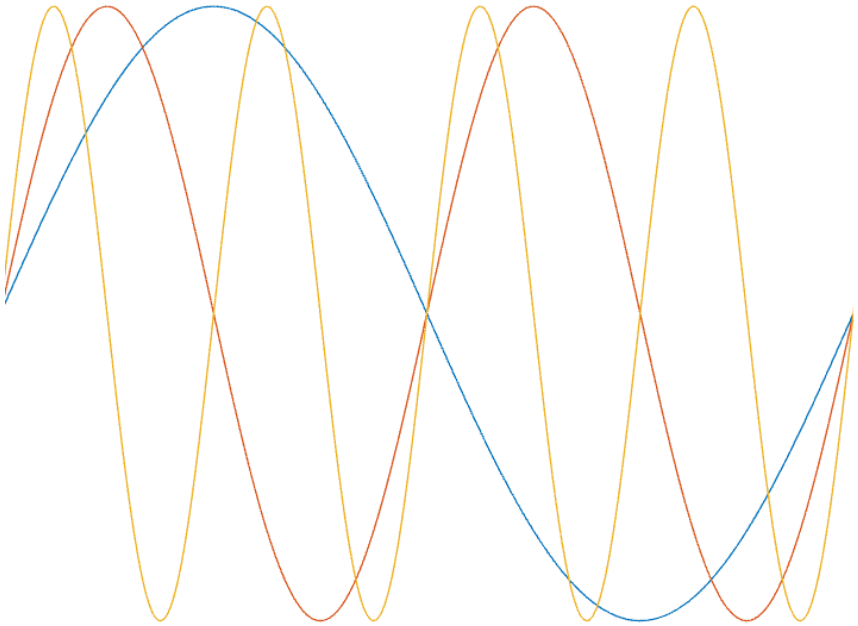
Oblíbenou formou granulární syntézy je forma softwarová. Ta se zasadila o používání jinak netradičních postupů tvorby zvuku. Konkrétněji jde o náhodné vybírání pozic vzorků, nebo míchání přehrávaných vzorků pozpátku se vzorky přehrávanými v běžném pořadí. V hardwarové podobě granulární syntéza tolik oblíbená není. Dostupných variant se na trhu nevyskytuje mnoho a většinou je jejich koupě finančně náročná. Za zmínku však stojí poměrně oblíbený výrobek brněnského výrobce Bastl Instruments pod jménem Microgranny [1]. Jde o malé zařízení využívající granulárních algoritmů s několika způsoby nastavení. Je však spíše zajímavostí pro audio nadšence než běžně užívaným nástrojem k tvorbě zvuku.

[1] Microgranny - `https://shop.bastl-instruments.com/diy/microgranny-laguna-diy.html`

1.7 Aditivní syntéza

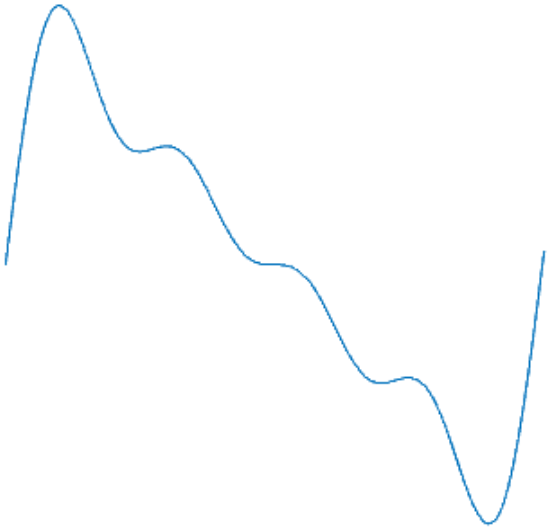
Za aditivní syntézu označujeme druh zvukové syntézy, při které vznikají komplexnější zvuky skládáním dvou a více jednodušších vln dohromady.

Historicky sahá aditivní syntéza na samotné počátky elektronických syntezátoru, které pro dosažení určité barvy hlasu kombinovaly sinusové vlny. V modernějších syntézátorech však není tato technika až tak běžná. Na místo ní se přímo generují signály již odvozené. V hardwarových syntézátorech se tak ušetří cenné místo, které by jinak zabíralo několik oscilátoru generujících pouze harmonický průběh[1]. Toto místo poté vyplní pouze jeden oscilátor, na kterém si lze zvolit tvar generovaného signálu.



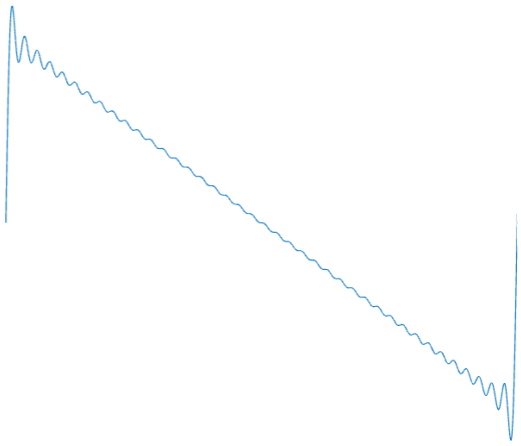
Obr. 1.7: Ukázka jednotlivých harmonických signálů.

Již zmíněné skládání probíhá jednoduchým součtem vln. Proto je tato syntéza často označována také jako součtová. Takovýmto komponováním harmonického signálu lze dosáhnout všech ostatních známých tvarů periodických vln jako jsou pila, trojúhelník, či obdélník. Kdy se každá harmonická složka do výsledného signálu vmíchá v jiném poměru.



Obr. 1.8: Kombinace prvních čtyř harmonických signálů.

Obecně platí, že čím větší počet harmonických složek, tím lepší rozlišení tvaru vlny. Jde především o co nejpřesnější přiblížení se skutečnému tvaru. Jako příklad může sloužit tvar signálu pily.

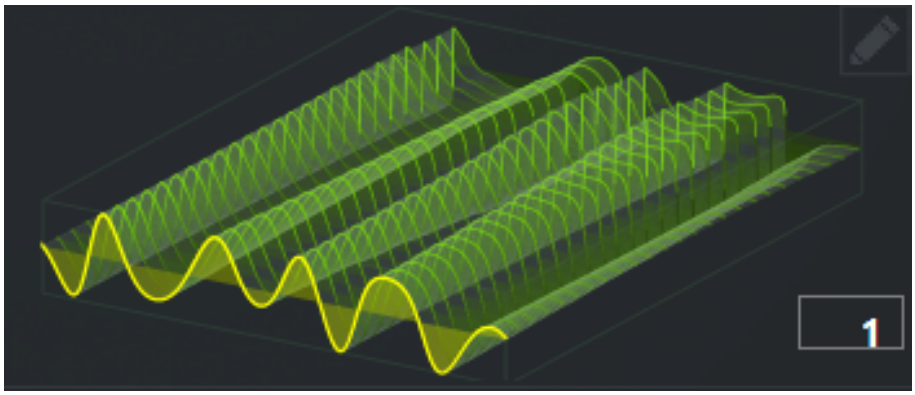


Obr. 1.9: Kombinace prvních čtyřiceti harmonických signálů.

Aditivní syntéza je po tvořivé stránce velmi lákavá, nabízí nespočet pozoruhodných variací zvuku plynoucí z obyčejného nastavení parametru frekvence a hlasitosti jedné složky.

1.8 Wavetable syntéza

Díky rozvoji digitálních technologií vznikl v roce 1958 pod záštitou Bell Labs program MUSIC II. Max Vernon Mathews, který následně spolupracoval i na počítačem generovaném zvuku lidského hlasu, přidal do druhé verze svého programu možnost generování až šestnácti různých zvukových vln. Tyto vlny se bez prodlev periodicky opakovaly a daly tak vzniknout novému druhu zvukové syntézy[2].



Obr. 1.10: Znázornění průchodu vln tabulky wavetable syntezátoru SERUM.

Princip wavetable syntézy tedy spočívá v použití sekvencí a tabulky obsahující jednodušší ale i komplexnější zvukové vlny. Je důležité zmínit, že se většinou jedná o jednu periodu takovéto vlny. Každá sekvence obsahuje několik po sobě jdoucích, avšak nezávislých vln, které jsou mezi sebou interpolovány, aby došlo k plynulejšímu zvukovému přechodu. Přehrávání sekvencí probíhá ve smyčce.

Modifikovat, který index tabulky se zvolí jako následující, a tudíž jaký tvar vlny bude zvuk obsahovat, lze například pomocí obálky ADSR při stisku klávesy. Měnit toto nastavení lze i za pomoci nízkofrekvenčního oscilátoru. Modifikační signál poté projde pozice s jednotlivými tvary vln v tabulce[3].

Obr. 1.11: Tabulka s indexy dostupných tvarů zásuvného modulu SERUM.

1.9 LFO

Low Frequency Oscillator nebo-li nízkofrekvenční oscilátor je speciálním druhem modifikačních oscilátorů. Vznikl za účelem periodické změny kontrolního napětí, které může ovládat parametry zvukových generátorů a efektů. Nejčastěji stejně jako běžný oscilátor generuje tvary vln sinu, pily, trojúhelníku a obdélníku. Takto vytvořený signál může automaticky měnit například hlasitost přehrávaného zvuku pro dosažení efektu tremola[3].

Obecně LFO vnáší do modulovaných zvuků a efektů značnou variabilitu, která může být ku prospěchu při delším poslechu jednotvárného zvuku, který se posluchači nemusí zamlouvat.

Zásadní vlastností je tedy oscilace v nižším kmitočtovém pásmu. Rychlost takovéhoto generování signálu se udává v hertzích a označuje kolikrát se celý průběh nastaveného tvaru vlny napěťově projeví na výstupu. Běžně se lze setkat s kmitočty od desetiny hertzu až do 20 Hz.

Parametry oscilátoru mohou být již zmíněný tvar vlny a frekvence oscilátoru. Dále také šířka pulzu pro případ, že si uživatel zvolí obdélníkový signál. Posledním běžně užívaným parametrem je velikost změny, často označovaná jako hloubka, známá pod anglickým názvem depth. Jedná se o míru, jakou výsledný signál LFO ovlivní modifikovaný parametr. Zřídka kdy se také používá parametr počáteční fáze, anglicky označovaný jako delay, offset, nebo přímo phase.

Je ale důležité zmínit, že napětí oscilátoru by v digitální oblasti nemělo moc význam a tak se jako zástupce nejčastěji volí desetinné hodnoty v rozsahu 0-1.

2 Příprava vývoje

Cílem přípravy je vhodně zvolit prostředky a způsob jakým bude následně plug-in modul vyvíjen. Nejdříve se v krátké části zaměří příprava na průzkum digitálních audio stanic, následně popíše možnosti technologií zásuvných modulů a nakonec přihlédne i k variantám vývojových frameworků.

2.1 Testovací DAW

Tato sekce popíše DAW programy, které budou zohledněny při uživatelském testování zásuvného modulu. Digitální audio stanice budou vybrány na základě osobních zkušeností i z důvodů popularity a odlišného typu plug-in hostitelů. Snahou je do jisté míry pokrýt co nejširší spektrum uživatelů, kteří by si případně zkompletovaný výsledek této práce chtěli vyzkoušet. Avšak podporovat všechny DAW není hlavní prioritou a tudíž je nutné zvolit si na které DAW se vývoj zaměří.

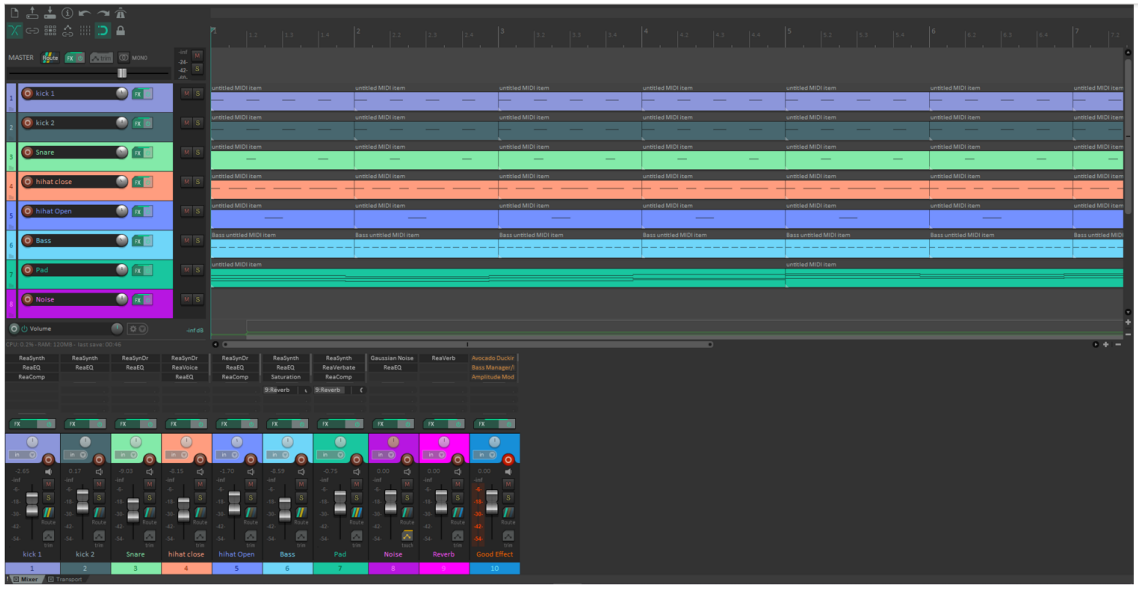
2.1.1 REAPER

Označení REAPER je zkratkou pro Rapid Environment for Audio Production Engineering and Recording. Název v tomto případě vystihuje vlastnosti REAPER. REAPER je rychlý, multiplatformní a nabízí i přenosnou verzi, kterou lze spustit například z USB nebo síťového úložiště. Podporuje MIDI zařízení a vícekanálové směrování zvuku. Zároveň také podporuje virtuální nástroje a plug-in efekty typu VST, VST3, AU a další[22]. REAPER lze využít ve velké škále projektů od masteringu až po studiové nahrávání. Lze využít k post produkci, podcastům i živým vystoupením. Splňuje tak požadavky začátečníků i pokročilejších. REAPER nabízí zkušební verzi na 60 dní, po jejich uplynutí vyžaduje zakoupení licence [1]
Zakoupit lze jednu z variant:

- **Discounted** - určena především pro soukromé a vzdělávací účely, ale i malé komerční projekty.
- **Commercial** - pro uživatele, kteří výdělkem přesahují 20 tisíc dolarů hrubého příjmu ročně.

Příjemné je, že po zakoupení jedné licence není potřeba platit měsíční poplatky. Veškeré nově zakoupené licence zahrnují zdarma upgrady REAPER až do verze 7.99.

[1] REAPER - `https://www.reaper.fm/purchase.php`



Obr. 2.1: Ukázka prostředí digitální audio stanice REAPER.

Velká výhoda REAPER je komunita. Vývojáři na svých stránkách[2] hostují repozitář, do kterého uživatelé přispívají s nejrůznějšími plug-iny, vzhledy uživatelského rozhraní i projekty a zvuky ke stažení.

2.1.2 FL Studio

FL Studio je intuitivní a přehledné DAW od vývojářů Image-Line. Podle průzkumu[21] z roku 2021 je FL Studio nejvíce využívané k produkci elektronické a Hip-hop hudby. Od ostatních DAW se svojí strukturou lehce liší.

Skládá se totiž ze 4 základních bloků:

- **PLAYLIST**
  - Obsahuje tzv. tracky, do kterých lze vložit pattern, zvukový soubor, nebo přímo nahrát stopu z mikrofonu. Také natáhnout nahrávky v časové ose, nebo editovat délku patternů.
- **PIANO ROLL**
  - Uvnitř piano rollu lze najít kromě samotné klaviatury s notami také různé parametry například velocity.
- **CHANNEL RACK**
  - Zde je možnost umístit virtuální nástroj, generátor, nebo zvukový vzorek. Samotný channel rack může mít několik vrstev nazývaných patterny. Ty lze v editoru zvolit a vložit je do playlistu. Každý pattern je notačně

[2]REAPER Resources - `https://stash.reaper.fm/`

oddělen a může tak obsahovat jiné noty pro stejné nástroje, aniž by bylo potřeba přidávat další zdroje.

- **MIXER**
  - V této části DAW je umístěno mixážní zařízení, které dovoluje do jednotlivých slotů – insertů vkládat zvukové efekty, upravovat hlasitost, rozložení do sterea a směrovat do dalších kanálů.
    Zde je zamýšleno vložení výsledku této práce, nejedná se sice o klasický zvukový efekt, avšak aby dokázal provádět granulární syntézu nad živým vstupem, musí být ve struktuře umístěn jako insert efekt.

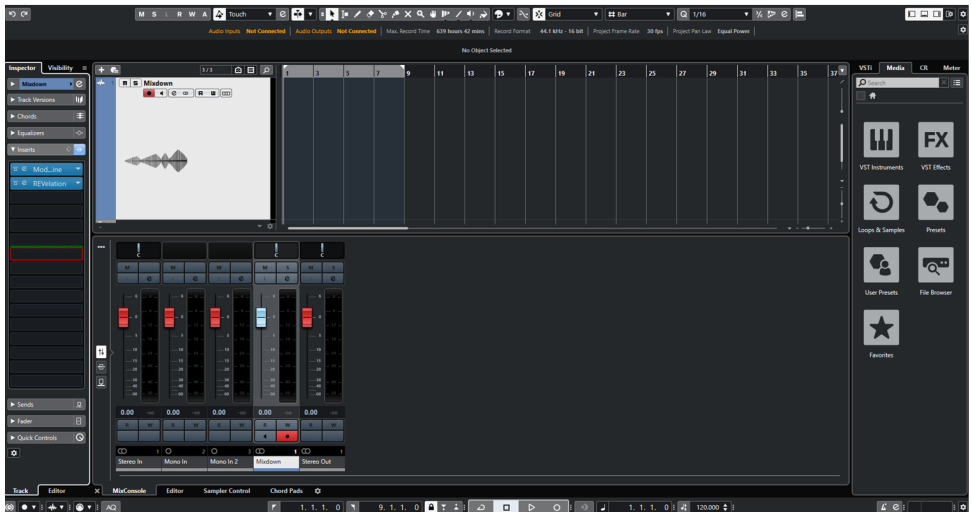
Obr. 2.2: Ukázka prostředí DAW programu FL Studio.

FL Studio je možné koupit v různých variantách od základní verze s mnoha omezeními a hrstkou nativních plug-inů, až po kompletní balíček všech rozšíření a modulů od výrobce. Stejně tak jako tomu bylo u REAPER, koupí-li si uživatel licenci, nemusí již platit měsíční poplatky a narozdíl od REAPER všechny budoucí upgrady, tj. bez omezení na verzi, získá zcela zdarma[3]. K FL Studiu taktéž existuje repozitář s balíčky obsahující zvuky, efekty, nastavení, či nástroje[4]. Ty jsou však placené, nicméně oproti REAPER jsou spravované organizacemi, nebo umělci a jsou na velmi vysoké úrovni. Obdobně jako u REAPER je komunita okolo FL Studia velmi aktivní. Své příspěvky ve velkém zadarmo sdílí po celém internetu, nejčastěji pak na platformě YouTube. Zde lze nalézt kromě ukázek projektů a zvukových balíčků i množství užitečných uživatelských tipů.

[3]FL Studio - `https://support.image-line.com/jshop/shop.php`
[4]FL Studio Sounds - `https://www.image-line.com/fl-studio/sounds/`

2.1.3 Cubase

Cubase je DAW vyvinuta německou společností Steinberg. Umožňuje snadno nahrávat nejen zvuk, ale i MIDI noty, události a různé automatizace parametrů. Cubase je v základním zobrazení rozdělen na několik zón. V okně projektu se na levé straně skrývá Inspector modul. Pomocí tohoto modulu je možno rychle získat informace a ovládat jeden zvolený track. Nachází se zde výběr z nastavení a ovládání z MixConsole. Pravá strana obsahuje modul Media, který zahrnuje VST instrumenty, efekty, zvukové smyčky, a uživatelská přednastavení. V dolní zóně se nachází MixConsole, zde lze nastavovat například panning, hlasitost a přehrávání automatizací.



Obr. 2.3: Ukázka prostředí DAW programu FL Studio.

Oproti FL Studiu lze v Cubase snadněji vkládat send efekty a nastavovat jejich parametry. Avšak nepodporuje zásuvné moduly typu AU. Cubase obsahuje spoustu speciálních prvků jako například Control Room, ten dovoluje nastavit monitorovací prostředí pro hudebníky. Neopomenutelnou součástí softwaru je i prostorový mix pro 5 kanálů Dolby Surround[18]. Cubase lze vyzkoušet na 30 dní zdarma, poté je nutnost za software zaplatit.

Pořidit si Cubase lze ve třech dostupných variantách:

- **Pro** - nulové limitace a omezení
- **Artist** - limitován na 32 fyzických vstupů a výstupů
- **Elements** - omezen například počet MIDI zařízení, audio tracků a VST instrumentů

Některé sady produktů od Steinbergu, či Yamahy obsahující například zvukové karty nebo kontrolery přidávají do balíčku zdarma Cubase AI. Jedná se o kompaktnější verzi Cubase používající stejné technologie jako varianta Cubase Pro[17].

2.2 Technologie zásuvných modulů

2.2.1 VST3

Jak již název napovídá VST3 je třetí verzí původního formátu plug-inů Virtual Studio Technology pocházejícího z roku 1996, kdy jej vydala firma Steinberg. Jedná se o nejčastěji používaný plug-in formát. Na oblíbenosti mu přidává fakt, že je multiplatformní. To znamená, že podporuje komunikaci s prostředky systému jak Windows, tak i macOS[13].

Obr. 2.4: Logo technologie VST3[34].

Tato technologie byla vyvinuta za účelem nahrazení hardwarových zařízení používaných ve studiu za pomoci počítačové emulace. Postupem času se tato technologie zdokonalila na tolik, že dokázala věrohodně napodobit i hudební nástroje, tomuto využití se potom konkrétněji říká VSTi, nebo-li VST instrument. Všeobecně se jedná o syntezátory a samplery. Jejich užití dalo vzniknout širokému spektru hudebních nástrojů, od klavírů a kytar, přes nástroje dechové, až po perkuse[11].

VST lze však použít i k vytvoření MIDI plug-inů, ty většinou zjednodušují práci s notami. Jako příklad lze uvést arpeggiator, který jako vstup vezme MIDI noty akordu. Ten nepřehraje jako několik not najednou, místo toho podle nastavení přehraje sekvenci těchto not o určité rychlosti a v určitém pořadí. Nelze opomenout ani využití VST plug-inů v zobrazování vlastností zvukového signálu, kde skvěle reprezentují například úroveň aktuálně přehrávaného signálu, nebo jeho frekvenční spektrum[10].

Třetí verze oblíbeného formátu oproti svým předchůdcům přinesla řadu pozitivních vylepšení. Podporuje nové konfigurace reproduktorů jako Ambisonic, Atmos nebo Auro 3D. Dovoluje přistupovat k informacím o aktuálním kanálu, kde je plug-in vložen. Důležitou novinkou je i způsob práce s přivedeným signálem, pokud na kanál s vloženým plug-inem není přiveden žádný signál, plug-in funguje mnohem úsporněji při zpracovávání bufferu.

Balíček pro vývoj je volně dostupný na stránce `https://github.com/steinbergmedia/vst3sdk`, obsahuje plug-in API, avšak navíc přidává pomocné třídy, vhodné k zabalení a implementaci formátů AAX AU i staršího VST2.

2.2.2 AUv3

AudioUnit verze 3, zkráceně AUv3, je formát plug-inů vyvíjen společností Apple především pro platformy iOS a macOS. Bývá označován za MAC ekvivalent VST na platformě Windows, i když to není úplně přesné přirovnání. Obsahuje třídy a nástroje umožňující využít Core Audio k vývoji zásuvného modulu. Core Audio je API, které zajišťuje komunikaci a přístup k infrastruktuře digitálního zvuku na macOS. Což ovšem znamená, že je AU limitováno pouze na tuto platformu a například na Windows fungovat nebude. Nicméně na macOS zajišťuje nízkou latenci a je velmi stabilní. Jedná se tedy o něco lepší variantu než-li VST pro macOS[12]. Pokud je však potřeba sdílet projekty s uživateli systému Windows, je lepší použít VST pro jeho univerzálnost[9]. Balíček pro vývoj softwaru je možné nalézt v repozitáři `https://github.com/apple/AudioUnitSDK`.

Obr. 2.5: Logo Apple AU[32].

2.2.3 AAX

AvidAudioeXtention jako nástupce předchozího formátu plug-inů RTAS vyvinula společnost AVID v roce 2011. Tato technologie může být bez omezení využita k vývoji efektů a hudebních nástrojů. AAX je schopen pracovat jak na počítači, tak i na externím akceleračním DSP hardwaru. Není totiž závislý na prostředí počítače,

což je jeho největší přednost a důvod proč byl tento formát vytvořen[6]. Zároveň je vhodný na rozsáhlejší projekty, které vyžadují větší množství plug-in modulů, zde u VST může docházet k chybám automatizací. Výhodu získává také pokud je použit v kombinaci s prostředím Pro Tools, zde si vývojáři dobře zahráli do karet. Vytvořit si vlastní formát pro vlastní DAW přináší jistě řadu výkonnostních výhod[8].

Velkou nevýhodou je fakt, že vývojářský balíček není volně dostupný jako u již zmíněných formátů a k jeho získání je nutno souhlasit s developerským programem AVID, což zahrnuje různé smluvní podmínky ohledně marketingu, poplatků a prodeje pluginu[7].

Obr. 2.6: Logo technologie AVID AAX [33].

2.2.4 Výběr technologie

Jednoznačnou volbou pro tuto práci byl formát VST3. Nejen že je multiplatformní, je také nejvíce používaný a podporovaný napříč DAW programy. Což značí i větší množství dokumentace a dostupných článků. Případné potíže a neočekávané komplikace při vývoji tak bude možné vyřešit snadněji.

2.3 Vývojové frameworky

Z důvodu času vymezeného na vypracování této práce, by nebylo možné napsat zamýšlený plug-in modul úplně od nuly. Je tedy nutné přiklonit se k využití frameworku, což je nástroj, který umožňuje práci s nejvíce základními prvky audio knihoven. Základním prvkem je myšlena především komunikace s audio infrastrukturou a externím hardwarem na nízké programovací úrovni. Za použití knihoven programátor nemusí vytvářet zmíněné prvky zcela od začátku, místo toho si může

pouze upravit základ, na kterém postaví svoji aplikaci[5]. Stejně jako tomu je při výrobě automobilu, je nesmyslné znovu vynalézat koncept kola, je však možné upravit jeho rozměry a dosáhnout tak lepších jízdních vlastností.

2.3.1 JUCE

JUCE je open-source framework určen k vývoji zásuvných modulů, ale i stand-alone aplikací. Vytvořen Julianem Storerem pod licencí GPL dává koncovému uživateli svobodu spouštět, sdílet a vytvářet změny ve zdrojovém kódu softwaru. Licence je přístupná na stránkách [5]. Pokud při vydání vývojář zvolí, že chce svoji aplikaci sdílet bez úmyslu finančního zisku, nemusí platit poplatky organizaci JUCE. Pokud by však vývojář chtěl svůj výtvor zpeněžit, musí tak učinit podle licenčních podmínek nacházejících se v repozitáři `https://github.com/juce-framework/JUCE/blob/master/LICENSE.md`.

Aktuální druhy licencí jsou:

- **Personal** - zdarma při výdělku pod $50'000
- **Indie** - pro firmy s výdělky pod $500'000 zpoplatněna $40 měsíčně
- **Pro** - bez limitace výdělku za $130 měsíčně

Psaní v JUCE vyžaduje znalost programovacího jazyka C++. Jak již bylo zmíněno dříve, tak i tento vývojový framework se skládá z tříd a metod poskytujících řešení základních problému, na které lze narazit při vývoji softwaru od úplného začátku. Konkrétněji jde o vykreslování grafických objektů, export a sestavení finálního plug-inu. Součástí jsou také nové datové typy, jenž nejsou zahrnuty v základu C++, jako například datový typ String.

Jednoduchost a přehlednost frameworku JUCE je velkou výhodou. Zásadní pro přehlednost je také dobrá dokumentace tříd a metod frameworku, tu lze nalézt na stránce `https://docs.juce.com/master/index.html`. Obsahuje všechny potřebné informace k použití tříd a metod. Občas i krátkou ukázku zamýšleného použití.

Za nejdůležitější článek frameworku lze považovat základní třídu AudioProcessor[6]. Je předurčena ke komunikaci s DAW hostitelem. Kromě jiných zahrnuje dvě zásadní metody prepareToPlay a processBlock. Metoda prepareToPlay je systémem volaná funkce, volá se při inicializaci plug-inu, nebo při změně vzorkovací frekvence v DAW. Obsahuje aktuální hodnotu vzorkovací frekvence a buffer size. Tyto dvě hodnoty se totiž mohou v průběhu používání DAW měnit a aplikace by pak nemusela fungovat správně. Opomenout nelze ani třídu AudioProcessorEditor[7], ta zajišťuje

[5] GPL-`https://www.gnu.org/licenses/gpl-3.0.en.html`

[6] AudioProcessor - `https://docs.juce.com/master/classAudioProcessor.html#details`

[7] AudioProcessorEditor - `https://docs.juce.com/master/classAudioProcessorEditor.html`

spojení s uživatelským rozhraním nebo-li GUI. GUI má za úkol vykreslovat objekty uživateli na obrazovce, ať už se jedná o to, jaké má tlačítko rozměry, tvar či barvu.

Okolo JUCE existuje taktéž značně aktivní komunita uživatelů. Pokud tedy po přečtení dokumentace není zřejmý způsob využití a implementace, může se vývojář obecně poradit na oficiálním diskuzním fóru[8]. Možností je také odkázat se na jeden z návodů dostupných na stránce[9]. Návody se dotýkají základních částí, které pravděpodobně každý plug-in obsahuje. JUCE podporuje export zkoumaných formátů VST3, AUv3 i AAX[10]. Výstupem JUCE, jak už bylo zmíněno, nemusí být jenom zásuvný modul, ale i stand-alone aplikace. Dále také statická, či dynamická knihovna, která může tvořit základ nebo rozšíření pro další vývoj audio softwaru.

Obr. 2.7: Logo frameworku JUCE[35].

2.3.2 iPlug2

Dalším vývojovým frameworkem je iPlug2. Stejně jako JUCE je cross-platform, což znamená že vývoj může probíhat na vícero operačních systémech. Druhá verze iPlug byla vydána v roce 2018. Taktéž podporuje formáty VST3, AUv3 i AAX a vývoj probíhá v jazyce C++. Framework rovněž umožňuje vývoj rozšíření REAPER extentions a webových audio API, taktéž známé pod názvem WAM. Od JUCE se liší podporou jazyku FAUST. Ten je popisován jako snadněji pochopitelná alternativa C++[11]. Po licenční stránce vývojáře iPlug2 potěší, že lze s frameworkem nakládat bez omezení i co se týče finančních odměn[12].

Obdobně jako u JUCE iPlug nabízí řadu kódových ukázek, které však slouží opravdu jen jako ukázky a mnohdy k nim není ani dokončený popis. Takže je pro

[8] Fórum JUCE -`https://forum.juce.com/`

[9] Návody JUCE -`https://juce.com/learn/tutorials`

[10] Exportování -`https://docs.juce.com/master/tutorial_manage_projucer_project.html#tutorial_manage_projucer_project_managing_configurations`

[11] FAUST -`https://faust.grame.fr/`

[12] Licence iPlug - `https://github.com/iPlug2/iPlug2/blob/master/LICENSE.txt`

nové vývojáře složitější se v tomto frameworku orientovat. Na znalostech tříd a metod nepřidá ani nekompletní dokumentace. Prozatím obsahuje sice krátký, ale dobrý popis základních částí. Například instalace frameworku, nebo výsledný export zásuvného modulu. I přes jeho kladné vlastnosti je největší nevýhodou velikost skupiny, která tento framework vyvíjí. Nejde o desítky vývojářů jako tomu bylo v organizaci JUCE a narozdíl od JUCE je iPlug skupina závislá na sponzorských darech. Dary lze posílat skrze stránky repozitáře. Vývojáři neposkytují žádnou oficiální uživatelskou podporu jako JUCE.

Obr. 2.8: Logo frameworku iPlug[36].

2.3.3 Výběr frameworku

Z důvodu perspektivy budoucího vývoje a podpory je pro tuto práci vhodnější volbou framework JUCE. Každopádně velký respekt patří oboum hlavním vývojářům iPlug2, kteří na svém frameworku pracují dobrovolně a za pomoci sponzorských darů se ve svém volném čase stále snaží tento framework vylepšovat.

2.4 Průzkum plug-inů

Při tvorbě této práce bylo nutné prozkoumat i již vytvořené a dostupné plug-iny zaměřující se na granulární syntézu. Následující zásuvné moduly reprezentují dva běžné typy softwarové granulární syntézy, které jsou často používáné i v ostatních plug-inech s touto tématikou.

2.4.1 Quanta 2

Prvním typem je použití syntezátoru přímo jako generátoru zvuku. Vložit jej lze na libovolný track a do provozu je uveden po vložení předem známého zvukového vzorku v podobě souboru. Dokáže přijímat MIDI noty, podle kterých přehrává následně jednotlivé granule.

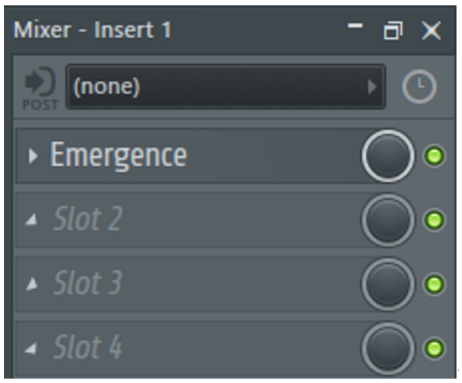
Obr. 2.9: Ukázka Quanta 2 v DAW FL Studio.

V Quanta 2 lze nastavit počet hlasů, syntezátor je totiž schopen hrát více hlasů najednou. Nastavení granulární syntézy lze modifikovat nízkofrekvenčními oscilátory, u kterých je například možnost nastavení fáze, tvar průběhu a frekvence opakování. Mezi hlavní parametry samotné syntézy patří například tvar obálky, výchozí pozice přehrávání a délka zrn. Plugin Quanta2 je možné zakoupit na stránkách `https://www.audiodamage.com/collections/plugin-instruments/`.

Obr. 2.10: Ukázka demo verze plug-inu Quanta 2.

2.4.2 Emergence

Druhým typem zásuvných modulů zabývajících se granulární syntézou je Emergence. K uvedení do provozu potřebuje být vložen na efektovou sběrnici libovolného tracku.



Obr. 2.11: Aplikování Emergence jako efektu v FL Studio.

Není třeba do něj načítat externí soubory, ani mu posílat MIDI zprávy. Jeho vstupem je samotný aktuálně přehrávaný zvuk tracku. Zvuk se neustále načítá do interního bufferu o konstantní rychlosti. Po načtení maximální možné délky vzorků do bufferu se přepisují předchozí načtené vzorky od začátku. Tímto způsobem má plug-in zajištěno dostatečné množství vzorků, bez nichž by nebyl schopen provést syntézu.

Zajímavé parametry, které Emergence obsahuje, jsou například rychlost, směr přehrávajícího kurzoru a počet přehrávaných granulí. Kurzor je výchozím bodem odkud vznikají další granule. Může se pohybovat od začátku až po konec celého bufferu v nekonečné smyčce. Podobnost s předchozím zkoumaným plug-inem lze nalézt ve využití nízkofrekvenčních oscilátorů. Obdobnou funkci hlasů v Quanta 2 zde zastupuje takzvaný Stream nebo-li tok. Stream sjednocuje parametry pro jeden zvukový tok a jeho výstup se přimíchá k hlavnímu zvukovému výstupu plug-inu. Plug-in Emergence od vývojáře Daniela Gergely je možné zadarmo získat na stránkách. `https://www.kvraudio.com/product/emergence-by-daniel-gergely`

Obr. 2.12: Grafické rozhraní plug-inu Emergence

2.4.3 Poznatky

Dle osobního testování výše zmíněných aplikací, lze říci, že spojení vizuální stránky se zvukovou je velmi uživatelsky přívětivé. Uživateli dají představu o aktuálně přehrávaném zvukovém vzorku a doplní jeho požitek z tvorby o drobné graficky zajímavé prvky a animace. Nezáleží tolik na dokonalosti technologických postupů, ale spíše na přehledném a funkčním uživatelském prostředí, od něhož uživatel očekává souhru se zmíněným algoritmem.

Jednoduše řečeno uživateli bývá daleko příjemnější, když je grafické prostředí přímočaré. Když nastíní, jaké parametry lze nastavit a jakými způsoby docílí onoho nastavení. Důležité je, aby prostředí reagovalo na uživatelské změny v co nejkratším čase a dalo tak uživateli zvukovou odezvu. To však nemusí znamenat zajištění kvalitnější tvorby zvuků. Nicméně je to věc, která zprostředkuje uživateli větší svobodu tvorby.

Parametry jsou pro granulární syntézu alfa a omega. Při vývoji bude tedy důležitá konzistence základních parametrů a označení rozsahů. Parametrů nesmí být velké množství, to ubírá na přehlednosti a intuitivnosti aplikace.

3 Implementace

Vývoj zásuvného modulu probíhal v několika fázích. Počáteční fáze zahrnovala sepsání požadavků na funkce plug-inu. V druhé fázi probíhalo navrhování obecných principů. Následně bylo přistoupeno ke psaní kódu. Při každém novém zjištění, se vývoj vrátil o krok zpět. Po promyšlení a aplikování získaných poznatků se fáze posunula opět vpřed a vývoj pokračoval dále.

Jelikož bylo nutné vytvořit vícero odlišných modulů ke splnění zadání, rozdělily se tyto moduly do jednotlivých projektů. Příkladem modulu je samostatná granulární syntéza nebo modul LFO. Přičemž všechny vycházely ze stejného základu, takže například funkce tlačítek a nastavení jsou stejné pro všechny moduly. V závěru vývoje byly tyto moduly implementovány do jednoho projektu z důvodu celistvosti zásuvného modulu.

Vývoj softwaru s sebou nese jistá úskalí, z toho důvodu bylo přistoupeno k využití verzovacího systému git. Ten zajistí zálohu a zachování všech historických verzí programu. Díky tomu se lze v případě výskytu problému velmi dobře dopátrat jaká změna vzniklý problém přinesla.

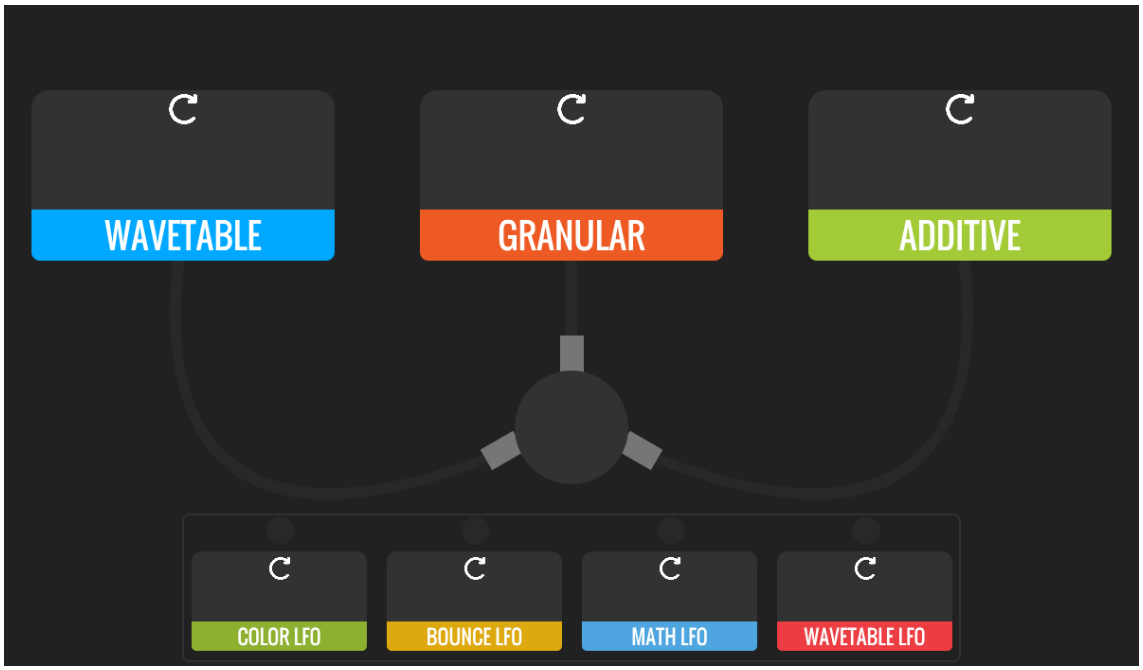
Kompletní kód všech modulů i finální plug-in je veřejně dostupný na stránce GitHub. `https://github.com/GranularFlow`

3.1 Wrapper modul

Wrapper je základní modul, který uživatel po přidání výsledného plug-inu do DAW uvidí. Obsahuje implementaci všech tří modulů zvukové syntézy. Ve spodní části se nachází čtyři moduly s experimentálními způsoby získávání hodnot pro následné nastavení parametrů.

Návrh grafického rozhraní udává obecnou představu o směrování zvuku v plug-inu. Výstupem finálního plug-in modulu je pouze zvukový tok. Nevrací žádné nastavení ani MIDI zprávy. Výstupní zvukový tok se skládá ze součtu signálů všech syntezátorových modulů. Úroveň hlasitosti zde není automaticky vyvažována, dá se však upravit jednotlivě v každém modulu. Tím uživatel dosáhne tíženého mixu syntetizovaných zvuků. To samé platí i pro stereo rozložení, jehož nastavení lze rovněž najít v jednotlivých modulech syntezátorů.

Funkce, kterou nelze z přiloženého obrázku úplně vypozorovat, je vypnutí jednotlivých modulů. Vypnout modul lze kliknutím myši na grafické vyobrazení kabelu. Po praktické stránce takovéto vypnutí znamená ušetření paměti a procesoru, jelikož se metody, které plní buffer generovaným zvukem a metody provádějící výpočty úplně přeskočí v hlavním bloku plug-inu.



Obr. 3.1: Grafické zpracování návrhu wrapper modulu.

Algoritmus celé aplikace probíhá ve dvou hlavních vláknech, jedná se o AudioThread a MessageThread. Vlákna jsou v rámci procesu úkony, které vykonávají činnost nezávisle na sobě. Pokud by se plug-in zpracovával pouze v jednom vlákně, docházelo by ke zvukovým výpadkům. Výpadky by byly způsobeny dlouhým čekáním na dokončení vykreslování uživatelského prostředí. Vlákna se v případě této práce zabývají například čtením hodnot po změně parametru z uživatelského rozhraní. Jsou využity také při funkcích časovače a překreslování uživatelského prostředí. Výsledkem práce vláken se stává také například překreslení stavu rotace knoflíků, nebo posun kurzoru.

Pomocné funkce

Některé výpočty bylo nutné provést několikrát. Aby nedocházelo k opakování, byla vytvořena statická třída Utils obsahující všeobecné funkce k výpočtu. Nejčastěji používanými funkcemi byly například převody času na dobu trvání vzorků a zpět. Dalšími často užívanými funkcemi byly převod desetinného čísla na procenta nebo přidání grafického prvku do mřížky.

Kromě třídy Utils byl vytvořen i soubor s konstantami. Obsahuje především konstantní definice barev a obecných parametrů jako šířka a výška okna v pixelech.

3.2 Modul granulární syntézy

V modulu granulární syntézy lze nalézt hned po jeho rozkliknutí několik základních nastavení. Jelikož by nebylo vhodné hodnoty některých nastavení měnit za chodu, jsou dostupné pouze před tím, než si uživatel vybere, zda chce granulární syntézu realizovat na živém vstupu, nebo na vzorcích ze souboru. Po tomto výběru již nelze nastavení měnit, jediný způsob jak dosáhnout obnovení tohoto nastavení je použití tlačítka reset, které se nachází v úvodním modulu wrapper. Po kliknutí na resetovací tlačítko se uživatelské nastavení pro granulární syntézu z dřívějška navrátí do původních hodnot. To zahrnuje i nastavení výběru živého vstupu, nebo načteného souboru.

Parametry modulu granulární syntézy

- **BUFFER / FILE** - tlačítka buffer i file zahájí inicializaci vzorků. Pokud uživatel klikne na tlačítko file, zobrazí se okno výběru zvukového souboru. Je nutné zmínit, že momentální výběr je omezen pouze na soubory WAVE. Pokud však uživatel klikne na tlačítko buffer, vzorky pro granulární syntézu se načtou až v průběhu používání ze zvukového kanálu, kam byl tento plugin umístěn.
- **Length** - nastavuje maximální velikost interního bufferu se vzorky. Důležité je upozornit, že tento buffer obsahuje maximální počet diskrétních vzorků, které aktuálně drží v paměti. Při syntéze ze souboru načte pouze tento počet vzorků souboru od začátku. V případě syntézy vzorků z živého vstupu modul načítá pouze do velikosti tohoto počtu vzorků. Při následném zaplnění přemazává vzorky již nahrané a jejich počet se nemění. To je obzvlášť žádané, jelikož se nezmění ani velikost proměnné, která by kvůli realokaci paměti v reálném čase způsobovala značné zpomalení a chybové stavy programu.
- **Count** - určuje kolik aktivních hlasů uživatel chce přehrávat najednou. Hlasy jsou v tomto plug-inu pojmenovány jako players. Player zahrnuje v grafickém prostředí kurzor, jeho nastavení a všechna jím generovaná zrna.
- **Select** - vybírá aktuální player, s nímž chce uživatel pracovat. Po zvolení playeru se ve spodní částí zobrazí jeho parametry a nastavení generovaných zrn.

Parametry granular playeru

- **MODE** – režim přehrávání granulí. ORDER přehrává granule od kurzoru směrem dopředu v čase. MIRROR zase průběh granule zrcadlí od kurzoru a vytváří tak vždy jednu granuli navíc. Možností je také REV.ORDER nebo-li přehrávání granulí pozpátku. Granule začíná vždy vzorkem na pozici kurzoru.
- **CURSOR** – možností tohoto parametru je uvedení kurzoru do chodu. Kur-

Obr. 3.2: Ukázka modulu granulární syntézy.

zor se v čase lineárně pohybuje po celé délce bufferu, ne však o stejný počet vzorků, o který se plní buffer. Tím vznikají velmi zajímavé zvuky, jenž v tichých pasážích tvoří příjemný a proměnlivý ambientní zvuk. Kurzor může také staticky stát na uživatelem určeném místě. Výhodou kurzoru je možnost posuvu v obou režimech. Uživatel si nemusí vybírat pozici kurzoru ovládacím knoflíkem, nýbrž si kurzor libovolně přetáhne myší.

- **MIDI** – parametr pro zapnutí příjmu MIDI, při režimu OFF syntéza probíhá neustále ve smyčce. S režimem ON se generování zvuku spustí až při MIDI události NOTE_ON.
- **WINDOW** – umožňuje zvolit si jeden ze tří typu zvukové obálky, která se aplikuje na jednotlivé granule. Dopomáhá plynulejšímu prolnutí granulí mezi sebou, bez této obálky by docházelo k vjemu praskání. Momentálně je na výběr z tvarů kladné půlvlny sinusoidy, Hannova okna a trojúhelníku.
- **LENGHT** – určuje délku jedné granule. Pro přehlednost je parametr udáván v milisekundách a může být nastaven až na jednu celou sekundu.
- **PITCH** – ovládá dodatečný posun výšky tónu, z praktického hlediska ovšem funguje zrychlením přehrávaného vzorku. Neimplementuje techniky, které by zajistili posun výšky při zachování stejného časovém úseku.
- **GRAINS** – nastavuje kolik granulí se má přehrávat. Nutno zmínit, že nová granule se vytvoří až po určitém časového úseku od vytvoření předchozí granule.

- **OFFSET** - nastavení již zmíněného časového úseku. Po vytvoření jedné granule určuje čas, po který se čeká, než je vytvořena granule další a mezi granulemi vytváří tedy časové mezery. V kombinaci s nastavením parametru GRAINS vzniká parametr, který v je v některých plug-inech označován jako density.
- **VOLUME** – udává hlasitost playeru a všech jeho zrn.
- **PAN** – parametr pro rozložení přehrávání granulí do sterea. Výchozí bod je v polovině otočného knoflíku a nastavení stran sterea se shoduje se směrem otočení tohoto knoflíku.

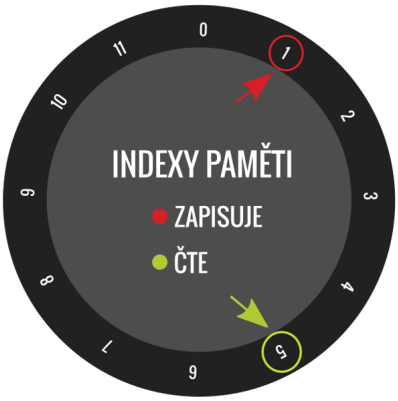
Algoritmus za návrhem modulu granulární syntézy je závislý především na počátečním výběru uživatele. Chce-li syntetizovat ze souboru nebo živého vstupu. Každá z těchto možností se následně zpracovává trochu odlišně.

Způsoby zpracování granulární syntézy

- **Syntéza ze souboru** – Pro tento typ syntézy byl vytvořen modul AudioLoad, ten při zavolání z jiné třídy vybere soubor a načte jej se správným kódováním do bufferu. Jelikož je při práci v programovacím jazyce C++ nutné být opatrný s alokací paměti. Do této metody se předává pouze odkaz na audio buffer ze třídy, která tuto metodu volá. Tím pádem není nutné vytvářet nový audio buffer.
- **Syntéza z živého vstupu** – Syntezovat živý vstup možná není úplně správný název. K syntéze je potřeba vícero vzorků, které nelze najednou získat. Proto bylo využito takzvaných circular bufferů. To jsou speciální druhy bufferů, které na jedné straně načítají datový vstup a na druhé jej dokáží číst ve stejném pořadí v jakém data přišly. Mají pevně danou velikost, což znamená, že nezatěžují tolik počítač realokací. V tomto případě se jedná o skupinu vzorků. Ty jsou na začátku všechny nastaveny na nulovou hodnotu a postupně se v čase přepíší na skutečné hodnoty načítané ze vstupu. Jakmile se načte určitý počet vzorků a buffer se zaplní, zapisování nekončí. Buffer přesune index opět na začátek a přepíše již předtím vloženou hodnotu, tím může zvuk zůstat relativně aktuální a více na sebe navazuje v čase.

Modul granulární syntézy pokaždé vyžaduje automatickou kontrolu a případnou korekci velikosti předávaného bufferu. Mohlo by totiž dojít k chybám ve zvukovém projevu. Když DAW zavolá VST API, očekává, že metoda hlavní části plug-inu processBlock naplní předaný buffer vzorky z interního bufferu granulární syntézy. Kvůli tomu se v této metodě volá pomocná metoda z modulu granulární syntézy. Ta zkontroluje přítomnost zdrojových vzorků, ošetří MIDI vstup a naplní předaný buffer všemi vzorky granulí instance player.

Každý player v průběhu zvukového vlákna zkontroluje počet vlastních zrn, pří-



Obr. 3.3: Ilustrace modelu circular bufferu.

padně vytvoří požadavek na vznik zrna nového a předá mu všechna dostupná nastavení. Mezi vytvořením požadavku na vznik jedné a druhé granule se čeká po počet vzorků nastavený parametrem OFFSET z uživatelského prostředí. Tato prodleva je přidána z důvodu, aby všechny granule nezačaly brát vzorky ze stejné pozice a se stejnými parametry najednou, ale s určitým časovým rozestupem. Pokud by zrna byla vytvořena najednou, došlo by pouze ke zvýšení úrovně hlasitosti a nejednalo by se úplně o granulární syntézu.

Při novém požadavku na přidání granule by dávalo smysl vložit její nově inicializovanou instanci do pole třídy player, avšak opět by se realokovala paměť s velikostí pole. Je tedy lepší mít předem definovanou velikost takovéhoto pole s granulemi a pouze kontrolovat, zda se má příslušná granule v tomto poli přehrát, nebo zda-li je připravena pro další použití. Pokud je granule označena k přeskočení, nejsou její vzorky předávány do hlavního bufferu. Při následném požadavku o přidání další granule je takto označená granule naplněna novými parametry, označení pro přeskočení se zruší a její vzorky se projeví ve výsledném zvuku. Po přehrání všech vzorků se granule sama označí k následnému přeskočení a tento cyklus se opakuje.

Na vzorky zrna se následně aplikuje obálka, podle toho v jaké fázi přehrávání se právě nachází. Poté se přičte hodnota desetinného čísla vzorku z aktuální pozice zrna s nastaveným stereo rozložením do obou kanálů předávaného bufferu. To je velmi důležité, jelikož stereo dodává poslechu granulární syntézy ten pravý požitek a nebylo by příjemné o tento prostorový vjem přijít a celou syntézu zpracovávat pouze mono.

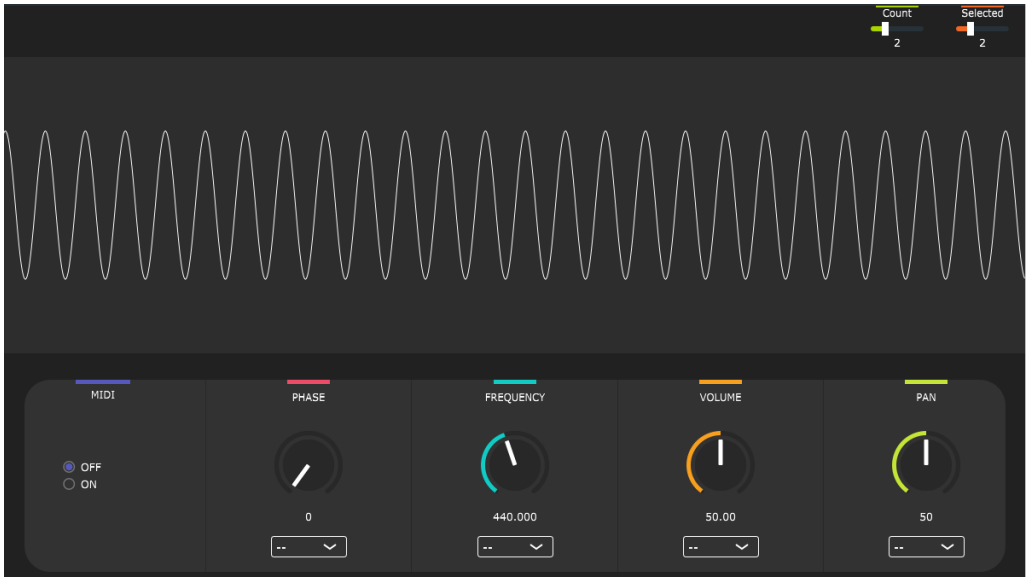
Co se týče zobrazení zvuku to probíhá za pomoci třídy GranularVisualiser. Převezme hodnoty vzorků zdrojového bufferu a vykreslí jejich amplitudu v nastaveném rozsahu. Vykreslení probíhá za pomoci pole desetinných čísel, není tedy využit všude

přítomný datový typ AudioBuffer. Ten totiž obsahuje informace, které nejsou pro vykreslení potřebné, což teoreticky ušetří další malý kousek výpočetní síly.

Do oblasti v horní části modulu granulární syntézy se vykresluje jeden zvukový kanál zmíněného zdrojového bufferu. Ten je však tvořen oběma kanály najednou. Zobrazuje totiž vzájemně překryté amplitudy z obou kanálů. V metodě, která plní pole vzorků pro vykreslení se algoritmus rozhoduje, která amplituda ze dvou vstupních kanálu je ta větší a následně ji zapíše jako další vzorek do tohoto pole. Uživatel tedy vidí jen tu nejvyšší hodnotu a nemůže tak dojít k přehrání nečekané hlasitosti v daném kanále, která by mohla uživatele nemile zaskočit. Nebyla by graficky vykreslena a uživatel by na ni nemusel být připraven.

3.3 Modul aditivní syntézy

Aditivní syntéza obsahuje v horní liště svého uživatelského rozhraní všeobecná nastavení složek. Ve spodní části lze poté nastavit vlastnosti vybrané složky.



Obr. 3.4: Návrh modulu aditivní syntézy.

Parametry aditivní syntézy

- **Count** – obdoba nastavení počtu playerů v modulu granulární syntézy. Count poskytuje možnost přidání či odebrání složky syntézy.
- **Selected** – volí zobrazení parametrů dané složky syntézy, které následně může uživatel modifikovat. Tyto změny zůstanou uložené až do chvíle, kdy se sníží

celkový počet složek. Při snížení počtu složek aditivní syntézy dochází k resetování hodnot poslední odebrané složky. Ostatní však zůstávají beze změn.

Parametry složky aditivní syntézy

- **MIDI** – tak jako u granulární syntézy je i zde možnost si nastavit pro každou složku syntézy, zda se má přehrávat v uzavřené smyčce, nebo až po přijmutí MIDI zprávy NOTE_ON.
- **PHASE** – nastavuje fázi signálu. Projevuje se hlavně v případě, že se přehrávají dvě a více složek najednou, protože každá složka aditivní syntézy je synchronizována tak, aby bez nastavení PHASE měla nulovou fázi. Tento parametr je udáván ve stupních. Tedy v případě, že na jedné složce bude nastavená PHASE na 0 a druhá složka bude mít nastaveno PHASE na 180, vlny se kompletně vyruší.
- **FREQUENCY** – jak již anglický název napovídá, tento parametr nastavuje frekvenci aktuální složky.
- **VOLUME** – udává hlasitost složky, je důležité vyvážit tuto hlasitost při přidání vícero složek, jelikož momentálně aditivní syntéza neobsahuje jedno hlavní nastavení hlasitosti.
- **PAN** – i zde parametr PAN nastavuje rozložení do sterea. Opět je nutné myslet na to, že modul additivní syntézy neobsahuje společný PAN a je nutné jej nastavit podle potřeby pro každou složku zvlášť.

Zde probíhá algoritmus zpracování signálu značně jednodušeji než-li tomu je u granulární syntézy. Modul aditivní syntézy obsahuje předem definované pole instancí takzvaných harmonických, avšak jde pouze o symbolické pojmenování, jejich frekvence je totiž určena výše zmíněným parametrem a nemusí se tak přesně jednat o násobky základní frekvence. Pole s těmito složkami je při volání metody processBlock postupně procházeno. Při každé takovéto iteraci je instance dotazována o naplnění hlavního bufferu příslušným počtem vzorků. V tomto dotazu se zároveň ošetřují MIDI data, které algoritmus nastaví podle čísla MIDI_NOTE_NUMBER frekvenci a vypočítá následný posun úhlu dalších vzorků. To je velmi důležité, pokud by se totiž nevypočítal rozdíl úhlů aktuálního a následujícího vzorku, nedocházelo by k hladkému přechodu mezi frekvencemi, nýbrž ke zřetelným skokům, doprovázených nepříjemným praskáním.

Při požadavku o přidání nové složky se jako u požadavku na vznik granulí nevytváří žádná nová proměnná, pouze se zvýší interní kontrolní počet iterovaných složek pole. Jak již bylo popsáno u parametru Selected, pokud dojde ke snížení počtu složek a uživatel editoval poslední složku z dostupného počtu, její parametry se nastaví na původní hodnoty a uživateli tak zruší jeho osobní nastavení. Pokud však uživatel použije k výběru z dostupných složek Selected všechny nastavené parametry

zachovají uživatelské nastavení.

3.4 Modul wavetable syntézy

Zajímavostí modulu wavetable syntézy je možnost kreslení vlastního průběhu vln. Uživatelské prostředí obsahuje v horní polovině tři bloky reprezentující pomyslná plátna, ve kterých může uživatel nakreslit až tři tvary vln stisknutím a táhnutím myši.

V této vymezené oblasti se v průběhu uživatelova tahu plní pole vzorků. Každé plátno představuje pole s obsahem 100 diskrétních vzorků. Jakmile uživatel svůj tah ukončí, automaticky se dopočítají chybějící vzorky, které by nemusely být zaznamenány rychlým tahem myši. Do tohoto procesu se zahrnují i vzorky kvůli výchozí pozici kurzoru při stisku a puštění myši, která by nemusela být přesně na začátku a konci bloku plátna. Všechny zmíněné vzorky se na vybraném plátně zobrazí a uživatel může pokračovat v kreslení další vlny. Pokud by se svou kresbou vlny nebyl spokojen, stačí tah opakovat, pole se při prvním stisku myši v oblasti plátna vyprázdní a čeká na pohyb kurzoru.

Po stisknutí tlačítka SYNTHESIZE, které se nachází nad zmíněnými plátny, se provede lineární interpolace mezi dvěma sousedními vlnami. Výsledný signál s dopočítanými vzorky je zobrazen na posledním plátně této řady. Je světlejší barvy a nedá se do něj samostatně kreslit.

Výsledný signál je tedy složen ze vzorků v tomto pořadí

- **FIRST WAVE** - počáteční nakreslená vlna
- dopočítané vlny - dopočítávají se z první a druhé nakreslené vlny
- **SECOND WAVE** - druhá nakreslená vlna
- dopočítané vlny - dopočítávají se z druhé a třetí nakreslené vlny
- **THIRD WAVE** - poslední uživatelem nakreslená vlna

Parametry wavetable syntézy

- **MIDI** – tak jako u předchozích syntéz je i zde možnost si nastavit, zda se má signál přehrávat ve smyčce, nebo až po přijmutí MIDI události NOTE_ON.
- **FREQUENCY** – tento parametr určuje s jakou četností se má výsledná vlna se všemi vzorky přehrát za jednu sekundu.
- **WAVE COUNT** – nastavuje počet celkových vln k dopočítání. Pro doplnění mezi dvěma nakreslenými vlnami je použit tedy jen poloviční počet dopočítaných vln.
- **INTERPOLATION** - k výběru jsou dva druhy interpolací, výchozí volbou je lineární interpolace, lze si však zvolit i interpolaci kubickou. V praxi se při

wavetable syntéze kubická interpolace oproti lineární chová jako filtr horní propusti, lze slyšet více složek vyššího spektra. Je nutné zmínit, že se jedná o interpolaci pouze výsledného signálu se všemi vzorky, nikoliv doplňujících vln.

- **VOLUME** - slouží k nastavení celkové hlasitosti syntezátoru, všechny nakreslené i dopočítané vlny jsou tímto parametrem ovlivněny rovnocenně.
- **PAN** – parametr umožňující rozložení výsledného zvuku do sterea.

Obr. 3.5: Grafický návrh modulu wavetable syntézy.

Pokud jde o zpracování zvuku v metodě processBlock modulu aditivní syntézy, není zde tolik zajímavých technik a vychytávek pro zpracování vzorků, jako je tomu u granulární syntézy. Všechny zpracovávané zvukové vzorky jsou v tomto syntézátoru předem zadány uživatelským kreslením a lineárním dopočítáním doplňujících vln. Důležitá je však v této metodě iterace polem výsledných vzorků, které plní hlavní buffer v momentě, co je provedena interpolace. Interpolace je zde hlavně z experimentálních důvodů. Kvůli nastavitelné frekvenci syntezátoru, je velmi pravděpodobné že krok, se kterým se pole iteruje, nebude celočíselný. V tomto případě nelze vrátit pouze vzorek z pole podle indexu, ale je nutné tento vzorek dopočítat. Původní myšlenka interpolací wavetable syntézy obsahovala i implementaci Hermitovy interpolace, avšak následná praxe ukázala, že její výpočet je příliš náročný a nelze jej provádět v reálném čase pro větší množství vzorků.

3.5 Moduly experimentálního získávání hodnot

Následující kapitola se zaměří na moduly, které získávají hodnoty netradičními způsoby. Pracují podobně jako LFO, v rozsahu od 0 po 1 nastavují v čase proměnlivou hodnotu určeným parametrům. Proto na ně bude v této části nahlíženo jako na nízkofrekvenční oscilátory, nemusí však vždy generovat pouze periodický signál.

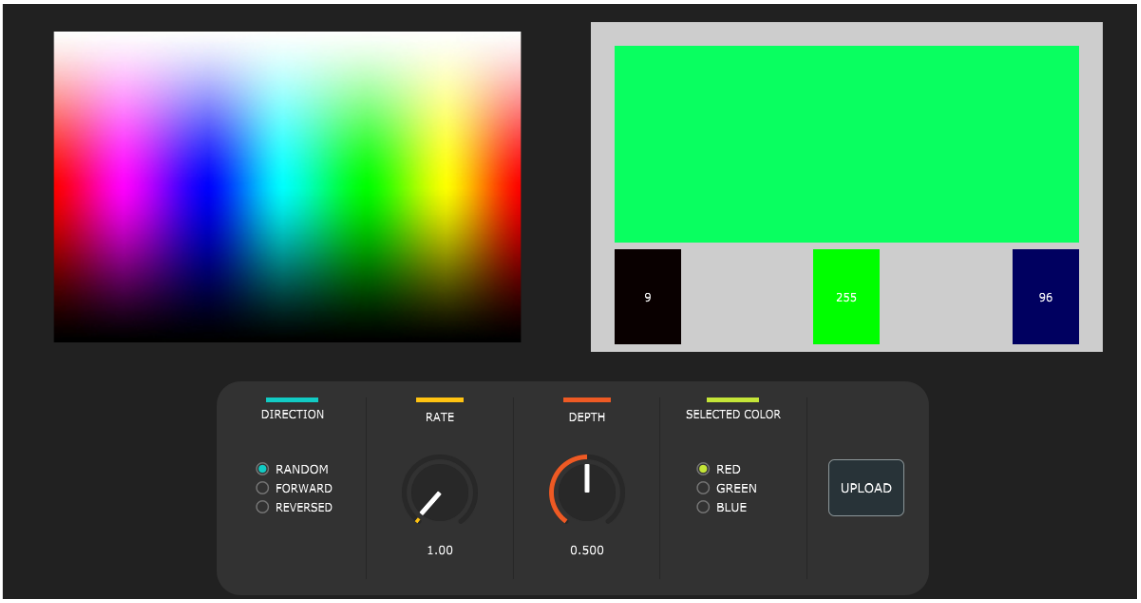
Každý modul LFO vyžaduje pravidelné volání metody updateKnobs, která nastaví vybraným parametrům aktuální hodnotu. To by však znamenalo implementovat hned několik časovačů, jelikož každé LFO může mít nastavenou jinou hodnotu RATE a tím pádem i jinou periodu pro opětovné volání updateKnobs. Každý časovač ale představuje potenciálně větší zatížení výpočetní síly.

Z tohoto důvodu bylo přistoupeno na systém s jedním globálním časovačem umístěným v modulu WRAPPER. Ten pravidelně volá metodu timerCallback, která kontroluje splnění podmínek pro samostatné moduly a následně rozhodne, zda se přistoupí k nastavení hodnoty vybraným parametrům.

Aby se docílilo různých časových prodlev, je každý časovač zastoupen proměnnou. Jednotlivý průchod timerCallback znamená zvýšení hodnoty zmíněných proměnných. Pokud splní taková proměnná podmínku, že její hodnota představující uplynulý čas je větší nebo rovna nastavené periodě vypočítané z parametru RATE, dojde k zavolání updateKnobs.

3.5.1 Color LFO

Color LFO je modul jehož myšlenkou je spojení zvuku a obrazového vjemu. Získávání hodnoty v rozsahu 0 až 1 probíhá za pomoci rozboru barev jednoho obrazového pixelu podle aditivního modelu barev RGB, obsahující červenou, zelenou a modrou barvu. Pokud se uživatel rozhodne využít tohoto modulu, musí nejprve pomocí tlačítka UPLOAD nahrát tížený obrázek. Momentálně lze vybrat soubory formátu JPEG nebo PNG.



Obr. 3.6: Nastavení LFO hodnot pomocí barvy z obrázku.

Parametry modulu Color LFO

- **DIRECTION** - zde má uživatel na výběr směr, jakým chce, aby se výběr další pozice na obrázku vydal. RANDOM ponechává volbu následující pozice náhodě. FORWARD začíná v levém horním rohu obrázku a postupuje po řádcích až do pravého dolního rohu, kde se pozice opět vynuluje a kurzor se přesune na začátek. REVERSED funguje na podobném principu, kurzor však postupuje opačným směrem.
- **SELECTED COLOR** - parametr udávající barvu, ze které se bude získávat výstupní hodnota. Na výběr jsou zmíněné základní barvy červená, zelená a modrá.
- **RATE** - klasický parametr, jenž nesmí chybět žádnému LFO modulu, zde však neudává kolikrát se zopakuje celý sled hodnot představující tvar vlnu, ale kolikrát se parametru nastaví další hodnota.

- **DEPTH** - nastavuje hloubku parametru, přímo násobí výstupní hodnotu a tím ji usměrňuje.

Po vybrání a nahrání souboru dochází při volání timeCallback k analýze jednoho obrazového pixelu. Poté je podle RGB modelu rozebrán na tři základní barvy. Podle zvoleného typu barvy parametru SELECTED COLOR se nastaví hodnota pro výstup modulu. Běžně se lze setkat s hodnotami v rozsahu 0-255, zde jsou však reprezentovány desetinným číslem souhlasným s výstupním rozsahem LFO.

3.5.2 Bounce LFO

Jedná se o interaktivní modul, ve kterém se malá kulička volně pohybuje po obrazovce ve vymezeném prostoru. Po dosažení stěny v tomto vymezeném prostoru se kulička dokáže odrazit a pokračovat v pohybu opačným směrem. Uživatel kuličce může nakreslit zábranu a tím ovládat její pozici X a Y ve vymezeném prostoru. V nastavení si pak vybere, jestli chce jako LFO hodnoty vzít podle jedné či druhé osy.

Obr. 3.7: Modul LFO využívající metodu odrazu objektu.

Parametry modulu Color LFO

- **BALL SPEED** - nastavení pro kontrolní metodu, určuje za jak dlouho má časovač posunout pozici kuličky.

- **DIRECTION** - vybere, zda výstupní hodnota vychází z pozice X či Y, kdy se za počáteční bod považuje levý horní roh vymezené oblasti.
- **RATE** - narozdíl od ostatních typů LFO určuje, za jak dlouho dojde k dalšímu nastavení hodnoty.
- **DEPTH** - násobí výstupní hodnotu a jako u ostatních LFO ji tak usměrňuje.

3.5.3 Math LFO

Tento modul se vyznačuje možností vytvořit vlastní tvar funkce. Uživatel může využít textového okna pro vepsání předpisu pomocí klávesnice. Použít ve svém zápisu může proměnnou x a konstantu pi. Po stisknutí tlačítka nedaleko tohoto textového pole program začne dopočítávat pole sta hodnot dle uživatelského vstupu. Proměnnou x poté nahradí hodnoty 0 až 10. Výsledek se po dopočítání zobrazí uprostřed grafického rozhraní.

Obr. 3.8: Modul LFO využívající metodu odrazu objektu.

Při tvorbě matematického modulu bylo nutné zodpovědět si, jak ošetřit tyto rovnice. Uživatel může do textového pole zadat v podstatě cokoliv. Některé výsledky takovéhoto zadávání by nemusely být úplně žádoucí.

Nejefektivnějším se zdálo uživatele usměrnit na tvar sinu, tím dosáhneme změny hodnot v rozmezí 0-1. Proto se zadaný předpis funkce vkládá jako argument funkce sinus. Druhou vrstvou ošetření je zkouška přeparsování textu knihovny exprtk. Ta

byla také použita pro umožnění převodů a výpočtů předpisu funkce na konkrétní číselné hodnoty.

Modul Math LFO obsahuje parametry podobné ostatním modulům RATE a DEPTH. Kromě nich pouze textové pole pro zadání vstupu a tlačítko pro vybuzení procesu výpočtů.

3.5.4 Wavetable LFO

Modul Wavetable LFO vychází z principů wavetable syntezátoru, kde si uživatel nakreslí průběh tří vln a nechá si programem dopočítat zadaný počet doplňujících vln pomocí lineární interpolace. Jeho výstupem jsou opět hodnoty v rozsahu 0-1. V průběhu fungování tohoto LFO jsou mezi vzorky dopočítány pouze lineárně narozdíl od modulu Wavetable syntézy, kde si uživatel může vybrat i kubickou metodu.

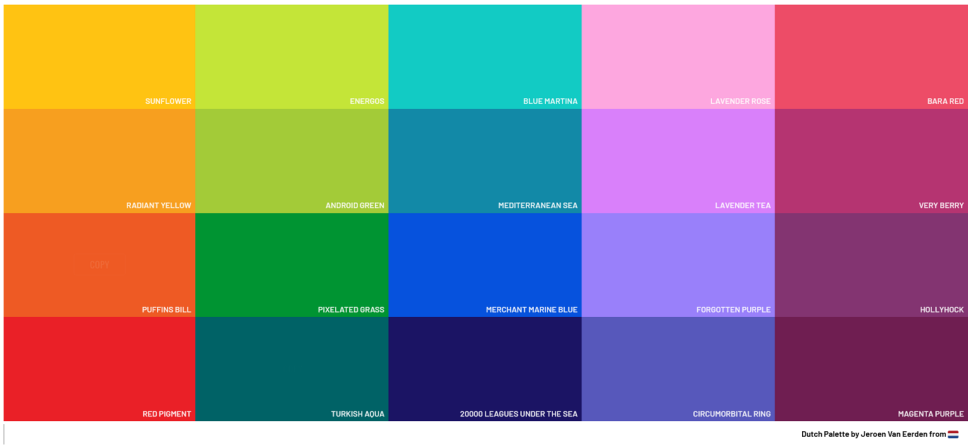
Modul je opět doplněn klasickými parametry RATE a DEPTH, k nimž se přidává i WAVE COUNT, který byl popsán v kapitole o modulu Wavtable syntézy.

Obr. 3.9: LFO modul s wavetable modifikací.

3.6 Grafický design

Při zkoumání již vytvořených pluginů bylo hned patrné, že se lépe pracuje s plug-iny, které se zamyslely nad přehledností svého grafického prostředí. To jako takové sice nemění, jakým způsobem plug-in funguje po technické stránce, může však ovlivnit uživatele při kreativní tvorbě.

Největší cíl, který si tedy práce po grafické stránce stanovila byla přehlednost a jednoduchost uživatelského rozhraní. Toho se dá docílit například vyhnutím se komplikovaným barevným schématům s gradientními přechody. Tímto způsobem je možné docílit větší intuitivnosti i se zobrazením mnoha potřebných ovládacích prvků. Aby k sobě prvky barevně ladily, byly zvoleny barvy z palety Dutch Palette ze stránek `https://flatuicolors.com/palette/nl`. Je nutné dodat, že ne vždy bylo možné využít těchto barev a bylo nutné si nějaké barvy odvodit ztmavováním, či zesvětlováním.



Obr. 3.10: Barevná paleta Dutch Pallete.

Pozadí

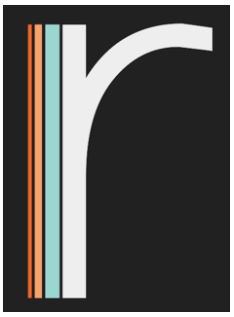
Častým trendem dnešní doby, je tmavý režim všech jinak světlých aplikací. Souvisí to hlavně s nepříjemným efektem, který má na oko velmi světlá barva při delším pozorování. Pozadí plug-inu bylo proto záměrně zvoleno tmavší. S tím přišlo také rozhodnutí o barvě textu, musel být rozhodně v kontrastu s pozadím kvůli dobré čitelnosti. Nejvíce se osvědčila bílá nebo lehce našedlá bílá. Šedá byla již na hranici čitelnosti.

Název

Název GranularFlow vznikl myšlenkou na jednotlivé zrna granulární syntézy, které si volně plynou, překrývají se a vytváří zvukový tok. Logo s názvem doplňují i jednotlivé barevné čáry znázorňující kurzory modulu granulární syntézy, jež jsou v této

Obr. 3.11: Logo plug-inu GranularFlow.

práci výchozím bodem vzniku nových granulí. Čáry jsou různě široké a kousek od sebe. Při zmenšeném pohledu mají vytvářet plynulý barevný přechod, jelikož i granule pro lidské uši splývají v celistvý zvuk. U písmena W jsou čáry nakloněny. Vícero čarami má grafika naznačit přechody mezi jednotlivými tvary vln wavetable syntézy.



Obr. 3.12: Detail barevných čar.

Otočný knoflík a posuvník

Základní otočné knoflíky frameworku JUCE jsou tvořeny zakulaceným obloukem, který se vyplňuje za označeným bodem tmavším odstínem barvy. Vyplnění a pozice bodu znázorňuje aktuální hodnotu nastaveného parametru. Tomuto bodu se říká thumb, ve výchozím nastavení je reprezentován modře vyplněným kolečkem.

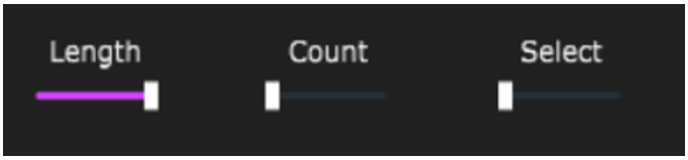
Výchozí grafický návrh však pro tuto práci nevypadal adekvátně, nepřipomínal na první pohled klasický otočný knoflík, se kterým se lze běžně setkat na audio zařízeních. Bylo tedy přistoupeno k vypracování upraveného rozhraní knoflíku. Po kódové stránce nové rozhraní umožňuje programátorovi lehčí změnu vyplňované barvy oblouku i pozadí thumb. Obě barvy jsou nyní sjednoceny a tvar kolečka thumb byl přepracován do lehce zakulaceného obdélníkového ukazatele.

Obr. 3.13: Výchozí vzhled točného knoflíku.

Obr. 3.14: Vytvořený grafický návrh otočného knoflíku.

Celý mechanismus knoflíku je navíc zabalen kódem do bloku parametru. Blok poté zobrazuje kromě názvu a barevně sjednoceného označení i aktuálně nastavenou číselnou hodnotu.

Úpravy byly provedeny i u posuvníku, zde šlo hlavně o změnu tvaru a barvy tlačítka na posouvání, kterému se rovněž říká thumb. I klasický posuvník byl kódově zabalen jako parametr.

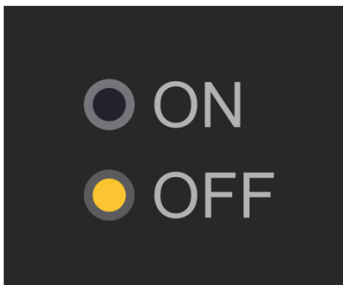


Obr. 3.15: Posuvníky horní lišty.

Výběr z vícero možností

Od původního vzhledu se změnil obrys čtverce se zakulacenými rohy a symbolem zaškrtnutí na kruh s výplní. Ten více napodobuje reálnou kontrolku hardwarového zařízení. Rovněž byl i tento způsob nastavení zabalen jako parametr a oddělen od ostatních separátorem, který je reprezentován jako svislá světlá čára.

Obr. 3.16: Původní grafika tlačítek výběru.



Obr. 3.17: Vytvořený vzhled výběrových tlačítek.

3.7 Poznatky z vývoje

Vývoj softwaru měl probíhat na operačních systémech Windows 10 a macOS, což však s sebou neslo značné problémy. Nejdříve byla snaha programovat ve vývojovém prostředí CLion. Je totiž dostupné na obou operačních systémech a jeho sdílené indexování souborů projektu je velikou výhodou. Stačilo by v podstatě změny synchronizovat s repozitářem a pokračovat v práci na druhém zařízení. Bohužel Clion a CMake k sestavení a testování nefungovalo na Windows nejlépe, a tak bylo zvoleno Visual Studio 2022. To díky integraci MSBuild výborně zvládalo sestavení a spouštění stand-alone software. Vývojová a testovací fáze tedy probíhala pouze na platformě Windows.

Důležité při programování bylo dávat si pozor na zbytečné zabírání paměti. Šlo především o dynamické vytváření nových proměnných a instancí tříd. Jako zásadní se v průběhu ukázalo zvolit vhodný datový typ. Stačilo si zanalyzovat využívané proměnné a zjistit, zda by se některé komplexnější algoritmy využívající tyto proměnné nemohly zjednodušit a využívat jen pár předem deklarovaných proměnných či polí. Tak tomu bylo hlavně u tříd Grain a AudioBuffer. Bylo nutné myslet na to, že proces zpracování zvuku trvá velmi krátkou dobu. Není tak možné mezi třídami předávat větší objem dat a vyžadovat jakékoliv kreslení. Bylo tím pádem lepší se vyhnout všem zásahům do grafického rozhraní z vlákna, které zpracovává zvuk.

Zároveň bylo nutné co nejvíce omezit překreslování rozhraní. Byla tedy omezena viditelnost otočného knoflíku, pokud na daný parametr bylo aplikováno LFO, takto se ušetřilo na zatíženosti procesoru a software běžel svižněji. Je určitě důležité zmínit i značné snížení vytíženosti procesoru nastavením neprůhlednosti objektů a ukládání každého komplexnějšího elementu do paměti cache [1]. Nejvíce se tato změna dotkla granulární syntézy. Všechny kurzory aktuální pozice totiž měly být zobrazeny zároveň, to se však ukázalo jako výkonnostně náročné. Všeobecně největší problémy s plynulostí běhu programu způsobovalo neustálé překreslování elementů. Což byla a stále je velká nevýhoda frameworku JUCE. Nepodporuje totiž vykreslování grafickými kartami a tuto práci tedy vykonává procesor počítače [2]. Bylo by mnohem snazší pracovat s grafickým rozhraním, pokud by se toto rozhraní vykreslovalo s výpočetní silou grafických karet. Jsou na takovéto záležitosti přímo určené a vývojář by tak měl více prostoru pro zajištění lepšího požitku koncového uživatele při používání softwaru.

[1] Vykreslování a cache - `https://forum.juce.com/t/repaint-without-parent-repaint/15550`

[2] Podpora překreslování za pomoci grafické karty - `https://forum.juce.com/t/gpu-supported-ui-rendering-in-2020/37288/9`

Nadějí na změnu bylo vydání JUCE verze 7, která nabízela vylepšené vykreslovací schopnosti. I když by měla tato nová verze být zpětně kompatibilní, tvořily se zde jisté problémy s otočnými knoflíky, tlačítky, a hlavně s textEditor třídou, která uživateli umožňovala psát argumenty vlastní funkce. Nevýhody vzniklé při praktickém testu nové verze převýšily pozitivní změny a projekt tak využívá stále frameworku JUCE verze 6 [3].

V neposlední řadě bylo nutné poradit si i s problémy memory leakage, nebo-li úniku paměti. To nastávalo při nedovoleném smazání instance, příkladem může být třída Grain. Zrno se sice správně smazalo po dokončení přehrávání, ale dříve, než bylo zamýšleno, a tak při následném přístupu v cyklu odkaz na něj již neexistoval. Řešením bylo kontrolovat počet granulí jak při nastavení parametru, tak i ve vláknu časovače, takže nemohlo docházet k nepřesnému počtu a tím pádem ani ke špatnému přístupu. Problém také vznikal při dynamické změně velikosti iterovaného pole granulí. Tento problém rovněž vyřešilo přidání proměnné kontrolního počtu a předem deklarovaná velikost pole se zrny.

Posledním větším problémem byla nekompatibilita s macOS. I přesto že vývojáři JUCE ujišťují na svých stránkách o přenositelnosti výsledných plug-in modulů, zapomínají dodat, že pro skutečné fungování na platformě macOS je nutné mít plugin podepsaný unikátním Developer ID certifikátem. Takový podpis lze získat pouze při koupi Apple Developer programu. Apple se tak chrání proti nechtěnému přepisu, zneužití a pirátství programů [4].

[3] JUCE 7 - `https://juce.com/juce-7-released/`

[4] Apple Codesign - `https://codeburst.io/how-and-why-to-codesign-applications-7d9b8be94ed`

Závěr

Tato práce se zaměřila na téma vývoje zásuvného modulu schopného granulární syntézy a experimentálních způsobů nastavení zvukové syntézy. Po získání informací z úvodního průzkumu, jenž odkryl možnosti plug-in modulů, se stanovily základy nutné pro vývoj softwaru.

Poté byl zahájen samotný proces vývoje jednotlivých částí programu zahrnující bloky granulární, aditivní a wavetable syntézy. Dále bylo vytvořeno několik typů experimentálních LFO modulů. Každý z nich implementoval odlišné kreativní techniky k získání hodnot potřebných pro nastavení parametrů syntéz. Software byl z důvodu zajištění kompatibility v průběhu vývoje manuálně testován na digitálních audio stanicích FL Studio, Cubase a REAPER.

Výstupem práce byl zásuvný modul využívající technologie VST3 zkompletovaný z výše zmíněných funkčních celků, který lze umístit jako efekt na efektovou sběrnici zvolené digitální audio stanice na platformě Windows.

Literatura

[1] DELTON T. HORN *MUSIC SYNTHESIZERS A MANUAL OF DESIGN & CONSTRUCTION*. 2nd ed. Blue Ridge Summit, Pennsylvania: TAB BOOKS, 1984. ISBN ISBN 0-8306-1565-2.

[2] Simon Crab-*MUSIC N-, Max Vernon Mathews, USA, 1957* [online]. London: Simon Crab, 2019 [cit. 2023-04-12]. Dostupné z: https://120years.net/wordpress/music-n-max-mathews-usa-1957/

[3] RUSS, Martin. *Sound synthesis and sampling*. 1. Oxford: Focal Press, 1996. Music technology series. ISBN 02-405-1429-7.

[4] Pitch Shifter: What It Is and How to Use It. *MUSICIAN ON A MISSION* [online]. London: Mayley Digital Limited, 2022 [cit. 2022-12-09]. Dostupné z: https://www.musicianonamission.com/pitch-shifter/

[5] What is a Framework in Programming & Why You Should Use One. *Net solutions* [online]. Chandigarh (Indie): Net Solutions, 2021 [cit. 2022-12-09]. Dostupné z: https://www.netsolutions.com/insights/what-is-a-framework-in-programming/

[6] What Are AAX Plugins? (And Do You Need Them?). *PRODUCER HIVE* [online]. Los Angeles (California): Producer Hive, 2022 [cit. 2022-12-09]. Dostupné z: https://producerhive.com/buyer-guides/vst/what-are-aax-plugins/

[7] AAX Connectivity Toolkit. *AVID* [online]. Burlington (Massachusetts): Avid, 2022 [cit. 2022-12-09]. Dostupné z: https://www.avid.com/alliance-partner-program/aax-connectivity-toolkit

[8] VST, VST, AU and AAX — What-s The Difference? Plugin Formats Explained. *Integraudio* [online]. Arizona: Integraudio, 2021 [cit. 2022-12-09]. Dostupné z: https://integraudio.com/vst-vst-au-and-aax-whats-the-difference/

[9] AU vs VST (Differences, Which To Use & Why). *PRODUCER HIVE* [online]. Los Angeles (California): Producer Hive, 2022 [cit. 2022-12-09]. Dostupné z: https://producerhive.com/buyer-guides/vst/au-vs-vst-differences-which-to-use

[10] What Are VST Plugins, And Why You Need Them?. *MusicProductionNerds* [online]. Paříž: MusicProductionNerds.com, 2017 [cit. 2022-12-09]. Dostupné z: https://musicproductionnerds.com/what-are-vst-plugins

[11] Virtual Studio Technology (VST). *Techopedia* [online]. Edmonton (Alberta): Techopedia, 2015 [cit. 2022-12-09]. Dostupné z: https://www.techopedia.com/definition/266/virtual-studio-technology-vst

[12] What Is Core Audio?. *Apple Developer* [online]. Cupertino (California): Apple, 2017 [cit. 2022-12-09]. Dostupné z: https://developer.apple.com/library/archive/documentation/MusicAudio/Conceptual/CoreAudioOverview/WhatisCoreAudio/WhatisCoreAudio.html

[13] Our technology. *Steinberg* [online]. Hamburg (Německo): Steinberg Media Technologies, 2022 [cit. 2022-12-09]. Dostupné z: https://www.steinberg.net/technology/

[14] Introduction to Uses of C++. *EDUCBA* [online]. Oregon: EDUCBA, 2018 [cit. 2022-12-09]. Dostupné z: https://www.educba.com/uses-of-c-plus-plus/

[15] *The audio programming book*. 1. Cambridge: MIT Press, 2011. ISBN 978-0-262-01446-5.

[16] Compare The Versions Of Cubase. *Steinberg* [online]. Hamburg (Německo): Steinberg Media Technologies, 2022 [cit. 2022-12-09]. Dostupné z: https://www.steinberg.net/cubase/compare-editions/

[17] What Is Cubase AI. *Steinberg* [online]. Hamburg (Německo): Steinberg Media Technologies, 2022 [cit. 2022-12-09]. Dostupné z: https://www.steinberg.net/cubase/ai/

[18] Discover All The Cubase Features. *Steinberg* [online]. Hamburg (Německo): Steinberg Media Technologies, 2022 [cit. 2022-12-09]. Dostupné z: https://www.steinberg.net/cubase/features/

[19] [online]. [cit. 2022-12-09]. Dostupné z: https://www.steinberg.net/cubase/

[20] *FL Studio* [online]. Gent (Belgie): Image Line Software, 2022 [cit. 2022-12-09]. Dostupné z: https://www.image-line.com/fl-studio

[21] Which genres are you proficient in producing. *Forum / FL Studio* [online]. Gent (Belgie): Image Line Software, 2021 [cit. 2022-12-09]. Dostupné z: https://forum.image-line.com/viewtopic.php?t=259534

[22] *REAPER* [online]. Rosendale (NewYork): Cockos Incorporated, 2022 [cit. 2022-12-09]. Dostupné z: https://www.reaper.fm/

[23] The Digital Audio Workstation. *CCRMA* [online]. Stanford (California): Center for Computer Research in Music and Acoustics Department of Music, 2016 [cit. 2022-12-09]. Dostupné z: https://ccrma.stanford.edu/courses/192b/Digital_Audio_Workstations.pdf

[24] Most Popular DAW Software 2022 — Which DAW Do Most Producers Use?. *Musician's HQ* [online]. Chicago: Musician's HQ, 2022 [cit. 2022-12-09]. Dostupné z: https://musicianshq.com/most-popular-daw-software-which-daw-do-most-producers-use/

[25] What Is A Good Buffer Size For Recording? [Buffer Size Explained]. *Orpheus Audio Academy* [online]. San Francisco: Orpheus Audio Academy, 2022 [cit. 2022-12-09]. Dostupné z: https://www.orpheusaudioacademy.com/buffer-size/

[26] What Buffer Size Should I Use? - Does It Affect Sound Quality?. *Integraudio* [online]. Arizona: Integraudio, 2021 [cit. 2022-12-09]. Dostupné z: https://integraudio.com/what-buffer-size-should-i-use/

[27] Goudard, Vincent and Muller, Remy - Real-time audio plugin architectures. *Research gate* [online]. Paříž: Sorbonne Université, 203n. l. [cit. 2023-04-14]. Dostupné z: https://www.researchgate.net/publication/265307102_Real-time_audio_plugin_architectures

[28] W. SMITH, Steven. *Digital Signal Processing* [online]. Second Edition. San Diego (California): California Technical Publishing, 1999 [cit. 2022-12-09]. ISBN ISBN 0-9660176-6-8. Dostupné z: https://users.dimi.uniud.it/ antonio.dangelo/MMS/materials/Guide_to_Digital_Signal_Process.pdf

[29] Digital Audio Basics: Sample Rate and Bit Depth. *PreSonus* [online]. Jacksonville(Florida): PreSonus Audio Electronics, 2022 [cit. 2022-12-09]. Dostupné z: https://www.presonus.com/learn/technical-articles/sample-rate-and-bit-depth

[30] Synthesis Definition & Meaning - Merriam-Webster. *Merriam-Webster* [online]. Chicago: Encyclopaedia Britannica, 2022 [cit. 2022-12-09]. Dostupné z: https://www.merriam-webster.com/dictionary/synthesis

[31] VST3. In: *Synthopia* [online]. Toronto (Ontario): Synthtopia, 2020 [cit. 2022-12-09]. Dostupné z: https://www.synthtopia.com/wp-content/uploads/2020/07/steinberg-vst-3.7-e1596209038442.png

[32] Audio units. In: *Apple Developer* [online]. Cupertino (California): Apple, 2022 [cit. 2022-12-09]. Dostupné z: https://developer.apple.com/licensing-trademarks/audio-units/

[33] AAX-plugin. In: *MusicProductionNerds* [online]. MusicProductionNerds.com, 2019 [cit. 2022-12-09]. Dostupné z: https://musicproductionnerds.com/wp-content/uploads/2019/01/AAX-plugin.png?ezimgfmt=ngcb2/notWebP

[34] VST3. In: *Synthopia* [online]. Toronto (Ontario): Synthtopia, 2020 [cit. 2022-12-09]. Dostupné z: https://www.synthtopia.com/wp-content/uploads/2020/07/steinberg-vst-3.7-e1596209038442.png

[35] JUCELOGO. In: *Github Avatars* [online]. San Francisco (California): JUCE, 2020 [cit. 2022-12-09]. Dostupné z: https://avatars.githubusercontent.com/u/62880632?s=200v=4

[36] IPlugLogo. In: *Github Avatars* [online]. San Francisco (California): iPlug, 2018 [cit. 2022-12-09]. Dostupné z: https://avatars.githubusercontent.com/u/36328372?s=200v=4