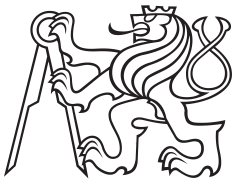
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Audio plugin VST jako akusticky analyzátor

Nazar Grigorenko

Vedoucí: Ing. Marek Brothánek, Ph.D.
Obor: Softwarové inženýrství a technologie
Květen 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Grigorenko** Jméno: **Nazar** Osobní číslo: **495643**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Audio plugin VST jako akustický analyzátor

Název bakalářské práce anglicky:

Audio plugin VST as an acoustic analyzer

Pokyny pro vypracování:

V rámci rešeršní části bakalářské práce porovnejte možnosti a vlastnosti jednotlivých audio pluginů. Následně pro pluginy typu VST (Virtual Studio Technology) porovnejte možnosti jejich vytvoření. Součástí rešeršní části bude i porovnání licenčních podmínek použití.
V praktické části vytvořte za použití vhodného frameworku audio plugin (ve zvoleném formátu) určený pro vyhodnocení akustického parametru přenosu řečového signálu – STIPA. Implementaci proveďte ve shodě s ČSN EN IEC 60268-16 a souvisejícími doporučeními. Pro tento účel si nastudujte problematiku digitální filtrace a frekvenční zpracování. Použití pluginu se předpokládá v softwaru Audacity. Vytvořený plugin po dohodě s vedoucím práce doplňte grafickým rozhraním a případně možností exportu (mezi)výsledku.
Výstupem bakalářské práce se předpokládá kromě funkčního pluginu komentovaný zdrojový kód, včetně provedených testování jak z pohledu implementace, tak funkčnosti.

Seznam doporučené literatury:

[1] STEINBERG MEDIA TECHNOLOGIES: VST Home, online,
Dostupné z URL: https://steinbergmedia.github.io/vst3_dev_portal/pages/index.html, [cit. 5. 1. 2023].
[2] RORABAUGH, C. Britton: Digital filter designer's handbook: featuring C routines, McGraw-Hill Professional, 1993.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Marek Brothánek, Ph.D. katedra fyziky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **13.02.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

Ing. Marek Brothánek, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studenta

Poděkování

Rád bych poděkoval svému vedoucímu práce za cenné rady a připomínky, které mi poskytl při psaní této práce.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a s použitím uvedené literatury.

Abstrakt

Tato bakalářská práce se zaměřuje na návrh a implementaci audio pluginu pro výpočet STIPA parametru. Obsahuje podrobnou analýzu exitujících formátů pluginů a vývojových prostředí, následovanou detailním postupem implementace pluginu a testování vytvořených algoritmů. Výsledkem práce je funkční audio plugin sloužící jako nástroj pro měření kvality přenosu řečového signálu.

Klíčová slova: VST, audio, plugin, zásuvný modul, STIPA, JUCE

Vedoucí: Ing. Marek Brothánek, Ph.D.

Abstract

This bachelor thesis focuses on the design and implementation of an audio plugin for the calculation of the STIPA parameter. It includes a detailed analysis of existing plugin formats and development environments, followed by a detailed procedure of plugin implementation and testing of the created algorithms. The result of the work is a functional audio plugin serving as a tool for measuring the quality of speech signal transmission.

Keywords: VST, audio, plugin, plug-in modul, STIPA, JUCE

Title translation: Audio plugin VST as an acoustic analyzer

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Zásuvné moduly pro audio, obecně známé jako audio pluginy, jsou autonomní softwarové komponenty. Tyto komponenty se dynamicky připojují k hostitelské aplikaci s cílem rozšířit její funkce nebo využít jejích schopností, a v oblasti zvukového zpracování se zaměřují především na digitální zpracování signálů a syntézu zvuku. Ve spojení s hostitelskými aplikacemi, jako jsou například DAW[1], poskytují možnost čtení a odebírání vstupních dat digitálního signálu, zatímco samotné audio pluginy se soustředí na zpracování tohoto signálu.

Technologie VST[2], představená společnostmi Steinberg a Propellerhead v roce 1996, byla revolučním krokem v simulaci tradičního hardwaru nahrávacího studia v softwaru prostřednictvím digitálního zpracování signálu. Audio pluginy VST pracují v reálném čase a zpracovávají zvuk prostřednictvím počítačových zdrojů, převážně CPU.

S rostoucí popularitou revoluční VST technologie se vyvinulo několik nových formátů pluginů. V současnosti existují tři hlavní formáty audio pluginů: VST, AU a AAX. Tyto formáty představují sady rozhraní API[3] a dalších instrukcí pro vytváření, zpracování, přehrávání a manipulaci se zvukovým signálem na různých operačních systémech. Audio pluginy VST zahrnují veškeré virtuální nástroje a efekty v libovolném formátu a mohou být implementovány prostřednictvím široké škály programovacích jazyků, frameworků a vývojových prostředí.

Tato bakalářská práce se soustředí na návrh a implementaci VST pluginu pro výpočet STIPA[4] parametru, který je klíčovým indikátorem kvality přenosu řečového signálu v různých prostředích. Hlavní cíle práce zahrnují vytvoření takového VST pluginu, který bude schopen analyzovat zvukový signál a poskytnout uživatele informace o kvalitě přenosu řečového signálu v daném prostředí, a zároveň se bude snažit optimalizovat výpočetní algoritmus a zajistit jeho kompatibilitu s různými DAW.

[1] Digital Audio Workstation je elektronický nebo počítačový systém určený k záznamu, ukládání, úpravám a přehrávání digitálního zvuku.

[2] Virtual Studio Technology je softwarové rozhraní pro komunikaci mezi hostitelským programem a pluginy, kde tyto pluginy slouží ke generování a úpravě digitálního audio signálu.

[3] Application Programming Interface je popis způsobů interakce jednoho počítačového programu s ostatními.

[4] Speech Transmission Index for Public Address systems.

V první části bakalářské práce je provedena analýza existujících formátů audio pluginů se zaměřením na jejich podporu, bitovou hloubku a postupy instalace. Tato analýza pomáhá při výběru nejvhodnějšího formátu pro implementaci návrhu pluginu, který je tématem práce.

Druhá část práce se zabývá porovnáním vývojových prostředí pro implementaci, včetně programovacích jazyků, frameworků, editorů a jejich licenčních politik. Tyto informace jsou klíčové pro výběr prostředí, které bude použito pro implementaci pluginu.

Třetí část je zaměřena na teoretické aspekty, které jsou důležité pro vývoj pluginu, jako jsou IIR filtry a vypočítání parametru STIPA. Ve čtvrté části je prezentován proces návrhu a implementace VST pluginu pro výpočet STIPA parametru, včetně jeho testování a optimalizace. Vytvořený plugin je kompatibilní s jakýmkoliv 64bitovým DAW a je otestován v DAW Audacity.

Tato bakalářská práce tedy představuje komplexní přístup k vývoji VST pluginu pro výpočet STIPA parametru, od teoretických aspektů, přes analýzu formátů a vývojových prostředí, až po samotný návrh, implementaci a testování pluginu.

Kapitola 2

Porovnání vývojových formátů

Tato kapitola se zaměřuje na existující formáty audio pluginů, jejich kompatibilitu s různými DAW a operačními systémy, a rozdíly v bitové hloubce. Popisuje také význam formátů v kontextu audio pluginů a jejich vliv na použitelnost a kompatibilitu s různými platformami.

2.1 Aktuální formáty

V této sekci jsou popsané aktuálně používané formáty pro vytváření audio pluginů.

Formát VST

Následující text vychází z [1]. VST (Virtual Studio Technology) je formát audio pluginů, který byl vyvinut společnostmi Steinberg a Propellerhead. Tento formát je kompatibilní se systémy Windows, macOS a Linux.

V roce 1999 společnost Steinberg představila nový formát VST 2.0, který zavedl pojem VSTi (Virtual Studio Technology Instrument). Na rozdíl od klasických VST pluginů fungují VSTi jako samostatné aplikace schopné generovat zvukové signály. V roce 2006 byl představen aktualizovaný formát VST 2.4, který začal podporovat 64bitové zpracování digitálních signálů. Od roku 2018 se společnost Steinberg soustředí na vývoj VST 3.0, známý jako VST3, představený v roce 2008. Hlavní vlastnosti VST3 oproti dřívějším verzím zahrnují:

- existuje pouze v 64bitovém formátu;
- zvýšená účinnost využití procesoru;
- virtuální audio výstupy;
- nabízí lepší práci s událostmi MIDI;
- podporuje větší počet MIDI vstupů a výstupů;
- možnost změny velikosti rozhraní pluginu;
- VSTXML protokol pro dálkové ovládání.

Ve verzi 3.0 došlo ke značnému vylepšení algoritmů pro zpracování pluginů, které nyní automaticky deaktivují zpracování, pokud na vstupu nejsou žádné signály. Díky tomu jsou VST3 audio pluginy energeticky efektivnější a zlepšují výkon celkového systému. Kromě toho mohou VST3 audio pluginy zpracovávat jak MIDI zprávy, tak audio vzorky, což jim umožňuje fungovat jak jako efekt, tak i nástroj.

Navíc rozšířený počet MIDI vstupů a výstupů přispívá k vytváření nástrojů s pokročilými zvukovými schopnostmi. Zatímco VST 2.4 poskytuje pouze jediný MIDI vstup a výstup s celkem 16 kanály pro přenos MIDI informací, VST3 podporuje více portů, přičemž každý z nich disponuje 16 kanály. Tato inovace je využívána například pro zpracování vícekanálového surround zvuku či implementaci pokročilých funkcí, jako je side-chain. Tím se VST3 stává flexibilnějším a výkonnějším formátem pro různé aplikace v oblasti zvukového inženýrství.

Přestože podpora VST 2.4 byla ukončena a VST3 nabízí značné výhody, mnoho vývojářů stále produkuje audio pluginy a virtuální nástroje ve formátu VST 2.4. Pro zajištění maximální kompatibility tak DAW (Digital Audio Workstation) nadále podporuje všechny tři formáty.

Formát AU

AU (Audio Units) je audio pluginový formát vyvinutý společností Apple, který je kompatibilní výhradně s operačními systémy macOS a iOS.

Formát AU byl uveden na trh v roce 2001 jako vylepšená verze, která poskytuje lepší kompatibilitu s macOS než původní formát VST. Díky své podobnosti s VST mohou vývojáři snadno konvertovat VST pluginy na AU pluginy, čímž zjednodušují přenos svých produktů do prostředí macOS. Postupem času, s nástupem smartphonů a tabletů, společnost Apple představila vylepšenou verzi formátu AU zvanou AUv3. Tato aktualizace přináší sadu API, která je kompatibilní s operačními systémy iOS a iPadOS.

AUv3 nabízí řadu vylepšení oproti předchozím verzím, jako je například lepší podpora pro multitasking, optimalizace využití systémových prostředků a zvýšená stabilita. Navíc poskytuje lepší integraci s audio aplikacemi na platformách iOS a iPadOS, což umožňuje hudebním tvůrcům a zvukovým inženýrům vytvářet sofistikovanější produkce na svých mobilních zařízeních. V současné době je formát AU široce používán v různých hudebních aplikacích a digitálních audio pracovních stanicích (DAW) pro platformu macOS, zatímco formát AUv3 se stává stále oblíbenějším v iOS a iPadOS prostředí. Více informací lze nalézt v archivní dokumentaci [2].

Formát AAX

AAX (Avid Audio eXtension) je audio pluginový formát, který byl vyvinut společností Avid v roce 2013 speciálně pro DAW Pro Tools 11. Tento formát je kompatibilní s operačními systémy Windows a macOS.

Formát AAX vznikl jako specializovaná varianta VST formátu, která byla navržena pro lepší kompatibilitu s Pro Tools. AAX se odlišuje od formátů

VST a AU následujícími vlastnostmi:

- exkluzivita pro DAW Pro Tools 11;
- lepší optimalizace pro určitý DAW;
- schopnost pracovat ve dvou režimech: *native* a *DSP*.

Podle [3] režim *native* využívá výkon procesoru počítače pro zpracování audiosignálu. Na druhou stranu, režim *DSP* je navržen tak, aby využíval DSP čipu externího audio rozhraní pro zpracování zvuku. V tomto režimu je audiosignál zpracován předtím, než vstoupí do počítače, což umožňuje uživateli slyšet již zpracovaný signál a zároveň šetří systémové zdroje počítače.

Díky svému specifickému zaměření a optimalizaci pro Pro Tools je formát AAX oblíbený mezi profesionálními hudebními producenty a zvukovými inženýry, kteří pracují s touto populární DAW. AAX pluginy poskytují vysokou úroveň stability a výkonu při práci s Pro Tools, což zajišťuje plynulý a efektivní pracovní postup.

Formát DX

Formát DX (DirectX Plugin) je vyvinut společnostmi Microsoft a Cakewalk a je založen na technologii DirectX. Tento formát je kompatibilní výhradně se systémem Windows.

Podobně jako u formátu VST 2.4, pluginy DX se dělí na dva typy: zvukové efekty (DX) a virtuální nástroje (DXi). Funkční princip DX pluginů je velmi podobný audio pluginům ve formátech VST, AU a AAX, s tím rozdílem, že DX formát využívá technologii DirectX. Klíčovým prvkem DX pluginů je schopnost použít DirectX knihovny pro manipulaci nejen s DAW, ale i s širokou škálou zvukových aplikací v prostředí Windows.

Díky integraci s DirectX mohou DX pluginy být připojeny ke zvukovým a video přehrávačům, video editorům a dalším kompatibilním softwarovým nástrojům. Tato flexibilita umožňuje producentům a zvukovým inženýrům využít DX pluginy napříč různými aplikacemi pro širokou škálu úkolů spojených se zpracováním zvuku a multimédií.

Ačkoli formát DX nenabízí takovou univerzálnost jako VST, AU či AAX, jeho schopnost integrace do různých Windows aplikací mu zajišťuje stále své místo na trhu. Tento formát je vhodný zejména pro ty, kteří pracují převážně na platformě Windows a potřebují efektivní způsob, jak propojit různé zvukové a multimediální aplikace.

Formát LV2/LADSPA Version 2

Nasledující text vyhází z [4]. Formát LV2 (Linux Audio Developer's Simple Plugin API Version 2) představuje otevřený standard pro audio pluginy, který se zaměřuje na poskytnutí bezplatné alternativy k formátům VST a AU. Tento formát je podporován v operačních systémech Linux (Ubuntu, Debian).

LV2 je nástupcem formátu LADSPA, který byl vytvořen s cílem překonat omezení předchozího formátu, jež způsobovaly potíže vývojářům usilujícím

o vytváření pokročilejších pluginů. Z hlediska technických specifikací LV2 formát sdílí vlastnosti s VST a AU formáty. Stejně jako tyto formáty, i LV2 umožňuje vývoj virtuálních nástrojů a zpracování signálu s podporou MIDI.

Přestože LV2 nabízí podobné funkce jako VST a AU, ve světě audio pluginů je tento formát méně rozšířený, což lze přičíst tomu, že Linux není nejpoužívanější operační systém pro hudební produkci. Nicméně pro hudební tvůrce a zvukové inženýry, kteří preferují Linux, může formát LV2 představovat užitečnou a cenově dostupnější alternativu k komerčním formátům. Jeho otevřená povaha umožňuje komunitě vyvíjet a sdílet pluginy, které mohou být přizpůsobeny specifickým potřebám uživatelů a vylepšovány v průběhu času.

Formát CLAP

Při tvorbě následujícího textu bylo čerpáno z [5]. Nejnovější formát CLAP (Clever Audio Plug-in) představuje otevřený standard audio pluginů, který byl uveden na trh v červnu 2022 výhradně pro DAW Bitwig Studio. Tento formát je podporován v operačních systémech Windows, macOS a Linux.

Na rozdíl od formátů VST a AU využívá formát CLAP vlastního systému metadat, což umožňuje rychlejší skenování audio pluginů a efektivnější správu. Tento formát také nabízí inovativní způsob ukládání souborů pluginů přímo v rámci projektu DAW místo samostatného adresáře na disku. Díky tomu jsou eliminovány problémy spojené s nemožností otevřít projekt kvůli chybějícím pluginům v operačním systému.

Formát CLAP navíc slibuje efektivnější zpracování signálu a snižuje latenci, což přispívá k lepšímu výkonu a zkušenosti uživatelů při práci s DAW Bitwig Studio. Vývojáři formátu CLAP doufají, že tato inovace přinese nové možnosti pro hudební produkci a umožní větší kompatibilitu mezi různými operačními systémy. Jelikož se jedná o otevřený standard, existuje možnost, že v budoucnu bude formát CLAP podporován i dalšími DAW a stane se konkurencí pro etablované formáty, jako jsou VST, AU a AAX.

2.2 Zastaralé formáty

Pluginy popsané v této části lze stále používat v některých starších operačních systémech a některých DAW, oficiálně však byla podpora těchto formátů ukončena ve prospěch nových verzí.

Formát RTAS

Formát RTAS (Real-Time Audio Suite) je předchůdce formátu AAX, vyvinutý společností Digidesign (později přejmenovanou na Avid) výhradně pro DAW Pro Tools. Tento formát je podporován v operačních systémech Windows a macOS.

Podobně jako u formátu AAX je i RTAS založen na formátu VST a byl vytvořen za účelem zvýšení kompatibility s DAW Pro Tools před uvedením

verze Pro Tools 11. Co se týče funkcí a vlastností, nejsou podle [6] mezi formáty RTAS, VST a AU žádné výrazné rozdíly, s výjimkou exkluzivity pro Pro Tools.

Jedním z důvodů, proč byl formát RTAS nahrazen formátem AAX, bylo zlepšení efektivity a výkonu pluginů, které AAX nabízí. Navzdory tomu, že RTAS již není aktivně podporován a vyvíjen, stále existuje řada starších pluginů a DAW Pro Tools, které tento formát používají. Uživatelé Pro Tools, kteří stále pracují s RTAS pluginy, mohou čelit omezenému výběru a nižšímu výkonu ve srovnání s novějšími formáty, jako je AAX.

Formát TDM

Při tvorbě následujícího textu bylo čerpáno z [7]. Formát TDM (Time Division Multiplexing) je historický formát audio pluginů, který byl vyvinut společností Digidesign (později Avid) pro Pro Tools HD systémy. Tento formát byl podporován na operačních systémech Windows a macOS. TDM pluginy byly navrženy pro spolupráci s proprietárním DSP hardwarem, který tvořil součást Pro Tools HD systémů. To umožňovalo zpracování signálů mimo počítačový procesor a poskytovalo uživatelům vysokou úroveň výkonu a spolehlivosti.

S příchodem formátu AAX DSP, který nabízí obdobné výhody jako TDM, byl formát TDM postupně vyřazován a již není aktivně podporován v novějších verzích Pro Tools. Přestože formát TDM není součástí moderních řešení, jeho důležitost v historii audio produkce a vývoji profesionálních DAW nelze přehlédnout. TDM představoval zásadní krok ve vývoji hardwarově akcelerovaných audio systémů, které poskytovaly nejvyšší možnou kvalitu zpracování zvuku a umožnily hudebním producentům a zvukovým inženýrům dosáhnout sofistikovaných a náročných projektů.

Formát ReFill

Reason ReFill je specifický formát, který byl vyvinut společností Propellerhead (nyní známou jako Reason Studios) výhradně pro DAW Reason.

ReFill zahrnuje komplexní zvukové knihovny, které kombinují vzorky, MIDI soubory, různé groovy, patche a dokonce i kompletní projekty skladeb. ReFill pluginy mohou fungovat jako virtuální nástroje s vlastní sadou efektů, jako simulace skutečných analogových zařízení nebo jako kolekce vzorků. Tento formát nabízí uživatelům DAW Reason bohaté možnosti pro práci se zvukem.

V roce 2019 společnost Reason Studios oznámila ukončení aktivního vývoje ReFill a zavedla podporu formátu VST do svého DAW Reason. Tímto krokem umožnila Reason fungovat jako audio plugin v jiných DAW a rozšířila dostupnost efektů a pluginů ReFill z DAW Reason pro uživatele ostatních DAW, které tento formát podporují.

I přes tuto změnu zůstává ReFill důležitou součástí historie DAW Reason a mnoho uživatelů stále využívá původní ReFill knihovny pro své hudební projekty. Díky této podpoře formátu VST se stává Reason ještě atraktivnějším nástrojem pro hudební tvorbu napříč různými platformami a DAW.

2.3 Specializované formáty

Následující část popisuje existující formáty pluginů, které jsou speciálně navrženy pro konkrétní použití a aplikace.

Formát NKI

NKI (Native Instruments Kontakt Instrument) je proprietární formát navržený společností Native Instruments specificky pro použití v jejich sampleru Native Instruments Kontakt. Tento formát zahrnuje archiv sestávající ze vzorků, zvuků a zpracování.

NKI soubory lze otevřít pouze prostřednictvím sampleru[1] Kontakt, který je k dispozici jako audio plugin ve formátech VST, AU a AAX. V rámci sampleru je soubor NKI vnímán jako samostatný virtuální nástroj. Bez sampleru Kontakt nelze soubory NKI otevřít v DAW, jelikož Kontakt slouží jako nezbytný prostředník mezi DAW a virtuálním nástrojem ve formátu NKI.

Je důležité poznamenat, že NKI formát je uzavřený a je striktně vázán na použití s produkty Native Instruments. Tento přístup zajišťuje maximální kompatibilitu a stabilitu při použití NKI souborů v rámci Kontakt sampleru, ale současně omezuje možnost použití těchto souborů v jiných prostředích nebo s jinými samplery.

Formát NKS

NKS (Native Kontrol Standard) nelze považovat za plnohodnotný formát audio pluginů. Místo toho se jedná o technologii, která umožňuje hlubší integraci hardwarových kontrolérů, jako jsou MIDI-klávesnice, s virtuálními knihovnami NKI a VST/AU pluginy.

NKS je vytvořen společností Native Instruments a slouží k zajištění lepšího spojení mezi softwarem a hardwarem, čímž se zjednodušuje ovládání a zlepšuje pracovní tok při hudební produkci. Ikona NKS u pluginů a hardwaru poukazuje na podporu integrace, ale je důležité si uvědomit, že NKS sám o sobě není samostatným formátem.

Jednou z výhod technologie NKS je, že umožňuje uživatelům snadno procházet zvuky a efekty pomocí hardwarových kontrolérů a zobrazovat relevantní informace na displejích zařízení. To umožňuje hudebníkům soustředit se více na tvorbu hudby, aniž by museli neustále přepínat mezi hardwarem a softwarem.

Formát SFZ

SFZ (Sforzando) je sofistikovaný formát určený pro uchovávání hudebních dat, který slouží jako standard pro definici chování virtuálních hudebních

[1] Sampler je zařízení nebo software, který umožňuje záznam, přehrávání a manipulaci se vzorky zvuků.

nástrojů. Formát SFZ specifikuje, jak budou zvukové knihovny a efekty znít a jak budou reagovat na různé hudební parametry.

Jako alternativa k formátu NKI poskytuje SFZ možnost vytvářet a upravovat virtuální zvukové knihovny a využívat je v různých samplerových aplikacích. Na rozdíl od NKI, který je vázán na Native Instruments Kontakt, je formát SFZ otevřený a lze ho používat s různými samplery, což umožňuje větší flexibilitu a širší kompatibilitu s různými hudebními softwary.

Formát SFZ zahrnuje textový soubor s příponou .sfz, který obsahuje instrukce a parametry pro jednotlivé vzorky, a samotné audio soubory ve formátech jako WAV nebo FLAC. Tento otevřený přístup umožňuje hudebníkům a vývojářům snadno upravovat a přizpůsobovat zvuky podle svých potřeb a preferencí, což podporuje kreativitu a inovaci v oblasti hudební produkce.

Formát JSFX

JSFX (Jesusonic Effects) představuje specializovaný formát audio pluginů, který byl vyvinutý pro použití v DAW REAPER od společnosti Cockos. Tento formát je kompatibilní s operačními systémy Windows, macOS a Linux. JSFX pluginy jsou založeny na skriptovacím jazyce EEL2, což poskytuje uživatelům snadný a přístupný způsob tvorby vlastních audio efektů a nástroj v této hostitelské aplikaci.

Formát JSFX nabízí uživatelům REAPERu řadu výhod, jako jsou například možnost rychlého prototypování, uživatelsky přívětivé prostředí pro vývoj a testování, a široké spektrum možností pro experimentování se zpracováním zvuku. Jeho jednoduchost a flexibilita přispívají k jeho popularitě mezi uživateli REAPERu, kteří si rádi vytvářejí své vlastní nástroje a efekty. Navíc, díky podpoře více operačních systémů, umožňuje tento formát rozšíření dostupnosti vytvořených pluginů pro širší spektrum uživatelů.

Formát MAS

Formát MAS (Modular Audio System) představuje proprietární formát audio pluginů, který byl vyvinut společností MOTU (Mark of the Unicorn) pro použití ve své DAW Digital Performer a dalších MOTU aplikacích. Tento formát je podporován na operačních systémech Windows a macOS. MAS nabízí uživatelům modulární prostředí, které umožňuje flexibilní propojení a konfiguraci audio a MIDI cest, což vede k lepšímu řízení a organizaci audio signálů.

Jelikož je formát MAS uzavřený a podporován pouze v MOTU aplikacích, jeho použití je omezeno na uživatele těchto softwarových řešení. Tento fakt může značně ovlivnit rozšíření a dostupnost MAS pluginů na trhu. Navzdory omezené kompatibilitě s jinými DAW a aplikacemi, formát MAS zůstává oblíbeným řešením pro uživatele MOTU softwaru, kterým nabízí robustní a modulární platformu pro tvorbu a správu audio projektů.

2.4 Podpora formátů audio pluginů

Tato podkapitola se zaměřuje na analýzu a porovnání podpory různých formátů audio pluginů v operačních systémech a DAW. Informace jsou prezentovány v tabulkách 5.2 a 2.2, které poskytují přehled o kompatibilitě jednotlivých formátů s různými DAW a operačními systémy.

	Windows	macOS	Linux
VST	✓	✓	✓
VST3	✓	✓	✓
AU		✓	
AAX	✓	✓	
CLAP	✓	✓	✓
DX	✓		
LV2			✓

Tabulka 2.1: Porovnání podpory formátů audio pluginů v operačních systémech.

	VST	VST3	AU	AAX	CLAP	DX	LV2
Ableton Live	✓	✓	✓				
Adobe Audition	✓	✓	✓			✓	
Audacity	✓	✓	✓				✓
Bitwig Studio	✓	✓	✓		✓		
Cubase	✓	✓				✓	
FL Studio	✓	✓	✓			✓	
GarageBand			✓				
LogicPro			✓				
MAGIX Samplitude	✓	✓				✓	
ProTools				✓			
REAPER	✓	✓	✓			✓	✓
Studio One	✓	✓	✓				

Tabulka 2.2: Porovnání kompatibility formátů audio pluginů s různými DAW.

Na základě informací z tabulek a analýzy článků lze odvést, že formáty VST, VST3 a AU jsou nejběžnější formáty pluginů s největší podporou jak v operačních systémech, tak i v DAW. Hlavní výhodou formátu VST oproti ostatním je, že pluginy v tomto formátu mohou fungovat ve všech operačních systémech a ve větším počtu DAW.

Z praktického hlediska mezi formáty neexistují skoro žádné rozdíly. Hlavní rozdíl spočívá v technických bodech, které neovlivňují práci uživatele – zvláštnostech vývoje, bitové hloubce pluginu a metodách jeho dodávky. Je důležité zvážit potřeby koncových uživatelů, kompatibilitu s různými DAW a podporu operačních systémů při výběru vhodného formátu pro implementaci pluginu.

Po hlubší analýze výhod a nevýhod jednotlivých formátů, včetně VST, VST3 a AU, lze pozorovat, že VST a VST3 nabízejí širokou podporu a

kompatibilitu. VST3 je vylepšená verze VST, která přináší větší efektivitu a pokročilejší funkce, jako je dynamické přidělování zdrojů, ale stále je to formát, který se drží svých kořenů v VST. V případě AU formátu je důležité zohlednit, že tento formát je určen výhradně pro macOS, což omezuje jeho univerzálnost.

Vzhledem k široké podpoře a kompatibilitě VST a VST3 formátů, se jeví tyto formáty jako ideální kandidáty pro implementaci VST pluginu pro výpočet STIPA parametru. Tímto způsobem lze zajistit maximální dostupnost a použitelnost pluginu pro co nejširší spektrum uživatelů v různých prostředích.

2.5 Bitová hloubka pluginů

Pokud jde o bitovou hloubku, audio pluginy se často dodávají ve dvou variantách – 32bitové a 64bitové. Každá z těchto verzí má svá omezení a výhody, které by měly být zohledněny při výběru kompatibilního pluginu pro DAW.

Audio pluginy s 32bitovou hloubkou mají omezení využití paměti RAM, která nepřesahuje 4 GB. Mohou být také nekompatibilní s 64bitovými operačními systémy a 64bitovými DAW. Pokud 32bitový plugin dosáhne svého paměťového limitu, může začít fungovat nestabilně. Mezi 32bitové formáty audio pluginů patří VST/VSTi, RTAS a DX.

Oproti tomu 64bitové audio pluginy nemají žádná podobná omezení. Tyto verze mohou využít veškerou dostupnou RAM v systému, což umožňuje operačnímu systému flexibilněji spravovat prostředky počítače a rozdělovat je mezi všechny nástroje a pluginy používané v projektu. Mezi 64bitové formáty audio pluginů patří VST3, AU, AAX a CLAP.

Pro optimální výkon je doporučeno používat 32bitové pluginy v 32bitových DAW a 64bitové pluginy v 64bitových DAW. Nicméně některé 64bitové DAW obsahují vestavěné 32bitové konvertory, které umožňují využití 32bitových pluginů v 64bitovém prostředí. Tyto pluginy však nejsou optimalizované pro 64bitové verze DAW a mohou se chovat nestabilně. Pro překlenutí omezení 32bitových formátů v jiných DAW lze použít nezávislé konvertory, jako je například jBridge.

Ve tabulce 5.2 všechny formáty pluginů mohou podporovat jak 32-bitové, tak 64-bitové architektury, v závislosti na konkrétní realizaci pluginu a hostitelské aplikace. Ačkoliv mnoho z těchto formátů technicky podporuje 32bitové a 64bitové verze, je důležité poznamenat, že průmysl jako celek se posouvá směrem k 64bitovým systémům. Toto je spojeno s tím, že 64bitové systémy mohou efektivněji spravovat větší množství paměti, což je zvláště důležité pro zpracování zvuku a hudební produkci.

2.6 Instalace pluginů

Na konci této kapitoly lze shrnout, že formáty instalačních souborů pro audio pluginy a virtuální nástroje se liší v závislosti na použitém operačním systému a vybraném formátu pluginu. Některé pluginy a nástroje jsou distribuovány

v komprimovaném archivu ve formátu ZIP, který obsahuje jeden nebo více souborů pluginu. Pro instalaci je nezbytné tyto soubory manuálně přesunout do příslušných adresářů určených pro ukládání pluginů v operačním systému.

Následující seznam uvádí běžné formáty instalačních souborů pro operační systémy Windows a macOS:

- soubory s příponou .dll jsou určeny pro instalaci VST pluginů v systému Windows.
- soubory s příponou .vst3 slouží pro instalaci VST3 pluginů v systému Windows.
- soubory s příponou .vst/.vst3 se používají pro instalaci VST/VST3 pluginů v systému macOS.
- soubory s příponou .component jsou potřebné pro instalaci AU pluginů v systému macOS.
- soubory s příponou .aax se využívají pro instalaci AAX pluginů v systémech Windows a macOS.

Při výběru vhodného audio pluginu je tedy důležité zohlednit nejen jeho formát a bitovou hloubku, ale také kompatibilitu s používaným operačním systémem. Pozornost věnovaná formátu instalačních souborů a dodržování instrukcí poskytnutých vývojářem pluginu může zajistit správnou instalaci a fungování v rámci digitální audio pracovní stanice (DAW). Tato skutečnost by měla být zohledněna při analýze a výběru vhodných pluginů pro konkrétní potřeby a aplikace. Stejně tak je důležité vědět licenční podmínky při výběru audio pluginu. Některé pluginy jsou distribuovány jako freeware, což znamená, že jsou k dispozici zdarma pro všechny uživatele. Jiné pluginy jsou k dispozici jako shareware, což znamená, že jsou poskytnuty zdarma na omezenou dobu nebo s omezenými funkcemi, a vyžadují nákup licence pro plné využití.

Kapitola 3

Porovnání vývojových prostředí

Tato kapitola se zaměřuje na různá prostředí, která mohou být využita při vytváření audio pluginů. Je zde popsána problematika programovacích jazyků, existujících frameworků a grafických editorů, které slouží k usnadnění procesu vývoje audio pluginů.

3.1 Programovací jazyky

Nasledující sekce popisuje jak specializované, tak i aktivně používané programovací jazyky, pomocí kterých lze implementovat audio pluginy.

C++

V kontextu vývoje audio pluginů je jazyk C++ jedním z nejvíce standardizovaných a běžně využívaných programovacích jazyků. Díky své stabilitě a flexibilitě je C++ jednou z nejlepších voleb pro vývoj komplexních a výkonných aplikací, jako jsou audio pluginy.

Vzhledem k tomu, že společnost Steinberg je tvůrcem technologie VST, je pro vývoj audio pluginu vhodné zaměřit se na jejich standardní přístup. Vývoj audio pluginů v C++ je často zaměřen na použití standardního SDK[1], který je volně dostupný na webových stránkách Steinbergu [8]. Verze 3.7.6 SDK, která je momentálně nejnovější, je specificky určena pro tvorbu pluginů ve formátu VST3.

Audio pluginy vytvořené pomocí jazyka C++ a Steinberg SDK jsou univerzálně kompatibilní a lze je snadno integrovat do různých DAW. Výsledný plugin může být spuštěn na různých operačních systémech, jako jsou Windows, macOS a Linux, avšak pouze v hostitelských aplikacích, které jsou kompatibilní s formátem VST3 (viz Tabulka 2.2).

Je třeba zmínit, že i když existují různé frameworky a nástroje usnadňující vývoj audio pluginů v C++, také je možné vytvářet pluginy *vanilla*, tedy bez použití těchto nástrojů. Pomocí C++ je možné vytvářet pluginy téměř ve všech možných formátech.

[1] Standard Developer Kit je standardní vývojářská sada pro vytvaření software.

Python

Python je populární vysokoúrovňový programovací jazyk, který je ceněn pro svou snadno čitelnou syntaxi a širokou podporu knihoven. I když Python nemusí být první volbou pro vývoj realtime audio pluginů kvůli svému nižšímu výkonu ve srovnání s nativními jazyky, stále existuje několik možností pro vývoj audio pluginů v Pythonu:

- `PyDub` je jedna z knihoven, která umožňuje manipulovat se zvukovými soubory a efekty. `PyDub` podporuje širokou škálu audio formátů a umožňuje provádět základní úpravy zvuku, jako je řezání, spojování a export zvuku. Tato knihovna je ideální pro jednoduché úpravy zvuku, ale nemusí být vhodná pro realtime aplikace.
- `PyAudio` je další knihovna pro Python, která poskytuje rozhraní pro vstup a výstup zvukových dat. `PyAudio` umožňuje práci se streamy zvuku v reálném čase a lze jej použít pro zpracování zvuku v interaktivních aplikacích.
- `PythonVSTWrapper` a `PyLV2` jsou knihovny, které umožňují vývojářům vytvářet audio pluginy ve formátech VST a LV2. Tyto knihovny poskytují rozhraní pro Python, které zjednodušuje vývoj pluginů a umožňuje vývojářům soustředit se na algoritmy zpracování zvuku.
- Grafická uživatelská rozhraní pro Python audio pluginy lze vytvořit pomocí knihoven, jako jsou `PyQt` a `PyGTK`. Tyto knihovny poskytují řadu nástrojů a widgetů pro tvorbu interaktivních uživatelských rozhraní, která lze snadno integrovat do audio pluginů.

Bez ohledu na to, zda je cílem jednoduchá manipulace se zvukem nebo vytváření sofistikovaných pluginů, Python nabízí nástroje a flexibilitu potřebnou k naplnění těchto požadavků.

Rust

Rust je moderní systémový programovací jazyk, který byl vyvinutý společností Mozilla Research. Tento jazyk je schopen kompilovat zdrojový kód přímo do nativního strojového kódu, což vede k výkonným aplikacím s nízkou latencí - faktor, který je pro audio vývoj zásadní.

Rustový kompilátor nabízí silnou statickou typovou kontrolu, což znamená, že většina chyb se odhalí již při kompilaci, a ne za běhu programu. Kromě toho Rust nabízí *zero-cost abstractions*, což znamená, že vývojáři mohou využívat vysokoúrovňové konstrukce, aniž by to negativně ovlivnilo výkon programu.

Rust poskytuje několik knihoven, které vývojářům usnadňují práci. Knihovna vst-rs umožňuje vývoj VST2 pluginů v Rustu, zatímco knihovna sample poskytuje soubor nástrojů pro práci se zvukovými daty. Pro vytváření uživatelských rozhraní lze využít knihovnu conrod. Rust se také dá integrovat s jazyky jako C++ nebo Python.

Java

Následující text vychází z [9]. Pro implementaci audio pluginů je další možností využití *jVSTwRapperu*, což je multiplatformní projekt založený na technologii VST. Tento projekt umožňuje vytvářet audio pluginy pomocí programovacího jazyka Java. Pluginy vytvořené s využitím jVSTwRapperu lze spustit na operačních systémech Windows, macOS a Linux, přičemž jeden univerzální kód je kompatibilní s formáty pluginů VST (2.4), AU a LADSPA.

Wrapper je vlastně zkompilovaný binární kód. Hostitelská DAW aplikace předává vzorky wrapperu, který následně volá JVM (Java Virtual Machine) a audio plugin napsaný v Javě. Hostitelská aplikace DAW spravuje vstupní a výstupní data, zatímco vývojář implementuje prostředí wrapperu a algoritmus pro zpracování audio signálu. Pro tvorbu grafického rozhraní je možné použít knihovny Swing nebo JavaFX.

jVSTwRapper navíc podporuje a umožňuje práci se skriptovacím jazykem JRuby a programovacím jazykem Clojure, což je moderní dialekt jazyka Lisp. Tímto způsobem poskytuje jVSTwRapper flexibilitu a širokou škálu možností pro vývojáře při vytváření audio pluginů.

Kotlin/Native

Kotlin/Native je variantou jazyka Kotlin umožňující kompilaci do nativního kódu. Kotlin/Native byl vyvinutý společností JetBrains s cílem poskytnout vývojářům nástroje pro snadnou tvorbu nativních aplikací. Tento jazyk umožňuje vývoj nativních aplikací pro široké spektrum operačních systémů, včetně Windows, macOS, Linux, Android a iOS.

Kotlin/Native je schopen kompilovat zdrojový kód do nativního strojového kódu, což vede k výkonným aplikacím s nízkou latencí - faktor, který je pro audio vývoj zásadní. Dále Kotlin/Native podporuje přímé volání platformy a přístup k nativním API, což umožňuje vývojářům využívat nativní knihovny a nástroje.

V rámci Kotlin/Native lze využit multiplatformní knihovny, jako je `korlibs`. Tato knihovna nabízí rozsáhlou podporu pro práci se zvukem, zpracování signálů a vytváření grafických rozhraní. Kromě toho Kotlin/Native podporuje integraci s jazyky, jako jsou C, C++ nebo Swift, což rozšiřuje možnosti vývoje a integrace.

Csound

Při tvorbě následujícího textu bylo čerpáno z [11]. Csound je programovací jazyk vyvinutý na základě jazyka C, který byl navržen spolupracovníkem MIT a skladatelem Barrym Vercoem v roce 1984. Je optimalizován pro generování zvuku a zpracování zvukových signálů.

Jazyk obsahuje více než 450 operačních kódů (opcodes), které vývojáři využívají k vytváření virtuálních nástrojů a patchů. Csound zpracovává dva speciálně formátované textové soubory jako vstup. Soubor typu *orchestra* (.orc) popisuje strukturu nástrojů, zatímco soubor typu *score* (.sco) definuje noty

a další parametry podle časové osy. Po zpracování instrukcí obsažených v těchto souborech se vytváří výstupní zvukový soubor nebo audio stream v reálném čase.

CsoundVST lze použít jako samostatné GUI[2] pro Csound, ale také jako VST plugin v hostitelských aplikacích. Pro tvorbu VST pluginů s využitím Csoundu se používá multifunkční frontend CsoundVST, který je založen na Csound API. Csound lze volat prostřednictvím jiných programovacích jazyků, jako jsou C++, Java a Python. K programování v Csoundu se často využívají IDE[3], jako jsou CsoundQt a Blue.

Je důležité poznamenat, že programovací jazyk Csound je primárně zaměřen na generování zvuku a jeho zpracování, nikoli na jeho editaci či měření parametrů. Díky tomu je Csound vhodný pro vývojáře zaměřené na syntézu a tvorbu zvukových efektů, ale méně vhodný pro úpravy zvuku nebo analýzu zvukových parametrů.

SuperCollider

SuperCollider představuje další efektivní způsob, jak implementovat audio plugin. Jedná se o virtuální prostředí a programovací jazyk, který vytvořil James McCartney v roce 1996 pro programování, konstrukci logických struktur a syntézu zvuku v reálném čase.

Podle [12] prostředí SuperCollider se skládá ze tří hlavních komponent: Scsynth (audio server v reálném čase), sclang (samotný programovací jazyk) a scide (editor pro sclang s integrovaným systémem nápovědy). Komunikace mezi těmito komponentami probíhá prostřednictvím protokolu OSC[4].

SuperCollider se nejčastěji využívá pro tvorbu živých vystoupení, avšak s pomocí SuperColliderAU lze toto prostředí také použít jako audio plugin. SuperColliderAU je AudioUnit wrapper, který umožňuje integrovat servery SuperCollider do hostitelských aplikací podporujících pluginy ve formátu AU, avšak pouze na operačním systému macOS. Vložený server lze následně ovládat prostřednictvím protokolu OSC.

Tato možnost poskytuje širokou škálu funkcí pro tvorbu a manipulaci se zvukem, což umožňuje vývojářům vytvářet složité a kreativní zvukové efekty a nástroje. Je třeba poznamenat, že SuperColliderAU je omezen na platformu macOS, což může omezit jeho použití v rámci multiplatformních projektů. Přesto je SuperCollider ceněný pro svou flexibilitu a možnosti, které nabízí v oblasti zpracování zvuku a živých vystoupení.

[2] Grafické uživatelské rozhraní je uživatelské rozhraní, které umožňuje uživateli ovládat aplikaci pomocí interaktivních grafických ovládacích prvků.

[3] Integrated Development Environment je sada softwarových nástrojů používaných programátory pro vývoj softwaru.

[4] Open Sound Control je paketový protokol pro komunikaci multimediálních zařízení (elektronické hudební nástroje, zejména syntezátory, počítače atd.)

FAUST

FAUST (Functional Audio Stream) je specializovaný jazyk pro vývoj audio aplikací a pluginů. Byl navržený na francouzské výzkumné instituci GRAME a je zaměřen na vysokoúrovňové programování zvukových procesů.

FAUST je unikátní tím, že umožňuje vývojářům popsat jakýkoliv audio algoritmus pomocí jednoduchých matematických operací a vzorců, které jsou pak kompilovány do vysoce optimalizovaného kódu v jazyce C++, C nebo jiných jazycích. Díky tomu lze generovat pluginy pro různé formáty, jako jsou VST, AU, LADSPA a další.

FAUST také podporuje vytváření grafických uživatelských rozhraní pomocí speciálních funkcí a skriptů. Díky tomu je možné vytvářet komplexní GUI s interaktivními prvky, jako jsou tlačítka, posuvníky, displeje a další.

Haskell

Haskell je čistě funkcionální programovací jazyk, který se zaměřuje na matematický přístup k programování a typovou bezpečnost. Jednou z unikátních vlastností Haskellu je snadné skládání funkcí a manipulaci s daty. To umožňuje efektivně modelovat signálové procesy a algoritmy pomocí matematických funkcí a výrazů.

Haskell také nabízí silnou statickou kontrolu typů, která zajišťuje, že většina chyb je odhalena během kompilace, než dojde k runtime chybám. To umožňuje vývojářům psát stabilnější kód, který je méně náchylný k chybám při zpracování audio signálu.

Přestože není primárně určen pro vývoj audio pluginů, existuje několik knihoven, které umožňují práci se zvukem v Haskellu, jako je *TidalCycles* pro živé kódování hudby nebo Euterpea pro zpracování a generování zvuku. Tyto knihovny poskytují širokou škálu funkcí pro práci se zvukem, včetně syntézy zvuku, zpracování signálů a interakce se zvukovými zařízeními.

Nim

Nim je moderní, efektivní a expresivní programovací jazyk, který kombinuje prvky z jazyků jako jsou Python, Ada a Modula. Jazyk Nim je staticky typovaný a kompiluje se do nativního kódu pro mnoho platforem, včetně Windows, macOS, Linux, Android a iOS.

Nim nabízí výhody nativního výkonu a efektivity při zachování přístupnosti a snadné syntaxe. Jazyk je navržen tak, aby byl co nejvíce expresivní a flexibilní, což umožňuje vývojářům psát kód, který je snadno pochopitelný a údržbový.

Jako jazyk s nativní kompilací je Nim schopen vytvářet výkonné aplikace s nízkou latencí, což je klíčové pro real-time audio vývoj. Navíc, jazyk Nim podporuje přímé volání nativních API a C knihoven, umožňující využívat již existující audio processing nástroje a knihovny.

Nyquist

Nyquist je programovací jazyk pro zvukovou syntézu a analýzu, který byl vyvinutý na Carnegie Mellon University. Jazyk je založen na Lispu, ale má speciální vlastnosti pro zvukové operace a hierarchickou sekvenci zvukových událostí.

Nyquist je schopen manipulovat s oběma, časem a frekvencí, což umožňuje široké spektrum audio zpracování, včetně time-stretching, pitch-shifting, convolving, a dalších. Jazyk obsahuje bohatou sadu nástrojů pro syntézu zvuku a zpracování zvukových signálů, a je optimalizován pro transformace zvuku.

Díky tomu, že Nyquist je interpretovaný jazyk, je možné měnit a testovat kód za běhu, což je výhodné při rychlém prototypování a interaktivní tvorbě zvuku. Nyquist podporuje vytváření VST pluginů prostřednictvím jeho rozšíření Nyquist Plug-in Support.

MATLAB

MATLAB (Matrix Laboratory), vyvinutý společností The MathWorks v roce 1984, je skriptovací programovací jazyk a interaktivní programovací prostředí, které se široce využívá v oblastech fyziky, matematiky a dalších technických disciplínách. Tento jazyk umožňuje práci s maticemi, vykreslování 2D a 3D grafů funkcí, implementaci algoritmů, počítačové simulace, analýzu a prezentaci dat, a také tvorbu aplikací včetně jednoduchého GUI.

Čerpáno z [10]. MATLAB nativně podporuje export vytvořených projektů a simulací ve formátu VST (2.4) a VST3. Kromě toho poskytuje možnost vytvořit .zip soubor obsahující generovaný kód C/C++ a soubor projektu JUCER, který je vhodný pro použití ve frameworku JUCE (viz sekce 3.2). Vygenerovaný soubor lze následně upravit pro potřeby konkrétního pluginu nebo zkompilovat do jiného formátu než VST. Tato možnost však vyžaduje speciální licenci MATLAB Coder.

Díky této nativní podpoře a možnostem exportu se MATLAB stává atraktivním vývojovým prostředím pro tvorbu audio pluginů, zejména pro vývojáře zaměřené na matematické modelování a analýzu zvukových signálů. Využití MATLABu při vývoji audio pluginů také usnadňuje přechod mezi různými fázemi vývoje, od teoretického modelování až po implementaci a optimalizaci algoritmů.

3.2 Framewroks

Tato sekce se zaměřuje na představení různých frameworků pro vytváření audio pluginů v různých programovacích jazycích.

Framework představuje softwarovou strukturu, která slouží jako nástroj podpory při programování, vývoji a organizaci různých softwarových projektů. Mezi jeho vlastnosti může patřit široká škála podpůrných programů, knihoven API, návrhových vzorů a doporučených postupů při vývoji.

JUCE

S rozvojem technologií audio pluginů se začaly objevovat různé frameworky a knihovny, a to jak pro zjednodušení vývoje, tak i pro rozšíření možností při implementaci pluginů a také jejich funkcionality. Jedním z takových nástrojů je framework JUCE, který byl vyvinutý společností ROLI v roce 2004.

JUCE je multiplatformní framework založený na svobodném kódu napsaném v programovacím jazyce C++. Jeho hlavní ambicí je poskytnout vývojářům nástroje pro vytvoření univerzálního zdrojového kódu, který je kompatibilní se všemi běžnými operačními systémy, a to prostřednictvím standardního SDK. K výhodám JUCE patří možnost napsat univerzální kód pro různé formáty pluginů (VST, VST3, AU, AAX a LV2), a to s využitím intuitivních nástrojů pro vývoj GUI, které do té doby nabízelo omezené možnosti. JUCE podporuje operační systémy Windows, macOS, Linux, iOS a Android. Více informací lze nalézt na [13].

Nerozlučnou součástí JUCE je aplikace ProJucer, která byla vytvořena pomocí tohoto frameworku a slouží k vizuálnímu návrhu a úpravám GUI. Jucer je schopen generovat C++ kód implementující vybranou strukturu GUI. JUCE sám o sobě není schopen překládat kód, proto spoléhá na překladače v rámci IDE a kompilátory pro finální překlad a spuštění kódu. Framework umožňuje tvorbu pluginů v moderních formátech (např. VST3) i v dřívějších verzích (např. VST ve verzi 2.4).

JUCE je distribuován pod licencí GNU GPLv3.

iPlug2

Další významný nástroj pro vývoj audio pluginů představuje framework iPlug2, který byl uveden na trh v roce 2018 vývojářem Oli Larkinem. Tato verze následuje původní iPlug, který byl vydán v roce 2008 jako součást knihovny WDL společnosti Cockos.

Podobně jako JUCE, i iPlug2 je framework založený na programovacím jazyce C++, a jeho hlavní funkcí je umožnit vývojářům vytvořit univerzální zdrojový kód pomocí standardního SDK od společnosti Steinberg. Oproti JUCE je však iPlug2 podporován pouze na operačních systémech Windows, macOS a iOS.

Mezi klíčové vlastnosti, které iPlug2 odlišují od JUCE, patří:

- podpora FAUST (viz 3.1);
- podpora Web Audio API – webový formát pro zvukové efekty a nástroje fungující na bázi prohlížeče;

Tento framework poskytuje zvýšenou flexibilitu při tvorbě a implementaci audio pluginů díky jeho podpoře pro širokou škálu technologií. Podobně jako JUCE, iPlug2 je distribuován pod licencí GNU GPLv3, což umožňuje vývojářům využívat a modifikovat tento framework v rámci jeho licenčních podmínek.

DPF

DISTRHO Plugin Framework (DPF), vytvořený Filipem Coelhem, je další významný framework, který zasluhuje pozornost v rámci vývoje audio pluginů. Tento framework, psaný v jazyce C++, se vyznačuje menším rozsahem funkcí v oblasti implementace a GUI ve srovnání s JUCE a iPlug2. Nicméně, jeho unikátní předností je možnost vytvářet audio pluginy pro Linux ve specifických formátech, jako jsou LADSPA a LV2. Toto je vlastnost, která ho odlišuje od ostatních a poskytuje významnou hodnotu pro vývojáře zaměřené na open-source a Linuxové prostředí.

DPF je distribuován pod licencí ISC, což vývojářům poskytuje docela velkou flexibilitu při využití tohoto frameworku.

RackAfx

Následující text vychází z [14]. RackAfx je framework vyvinutý programátorem Willemem Pirklem, který slouží k návrhu a testování zvukových algoritmů v C++ v reálném čase. Tento nástroj funguje jako prototypovací panel umožňující rychlé vytváření a konfiguraci prototypových rozhraní. Uživatelé mohou exportovat své algoritmy ve formátu ASPiK a poté pokračovat ve vývoji pluginu v různých operačních systémech a DAW.

RackAfx poskytuje 1044 ovládacích slotů, do kterých může být umístěn jednotlivý ovládací prvek, jako je knoflík, tlačítko nebo měřič. Je plně kompatibilní s MIDI a integruje grafický návrhářský nástroj pro vytváření komplexnějších designů a grafických rozhraní. Vytvořené GUI je nezávislé na platformě a zobrazuje se konzistentně napříč různými platformami.

RackAfx se vyznačuje dvěma specifickými moduly, které jsou integrovány jako systémové moduly: `IR Convolver` a `FIR Filter Designer`. Tyto moduly umožňují ukládat impulsní odezvy pluginů jako .wav soubor. Framework také umožňuje vytvářet pluginy jako efekty typu Insert nebo AUX a následně analyzovat jejich frekvenční, fázové, pulzní a krokové charakteristiky a zobrazit vstupy a výstupy v čase nebo frekvenci.

I přes mnoho výhod, RackAfx má své nevýhody, jednou z nich je nízká kvalita návrhu kódu DSP[5]. Přesto lze pluginy implementované pomocí RackAfx exportovat ve formátech VST3, AU a AAX.

RackAfx je distribuován pod modifikovanou licencí BSD.

VST.NET v2

VST.NET v2 je framework založený na programovacím jazyce C#. Framework byl navržen s využitím technologie .NET 6, kterou vyvinula společnost Microsoft v roce 2022. Klíčovou vlastností technologie .NET je její schopnost umožnit kompatibilitu a interoperabilitu mezi různými programovacími jazyky. Tím pádem, vývojáři mohou využívat jednotné jmenné prostory, knihovny a API napříč různými programovacími jazyky.

[5]Digital Signal Processing je využití digitálního zpracování signálu pomocí počítačů nebo specializovaných procesorů pro širokou škálu operací zpracování signálu.

VST.NET v2 využívá tzv. *interop* vrstvu technologie .NET, která je postavena na technologii COM Interop. Tato vrstva usnadňuje přechod mezi jazyky C# a C++, což zvyšuje flexibilitu při vývoji pluginů.

Významnou vlastností frameworku VST.NET v2 je možnost vytvářet pluginy pro Steinberg VST 2.0 až VST 2.4 API. Tento framework také poskytuje možnost vývoje hostitelských aplikací, což jej odlišuje od většiny ostatních dostupných frameworků.

VST.NET v2 je distribuován pod licencí GNU LGPLv2.1.

Cabbage

Cabbage je framework, který je založen na programovacím jazyce Csound. Podobně jako JUCE, jednou z jeho významných výhod je široké možnosti implementace GUI. Jak uvádí [15], Cabbage umožňuje exportovat projekty do formátů VST, AU, ale také jako samostatné aplikace.

Cabbage nabízí několik klíčových funkcí, které zvyšují jeho užitečnost a flexibilitu. Mezi tyto funkce patří:

- integrovaný patcher pro zpracovaní složitějších procesních řetězců;
- podpora skriptu napsaných v jazyce Python;
- vícevlaknový software;
- možnost přizpůsobení GUI pomocí SVG;
- podpora OSC protokolu;

Framework Cabbage je distribuován pod licencí GNU GPLv3.

3.3 Editory

Tato sekce se zaměřuje na nástroje, které nabízí alternativní přístup k vývoji audio pluginů, konkrétně prostřednictvím vizuálního programování. Tyto nástroje umožňují vytvářet pluginy bez nutnosti ručního psaní kódu, což může značně zjednodušit proces vývoje pro některé uživatele.

Max/MSP

Jedním z význačných příkladů těchto nástrojů je grafický editor Max (dříve známý jako MSP), vyvinutý společností Cycling'74 v roce 1998. Max je v současné době komerční software, který poskytuje vizuální programovací jazyk a prostředí určené pro tvorbu a zpracování hudby, zvuku a multimédií.

Podle zdroje [16], Max využívá takzvané okno *Patcher* pro vizuální programování. Toto prostředí je tvořeno řadou modulů, nebo *objektů*, které lze chápat jako funkce zpracovávající vstupní data a generující výstupní data. Tyto objekty jsou reprezentovány jako vizuální elementy v okně Patcher a jsou vzájemně propojeny vizuálními kabely, které umožňují vytváření komplexních algoritmů.

V roce 2022 společnost Cycling '74 představila nové patching prostředí nazvané RNBO. Toto nové prostředí poskytuje uživatelům možnost exportovat jejich projekty jako kompilovaný kód ve formátech VST3 a AU, nebo jako zdrojový kód v jazyce C++.

SynthEdit

SynthEdit, komerční software vytvořený programátorem Jeffem McClintockem v roce 1987, je dalším příkladem vizuálního programovacího prostředí určeného pro vývoj audio pluginů. Pracuje na podobném principu jako Max, avšak klíčovým rozdílem je možnost přidávat do projektů moduly napsané v jazyce C++.

Od svého uvedení na trh v roce 2005 SynthEdit podporoval export projektů výhradně ve formátu VST2. Nicméně, s pokračujícím vývojem technologie VST byla přidána také podpora formátu VST3, což poskytlo SynthEditu výhodu, jelikož se stal na dlouhou dobu jediným vizuálním programovacím prostředím s podporou 64bitového formátu.

FlowStone (SynthMaker)

FlowStone, nástroj pro vývoj VST pluginů vyvinutý společností DSP Robotics, představuje unikátní kombinaci grafického a textového programování s využitím jazyka Ruby.

Podobně jako Max a SynthEdit, FlowStone umožňuje uživatelům vytvářet algoritmy propojením vizuálních funkčních stavebních bloků, nazývaných komponenty. Význačnou vlastností FlowStone jsou uživatelsky definované moduly, které umožňují rychlé sestavení složitějších algoritmů prostřednictvím vytváření hierarchie modulů. Kromě standardních funkcí, FlowStone podporuje komunikaci s řadou externích zařízení, jako jsou USB zařízení, I/O karty, webové kamery a různé audio hardwary, jak je uvedeno v [17].

FlowStone je komerčním produktem, ale je také součástí ekosystému audio pluginů pro DAW FL Studio od společnosti Imagine-Line Software. Zatímco export projektu ve formátu VST2 je možný pouze v samostatné verzi FlowStone, uživatelé mohou využívat pluginy vytvořené v FlowStone přímo v rámci pluginu FlowStone pro FL Studio. Je důležité poznamenat, že FlowStone je dostupný výhradně ve 32bitovém formátu.

PureData

Následující text vychází z [18]. PureData, otevřený vizuální programovací jazyk, nabízí alternativní metodu pro implementaci audio pluginů prostřednictvím paradigmatu programování datových toků v reálném čase. Tento jazyk, vytvořený Millerem Puckettem s cílem umožnit vytváření interaktivních počítačových hudebních a multimediálních děl, představuje volnou alternativu k Max/MSP.

Zatímco zpracování zvuku je hlavním zaměřením PureData, tento jazyk byl v průběhu času rozšířen pro použití v širší oblasti digitálního zpracování

signálu, včetně zpracování obrazu a videa. Mezi další významné vlastnosti PureData patří jeho schopnost interagovat s fyzickým světem, jak je to například možné prostřednictvím zařízení jako je Arduino nebo Raspberry Pi.

Pokud jde o export projektů do formátů vhodných pro audio pluginy, existují pro PureData specifická rozšíření. PdVST umožňuje spouštět PureData patchy (a programy pd.exe) jako VST pluginy, avšak tato funkce je k dispozici pouze pro operační systém Windows. Pro další formáty může být použit meta-plugin Camomile, který podporuje formáty VST3, AU a LV2 na operačních systémech Windows, macOS a Linux.

3.4 Přehled licenčních politik

V kontextu vývojových nástrojů a frameworků, které byly popsány v předchozích sekcích, je klíčové porozumět licenčním politikám, které se na tyto produkty aplikují. Tato sekce se zabývá stručným přehledem některých základních licenčních modelů, které jsou běžně používány v rámci open-source i komerčního softwaru. Následující text byl čerpán z [20], [21], [22] a [23].

GNU GPLv3

GNU General Public License verze 3 (GPLv3) je copyleftová licence vyvinutá Free Software Foundation. Je to bezplatná licence, která poskytuje uživatelům právo používat, sdílet a modifikovat software pro jakékoli účely, včetně komerčních. GPL zajišťuje, že uživatelé odvozených programů mají stejná práva jako u původního software. Pokud software pod touto licencí distribuujete, musíte zveřejnit zdrojový kód a zachovat licenci GPL.

GNU LGPLv2.1

GNU Lesser General Public License verze 2.1 (LGPLv2.1) je varianta GNU GPL určená speciálně pro knihovny a rámce. Tato licence umožňuje software používat, sdílet a modifikovat za stejných podmínek jako GNU GPL, ale s menšími omezeními, co se týče integrace do proprietárního software. Jestliže modifikace distribuujete, musí být zveřejněny ve formě zdrojového kódu.

BSD

BSD licence je sada permisivních volných softwarových licencí. Na rozdíl od GNU GPL, BSD licence nejsou copyleftové. To znamená, že odvozená díla mohou být distribuována jako svobodný nebo proprietární software. Tyto licence obecně vyžadují pouze zachování původního uznání autorství a zřeknutí se odpovědnosti.

ISC

ISC licence je permisivní svobodná softwarová licence vytvořená společností Internet Systems Consortium. Jedná se o zjednodušenou formu BSD licence, která vynechává některé formulace považované za zbytečné. ISC licence umožňuje používat, kopírovat, modifikovat a/nebo distribuovat software s podmínkou zachování původního uznání autorství.

Kapitola 4

Teoretická analýza

Tato kapitola je zaměřena na teoretické základy a koncepty, které jsou klíčové pro porozumění implementace audio pluginu.

4.1 Digitální filtrace

Infinite Impulse Response (IIR) filtry, nebo filtry s nekonečnou impulsní odezvou, jsou klíčovým konceptem v oblasti digitálního zpracování signálů. IIR filtry jsou charakterizovány tím, že jejich impulsní odezva teoriticky pokračuje do nekonečna. Tato vlastnost jim poskytuje možnost efektivně a flexibilně modelovat různé frekvenční charakteristiky.

IIR filtry mohou být popisovány pomocí obecné diferenční rovnice následující formy:

$$y[n] = b_0x[n]+b_1x[n-1]+\ldots+b_Nx[n-N]-a_1y[n-1]-\ldots-a_My[n-M] \quad (4.1)$$

IIR filtry jsou široce používány v různých oblastech digitálního zpracování signálů, včetně audio zpracování, kvůli jejich efektivitě a flexibilitě. Nicméně, stojí za zmínku, že IIR filtry mohou být náchylné k nestabilitě, a je tedy nutné přistupovat k jejich návrhu s určitou opatrností.

Při implementaci IIR filtrů je jedním z klíčových aspektů návrh a výpočet správných koeficientů, které určují chování filtru. V případě oktávového filtru šestého řádu, který je klíčový pro tento projekt, je filtr rozdělen do tří sekcí, kde každá sekce je definována sedmi koeficienty: třemi koeficienty čitatele, třemi koeficienty jmenovatele a jedním zesilovacím koeficientem. Celkově pro implementaci oktávového filtru šestého řádu potřebujeme 21 koeficientů.

Návrh oktávových filtrů

V souladu se specifikacemi normy ČSN EN 61260-1 [25] pro elektroakustiku a normy pro oktávové a třetinové oktávové filtry byla provedena implementace sedmi oktávových filtrů šestého řádu. Tyto filtry jsou navrženy tak, aby odpovídaly standardním oktávovým frekvencím: 125 Hz, 250 Hz, 500 Hz, 1000 Hz, 2000 Hz, 4000 Hz a 8000Hz.

Tyto frekvence nebyly vybrány náhodně. Při návrhu audiosignálových systémů je třeba vzít v úvahu frekvenční charakteristiky lidského sluchu,

který je citlivý na frekvence v rozmezí od přibližně 20 Hz do 20 kHz. Vybrané frekvence pro oktávové filtry tak pokrývají široké spektrum lidského sluchu a jsou také široce používány v audio inženýrství a akustickém návrhu.

Kromě toho, tato volba frekvencí je v souladu s normou ČSN EN IEC 60268-16 [26], která stanovuje parametry pro STIPA. Tato norma vyžaduje, aby byly provedeny měření na těchto konkrétních frekvencích.

Export koeficientů

Pro návrh a výpočet potřebných koeficientů pro oktávové filtry jsem použil prototypovací prostředí MATLAB. Toto prostředí poskytuje robustní a přesné nástroje pro návrh a analýzu digitálních filtrů a umožňuje snadný export potřebných koeficientů.

Příklad skriptu v MATLABu pro návrh oktávového filtru šestého řádu se středem na 1000 Hz:

```
d = fdesign.octave(1,'Class 0','N,F0',6,1000,44100)
Hd = design(d)
fvtool(Hd,'Fs',44100,'FrequencyScale','log')
```

Tento kód využívá funkci *fdesign.octave()*, která umožňuje definovat specifikace filtru, včetně řádu filtru, středové frekvence a vzorkovací frekvence. Poté používá funkci *design()*, aby navrhla filtr podle těchto specifikací. Nakonec funkce *fvtool()* zobrazuje frekvenční charakteristiku navrženého filtru. Pro nalezení požadovaných koeficientů lze přejít do sekce Filter Coefficients v otevřeném okně Filter Vizualization Tool.

Je důležité poznamenat, že filtr, který je výsledkem tohoto kódu, je Butterworthův filtr. Butterworthovy filtry jsou známé svou maximálně plochou frekvenční charakteristikou v průchozím pásmu, což je výhodné pro mnoho aplikací, včetně analýzy audio signálů.

4.2 STIPA

Tato sekce se podrobněji zabývá metrikou STIPA , která je specifickou podobou plného Indexu přenosu řeči (STI).

Koncepce parametrů STI a STIPA

STI (Speech Transmission Index) je významnou metrikou, která umožňuje hodnotit a kvantifikovat srozumitelnost řeči v různých prostředích. Tento index poskytuje základ pro objektivní hodnocení, jak dobře může být přenesen řečový signál mezi mluvčím a posluchačem. Tato metrika získala mezinárodní uznání pro svou schopnost odhalit, jak různé faktory, jako jsou rušení a omezení přenosových kanálů, ovlivňují srozumitelnost řeči.

Parametr STI se v praxi využívá k hodnocení široké škály elektronických systémů a akustických prostředí. Příklady jeho aplikací zahrnují měření systémů veřejného rozhlasu, nouzových zvukových a komunikačních systémů,

komunikačních kanálů a systémů jako jsou interkomy a bezdrátová komunikace, stejně jako hodnocení potenciální srozumitelnosti řeči a komunikace v místnostech a sálech.

STIPA (Speech Transmission Index for Public Address systems) je zjednodušená forma parametra STI, která vychází z měření s nižším počtem modulačních indexů. Používá předdefinovaný testovací signál s dvěma modulacemi na oktávový pás generovanými současně. Tato metoda nabízí kratší dobu měření než plný STI a je primárně využívána v přímé metodě měření.

Existuje také varianta STIPA odvozená z impulsní odezvy, označovaná jako STIPA(IR), která se vypočítává pomocí predikce. Je důležité si uvědomit, že standardní STIPA signál je založen na spektru mužské řeči. Pro posluchače se sluchovým postižením však STIPA není spolehlivým prediktorem srozumitelnosti řeči, pokud nejsou provedeny konkrétní korekce. Přesto je možné provádět měření asistivních sluchových systémů a kanálů, i když mohou být vyžadovány specifické korekce.

Výpočet parametru STIPA

Metoda STIPA využívá dva unikátní modulační frekvence aplikované současně na každý z sedmi oktávových pásem (viz Tabulka 4.1). Tato metoda je validována pouze pro mužské řečové spektrum. Doba měření má doporučené rozmezí 15 až 25 vteřin.

Centrální frekvence oktávového filtru (Hz)	125	250	500	1000	2000	4000	8000
První modulační frekvence (Hz)	1,60	1,00	0,63	2,00	1,25	0,80	2,50
Druhá modulační frekvence (Hz)	8,00	5,00	3,15	10,0	6,25	4,00	12,5

Tabulka 4.1: Frekvenční modulace pro metodu STIPA

Při použití STIPA metody je zásadní kalkulace modulační přenosové funkce (MTF). Pro jednotlivá oktávová pásma je MTF odvozena korelací intenzity obálky s sinusovými a kosinusovými signály, jejichž trvání a modulační frekvence jsou dané předem. Z toho je možné vypočítat modulační hloubku přijatého signálu $m_o(k, f_m)$ pro každé oktávové pásmo k

$$m_o(k, f_m) = 2 \cdot \frac{\sqrt{[\sum I_k(t) \cdot \sin(2\pi f_m t)]^2 + [\sum I_k(t) \cdot \cos(2\pi f_m t)]^2}}{\sum I_k(t)}, \quad (4.2)$$

kde
f_m je modulační frekvence v Hz,
t je čas v sekundách,
$I_k(t)$ je intenzita obálky jako funkce času pro oktávové pásmo k.

Na základě modulačních indexů přijatého a vysílaného signálu lze pak vypočítat modulační přenosový poměr $m(k, fm)$

$$m(k, f_m) = \frac{m_o(k, f_m)}{m_i(k, f_m)}, \tag{4.3}$$

kde
$m_o(k, fm)$ je hloubka modulace přijímaného testovacího signálu pro oktávové pásmo k a modulační frekvenci f_m,
$m_i(k, fm)$ je hloubka modulace vysílaného testovacího signálu pro oktávové pásmo k a modulační frekvenci f_m.

Tato m-hodnota se pak používá pro výpočet signál-šumového poměru (SNR) pro každé oktávové pásmo a modulační frekvenci

$$SNR_{\text{eff}\,k,f_m} = 10 \times \log \frac{m'_{k,f_m}}{1 - m'_{k,f_m}}, \tag{4.4}$$

kde
m_{k,f_m} je korigovaná hodnota poměru modulačního přenosu pro oktávové pásmo k a modulační frekvenci f_m.

Po výpočtu SNR je možné stanovit přenosový index (TI) pro každé oktávové pásmo a modulační frekvenci

$$TI_{k,f_m} = \frac{SNR_{\text{eff}\,k,f_m} + 15}{30}, \tag{4.5}$$

kde
SNR_{k,f_m} je efektivní poměr signál-šum pro každé oktávové pásmo k a modulační frekvenci fm vyjádřený v dB.

Modulační přenosový index (MTI) pro každé oktávové pásmo je poté získán průměrováním přenosových indexů (TI) přes modulační frekvence

$$MTI_k = \frac{1}{n} \sum_{m=1}^{n} TI_{(k,f_m)}, \tag{4.6}$$

kde
TI_{k,f_m} je index přenosu pro každé oktávové pásmo k a modulační frekvenci f_m,
m je index modulační frekvence,
n je počet modulačních frekvencí na oktávové pásmo.

Konečně, STIPA je vypočten z MTI pro každé oktávové pásmo pomocí koeficientů α a β pro mužské řečové spektrum, které jsou uvedeny v následující tabulce:

Oktávové pásmo (Hz)		125	250	500	1000	2000	4000	8000
Males	α	0,085	0,127	0,230	0,233	0,309	0,224	0,173
	β	0,085	0,078	0,065	0,011	0,047	0,095	-

Tabulka 4.2: Váhové faktory oktávových pásem pro MTI

Výpočet STIPA je prováděn sumací přes všechna oktávová pásma

$$STIPA = \sum_{k=1}^{7} \alpha_k \times M_k - \sum_{k=1}^{6} \beta_k \times \sqrt{M_k \times M_{k+1}}, \tag{4.7}$$

kde
M je index modulačního přenosu pro oktávové pásmo k,
α_k je gender-specific váhový faktor pro oktávové pásmo k,
β_k je gender-specific redundanční faktor mezi oktávovým pásmem k a oktávovým pásmem $k + 1$.

Výsledné hodnoty STIPA vyšší než 1,0 by měly být nastaveny na 1,0. Tento postup vede ke kalkulaci STIPA hodnoty, která poskytuje kvantitativní ukazatel kvality přenosu řeči v daném prostředí.

Kapitola 5

Implementace a testování

Tato kapitola se zabývá konkrétní implementací audio pluginu. Postupně popisuji jednotlivé kroky vývoje, klíčové aspekty a základní funkčnosti vytvořeného pluginu. Součástí kapitoly jsou také sekce zaměřené na testování pluginu, které ověřuje jeho správnou funkčnost a spravnost vypočtů.

5.1 Návrh a struktura pluginu

Tato sekce popisuje výběr vývojového prostředí, strukturu pluginu a přípravu na jeho implementaci.

Volba vývojového prostředí

Volba vývojového prostředí byla provedena na základě detailní analýzy a zhodnocení informací získaných v předchozích kapitolách. Nakonec bylo rozhodnuto vytvořit prototyp pluginu v jazyce C++ s využitím standardního SDK a frameworku JUCE. Tato kombinace nabízí několik klíčových výhod:

- Programovací jazyk C++ je známý svou vysokou úrovní optimalizace a schopností řídit nízkoúrovňové operace, což je nezbytné pro efektivní zpracování signálu v reálném čase. C++ také nabízí plnou kontrolu nad kódem, což umožňuje flexibilitu a jistotu v implementaci komplexních algoritmů a datových struktur.
- Framework JUCE, který je navržen speciálně pro vývoj audio aplikací, poskytuje rozsáhlou API a předdefinované třídy pro práci s audio daty. JUCE zároveň zjednodušuje integraci s různými audio standardy, jako je například VST a VST3, a podporuje multiplatformní vývoj, což je další důležitý aspekt při výběru vývojového prostředí.

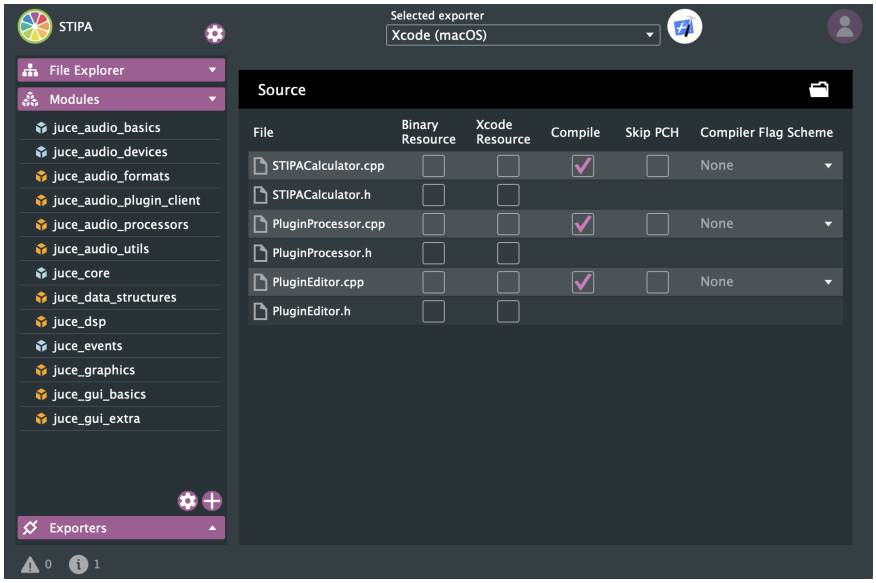
Při vývoji tohoto pluginu byl kladen důraz především na kompatibilitu s DAW Audacity. Plugin byl navržen a testován výhradně v tomto prostředí, protože to bylo předpokládáno v zadání bakalářské práce. I když tento plugin může být kompatibilní i s jinými DAW, které podporují formát VST3, nebyla tato kompatibilita hlavním cílem při vývoji. Audacity podporuje formát VST3 od listopadu 2022, což znamená, že tento plugin nebude fungovat na starších verzích Audacity.

Je důležité zdůraznit, že většina jiných programovacích jazyků a frameworků, které nepoužívají standardní SDK, nemůže zaručit plnou funkčnost pluginu v souladu se standardem VST. Některé z těchto prostředí dokonce nabízejí pouze možnost exportu interních projektů jako audio pluginů, což může výrazně snižovat kvalitu konečného produktu. Navíc, mnoho z těchto editorů je často používáno především pro tvorbu syntezátorů nebo prototypování hardwaru, což není cílem tohoto projektu.

Příprava na implementaci

Jak bylo již zmíněno v sekci 3.2, nezbytnou součástí frameworku JUCE je aplikace ProJucer. Tato aplikace umožňuje konfiguraci projektu ještě před jeho spuštěním, a to prostřednictvím výběru vhodné šablony z modulů frameworku. JUCE obsahuje velké množství různých modulů pro vytváření různých typů pluginů. Kombinací těchto modulů lze dosáhnout optimální základní struktury pro vývoj pluginu, která odpovídá cílům projektu. Pro naše účely byly vybrány následující moduly:

- `juce_audio_basics` - tento modul poskytuje základní třídy a struktury pro práci se zvukem;
- `juce_audio_devices` - modul slouží k interakci s audio zařízeními;
- `juce_audio_formats` - podporuje různé audio formáty;
- `juce_audio_plugin_client` - tento modul je základním modulem pro tvorbu audio pluginů;
- `juce_audio_processors` - obsahuje třídy pro zpracování audio dat;
- `juce_audio_utils` - nabízí různé nástroje pro práci se zvukem;
- `juce_core` - jde o základní modul, který poskytuje různé jádrové třídy a funkce;
- `juce_data_structures` - poskytuje struktury pro ukládání dat.;
- `juce_dsp` - obsahuje třídy pro digitální zpracování signálů;
- `juce_events` - poskytuje podporu pro události;
- `juce_graphics` - poskytuje nástroje pro tvorbu grafických prvků;
- `juce_gui_basics` a `juce_gui_extra` - tyto dva moduly obsahují třídy pro tvorbu uživatelského rozhraní.



Obrázek 5.1: Ukázka hlavního okna proketu v ProJucer.

Kromě speciálních modulů lze v hlavním okně projektu nastavit parametry samotného pluginu, jeho typ, počet zpracovávaných kanálů, stejně jako vybrat formáty pro budoucí export pluginu, kliknutím na ikonu ozubeného kola v levém horním rohu.

Jako součást nastavení pluginu byla vybrána funkce Analyzer, která se zaměřuje na analýzu a zobrazení spektrálních a časových charakteristik zvuku, což pomáhá uživatelům lépe pochopit, jakým způsobem zvukový signál ovlivňuje zpracování pluginem. Tento typ pluginu je často používán pro účely monitoringu, diagnostiky a ladění zvukového zpracování.

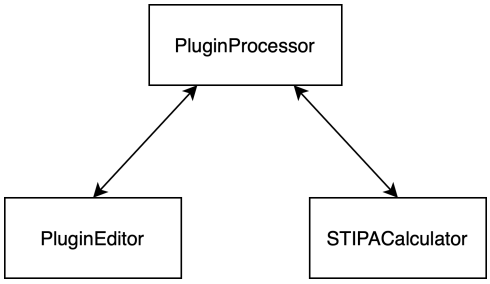
V nastavení projektu byl také specifikován počet zpracovávaných kanálů, který se přizpůsobuje podle potřeb konkrétního projektu. Pro tento případ byly nastaveny osm kanálů, což umožňuje přepínání mezi osmi kanály a zpracování každého zvlášť, aniž by byl signál ve formátu osmikanálového zvuku.

Dále lze v ProJuceru vybrat formáty pro budoucí export pluginu. Pro tento projekt byl vybrán formát VST3. Jako IDE byla v tomto projektu využita aplikace XCode.

Struktura pluginu

Jednou z největších výzev tohoto projektu je ne tak počet tříd, jako spíše implementace matematických výpočtů a hluboké porozumění specifické technologii VST. Jak je vidět na diagramu 5.4, celý projekt tedy sestává ze tří tříd: PluginEditor, PluginProcessor a STIPACalculator. Tuto strukturu lze považovat za základní příklad architektury MVC[1]:

[1]Model-View-Controller je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní.



Obrázek 5.2: UML diagram vztahů mezi komponentami pluginu.

- **PluginEditor.** V kontextu architektury MVC slouží třída PluginEditor jako View. Tato třída interaguje s uživatelem prostřednictvím GUI. Nejdůležitější metoda v této třídě je *paint()*, která se stará o vykreslení uživatelského rozhraní, a *resized()*, která upravuje rozměry a umístění komponent v GUI podle potřeby;
- **STIPACalculator.** Třída STIPACalculator představuje Model. Tato třída se stará o vytváření oktávových filtrů a veškeré potřebné matematické výpočty. Klíčové metody v této třídě jsou například *calculateSTIPA()*, která provádí výpočet parametrů STIPA, a metody pro vytváření a aplikaci oktávových filtrů *createOctaveBandFilters()*, *createFilterFromCoefficients()* a *processSingleFilter()*;
- **PluginProcessor.** Třída PluginProcessor funguje jako Controller. Tato třída je odpovědná za fungování pluginu, jeho interakci s DAW a také propojuje třídy PluginEditor a STIPACalculator pro zobrazení získaných hodnot v okně pluginu a získávání informací o zpracovávaném kanálu. Metoda *processBlock()* je zde zásadní, protože řídí zpracování vstupních a výstupních audio dat.

5.2 Zpracování signálu

Tato kapitola se zaměřuje na detailní prozkoumání procesu zpracování signálu v kontextu VST pluginu.

Základy zpracování signálu ve VST pluginech

Většina audio pluginů, které pracují s technologií VST, se zaměřuje na zpracování zvuku v reálném čase, což je jeden z klíčových aspektů této technologie. Tyto pluginy zpracovávají vstupní signál ve formě malých datových bloků, neboli bufferů. Takové zpracování umožňuje pluginům efektivně manipulovat s daty, provádět potřebné výpočty a aplikovat jejich funkcionalitu na výstupní signál.

V tomto kontextu je důležité zmínit pojem *buffer tails*. Tyto *tails* představují zbytky zvukového signálu, které zůstávají v bufferu po zpracování. V rámci našeho pluginu je tato problematika řešena tak, aby tyto "tails"byly pečlivě

zpracovány a eliminovány, a tím bylo zabráněno jejich hromadění, které by mohlo narušit celkovou kvalitu zvuku.

Pro výpočet parametru STIPA, konkrétně pro výpočet MTF, je třeba mít k dispozici kompletní blok dat. Vzhledem k omezením standardní technologie VST to však není přímo možné. Uživatel tedy musí nejprve spustit plugin a nastavit preferovaný počet sekund pro výpočet (viz 4.2), a poté může plugin začít zaznamenávat vzorky do své paměti, propouštět signál přes všechny oktávové filtry a provádět potřebné výpočty.

Řešení specifických problémů při zpracování signálu

Při zpracování signálu v kontextu VST pluginu jsem narazil na některé specifické problémy. Jeden z nich se týkal testování pluginu DAW. Některé DAW otestují plugin hned po jeho spuštění uživatelem tím, že pošlou testovací datové bloky složené z nulových vzorků. Tyto testovací datové bloky jsou posílány za účelem ověření, zda plugin neprovádí žádné neobvyklé operace s budoucím signálem, které by mohly vést k pádu aplikace nebo chybnému zpracování vstupního signálu.

Tyto testovací datové bloky však mohou ovlivnit konečné výpočty. Během zápisu vzorků do bufferu mohou nulové hodnoty z testovacích bloků zůstat na začátku bufferu, což by mohlo vést k velkému množství nulových hodnot v zpracovávaném signálu. Taková situace by mohla narušit výpočty parametrů STIPA a ovlivnit tak kvalitu a přesnost výstupů pluginu.

Z tohoto důvodu jsem do své implementace zahrnul ochranný mechanismus. V třídě PluginProcessor jsem vytvořil specifické části kódu, které efektivně omezují použití testovacích bufferů s nulovými hodnotami a ignorují signál, který se skládá výhradně z nulových vzorků:

```
namespace {
bool containsOnlyZeros(juce::AudioBuffer<float>& buffer, int
    currentSelectedChannel) {const auto readPointer = buffer
    .getReadPointer(currentSelectedChannel); return std::
    all_of(readPointer, readPointer + buffer.getNumSamples(),
    [](float sample) { return sample == 0.f; });}
}

void PluginProcessor::processBlock(juce::AudioBuffer<float>&
    buffer, juce::MidiBuffer& midiMessages)
{
    //...other code

    if (containsOnlyZeros(buffer, currentSelectedChannel)) {
        return;
    }

    //...other code
}
```

Toto řešení zajišťuje, že testovací bloky poslané DAW nebudou mít negativní vliv na výsledné výpočty parametrů STIPA.

5.3 Oktávové filtry

Tato podkapitola se zabývá implementací a testováním oktávových filtrů v rámci VST pluginu.

Implementace oktávových filtrů

Implementace oktávových filtrů probíhá v několika fázích. Při otevření pluginu, je spuštěna funkce *prepareToPlay()*, která se nachází ve třídě *PluginProcessor*. Tato funkce předává informace o vzorkovací frekvenci z DAW do třídy *STIPACalculator* pro další zpracování signálu a výpočty. Současně jsou vytvořeny všechny oktávové filtry pomocí funkcí *createOctaveBandFilters()* a *createFilterFromCoefficients()*, které se nachází také v třídě *STIPACalculator*.

Pokud se podíváme na samotný kód, funkce *createOctaveBandFilters()* definuje koeficienty filtrů a předává je funkci *createFilterFromCoefficients()*. Je třeba poznamenat, že tyto filtry jsou vytvořeny pro specifické frekvence, které jsou předdefinované a odpovídají oktávovým pásmům. Oktávová pásma jsou definována na frekvencích 125 Hz, 250 Hz, 500 Hz, 1000 Hz, 2000 Hz, 4000 Hz a 8000 Hz.

Funkce *createFilterFromCoefficients()* pak vytvoří IIR filtr (viz 4.1) z daných koeficientů. Pro každý soubor koeficientů filtrační sekce v předaném dvourozměrném poli iteruje a vytvoří filtrační sekci. Tyto sekce filtrů jsou uloženy do vektoru *filterSectionsVector*.

Výsledné filtry jsou poté uloženy do vektoru octaveBandFilters a jsou připraveny k dalšímu zpracování. Je důležité poznamenat, že tento proces vytváření filtrů je proveden pouze jednou při inicializaci pluginu, čímž se zvyšuje efektivita a snižuje se procesorová náročnost během zpracování audio signálu. Každý filtr je navržen tak, aby měl co nejmenší dopad na audio signál mimo své cílové frekvenční pásmo. To zajišťuje, že každý filtr bude správně analyzovat pouze své specifické frekvenční pásmo a nebude ovlivňovat ani být ovlivněn jinými frekvencemi.

Zpracování signálu přes filtry

Po inicializaci filtrů následuje zpracování vstupního signálu prostřednictvím všech sedmi filtrů. Tento proces je řízen funkcí processBlock, která je součástí třídy *PluginProcessor* a je zodpovědná za provoz celého pluginu. Hned na začátku této funkce je volána funkce *processSingleFilter()*, která je součástí třídy *STIPACalculator*. Tímto začíná zpracování signálu v reálném čase. Signál je současně zpracováván všemi filtry a zpracované vzorky jsou ukládány do paměti pluginu.

Funkce *processSingleFilter()* je klíčová pro řízení tohoto procesu. Tato funkce prochází jednotlivé vzorky zadaného kanálu a zpracovává je prostřednictvím jednotlivých oktávových filtrů. Každý vzorek je postupně procházen a zpracováván prostřednictvím jednotlivých sekcí každého oktávového filtru. Zpracovaný vzorek je pak uložen do výstupního vektoru filtrů. Tento proces

je opakován pro všechny vzorky na daném kanálu a pro všechny oktávové filtry. Konečný výstup tedy představuje skupinu zpracovaných vzorků pro každý oktávový filtr, které jsou připraveny k dalšímu zpracování a výpočtu MTF.

Jak lze pochopit z kódu funkce *processBlock()*, výpočet parametru STIPA začíná až po získání dostatečného počtu vzorků, v závislosti na počtu sekund, který si uživatel vybere v okně pluginu. Toto je klíčový krok, který umožňuje přesné a spolehlivé výpočty parametru STIPA, neboť vyžaduje dostatečný počet vzorků pro správné výpočty. Tímto způsobem je zajištěna preciznost a spolehlivost výsledků, což je klíčové pro kvalitní analýzu akustického signálu a následnou interpretaci výsledků.

Testování oktávových filtrů

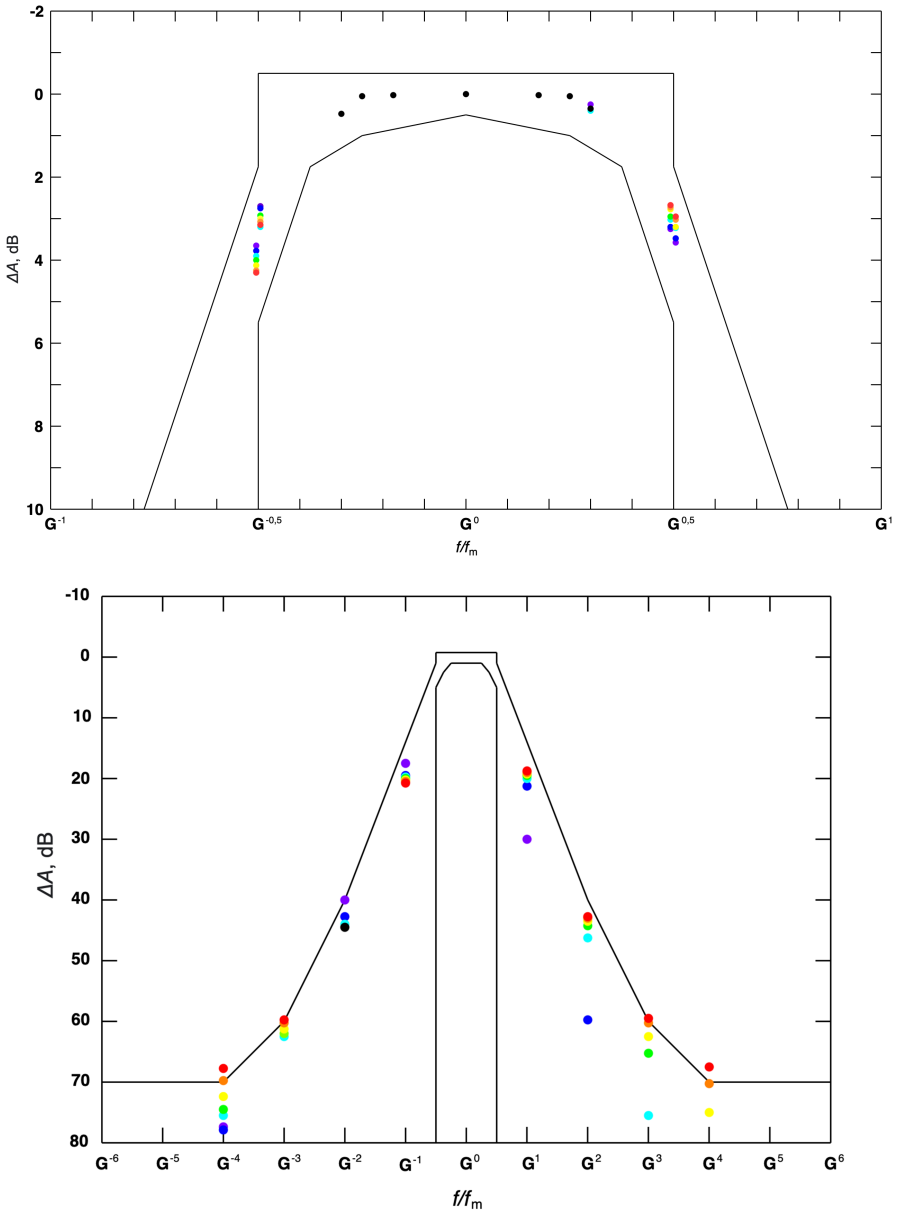
Testování oktávových filtrů je klíčovou součástí analýzy signálů a jejich kvality. Následující tabulka shrnuje výsledky našeho testování. Tato data byla získána porovnáním hodnot RMS pro oktávový filtr se středem na různých frekvencích a porovnáním rozdílů s tolerancí podle normy ČSN EN 61260-1 [25].

Frekvence signálu (Hz)	Rozdíl (dB) mezi hodnotami RMS pro oktávový filtr se středem na							Tolerance (dB)
	125 Hz	250 Hz	500 Hz	1000 Hz	2000 Hz	4000 Hz	8000 Hz	
$\Omega_I = G^{-4}$	66,99	69,85	72,51	74,88	76,88	78,03	77,21	70; $+\infty$
$\Omega_I = G^{-3}$	58,89	60,45	61,48	62,06	62,26	61,89	59,86	60; $+\infty$
$\Omega_I = G^{-2}$	43,41	43,52	43,53	43,48	43,32	42,78	40,78	40,5; $+\infty$
$\Omega_I = G^{-1}$	20,03	19,93	19,82	19,71	19,53	19,13	17,79	16,6; $+\infty$
$\Omega_{1-\varepsilon} = G^{-1/2} - \varepsilon$	4,35	4,24	4,13	4,03	3,92	3,80	3,63	1,2; $+\infty$
$\Omega_{1+\varepsilon} = G^{-1/2} + \varepsilon$	3,18	3,09	2,99	2,90	3,22	2,73	2,67	-0,4; 5,3
$\Omega_I = G^{-3/8}$	0,42	0,38	0,36	0,35	0,34	0,34	0,40	-0,4; 1,4
$\Omega_I = G^{-1/4}$	0,08	0,08	0,07	0,06	0,06	0,06	0,09	-0,4; 0,7
$\Omega_I = G^{-1/8}$	0,01	0,01	0,01	0,01	0,01	0,01	0,01	-0,4; 0,5
$\Omega_I, \Omega_h = G^0 = 1$	0,00	0,00	0,00	0,00	0,00	0,00	0,00	-0,4; +0,4
$\Omega_h = G^{1/8}$	0,01	0,00	0,01	0,01	0,01	0,00	0,00	-0,4; 0,5
$\Omega_h = G^{1/4}$	0,05	0,05	0,06	0,06	0,07	0,06	0,03	-0,4; 0,7
$\Omega_h = G^{3/8}$	0,29	0,30	0,32	0,34	0,35	0,34	0,24	-0,4; 1,4
$\Omega_{2-\varepsilon} = G^{1/2} - \varepsilon$	2,67	2,76	2,85	2,94	3,03	3,13	3,28	-0,4; 5,3
$\Omega_{2+\varepsilon} = G^{1/2} + \varepsilon$	2,94	3,03	3,13	3,22	3,32	3,44	3,67	1,2; $+\infty$
$\Omega_h = G^1$	19,42	19,53	19,66	19,84	20,26	21,75	30,83	16,6; $+\infty$
$\Omega_h = G^2$	43,09	43,32	43,58	44,18	46,38	59,12	-	40,5; $+\infty$
$\Omega_h = G^3$	59,27	60,82	62,24	64,92	76,46	-	-	60; $+\infty$
$\Omega_h = G^4$	67,85	70,92	74,55	82,59	-	-	-	70; $+\infty$

Tabulka 5.1: Výsledky testu oktávových filtrů.

Po analýze tabulky lze odvést, že ne všechny hodnoty (zejména na 125 Hz) odpovídají dané toleranci. Nicméně, tato odchylka je přijatelná a neměla by mít významný dopad na kvalitu zpracování signálu.

Vzhledem k těmto výsledkům můžeme konstatovat, že testování oktávových filtrů proběhlo úspěšně a filtry fungují dle očekávání pro většinu frekvencí. Toto potvrzuje, že implementované metody zpracování signálu jsou efektivní a docela přesné.



Obrázek 5.3: Minimální a maximální meze poměrného útlumu jako funkce f/f_m pro oktávové filtry 1. třídy.

Na obrázku 5.3 jsou ilustrovány hodnoty prezentované v tabulce 5.1. Každý oktávový filtr je zastoupen specifickou barvou: filtr se středem na 125 Hz je reprezentován červenou barvou, filtr 250 Hz oranžovou, filtr 500 Hz žlutou, filtr 1000 Hz zelenou, filtr 2000 Hz světle modrou, filtr 4000 Hz tmavě modrou a filtr 8000 Hz je zastoupen fialovou barvou. V případě, že se v jednom bodě vyskytuje více hodnot, jsou tato místa na obrázku reprezentována černými body.

5.4 Určení MTF

Tato podkapitola podrobně popisuje implementaci výpočtu modulačních přenosových funkcí a následující proces testování tohoto výpočtu.

Implementace výpočtu MTF

Výpočet MTF (viz 4.2) je realizován v třídě *STIPACalculator* metodou *calculateMTFs()*. Tato metoda je volána s polem vektorů floatů `filtersOutput`, které reprezentuje výstupy jednotlivých oktávových filtrů a také vzorkovací frekvencí `sampleRate`.

Nejprve jsou definovány modulační frekvence pro jednotlivá oktávová pásma ve formátu dvojrozměrného pole. Každé oktávové pásmo má dvě specifické modulační frekvence, které jsou přiřazeny na základě akustických vlastností tohoto pásma. Dále je vytvořena proměnná `MTFs`, která bude obsahovat výsledky výpočtu MTF pro každou modulační frekvenci a oktávové pásmo.

Následuje cyklus, který prochází všechna oktávová pásma. Uvnitř tohoto cyklu je získán filtrovaný signál pro aktuální oktávové pásmo. V rámci tohoto cyklu probíhá další cyklus, který prochází obě modulační frekvence pro dané oktávové pásmo. V tomto cyklu je vypočítána doba zpracování a počet vzorků pro zpracování na základě modulační frekvence a délky signálu. Doba zpracování je určena tak, aby odpovídala celému počtu period dané modulační frekvence, což umožňuje synchronizaci zpracování signálu s modulační frekvencí.

Poté jsou inicializovány proměnné `sum_I_k`, `sum_sin` a `sum_cos`, které budou použity pro výpočet MTF. Další cyklus pak prochází všechny vzorky k zpracování. V rámci tohoto cyklu je vypočítána intenzita signálu v daném čase `I_k_t`, která je poté využita pro výpočet `sum_sin` a `sum_cos`. Také je současně počítán součet všech intenzit `sum_I_k`.

Po provedení těchto výpočtů je MTF pro aktuální modulační frekvenci a oktávové pásmo vypočítána jako dvojnásobek druhé odmocniny součtu čtverců `sum_sin` a `sum_cos`, děleno `sum_I_k`. Tato hodnota je poté normalizována dělením konstantou 0.55, protože každé oktávové pásmo je současně modulováno dvěma modulačními frekvencemi s frekvenčním poměrem 5. Pro sinusové sčítání těchto dvou komponent s fázovým rozdílem 180° mezi komponentami by měla být hloubka modulace pro každou modulační frekvenci 0.55. Po normalizaci je hodnota omezena na maximální hodnotu 1.

Výsledné hodnoty MTF pro všechny modulační frekvence a oktávová pásma jsou poté vráceny jako výstup metody *calculateMTFs()* a následně se využívají v uživatelském rozhraní pluginu. Tyto hodnoty MTF jsou zobrazeny a aktualizovány v hlavním okně pluginu, poskytují tak uživatelům přehled o aktuálním stavu zpracování signálu.

Tímto způsobem je realizován výpočet MTF, který je klíčový pro následný výpočet parametru STIPA.

Testování výpočtu MTF

Jak bylo popsáno v sekci 4.2, MTF jsou vypočítány pomocí přímé metody. Výsledky tohoto výpočtu pro jednotlivá oktávová pásma a jejich modulační frekvence jsou prezentovány v tabulce 5.4. Všechny hodnoty MTF v tabulce byly vypočítány pomocí metody *calculateMTFs()* v třídě *STIPACalculator*. Pro výpočet byly použity koeficienty pro 6. řád oktávového filtru, které byly exportovány z MATLABu. Tyto koeficienty jsou konkrétně pro vzorkovací frekvenci 44100 Hz. Příklad exportovaných koeficientů pro oktávový filtr se středem na 1000 Hz:

```
const std::array<std::array<double, 7>, 3>
    filterCoefficients1000 = {{
    {1.0, 0.0, -1.0, 1.0,
        -1.9011048443041209665693713759537786245 35,
        0.9371729063792801905918850025045685470 1,
        0.0489608778079507270253323270026157842 95},
    {1.0, 0.0, -1.0, 1.0,
        -1.9543708585925725351728488021763041615 49,
        0.9652044686244238835826081412960775196 55,
        0.0489608778079507270253323270026157842 95},
    {1.0, 0.0, -1.0, 1.0,
        -1.8850222170086214923401257692603394389 15,
        0.9043341500393069498642262260545976459 98,
        0.0478329249803464903734173674365592887 63}
    }};
```

Je důležité poznamenat, že typ double v C++ má přesnost až na 15 desetinných míst. To znamená, že přesnost koeficientů je omezena na 15 desetinných míst, což by mohlo mírně ovlivnit výsledky výpočtu MTF, pokud by původní hodnoty měly vyšší přesnost.

Pro ověření algoritmu pro výpočet MTF byly generovány testovací signály na základě sinusových nosičů s hloubkou intenzitní modulace od 0,0 do 1,0 v krocích 0,1 pro vzorkovací frekvenci 44100 Hz. Sinusové nosiče byly použity ke snížení nejistoty. Vzhledem k tomu, že STIPA má nelineární vztah s m-hodnotami, nebude STIPA v krocích 0,1.

Podle normy ČSN EN IEC 60268-16 [26] je přijatelná odchylka hodnot MTF od očekávaných hodnot stanovena na 0,05. Z tabulky 5.4 je vidět, že ve většině případů jsou odchylky od očekávaných hodnot MTF v rámci této přijatelné odchylky. Nicméně v některých případech jsou odchylky o něco větší. Toto může být způsobeno zašuměním signálu, které může mít vliv na výsledky výpočtu MTF.

Je také důležité si uvědomit, že odchylky jsou obecně větší pro vyšší hodnoty MTF. Absolutní hodnoty odchylek jsou pro vyšší hodnoty MTF větší, i když relativní odchylky zůstávají v přijatelných mezích. Tato skutečnost odráží složitost a náročnost přesného výpočtu MTF, zejména při vyšších hodnotách.

Přestože jsou zaznamenány některé odchylky, je zřejmé, že implementace metody výpočtu MTF je docela přesná a slouží jako klíčový nástroj pro celkový proces výpočtu parametru STIPA.

Test vážících koeficientů

Metoda STI dokáže rozlišovat mezi mužským a ženským řečovým signálem. Avšak v praxi, a pro zjednodušení predikce a měřicího procesu, by měla být použita pouze mužská řeč. Váhové faktory (α) a redundanční faktory (β) pro mužskou řeč jsou zobrazeny v tabulce 4.2 jako funkce oktávových pásem.

Pro ověření, zda byly použity správné váhové a redundanční faktory, byly generovány modulované sinusové nosiče v oktávových pásmových párech:

Dvojice oktávových filtrů se středem na	Získaná hodnota STIPA	Očekávaná hodnota STIPA
125 Hz/250 Hz	0,13	0,13
250 Hz/500 Hz	0,28	0,28
500 Hz/1000 Hz	0,40	0,40
1000 Hz/2000 Hz	0,53	0,53
2000 Hz/4000 Hz	0,49	0,49
4000 Hz/8000 Hz	0,30	0,30

Tabulka 5.2: Výsledky testu vážících koeficientů.

Testování hmotnostního faktoru MTF ukázalo dobré shody mezi získanými a očekávanými hodnotami STIPA pro všechny testované dvojice oktávových filtrů. Z těchto výsledků lze usoudit, že test proběhl úspěšně.

Test fázového zkreslení oktávových filtrů

Testování fázového zkreslení oktávových filtrů je nezbytné pro zajištění přesnosti měření. Úzké filtrační banky mohou způsobit předstíhání a zpoždění určitých frekvencí, což by mohlo vést k výrazně nižším hodnotám MTF. Filtrační banka by neměla vykazovat zkreslení STIPA o hodnotě 0,01 nebo vyšší v rozsahu STIPA mezi 0,1 a 0,9. Za tímto účelem byly generovány testovací signály na základě dvou sinových nosičů pro každou oktávovou pásmo, umístěné na okraji střední 1/2 oktávy. Odchylky by neměli představovat systematickou chybu (to jest, že všechny hodnoty jsou nižší, nebo všechny jsou vyšší).

Podle získaných dat v tabulce 5.5 je vidět, že odchylky jsou přijatelné pouze do určité míry. Nicméně, problémy začínají se výrazněji projevovat v vyšších frekvenčních pásmech, což naznačuje, že použité oktávové filtry možná nejsou ideální pro tento účel.

Test strmosti oktávových filtrů

Filtrační banka pro analýzu oktáv by měla v podstatě splňovat charakteristiky oktávových filtrů 1. třídy. Je vyžadováno, aby sklon filtrů byl alespoň 41 dB nižší v přilehlém oktávovém pásmu. Avšak, kvůli omezením filtrů navržených podle normy ČSNEN61260-1 [25], nelze tento požadavek plně splnit a sklon filtrů bude v souladu s touto normou pouze 20 dB nižší. Z tohoto důvodu bylo generováno několik testovacích signálů pomocí modulovaného nosiče v pozorovaných pásmech (od 125 Hz do 8000 Hz), zatímco v sousedním pásmu

(dolní a horní) byl vytvářen nemodulovaný tón s celkovou úrovní vyšší o 20 dB. Výsledky testu by měly dosahovat hodnoty MTF alespoň 0,5 nebo vyšší v pozorovaném pásmu s přijatelnou odchylkou 0,05. Podrobný seznam modulačních frekvencí pro jednotlivá oktávová pásma lze nalézt v tabulce 4.1.

Generovaný signál	Hodnota MTF na první modulační frekvenci	Hodnota MTF na druhé modulační frekvenci
Lowslope 125 Hz	0,50	0,50
Highslope 125 Hz	0,47	0,47
Lowslope 250 Hz	0,50	0,50
Highslope 250 Hz	0,47	0,47
Lowslope 500 Hz	0,49	0,49
Highslope 500 Hz	0,48	0,48
Lowslope 1000 Hz	0,48	0,48
Highslope 1000 Hz	0,49	0,49
Lowslope 2000 Hz	0,47	0,47
Highslope 2000 Hz	0,51	0,51
Lowslope 4000 Hz	0,45	0,45
Highslope 4000 Hz	0,60	0,60
Lowslope 8000 Hz	0,38	0,37
Highslope 8000 Hz	0,94	0,94

Tabulka 5.3: Výsledky testu strmosti oktávových filtrů.

Na základě hodnot uvedených v testovacích tabulkách je zřejmé, že stanovené požadavky nebyly plně splněny. Tato skutečnost však neznamená, že je celková kvalita pluginu nedostatečná a využité oktávové filtry nejsou pro daný účel optimální. Tyto testy byly navrženy pro vyhodnocení extrémních scénářů a jejich výsledky naznačují, že striktní dodržení parametrů oktávových filtrů dle normy ČSN EN 61260-1 [25] nemusí nutně splnit přesnost požadovanou normou ČSN IEC 60268-16 [26]. Změny v požadavcích na oktávové filtry a detailní popis provedených testů byly zavedeny až v nejnovější verzi ČSN IEC 60268-16 [26], zatímco dřívější verze vyžadovaly jen soulad s normou ČSN EN 61260-1 [25].

5.5 Výpočet STIPA

Po výpočtu hodnot MTF pro každé oktávové pásmo a modulační frekvenci, pokračuje proces výpočtem hodnoty STIPA. Teoretický popis algoritmu a vzorce jsou detailně popsány v sekci 4.2.

Nejprve je zavolána funkce *calculateSNRs()*. Tato funkce na základě již vypočítaných hodnot MTF vytvoří hodnoty SNR pro každé oktávové pásmo a modulační frekvenci. Výsledky tohoto výpočtu jsou však omezeny tak, aby hodnota SNR nepřekročila rozsah od -15 dB do +15 dB. Následně je zavolána funkce *calculateTIs()*, která pro každé oktávové pásmo a modulační frekvenci vypočítá hodnoty TI na základě předchozích výpočtů SNR. Další krok zahrnuje volání funkce *calculateMTIs()*, která pro každé oktávové pásmo vypočítá průměrnou hodnotu TI na základě předchozích výpočtů TI. Nakonec je zavolána funkce *calculateSTIPA()*, která vypočítá hodnotu STIPA jakožto vážený průměr hodnot MTI pro jednotlivé oktávové pásma. Tato hodnota STIPA je následně omezena na maximální hodnotu 1.

Výsledná hodnota STIPA je pak vrácena a použita v uživatelském rozhraní. Zde je tato hodnota zobrazována a průběžně aktualizována, aby poskytovala nejnovější informace o hodnotě STIPA během používání pluginu.

5.6 GUI

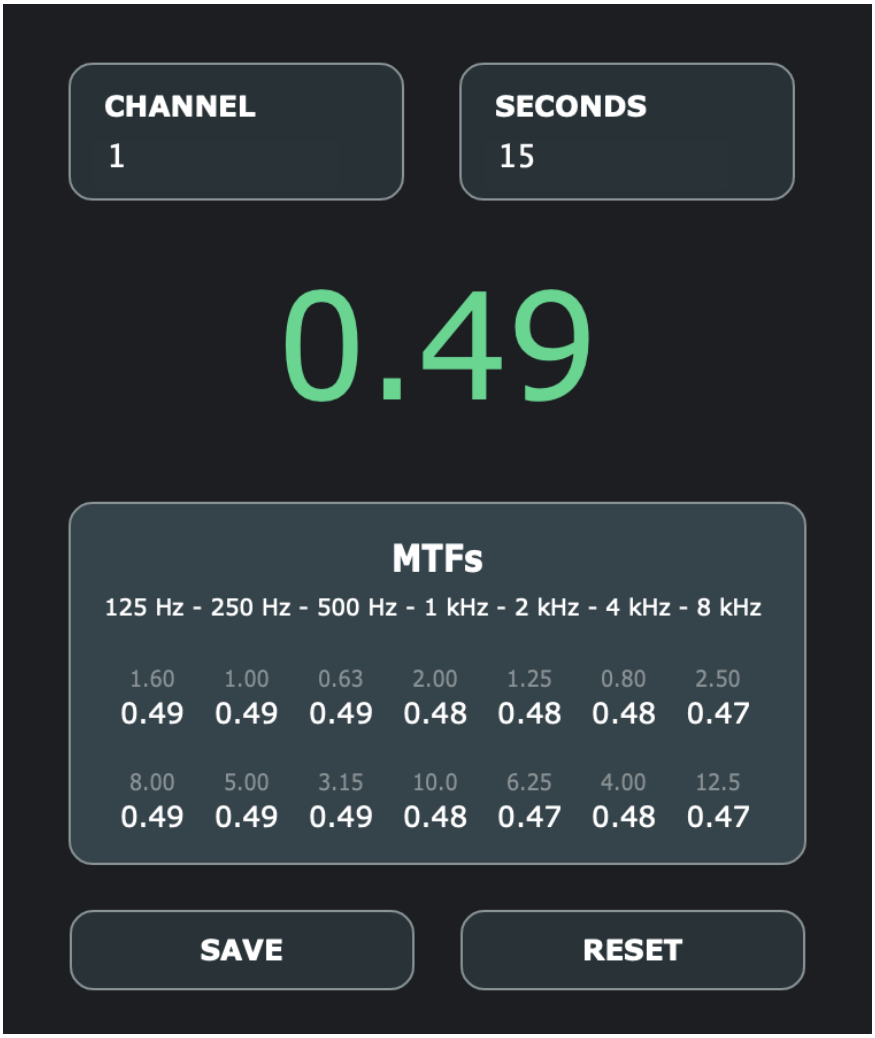
Tato sekce se věnuje popisu implementace uživatelského rozhraní pluginu. Tato implementace využívá kombinaci originálních prvků a standardních komponent poskytovaných knihovnou JUCE.. Základem našeho GUI je třída *PluginEditor*, která slouží jako kontejner pro všechny prvky GUI a řídí jejich správnou organizaci a interakci.

Tlačítka

Jedním z klíčových prvků uživatelského rozhraní pluginu jsou tlačítka. Konkrétně se jedná o tlačítka SAVE a RESET, která jsou založena na třídě *TextButton* ze standardní knihovny JUCE.

Při interakci s těmito tlačítky jsou vyvolány specifické funkce. Při kliknutí na tlačítko SAVE se aktivuje funkci *saveStipaToFile()*, která otevírá standardní dialogové okno pro ukládání souborů. Uživateli tak umožňuje uložit hodnoty STIPA a MTF do textového souboru. Naopak tlačítko RESET spouští metodu *resetFilters()* v rámci třídy *STIPACalculator*. Tato funkce resetuje všechny filtry a výstupy filtrů, čímž připravuje prostor pro další výpočty.

V rámci implementace těchto tlačítek bylo předefinováno standardní chování třídy *TextButton*. Pro dosažení požadované vizuální prezentace tlačítek byla upravena metoda *drawButtonText()*, která je zodpovědná za nastavení fontu, barvy a pozice textu tlačítka. Tato funkce také dynamicky mění barvu textu tlačítka v závislosti na jeho stavu (aktivní, neaktivní), což vede k lepší uživatelské interakci a pochopení funkcí jednotlivých tlačítek.



Obrázek 5.4: Hlavní okno pluginu.

ComboBox

Další důležitou součástí uživatelského rozhraní pluginu je prvek ComboBox, který slouží k nabídce mnoha možností pro uživatele. Tento prvek je implementován s využitím třídy *ComboBox* ze standardní knihovny JUCE.

V případě tohoto pluginu slouží ComboBox k výběru specifického kanálu a počtu sekund určených pro zpracování vstupního signálu. K řízení interakce s ComboBoxem slouží metoda *comboBoxChanged()*, která je automaticky volána pokaždé, když dojde ke změně výběru uživatelem.

Výběr kanálu a doba zpracování jsou pak využity v metodě *processBlock()* třídy *PluginProcessor*. Tato metoda představuje jádro našeho pluginu, kde probíhá zpracování audio dat. Zde se využívá vybraný kanál pro zpracování příslušného kanálu vstupního bufferu a doba zpracování určuje minimální počet vzorků, které jsou nutné k zpracování. V rámci této metody dochází také k výpočtu hodnot MTF, SNR, TI, MTI a STIPA.

V rámci implementace ComboBoxu byly provedeny několik úprav standardní třídy *ComboBox*. Předefinoval jsem funkci *drawComboBox()*, která upravuje vizuální prezentaci ComboBoxu. Tato funkce mění rozměry a barvu ComboBoxu, čímž přizpůsobuje jeho vzhled potřebám našeho pluginu.

Labely

Labely jsou textové prvky sloužící k zobrazení informací pro uživatele. V našem případě jsou použity k zobrazení hodnoty STIPA a hodnot MTF. Aktualizace těchto hodnot probíhá pravidelně pomocí metody *timerCallback()*.

Pro pravidelnou aktualizaci hodnot zobrazených v našem pluginu využíváme mechanismus časovače poskytovaný knihovnou JUCE. V rámci třídy *PluginProcessorEditor* je implementována třída *Timer*, která umožňuje pravidelné spouštění určitých funkcí. Metoda *timerCallback()* je volána po určitých intervalech, které jsou definovány v konstruktoru třídy pomocí metody *startTimer()*. V případě tohoto pluginu metoda *timerCallback()* kontroluje, zda byl výpočet STIPA dokončen (což je určeno boolean hodnotou *isCalculated* v třídě *STIPACalculator*). Pokud ano, labely jsou aktualizovány hodnotami STIPA a MTF. Tímto způsobem je uživateli pravidelně prezentován aktuální stav výpočtů, který se dynamicky mění v závislosti na vstupním signálu.

Celkově je uživatelské rozhraní pluginu navrženo tak, aby bylo přehledné a intuitivní pro uživatele. Důraz je kladen na prezentaci klíčových informací a umožnění jednoduché interakce s funkcemi pluginu. Je důležité zmínit, že plugin uchovává a zobrazuje informace o spočítaných hodnotách, a to i při minimalizaci okna pluginu. To umožňuje uživateli vrátit se k datům kdykoli, aniž by musel znovu spouštět analýzu.

Očekávaná hodnota MTF	Získané hodnoty MTF pro signál se středem na														Získaná hodnota STIPA	Očekávná hodnota STIPA
	125 Hz		250 Hz		500 Hz		1000 Hz		2000 Hz		4000 Hz		8000 Hz			
	1,60 Hz	8,00 Hz	1,00 Hz	5,00 Hz	0,63 Hz	3,15 Hz	2,00 Hz	10,00 Hz	1,25 Hz	6,25 Hz	0,80 Hz	4,00 Hz	2,50 Hz	12,50 Hz		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0,10	0,10	0,10	0	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,09	0,09	0,18	0,18
0,20	0,20	0,20	0,20	0,20	0,20	0,20	0,19	0,19	0,19	0,19	0,19	0,19	0,19	0,19	0,29	0,30
0,30	0,29	0,29	0,30	0,30	0,30	0,30	0,29	0,29	0,29	0,29	0,29	0,29	0,28	0,28	0,37	0,38
0,40	0,39	0,39	0,39	0,39	0,49	0,39	0,38	0,38	0,38	0,38	0,38	0,38	0,37	0,37	0,43	0,44
0,50	0,49	0,49	0,49	0,49	0,49	0,49	0,48	0,48	0,48	0,47	0,48	0,48	0,47	0,47	0,49	0,50
0,60	0,59	0,59	0,59	0,59	0,59	0,59	0,57	0,57	0,57	0,57	0,57	0,57	0,56	0,56	0,54	0,56
0,70	0,68	0,68	0,69	0,69	0,69	0,69	0,67	0,67	0,67	0,66	0,67	0,67	0,66	0,65	0,60	0,62
0,80	0,78	0,78	0,79	0,79	0,79	0,78	0,76	0,76	0,76	0,76	0,76	0,76	0,75	0,75	0,67	0,70
0,90	0,88	0,88	0,89	0,89	0,89	0,88	0,86	0,86	0,86	0,85	0,86	0,86	0,84	0,84	0,77	0,82
1,00	0,98	0,98	0,99	0,99	0,99	0,97	0,96	0,96	0,95	0,94	0,95	0,95	0,94	0,93	0,94	1,00

Tabulka 5.4: Výsledky testu modulační hloubky počítané přímou metodou.

Očekávaná hodnota MTF	Získané hodnoty MTF pro signál se středem na														Získaná hodnota STIPA	Očekávaná hodnota STIPA
	125 Hz		250 Hz		500 Hz		1000 Hz		2000 Hz		4000 Hz		8000 Hz			
	1,60 Hz	8,00 Hz	1,00 Hz	5,00 Hz	0,63 Hz	3,15 Hz	2,00 Hz	10,00 Hz	1,25 Hz	6,25 Hz	0,80 Hz	4,00 Hz	2,50 Hz	12,50 Hz		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,06	0,09	0,10
0,11	0,11	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,10	0,11	0,11	0,19	0,20
0,20	0,19	0,19	0,19	0,19	0,19	0,19	0,19	0,19	0,19	0,19	0,19	0,19	0,19	0,19	0,29	0,30
0,33	0,32	0,31	0,31	0,31	0,31	0,31	0,31	0,31	0,31	0,31	0,31	0,31	0,32	0,32	0,39	0,40
0,50	0,48	0,47	0,47	0,47	0,47	0,46	0,47	0,47	0,47	0,46	0,47	0,47	0,48	0,48	0,48	0,50
0,67	0,64	0,63	0,62	0,62	0,62	0,62	0,62	0,62	0,62	0,62	0,62	0,62	0,63	0,63	0,57	0,60
0,80	0,77	0,75	0,74	0,74	0,75	0,74	0,75	0,74	0,75	0,74	0,75	0,75	0,76	0,76	0,66	0,70
0,89	0,78	0,78	0,79	0,79	0,79	0,78	0,76	0,76	0,76	0,76	0,76	0,76	0,75	0,75	0,73	0,80
0,94	0,91	0,89	0,88	0,88	0,88	0,87	0,88	0,88	0,88	0,87	0,88	0,88	0,90	0,89	0,79	0,90
1,00	0,97	0,95	0,93	0,93	0,93	0,92	0,93	0,93	0,93	0,92	0,93	0,94	0,95	0,95	0,89	1,00

Tabulka 5.5: Výsledky testu fázového zkreslení oktávových filtrů.

Kapitola 6

Závěr

V rámci této bakalářské práce byl navržen a implementován VST plugin pro výpočet hodnoty STIPA. Výsledkem je užitečný nástroj, který nabízí rychlý a snadno interpretovatelný přehled o přenosových parametrech řeči. Je důležité zdůraznit, že tento plugin byl využit výhradně pro účely měření, což není obvyklá praxe pro VST pluginy.

Na začátku práce byla provedena podrobná analýza existujících formátů audio pluginů a vývojových prostředí. Tato analýza poskytla potřebné podklady pro výběr nejvhodnějších nástrojů a technologií pro realizaci daného projektu. Na základě těchto analýz byla vybrána technologie VST3 pro formát pluginu a framework JUCE pro jeho implementaci.

V rámci práce byl navržen a implementován komplexní algoritmus pro výpočet hodnoty STIPA, zahrnující implementaci oktávových filtrů a určení modulačních přenosových funkcí. Uživatelské rozhraní pluginu bylo navrženo tak, aby bylo intuitivní a snadno použitelné. Plugin kromě hodnoty STIPA zobrazuje také všechny hodnoty MTF a umožňuje export těchto hodnot do textového souboru pro další analýzu.

Nicméně, je třeba zdůraznit, že plugin je omezen svým zaměřením na vzorkovací frekvenci 44100 Hz. Jeho interakce s technologií VST navíc neumožňuje načítání celého bloku dat na začátku. Tento fakt znamená, že je nutné přehrávat signál po určité době, aby se data načetla do paměti. Tato omezení by mohla být řešena v budoucích verzích pluginu technologií ARA, která umožňuje získání celého bloku dat na začátku. Tato technologie je však stále relativně nová a dosud je využívána jen omezeným počtem pluginů. Je tedy nutné důkladně otestovat její možnosti a omezení pro tento konkrétní účel.

Dále je důležité zmínit, že během testování oktávových filtrů a výpočtu MTF byly objeveny nesrovnalosti v požadavcích na návrh oktávových filtrů mezi jednotlivými normami. Detaily jsou uvedeny v podkapitole 5.4.

Přínos této bakalářské práce spočívá v návrhu a implementaci nástroje, který může významně usnadnit analýzu a hodnocení přenosu srozumitelnosti pomocí parametru STIPA. Navíc práce přispívá k aktuálnímu stavu poznání v oblasti formátů audio pluginů a vývojových prostředí a může sloužit jako užitečný zdroj informací pro budoucí vývojáře audio pluginů.

Všechna stanovená kritéria a cíle bakalářské práce byla úspěšně splněna.

Literatura

[1] MCQUARRIE, J. G. *VST vs VST3: What's the difference between this two?*. Online. 2020-08-15. Dostupné z: `https://crumplepop.com/what-is-the-difference-between-vst-vs-vst3`. [cit. 2022-10-12].

[2] APPLE, INC. *About Audio Units*. Online. Dostupné z: `https://developer.apple.com/library/archive/documentation/MusicAudio/Conceptual/AudioUnitProgrammingGuide/Introduction/Introduction.html`. [cit. 2022-10-12].

[3] O'TOOLE, James. *Native vs. DSP-Powered Plugins: A Music Producer's Guide*. Online. 2020-12-04. Dostupné z: `https://www.avid.com/resource-center/native-vs-dsp-powered-plugins_a-music-producers-guide`. [cit. 2022-10-13].

[4] LV2. *LV2*. Online. Dostupné z: `https://lv2plug.in`. [cit. 2022-10-15].

[5] KIRN, Peter. *CLAP is a new open-source plug-in format from Bitwig, u-he - do we need it, and who will use it?*. Online. 2022-06-15. Dostupné z: `https://cdm.link/2022/06/clap-is-a-new-open-source-plug-in-format-from-bitwig-u-he-do-we-need-it-and-who-will-use-it`. [cit. 2022-11-30].

[6] PRODUCER HIVE. *WHAT ARE RTAS PLUGINS?*. Online. 2022-12-09. Dostupné z: `https://producerhive.com/buyer-guides/vst/what-are-rtas-plugins/`. [cit. 2022-12-19].

[7] MUSICPRODUCTIONNERDS. *What Are TDM Plugins? – Audio Plugin Formats Explained*. Online. Dostupné z: `https://musicproductionnerds.com/tdm-plugins-explained`. [cit. 2023-02-17].

[8] STEINBERG MEDIA TECHNOLOGIES. *3rd Party Developers*. Online. Dostupné z: `https://www.steinberg.net/developers`. [cit. 2022-11-12].

[9] SOURCEFORGE. *Java-Based Audio Plug-Ins*. Webové sídlo. Dostupné z: `https://jvstwrapper.sourceforge.net`. [cit. 2022-11-12].

[10] THE MATHWORKS, INC. *Audio Plugin Creation and Hosting*. Online. Dostupné z: `https://www.mathworks.com/help/audio/audio-plugin-creation-and-hosting.html`. [cit. 2022-11-24].

[11] SWEETWATER. *Csound*. Online. 2005-11-18. Dostupné z: `https://www.sweetwater.com/insync/csound`. [cit. 2022-11-09].

[12] SUPERCOLLIDER. *SuperCollider: A Platform For Audio Synthesis And Algoritmic Composition, Used By Musicians, Artists And Researches Working With Sound*. Online. Dostupné z: `https://supercollider.github.io`. [cit. 2022-11-10].

[13] RAW MATERIAL SOFTWARE LIMITED. *JUCE: AudioPluginFormat Class Reference*. Online. Dostupné z: `https://docs.juce.com/master/classAudioPluginFormat.html`. [cit. 2022-11-17].

[14] PIRKLE, Will. *Will Pirkle Audio Technology*. Webové sídlo. Dostupné z: `https://www.willpirkle.com`. [cit. 2022-11-22].

[15] *Cabbage: A Framework For Audio Software Development*. Webové sídlo. Dostupné z: `https://cabbageaudio.com`. [cit. 2022-11-18].

[16] FADIEIEV, Heorhii. *Experimentální softwarový syntetizér a sekvencer*. Online. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Dostupné z: `http://hdl.handle.net/11012/206261`. [cit. 2022-12-02].

[17] DSP ROBOTICS. *Flowstone*. Webové sídlo. Dostupné z: `http://www.dsprobotics.com/flowstone.html`. [cit. 2022-12-02].

[18] INSTITUT FÜR ELEKTRONISCHE MUSIK UND AKUSTIK - IEM. *Pure Data community, Documentation.*. Online. Dostupné z: `https://puredata.info/docs`. [cit. 2022-12-02].

[19] OPEN SOURCE INITIATIVE. *Licenses by Name*. Online. Dostupné z: `https://opensource.org/licenses/alphabetical`. [cit. 2022-12-18].

[20] FREE SOFTWARE FOUNDATION. *GNU General Public License*. Online. Dostupné z: `https://www.gnu.org/licenses/gpl-3.0.en.html`. [cit. 2022-12-18].

[21] FREE SOFTWARE FOUNDATION. *GNU Lesser General Public License*. Online. Dostupné z: `https://www.gnu.org/licenses/lgpl-2.1.html`. [cit. 2022-12-18].

[22] OPEN SOURCE INITIATIVE. *The 2-Clause BSD License*. Online. Dostupné z: `https://opensource.org/licenses/BSD-2-Clause`. [cit. 2022-12-18].

[23] OPEN SOURCE INITIATIVE. *ISC License*. Online. Dostupné z: `https://opensource.org/licenses/ISC`. [cit. 2022-12-18].

[24] THE MATHWORKS, INC. *Octave filter design - MATLAB fdesign.octave*. Online. Dostupné z: `https://www.mathworks.com/help/dsp/ref/fdesign.octave.html`. [cit. 2023-03-27].

[25] ČSN EN 61260-1. *Elektroakustika - Oktávové a zlomkooktávové pásmové filtry - Část 1: Technické požadavky*. Český normalizační institut, 2014. [cit. 2023-04-02]

[26] ČSN EN IEC 60268-1. *Elektroakustická zařízení - Část 16: Objektivní hodnocení srozumitelnosti řeči indexem přenosu řeči*. Český normalizační institut, 2020. [cit. 2023-04-15]

Příloha A

Obsah přiloženého média

- Priloha_A.zip
  - JuceLibraryCode
    - Libraries.zip
  - Source
    - PluginEditor.cpp
    - PluginEditor.h
    - PluginProcessor.cpp
    - PluginProcessor.h
    - STIPACalculator.cpp
    - STIPACalculator.h
  - STIPA.jucer
  - STIPA.vst3