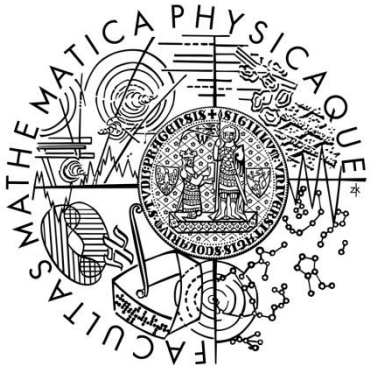
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Aleš Woska

M-lizer

Katedra spolehlivých a distribuovaných systémů

Vedoucí bakalářské práce: RNDr. Jan Kofroň, Ph.D.

Studijní program: Informatika

Studijní obor: Programování

Praha 2011

Chtěl bych poděkovat mému vedoucímu bakalářské práce a předmětu Softwarová praxe, RNDr. Janu Kofroňovi, Ph.D., za konzultaci, pomoc a vedení v průběhu tvorby práce.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne .....

podpis

Název práce: M-lizer

Autor: Aleš Woska

Katedra / Ústav: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: RNDr. Jan Kofroň, Ph.D

Abstrakt: Tato bakalářská práce se zabývá problematikou vývoje programového zvukového syntetizéru nad platformou .NET. Cílem je vytvořit programový nástroj, který z MIDI dat vytvoří data, která mohou být přímo předaná zvukové kartě, nebo podobnému zařízení, a být přehrána na výstupním zvukovém zařízení s důrazem na věrohodnost zvukového výstupu. Vytvořený zvuk by se tedy měl co nejvíce podobat zvuku skutečných nástrojů. Dalším cílem je vytvořit návod, jak psát programový zvukový syntetizér podporující technologii VST.

Klíčová slova: VST, .NET, SoundFont, zvuková syntéza, instrument, syntetizér

Title: M-lizer

Author: Aleš Woska

Department: Department of Distributed and Dependable Systems

Supervisor: RNDr. Jan Kofroň, Ph.D

Abstract: The thesis deals with the development of a software sound synthesizer of the .NET platform. The goal is to create a software tool which creates data, that are passed to a sound card or a similar device and that can by played on an output sound device, from MIDI data. Emphasis is placed on a authenticity of a sound output. Created sound would therefore be most resemble the sound of real instruments. Another goal is to create a guide, on how to write a software sound synthesizer that supports VST technology.

Keywords: VST, .NET, SoundFont, sound synthesis, instrument, synthesizer

Obsah

1. Úvod

Cílem bakalářské práce je vytvořit nástroj, který bude na základě vstupních MIDI dat generovat výstup, po jehož přehrání na příslušném zařízení a poslechu, bude pozorovatel nabývat dojmu, že vygenerovaný zvuk zní stejně, jako zvuk skutečného hudebního nástroje. V současné době je k dispozici mnoho zvukových syntetizérů. V této práci bych chtěl navrhnout takový nástroj, který by byl co nejpoužitelnější, odlišoval se od ostatních programových zvukových syntetizérů a jehož výsledný zvuk by zněl co nejvěrohodněji.

Při návrhu jsem procházel řešením několika úrovní problémů. Jako první přichází na řadu způsob obstarání vstupu. MIDI data může program získávat buďto z nástroje připojeného k počítači pomocí MIDI portu nebo z MIDI souboru uloženého v paměťi počítače. Program musí mít prostředky, pomocí kterých by měly přístup k oběma typům vstupu.

Další problém k rozmyšlení je způsob, jakým aplikace ze vstupu vytvoří data s požadovaným výsledkem. Program musí obsahovat obecné mechanizmy, kterými bude řešit převod syntaktických MIDI dat (MIDI technologie nedefinuje způsob, jakým mají být data zpracovávána, definuje pouze jejich význam) na sémantická data výstupu, které nemají žádnou syntaktickou strukturu (jedná se o posloupnost hodnot), ale po odeslání dat do zvukové karty a následném přehrání na výstupním zvukovém zařízení vznikne zvuk s požadovanými vlastnostmi. Tento proces se nazývá zvuková syntéza a program musí alespoň jednu konkrétní zvukovou syntézu implementovat.

Poslední věc k rozmyšlení je zpracování výstupu aplikace. Data musí být aplikací buďto pomocí systémového rozhraní přenesena do zvukové karty, aby pak mohla být zvukovou kartou přehrána na zvukovém výstupu počítače, nebo být uložena do zvukového souboru. Program musí disponovat prostředky pro komunikaci se zvukovým výstupem i pro práci se zvukovými soubory a jejich formáty.

Tato práce bude rozebírat všechny problémy, které je nutné řešit při vývoji aplikace tohoto typu. Popíše a představí technologie potřebné pro korektní a efektivní funkčnost aplikace. Klíčovým pojmem v práci je zvuková syntéza. V práci vysvětlím výběr konkrétního typu syntézy, a tuto syntézu porovnám s jinými typy,

osvětlím její výhody a nevýhody vůči ostatním typům. V práci dále popíši strukturu samotné aplikace s odůvodněním způsobu jejího řešení. Vysvětlím způsob využití technologií VST, MIDI, ASIO a Soundfont v práci pro efektivní a snadnější řešení problému, včetně propojení těchto technologií do jednoho celku.

2. Dekompozice problému

Psaní aplikace takového typu je technologicky náročné, protože proces tvorby zvuku prochází různými stádii, ve kterých využívá rozdílné technologie, a pro požadovaný výsledek je nutné ovládat každou technologii. Každá technologie je specifická a s každou se pracuje rozdílným způsobem. Program se proto bude skládat z několika vzájemně komunikujících modulů, přičemž každý modul bude řešit práci s rozdílnou technologií nutnou pro funkčnost celku.

Při řešení problému je třeba se zaměřit na různé aspekty úlohy. Nejprve je nutné zvolit vhodnou koncepci programu. Psaní této aplikace jako samostatného programu má značné nevýhody. Uživatel programu ho pro korektní spuštění musí nainstalovat (ve většině případů), což mnoho uživatelů obtěžuje. Uživatel pracující se zvukem obvykle používá svou oblíbenou aplikaci (většinou multifunkční) a proto mu nemusí vyhovovat instalace nové aplikace. Další problém může být ten, že samotná aplikace by naplno nevyužila svůj potenciál. Pro komplexnější práci se zvukem by musela ještě navíc implementovat řadu dalších funkcí.

Oba problémy řeší koncepce aplikace ve formě pluginu nebo knihovny. Zaprvé, uživatel nemusí instalovat další samostatnou aplikaci, ale postačí, když nahraje knihovnu do své oblíbené aplikace. K pluginu poté uživatel bude mít přístup z této aplikace. Zadruhé, hostující aplikace bude poskytovat vlastní prostředky pluginu (např. nahrávání a ukládání souborů, přístup ke zvukovým efektům, komunikace s dalšimi pluginy, …), takže funkcionalita a použitelnost pluginu tím vzroste, aniž by plugin nějaké další funkce musel implementovat.

Vzhledem k uvedeným výhodám jsem se rozhodl aplikaci konceptovat jako plugin. Pro plugin je klíčové, aby ho bylo schopno rozpoznat a nahrát co nejvíce aplikací. Měl by tedy implementovat určité rozhraní, přes které s pluginem bude schopno komunikovat co nejvíce hostujících zvukových aplikací. Ideální by tedy bylo implementovat nějaké standardizované rozhraní, které by mělo být schopno rozeznat více aplikací. Tento popsaný problém řeší několik technologií: Audio Units od firmy Apple, DirectX od firmy Microsoft, VST od firmy Steinberg a další (uvedl jsem pouze nejpoužívanější). Všechny zmíněné technologie poskytují standardizované rozhraní, přes které aplikace může komunikovat s pluginem. Zmíněné technologie se používají většinou komerčně ve zvukovém průmyslu

v různých nahrávacích studiích, audio stanicích a podobně. Rozhodl jsem se pro poslední zmíněnou technologii - VST, protože je ze všech zmíněných technologií nejpoužívanější. Využívá ho většina profesionálních aplikací (1) a je k dispozici i mnoho bezplatných (freeware) programů (Audacity, Wavosaur, VSTHost, …). Navíc, VST má oproti podobné zmíněné technologii (DirectX plugin) několik dalších výhod. VST je přenositelnější mezi platformami, protože DirectX plugin vyvíjí přímo firma Microsoft pro platformu Windows. Kód psaný pro VST je jednodušší, protože DirectX má složitou strukturu a vyžaduje množství kódu navíc. Mnoho profesionálních VST aplikací dokáže vygenerovat uživatelské rozhraní na základě kódu pluginu. Tato výhoda je ale pro tuto práci nedůležitá, protože uživatelské rozhraní jsem tvořil sám. Dalším důvodem pro tento výběr je, že technologie VST obsahuje přehledné a stručné programátorské rozhraní (VST SDK), pomocí kterého je tvorba pluginu snadnější. Mimo tyto popsané rozdíly mezi technologiemi VST a DirextX plugin není výrazný funkční rozdíl.

Programování VST pluginu se téměř automaticky rovná psaní kódu v jazyce C++. Funkce a rozhraní VST jsou psány v jazyce C++ a zpřístupňuje je VST SDK API. (VST Sofwtare Development Kit API). Existují nástroje, pomocí kterých je programátor schopný psát VST plugin nad platformou .NET. Technologii VST a platformu .NET kloubí dohromady projekt s názvem VST .NET, který umožňuje efektivní psaní pluginu v libovolném programovacím jazyce podporující platformu .NET. Alternativou k VST .NET je projekt s názvem Naudio, který jsem nepoužil, protože na rozdíl od VST .NET neimplementuje třídy a metody pro práci s VST. Nabízí se ještě možnost využití možností jazyka C++/CLI pro volání nativních funkcí VST SDK API přímo v C++. Tuto možnost jsem zamítl, protože bych psal kód, který je už implementovaný a optimalizovaný tvůrci VST .NET. Pro psaní pluginu jsem si vybral jazyk C# a platformu .NET, protože tato platforma je moderní, perspektivní a umožňuje psaní bezpečného managed kódu. Využitím existujících knihoven jsem získal více času a prostoru pro psaní samotné funkčnosti programu a zaměřením se na zajímavější části problému, které samotné rozhraní VST neřeší.

Výstup pluginu bude pomocí systémových funkcí posílán do bufferu zvukové karty a ta jej bude přehrávat pomocí vhodného výstupního zvukového zařízení. Výstup představuje posloupnost hodnot reprezentující diskrétní hodnoty průběhu

zvukového signálu. Zvuková karta tyto číslicové hodnoty pomocí A-D převodníku převede na analogový spojitý zvukový signál. Pluginu stačí řešit pouze problém přenosu dat z výstupu do zvukové karty. K tomuto přenosu potřebuje rozhraní, přes které bude komunikovat se zvukovou kartou. Přitom potřebuji docílit nízké odezvy při komunikaci s kartou, protože krátká odezva je při práci se zvukem nezbytná. Nativní ovladače systému Windows nezajišťují krátkou odezvu při komunikaci se zvukovou kartou. To je způsebeno nutností data několikrát převzorkovat, než je systém doručí do karty. Řešení nabízí mnoho typů technik.

První technikou je Windows Driver Model (WDM), což je rozhraní pro komunikaci s ovladači zvukové karty. Tento typ jsem nezvolil, protože má značné množství nevýhod - špatná dokumentace, neefektivní kód, žádná podpora čistě uživatelských ovladačů, způsobuje chyby v komunikaci s hardware (8). Naproti tomu neposkytuje žádné znatelné výhody.

Druhou technikou je Kernel Streaming (KS), který je součástí modelu zařízení systému Windows. Tato technika umožňuje přímý přístup k ovladačům zvukové karty. Podobnou funkci nabízí protokol Audio Stream Input/Output (ASIO), který také umožňuje přístup k ovladačům zvukové karty s krátkou odezvou a přijatelným rozhraním.

Z těchto dvou podobných technik (KS a ASIO) jsem zvolil ASIO, protože zajišťuje ještě kratší odezvu, než KS (ASIO předává zvukové kartě nezpracovaná data, kdežto KS data převzorkovává ze 44,1 kHz na 48 kHz).

Posledním důležitým krokem je rozmyslet si způsob generování výstupu aplikace. Generování zvuku řeší proces s názvem zvuková syntéza. Existuje mnoho typů zvukových syntéz (7). V aplikaci proto musí být přítomná komponenta, která implementuje nějakou zvukovou syntézu. Později v práci (kapitola 3.5) vyberu jeden konkrétní typ, který bude implementovaný v aplikaci. V kapitole 3.5 také zdůvodním výběr této metody a uvedu její hlavní výhody a nevýhody. Pro práci jsem zvolil syntézu zvukových tabulek (angl. Wavetable synthesis).

S výběrem zvolené syntézy souvisí způsob uložení dat, vzorků, s kterými bude syntéza pracovat. Nejperspektivnější možností je souborový formát Sound Font 2. Tento formát se vyznačuje tím, že kromě samotných vzorků potřebných k syntéze obsahuje také velkou míru metadat. Metadata usnadňují práci

se vzorky a celkově zjednodušují implementaci zvukové syntézy, např. obsahují informace o smyčkách, efektech, hlasitosti nebo míru citlivosti. Obsahuje tedy syntaktické i sémentické informace ke zpracovávaným vzorkům, a proto výrazně napomůže k implementaci problému. Alternativou k tomuto souborovému formátu se mohou nabízet například formáty WAV nebo AIFF. Ačkoliv také obsahují metadata, jejich množství je znatelně menší, než u formátu Sound Font 2. Vyžadují navíc použití platformy EMP (Extensible Metadata Platform). Ačkoliv oba formáty využívají podobný způsob uložení vzorku (PCM formát), Sound Font 2 převažuje obsahem dalších zmíněných parametrů potřebných pro zvukovou syntézu a díky tomu mohou zařízení, které Sound Font 2 podporují, generovat zvukový obsah ve stejné kvalitě, jako profesionální digitální samplery. Kromě zmíněných předností jsem vybral tento souborový formát také proto, že je standardizovaný, moderní, jeho použití podporují některé moderní hudební nástroje a je hojně využíván v hudebním průmyslu.

3. Použité technologie

3.1. Virtual Studio Technology

Virtual Studio Technology (dále jen VST), vyvinuto firmou Steinberg GmbH, je rozhraní pro integrování softwarových syntetizérů, zvukových efektů a nahrávacích systémů. VST a podobné systémy využívají digitální zpracování zvukového signálu pro simulování tradičního vybavení nahrávacích studií. VST je podporována většinou profesionálních zvukových aplikací (např. Cubase, Cakewalk, Adobe Premiere, …).

VST je platformně nezávislé. Nativní platforma pro VST je Windows, ale pluginy VST dokáží pracovat i pod platformou UNIX za použití speciálního software[1]. Podobně to je i s platformou OS X, která má také speciální software pro zpřístupnění VST[2].

Tvorba nástrojů pomocí rozhraní VST spočívá v psaní pluginů. Tyto pluginy může jejich uživatel nahrát do hostující aplikace, která tuto technologii podporuje. VST pluginy se dělí na nástroje (instrumenty) a efekty.

Nástroj je softwarová emulace některého z existujících nástrojů (zejména syntetizérů a samplerů, ale také klasických nástrojů), označují se zkratkou VSTi. Primární funkcí nástroje je generování zvuku. Například, jako vstup plugin načítá MIDI data (získaná ze souboru, z nástroje připojeného k počítači, atd.) a plugin na jeho základě generuje data, která jsou předáváná dále, až do výstupního zvukového zařízení a přehrána.

Plugin typu efekt pracuje jiným způsobem. Nevytváří nový typ dat, ale pouze upravuje ta ze vstupu a výstup je homogenní se vstupem (na rozdíl od instrumentu). Upravená data může distribuovat dál, např. pro vstup dalších efektů. Příklad efektu je ozvěna, wah-efekt nebo distorze.

Koncepci programu jsem tedy navrhl jako plugin nad technologií VST. Bude tak kompatibilní s aplikacemi i dalšími pluginy podporujícími VST. Díky tomu odpadá nutnost psát některé funkční části programu. Například není nutné řešit

[1] http://quicktoots.linuxaudio.org/toots/vst-pl

[2] http://www.fxpansion.com/index.php?page=5&tab=22

zpracování MIDI vstupu. Tuto funkci zajistí hostující aplikace, a to buď ze souboru, nebo přímo ze vstupu reálného MIDI nástroje připojeného k počítači. Plugin zažádá přes rozhraní VST o data a hostující aplikace je poskytne (nebo opačně). Vše je tedy řešené na úrovní volání funkcí mezi pluginem a hostující aplikací. Dále plugin nebude muset řešit ukládání výstupních dat do souboru, to také zajistí hostující aplikace přes rozhraní VST.

3.2. VST .NET

VST .NET je projekt umožňující psaní pluginu VST v libovolném .NET jazyce a vyplňuje tak mezeru mezi nativním jazykem VST, čímž je C++. Nabízí řadu rozhraní, pomocí kterých můžeme psát přehledný kód, bez podrobných znalostí standardů VST, které jsou zajištěné v rozhraních nabízející VST .NET. Současná verze je 0.9 a dále se pokračuje na vývoji.

K dispozici jsou rozhraní na dvou úrovních, a to na úrovni jádra a na úrovni frameworku.

Využití rozhraní na úrovni jádra umožňuje volání řízené (managed) verze nativních metod VST. Díky tomu bude mít kód minimální režii navíc, ale zato musí programátor podrobně znát standardy VST.

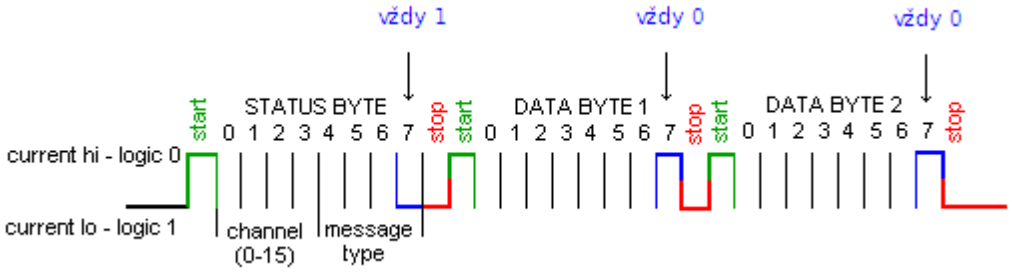
Využívání rozhraní na úrovni frameworku bude program stát režii navíc, zato může programátor vycházet z již definovaných rozhraní, tříd, metod a datových struktur, které stačí v případě rozhraní implementovat a v případě tříd instanciovat. VST .NET se skládá z několika knihoven (celkem ze tří, přičemž jenom dvě knihovny jsou nezbytné pro chod), které stačí, které stačí připojit k programu.

3.3. MIDI

„Musical Instrument Digital Interface (MIDI) je mezinárodní standard používaný v hudebním průmyslu jako elektronický komunikační protokol, který dovoluje hudebním nástrojům, počítačům i dalším přístrojům komunikovat v reálném čase prostřednictvím definovaného sériového rozhraní. Standard spravuje organizace MIDI Manufacturers Association.“ (1)

3.3.1. Přenosový protokol

„U MIDI je použit jednoduchý asynchronní sériový přenos dat (viz RS-232C) s využitím výše popsané proudové smyčky. Data jsou rozdělena do bajtů (osmice bitů, nejdříve se přenáší bit s nejmenší váhou) doplněných o start bit a stop bit. Start bit má vždy logickou hodnotu 0, stop bit logickou hodnotu 1. Hodinová frekvence vysílače je rovna 31250 Hz, což znamená, že se celý bajt i se start bitem a stop bitem přenese za 320 µs. Jedná se o poměrně netypickou hodnotu, mnoho ostatních sériových rozhraní se drží spíše celočíselných násobků 75 Hz. Z přenášených bajtů jsou skládané MIDI zprávy. Jejich délka je rovna jednomu, dvěma či třem bajtům, ovšem jeden speciální typ zprávy má neomezenou délku. První bajt zprávy se nazývá stavový bajt (status byte). Jeho zvláštností je to, že má nastavený nejvyšší bit na logickou hodnotu 1 (jeho dekadická hodnota je vždy větší než 127). Všechny ostatní bajty mají tento bit nulový (jejich hodnota je menší než 128), takže i v případě, že MIDI zařízení ztratí synchronizaci (vytažení konektoru apod.), může poměrně jednoduše detekovat začátek další zprávy.“ (2)



Obrázek 1: *Přenos MIDI zprávy o délce tří bajtů. Tři bajty je obvyklá délka většiny MIDI zpráv. (2)*

3.3.2. Typy MIDI zpráv

V každé MIDI zprávě její první bajt určuje typ zprávy a číslo kanálu[3]. Kanál se dá abstrahovat jako jeden primitivní nástroj (např. nějaký hudební software emulující MIDI nástroje přiřadí prvnímu kanálu zvuk klavíru a druhému kanálu zvuk houslí, atd.). V prvním bajtu zprávy je využito 7 bitů, protože osmý bit je konstantně logická jednička. V MIDI je definováno 16 kanálů, na jejich zakódování do binární hodnoty postačí 4 bity. Zbylé tři bity slouží pro rozpoznání typu MIDI zprávy, kterých je logicky právě 8. Ve zbývajících bajtech zprávy jsou přenášeny parametry

[3] Až na kód příkazu pro nehudební příkazy (s kódem příkazu 0xf0), který v prvním bajtu neobsahuje číslo kanálu.

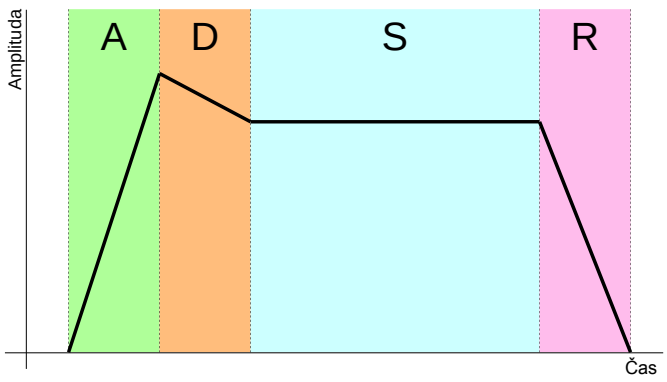
MIDI zprávy. Některé typy zpráv využívají dva parametry po sedmi bitech (osmý bit je vždy logická nula) a jiné typy zpráv využívají jeden čtrnáctibitový parametr (v každém ze dvou bajtů zprávy je poslední bit logická nula). Struktura MIDI zprávy je znázorněna v Obrázku 1. V případě zprávy typu System Exclusive uvozené bajtem 0xf0 následuje libovolný počet bajtů s daty, sekvence dat musí být ukončena bajtem 0xf7. Následující tabulka uvádí kódy a významy všech typů MIDI zpráv.

Kód příkazu (hexadecimálně)	**Název/význam**	**Počet parametrů**	**Účel prvního parametru**	**Účel druhého parametru** (jestli je přítomen)
0×80	**Note off**	2	číslo noty	rychlost doznění
0×90	**Note on**	2	číslo noty	hlasitost přehrávání
0×a0	**Aftertouch**	2	číslo noty	síla stisku
0×b0	**Controller**	2	číslo MIDI kontroleru	hodnota, která se má nastavit
0×c0	**Program change**	1	číslo programu	-
0×d0	**Channel Pressure**	1	síla stisku	-
0×e0	**Pitch Wheel**	1	hodnota Pitch Wheel	-
0×f0	**System Exclusive**	více bajtů ukončených 0xf7	hodnota zprávy	… více hodnot

Tabulka 1: Tabulka kódů a významů MIDI zpráv.

3.3.3. Obálka ADSR

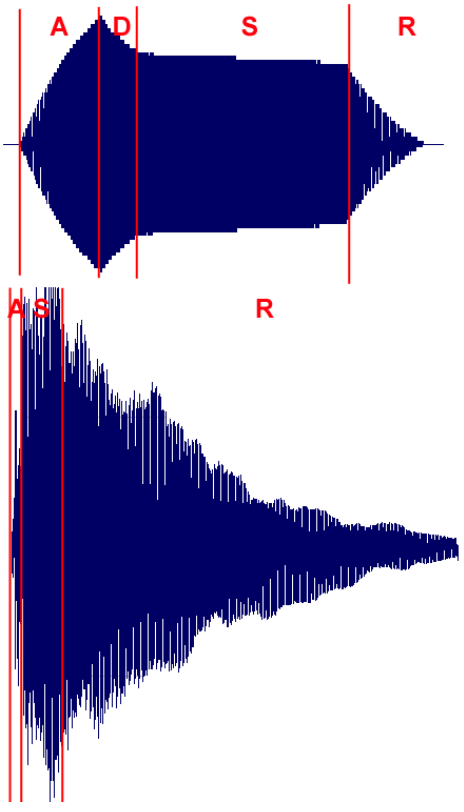
Obálka ADSR popisuje závislost amplitudy generovaného signálu na čase (průběh hlasitosti). Zkratka ADSR je odvozena od názvů čtyř částí průběhu signálu: Attack (náběh), Decay (pokles), Sustain (ustálená hodnota), Release (doznění).



Obrázek 2: Obálka ADSR signálu představujícího přehrávanou notu.

V okamžiku stisku klávesy (tím je vygenerovaná Note On MIDI zpráva, která je předána ke zpracování) nastává část A. Během této události signálu roste amplituda až na definovanou hodnotu, poté v části D se amplituda ustálí na hodnotu definovanou ve zprávě Note On, tím se přejde do části S. Jeden typ nástrojů má v části S smyčku, a ta je přehrávána, dokud není přijata zpráva Note Off (většinou se jedná o syntetizéry a samplery). Druhý typ nástrojů přejde z části S plynule do části R (většinou klasické nástroje). V části R je amplituda signálu zmenšována až na nulovou hodnotu, a tím je ukončeno přehrávání tónu.

Pro lepší pochopení obálky ADSR poslouží následující obrázek, který představuje obálky ADSR dvou nástrojů. Horní část obrázku představuje roh, který má poměrně dlouhou část S (sustain). Druhá část obrázku představuje klavír, který má naopak část S velice krátkou a téměř po celou dobu znění nástroje se signál pohybuje v části R (release). Z obálky lze tedy vyčíst povaha nástroje, roh zní dlouho dobu monotónně, kromě náběhu. Klavír zní nejsilněji při úderu kladívka a jeho hlasitost poté pouze slábne.



Obrázek 3: Obálka ADSR pro roh a klavír.

3.3.4. Přehled různých MIDI zpráv

Asi nejčastěji posílaná MIDI zpráva je Note On. Reprezentuje stisknutí klávesy na nástroji. Parametry této zprávy jsou číslo stisknuté klávesy a úroveň hlasitosti části S v ADSR obálce. Nástroje, které nepodporují snímání úrovně hlasitosti, odesílají hodnotu 64 (přičemž nejhlasitější je maximální hodnota 127). V případě, že je úroveň hlasitosti nulová, pak tato zpráva musí být zpracována jako zpráva Note Off (podle specifikace MIDI).

Zpráva Note Off je opačná zpráva k Note On. Její parametry jsou číslo uvolněné klávesy a rychlost doznění (v obálce ADSR část R). Jestliže nástroj tuto možnost neposkytuje, pak odesílá hodnotu 64 (nejrychlejší je maximální hodnota 127).

Zpráva After Touch reprezentuje sílu stisknutí přehrávané noty. Ne všechny nástroje toto podporují, protože je potřeba zvláštní senzory pro každou klávesu, které tento tlak dokáží měřit. V obálce ADSR tato zpráva ovlivňuje amplitudu všech částí. Ve druhém bajtu zprávy je uloženo číslo noty, které se zpráva týká, a ve třetím bajtu zprávy je uložena síla stisku klávesy z rozsahu 0 - 127. Některé nástroje nedokáží rozlišit všechny úrovně hlasitosti, v tomto případě musí úrovně odpovídat hodnotám násobku osmi. Realizace tohoto typu zprávy je nákladná, proto se častěji využívá zpráva Channel Presure, pro kterou stačí, aby senzory měřili tlak na všech klávesách najednou a posílájí jejich průměr. Tato zpráva má pouze jeden parametr, který určuje sílu stisku (opět je 127 maximální hodnota). Nepřítomnost druhého parametru (číslo klávesy) je logická, protože zpráva se netýká jedná klávesy, ale všech.

Zpráva Controller reprezentuje změnu hodnoty nějakého kontroleru (kontroler může být kterýkoliv ovládací prvek na nástroji, který má nějakou funkci, např. Modulation Wheel, různé druhy pedálů, ozvěna, atd.). Tyto kontrolery tedy způsobují nějaký zajímavý zvukový efekt. První parametr značí číslo kontroleru. Přitom číslo 1 má kontroler Modulation Wheel a další čísla kontrolerů může nástroj interpretovat libovolně. Jsou ale definovány čísla jednotlivých kontrolerů (9) a nástroj by je měl dodržovat, pokud možno, co nejvíce. Není to ale vyložené nutné, až na kontroler číslo 1. Druhý parametr značí hodnotu, na kterou by měl být nastavený tento kontroler.

Další zpráva, Program Change, reprezentuje změnu zvuku nástroje (tedy programu). Například, pokud chce uživatel změnit zvuk svého nástroje z klavíru na zvuk flétny. Tato zpráva ma jediný parametr, který značí číslo programu nástroje (z rozmezí 0 - 127).

Zpráva Pitch Wheel reprezentuje změnu výšky všech tónů v daném kanálu pomocí speciálního ovládacího prvku. Ve druhém a třetím bajtu zprávy je uložena čtrnáctibitová hodnota, která reprezentuje posun tónů ve stupnici o 0 až $2^{14} = 16384$ centů. Přitom střední poloha je $2^{13} = 8192$ centů. Tedy, pokud je hodnota parametru Pitch Wheel zprávy větší než 8192, znamená to posun výšky tónů směrem nahoru a pokud je menší než 8192, znamená to posun výšky tónů směrem dolů. Cent je setina půtónu. Proto posun výšky tónů a jeden cent odpovídá posunu o $\frac{1}{100}$ půltónu, neboli o $\frac{1}{1200}$ oktávy (oktáva má 12 půtónů). Kvůli kódování MIDI zprávy nelze parametr této zprávy získat přímo, ale za použití jednoduchého algoritmu používajícího bitové operace:

```
public UInt16 PitchWheelArgument(byte prvniBajt, byte druhyBajt)
{
      UInt16 argument = druhyBajt;
      argument <<= 7;
      argument |= prvniBajt;
      return argument;
}
```

Poslední typ zprávy, System Exclusive, reprezentuje speciální stystémové zprávy nástroje. Těmito zprávami může nástroj např. měnit celkovou hlasitost (master volume), řídit přehrávání skladeb uložených v nástroji nebo posílat informace o sobě.

3.3.5. Převod mezi číslem tónu a frekvencí tónů

Většina zpráv se odkazuje na číslo klávesy. Pro identifikaci čísla tónu má MIDI zpráva k dispozici $2^7 = 128$ hodnot, tedy čísla tónů jsou z rozsahu 0 - 127. Frekvence tónů se odvozuje od frekvence tzv. komorního A, jehož frekvence je 440 Hz. Tento tón odpovídá číslu 69. Posun výšky tónu o oktávu nahoru znamená změnu frekvence tónu na dvojnásobek. Naopak, posun tónu o oktávu dolů znamená změnu frekvence tónu na polovinu.

Protože oktáva má 12 půtónu, je lehké odvodit, že posun o jeden půltón znamená změnu frekvence o $2^{\frac{1}{12}}$- násobek původní frekvence. Obecně, jestliže x je číslo tónu odpovídající tónu o známé frekvenci f_x a y je číslo klávesy odpovídající tónu o hledané frekvenci f_y, potom je hodnota této frekvence určená vztahem:

$$f_y = f_x \cdot 2^{\frac{y-x}{12}}$$

Vzorec je platný jak pro změnu frekvence směrem nahoru (vychází kladný exponent, a proto se f_x násobí), tak pro posun frekvence směrem dolů (poté vychází záporný exponent a f_x se dělí).

3.4. Audio Stream Input/Output

Audio Stream Input/Output (zkratka ASIO) je platformně nezávislý vícekanálový protokol pro ovladače zvukových karet vyvinutý firmou Steinberg GmbH. Zajišťuje velmi nízkou odezvu a věrné rozhraní mezi softwarovou aplikací a zvukovou kartu. Umožňuje tedy přímý přístup ke zdrojům zvukové karty. Díky těmto vlastnostem jsou ovladače ASIO ideální pro použití s VST nástroji a pluginy. Nízká odezva je nutná pro přehrávání dat v reálném čase. Jestliže je odezva mezi příchozí MIDI událostí a přehráním zpracovaných dat velká, je to v hudební praxi naprosto nepoužitelné. Je proto nutné generovaná data předat bufferu zvukové karty co nejrychleji.

Aby plugin zajistil nízkou odezvu při komunikaci se zvukovou kartou, musí využívat protokoly ASIO. Jelikož komunikuje s hostující aplikací, nemusí s nimi pracovat přímo. Sama hostující aplikace zajistí komunikaci a využití protokolu ASIO pro předávání dat zvukové kartě. Jedinou podmínkou samozřejmě je, aby uživatel měl protokoly ASIO nainstalované, a tudíž je mohla hostující aplikace využívat.

Původní protokoly ASIO jsou určené pro výkonné profesionální zvukové karty a jejích používání je zpoplatněno. Pro běžné zvukové karty, a také pro zvukové čipy integrované na základních deskách jsou k dispozici nezávislé univerzální ovladače ASIO. Nejpoužívanější se nazývá ASIO4ALL, je volně ke stažení a jeho používání je bezplatné.

3.5. Zvuková syntéza

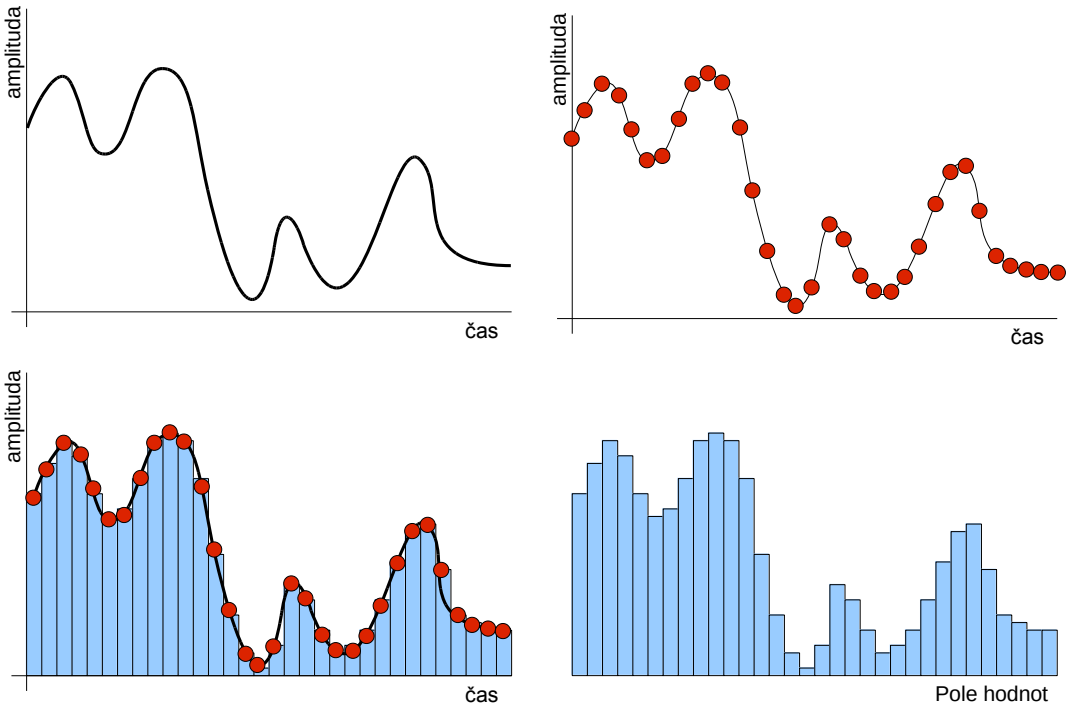
Zvuková syntéza je technika generování zvuku za použití elektronického hardware nebo software. Nejběžnější použití zvukové syntézy je hudební, při kterém jsou elektronické nástroje, zvané syntetizéry, používané při produkci a nahrávání hudby. Zvuková syntéza má mnoho využití, jak v akademické, tak umělecké oblasti. Běžně je využívaná několika způsoby.

První způsob je generování unikátního a zajímavého zvuku nebo zvukového zabarvení, kterého nelze dosáhnout za použití akustických nástrojů.

Druhý způsob je reprodukce nebo modelování zvuků akustických nástrojů, nebo jiných zvukových zdrojů (lidský hlas, zvuky prostředí, atd.), ze skutečného světa.

Třetí způsob je usnadnění automatizace systémů a procesů (syntéza lidského hlasu, která se dá použít např. pro rozhlas ve vlakových stanicích, usnadnění čtení textu nevidomým, atd.) (10). V práci bych se chtěl primárně zaměřit na druhý způsob syntézy, to je reprodukce zvuků akustických nástrojů.

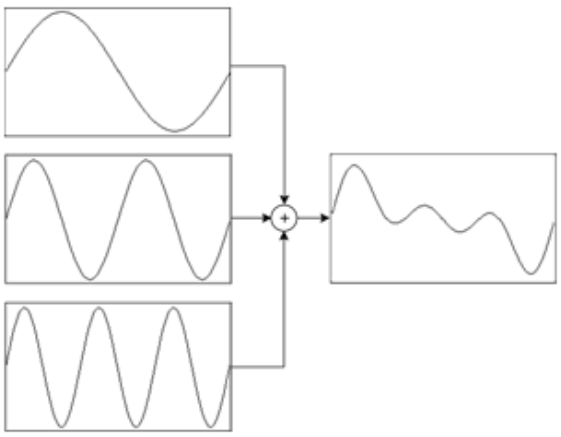
Existuje celá řada typů zvukových syntéz. Každá z nich má své výhody a nevýhody, tedy každá je vhodná k jinému účelu. V následujícím textu budu používat výraz křivka. Křivka je reprezentace průběhu zvukového signálu. Ukazuje průběh amplitudy v průběhu času. Amplitudu signálu vyznačuje vertikální osa, zatímco časový průběh reprezentuje horizontální osa. Křivka může být v textu chápana jako abstraktní pojem (v tom případě se jedná o popsaný průběh amplitudy signálu v čase) nebo může mít konkrétní implementaci v počítači. Téměř vždy se jedná o pole hodnot. Tyto hodnoty se dají získat pomocí audio-digitálního převodu, kdy jsou analogové hodnoty nějakým nahrávacím zařízením změřeny, během měření jsou tyto naměřené hodnoty navzorkovány a pomocí analogově-digitální konverze převedeny na digitální hodnoty, které se dají uložit do počítače. Při tomto převodu dochází k chybám měření, chybám při vzorkování a k chybám zaokrouhlování. Jedná se stále o křivku, která je konkrétní implementací uložená v počítači. Při reprezentaci v poli se nedá přímočaře zaznamenat průběh v čase. Ten se dá jednuduše odvodit, jestliže je známá vzorkovací frekvence.



Obrázek 4: Reprezentace a průběh převodu křivky (waveform)

3.5.1. Typy syntéz

Aditivní syntéza vytváří zvuky sčítáním křivek, které jsou obvykle harmonicky spojené. Její princip je tedy skládání harmonických signálů, nejčastěji sinusového průběhu. Tato syntéza je početně nenáročná. V práci tuto metodu nepoužiji, protože je spíše vhodná pro „typický zvuk“ syntetizátorů, ale já chci dosáhnout realistického zvuku skutečných akustických nástrojů. Přesto je tato syntéza využívaná ve velkém množství syntetizátorů.



Obrázek 5: Znázornění principu aditivní syntézy

Subtraktivní syntéza je založena na filtrování harmonicky bohatých křivek a využívá jednoduchý akustický model, který předpokládá, že nástroj může být napodoben generátorem jednoduchého signálu (generující signály tvaru vlny, čtverce, trojúhelníku, atd.) následujícím filtrem. Kombinace jednoduchých modulačních okruhů (např. pulsová šířková modulace a synchronizace oscilátoru), spolu s fyzicky nerealistickými lowpass filtry, jsou zodpovědné za zvuk klasických syntetizátorů obyčejně spojených s analogovou syntézou (10). Podobně jako aditivní syntéza i tato je nevhodná pro mou práci, protože způsobuje typický syntetizérový zvuk. V některých případech může být takový druh zvuku žádoucí, ale ne v mém případě.

Syntéza frekvenční modulace (angl. Frequency Modulation Synthesis, neboli FM Synthesis) pracuje na principu frekvenční modulace signálu z jednoho oscilátoru signálem z druhého oscilátoru. Pomocí složitých matematických funkcí dochází při modulaci ke vzniku nových harmonických frekvencí, které se přidávají k původnímu nosnému signálu. FM syntetizér může obsahovat až desítky oscilátorů, které lze mezi sebou libovolně propojovat. FM syntéza je obzvlášť vhodná pro vytváření kovových a řinčivých zvuky zvonů, činelů a jiných bicích nástrojů (11). Podobně jako u předešlých dvou typů, je pro mou práci tato metoda nevhodná. Způsobuje zvuk charakteristický jen pro určitou skupinu nástrojů.

Granulární syntéza manipuluje s velmi malými vzorky (o délce 1 až 50 milisekund). Mnoho těchto vzorků je navrstveno a jsou přehrávány s rozdílnou rychlostí, fází, hlasitostí a výškou. Díky tomu může dosáhnout mnoha rozdílných zvuků. Přehrávání při nízké rychlosti způsobuje zvukovou paletu přírodních zvuků a přehrávání při vysoké rychlosti produkuje zvuk neobvyklé zvukové palety. Tato metoda je vhodná pro různé zvukové efekty nebo jako materiál pro další zpracování, ale není vhodná pro syntetizéry.

Syntéza fyzikálního modelování (Physical modeling synthesis, PhM Synthesis) používá množinu rovnic a algoritmů, pomocí kterých simuluje skutečné nástroje nebo jiný fyzikální zdroj zvuku. Tento typ syntézy provádí simulaci fyzikálních vlastností nástroje a díky tomu dokáže produkovat velmi realistické výsledky. Díky tomu by tato metoda mohla být vhodná pro mou práci, ale je výpočetně náročná a pro syntézu v reálném čase musí být její funkce omezené. Pro syntézu v reálném čase proto tato syntéza generuje méně realistický výstup.

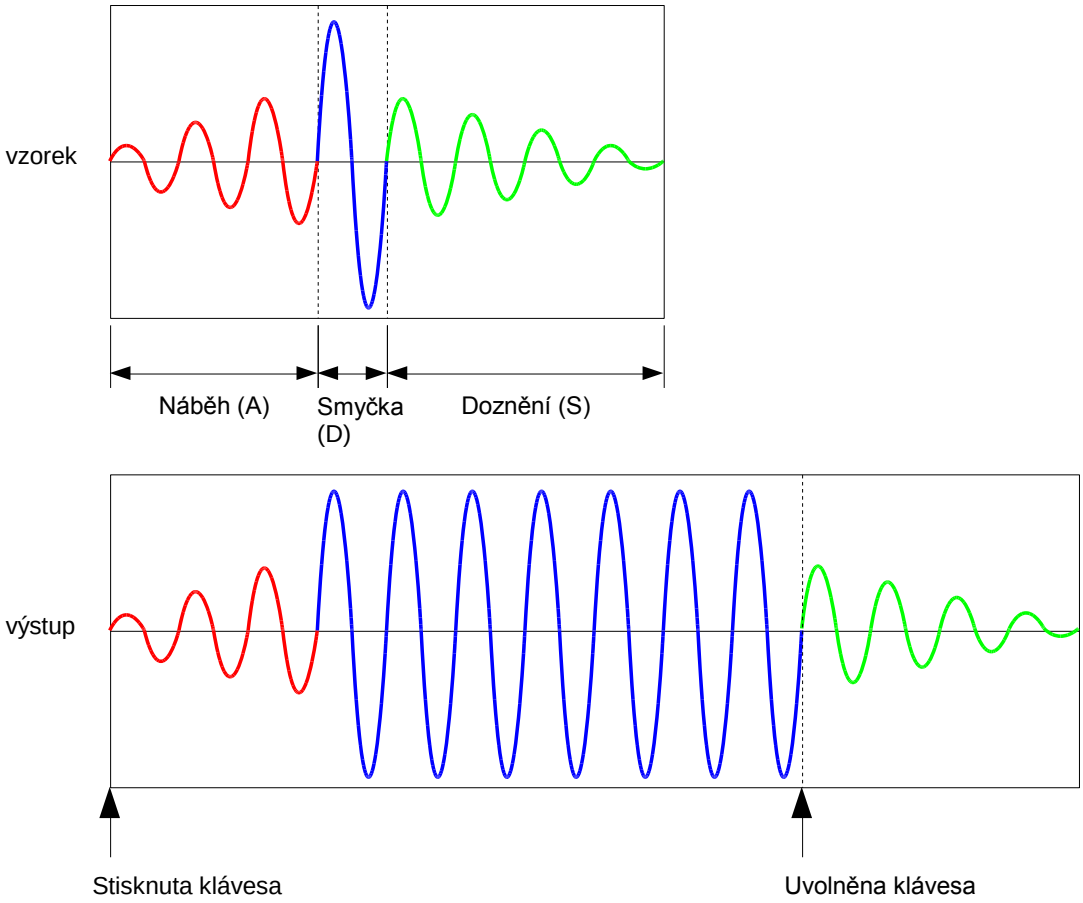
3.5.2. Wavetable syntéza

Jako velice perspektivní se jeví syntéza zvukových tabulek (angl. Wavetable Synthesis). Princip této syntézy je periodické přehrávání předuložených vzorků[4] a dokáže produkovat zvuk skutečných nástrojů i typický syntetizérový zvuk. Tyto vzorky jsou uložené v paměti a mohou být přehrávány s rozdílnou rychlostí pro vyprodukování všech výšek tónů (protože zpravidla jsou k dispozici pouze vzorky o několika málo výškách). Každý vzorek obsahuje tři části, které odpovídají obálce ADSR, která je popsána v kapitole 3.3.3, ale chybí zde část D (decay). V části S se nachází smyčka, která je syntetizérem přehrávaná tak dlouho, dokud je držena klávesa. Díky smyčce může být každý vzorek přehrávan libovolně dlouho (jestliže některé nástroje nemají logickou smyčku, jako např. klavír, pak je ve vzorku tato smyčka umístěna na konec vzorku a opakuje se pouze „prázdný“ zvuk)[5]. Wavetable syntéza je ve své podstatě speciálním případem aditivní syntézy, takže si zachovala její výhodu výpočetní nenáročnosti. Díky využití vzorků ale stírá její nevýhodu a dokáže produkovat realistický zvuk. Kromě realistických zvuků dokáže produkovat i typický syntetizérový zvuk (protože vzorky nerozlišují druh nástroje a může v něm být uloženo cokoliv). Díky popsaným výhodám jsem zvolil právě tento typ syntézy. Je výpočetně nenáročná a zároveň dokáže produkovat realistický zvuk akustických i elektronických nástrojů. Na druhou stranu, může být obtížné sehnat kvalitní zvukový vzorek (kvalitní a profesionální vzorky bývají placené) a v případě kvalitního vzorku může nastat větší spotřeba paměti (v nejhorším případě v řádech několika desítek až stovek MB).

V textu pracuji s pojmy vzorek (angl. sample) a vlna (angl. waveform). Na abstraktní úrovni se jedná o stejné pojmy. Vlna popisuje půběh amplitudy signálu v čase a má spíše fyzikální význam, zatímco vzorek jsou konkrétní hodnoty signálu uložené v počítači a má především informatický význam.

[4] Přitom tato syntéza nerozlišuje vzorky s jedním cyklem, neboli s jednou periodou (single-cycle) a vzorky s více cykly.

[5] Syntéza dokáže pracovat pouze se smyčkou a vzorek nemusí obsahovat části A a S. Naopak přítomnost smyčky ve vzorku je nezbytná.



Obrázek 6: Příklad vzorku a výstupu Wavetable syntézy

3.6. SoundFont

SoundFont je obchodní značka vztahující se k souborovému formátu a související technologii vytvořené pro vyplnění mezery mezi nahrávaným a syntetizovaným zvukem. SoundFont obsahuje zvukové vzorky a informace pro jejich organizaci a zpracování. Podle těchto informací může nástroj, obvykle řízený MIDI daty, generovat tóny požadovaných barev, výšek, hlasitostí a dalších parametrů (4). Tato značka využívá soubory formát s názvem Sound Font 2.

3.6.1. Struktura souborového formátu Sound Font 2

Souborový formát Sound Font 2 je organizovaný do formátu nazývaný RIFF (Resource Interchange File Format), je to souborový formát firmy Microsoft sloužící pro ukládání multimediálních zvukových a obrazových předloh. Formát RIFF definuje strukturu uložení dat do souboru pro různé typy a formáty dat (multimediální kontejner). RIFF se skládá z datových struktur zvaných shluky (angl. chunks), každý shluk má svoji čtyřznakovou signaturu (ID) definovanou v hlavičce

shluku. Za shlukem následují data do velikosti dané v hlavičce shluku. Shluk může obsahovat tzv. podshluk (subchunk). Každý RIFF soubor začíná shlukem se signaturou RIFF, dále soubor obsahuje jeden nebo několik shluků se signaturou LIST, které obsahují dodatečný identifikátor formátu dat následujících dat v souboru. Existuje ještě shluk se signaturou JUNK, používaný jako výplň dat, pro zarovnání dat na velikost čteného bloku. Data jsou v souborech řazená v pořadí little endian. Existuje i varianta RIFF ve tvaru big endian s příponou souboru RIFX. (5)

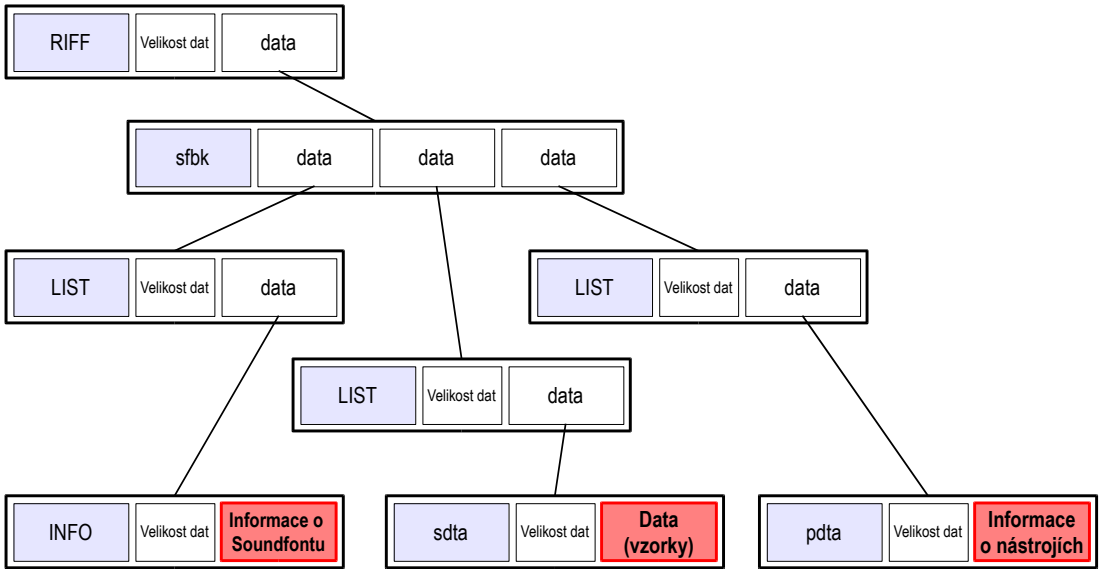
Název bloku	Velikost bloku
ChunkID	4 B
velikost	4 B
data	|velikost| B

Tabulka 2: Struktura shluku (chunku) používaná ve formátu RIFF

Celý souborový formát Sound Font 2 je tedy členěn do určitých bloků. Ze specifikace formátu RIFF musí každý takový soubor začínat se shlukem se signaturou „RIFF“. V případě souborového formátu Sound Font 2 se v datech tohoto shluku nachází podshluk se signaturou „sfbk“, který je jednoznačný identifikátor tohoto formátu. Tento shluk obsahuje tři další podshluky se signaturou „LIST“. První tento podshluk obsahuje obecné informace o souboru[6]. V jeho datech se nachází další podshluk se signaturou „INFO“. Druhý podshluk se signaturou „LIST“ obsahuje podshluk se signaturou „sdta“, který reprezentuje vzorky v čisté podobě. Tato data jsou uložena sekvenčně vcelku, bez zvláštního sémantického významu. Význam těchto dat určuje až třetí podshluk se signaturou „LIST“. Ten obsahuje podshluk se signaturou „pdta“. V tomto posledním podshluku je uložen seznam zón, nástrojů, generátorů a modulátorů. Zmíněné seznamy se odkazují na data ve vzorku. Celou situaci nejlépe přehledně vyjadřuje Obrázek 7. Veškeré podrobnosti a technické specifikace souborového formátu Sound Font 2 jsou k dispozici v jeho dokumentaci[7].

[6] verze souboru, jméno SoundFontu, datum vytvoření, komentáře, atd.

[7] http://connect.creativelabs.com/developer/SoundFont/sfspec21.pdf



Obrázek 7: Struktura souborového formátu Sound Font 2

3.6.2. Reprezentace nástrojů

V abstraktní struktuře SoundFontu jsou hlavním objektem nástroje (angl. instruments), které mohou být sloučeny do větších celků, přednastavení (angl. preset). Jako preset si lze představit zvukovou banku nějakého konkrétního typu nástroje (např. klavír) a nástroje zahrnuté v tomto presetu zastupují skutečné jednotlivé nástroje (např. klasický klavír, hapsichord, varhany, atd.).

Nástroj se skládá z množiny zón, přičemž tyto zóny nemusí být disjunktní. Na zónu se může nahlížet, jako na množinu tónů se stejnými vlastnostmi (stejné, až na výšku tónů). Přitom je důležité, že uložené zvukové vzorky se vztahují na zónu a ne na celý nástroj. To znamená, že každý nástroj bývá tvořen více zvukovými vzorky. Zóny nemusí pokrývat celý rozsah nástroje, takže pro některé krajní tóny nemusejí být definována data. Každá zóna obsahuje množinu generátorů a modulátorů, které se vztahují pouze pro danou zónu.

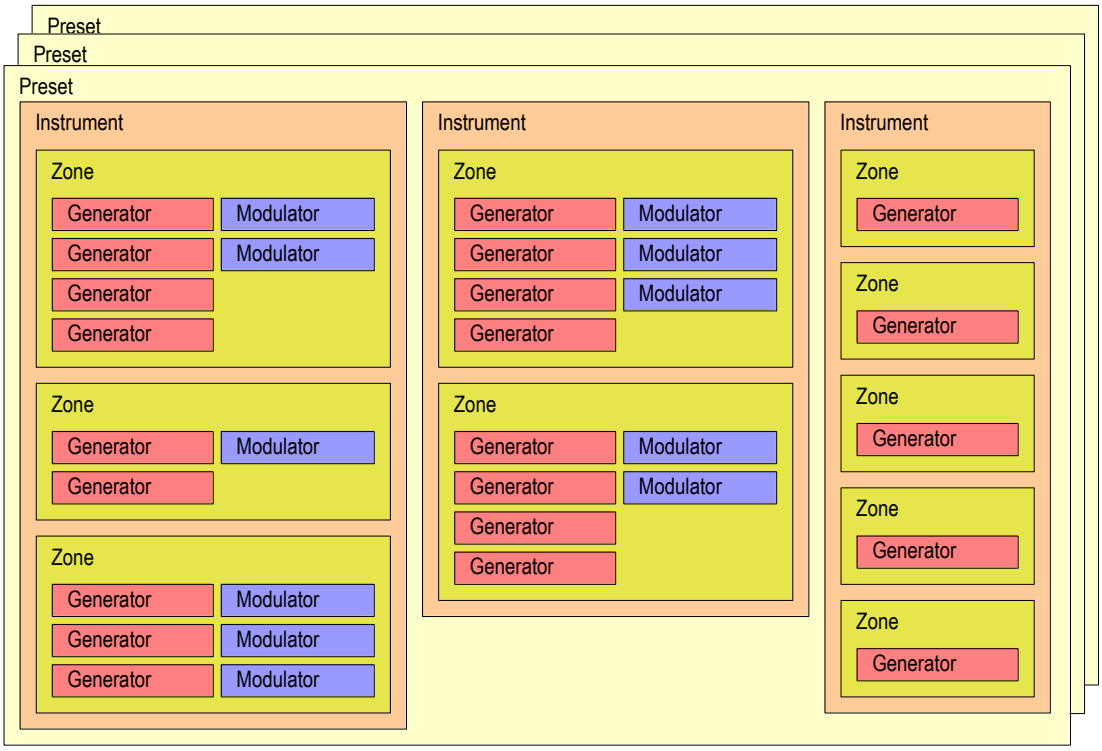
Generátory jsou potřené pro základní generování zvuku. Právě generátory definují základní informace o vzorku a práci s ním. Obsahují, mimo jiné, informace jako:

- rozsah tónů zóny (na kterých tónech je definovaná daná zóna)
- offset nástroje v datech (data jsou uložená jako souvislý blok, tento generátor definuje, na které pozici se nachází vzorek pro danou zónu)
- velikost vzorku dané zóny

- definování části attack, release, začátek a konec smyčky
- výška tónu vzorku (od které se počítají výšky ostatních tónů)

Modulátory definují data a vlastnosti pro úpravu již definovaných dat. Většinou se jedná o modifikaci různými MIDI kontrolery.

Generátory a modulátory neobsahují funkce či algoritmy pro generování a úpravu zvuku. Jsou v nich pouze uloženy údaje, pomocí kterých syntetizér tyto funkce provádí.



Obrázek 8: Příklad abstraktní struktury obsahu souboru formátu Sound Font 2.

4. Návrh řešení

4.1. Struktura VST .NET

VST .NET se skládá ze tří knihoven, a to Interop, Core a Framework.

Knihovna Interop je zodpovědná za vystupování pluginu jako nativního VST pluginu a přenáší komunikaci mezi hostující aplikací a pluginem k managed (řízenému) pluginu VST .NET, který implementuje skutečnou funkčnost. Provádí typovou konverzi nutnou pro přechod mezi řízeným a nežízeným kódem. Tato knihovna obsahuje funkci VSTPluginMain, kterou volá hostující aplikace. Vrací datovou strukturu AEffect obsahující inicializační informace o pluginu. Knihovna Interop zachytí volání hostující aplikace a přepošle je dvojici tříd Plugin Command Proxy a Plugin Command Stub.

Knihovna Core obsahuje definici běžných řízených datových struktur sdílených mezi pluginem a knihovnou Interop. Všechny tyto struktury jsou přímou reprezentací nativních neřízených struktur z C++.

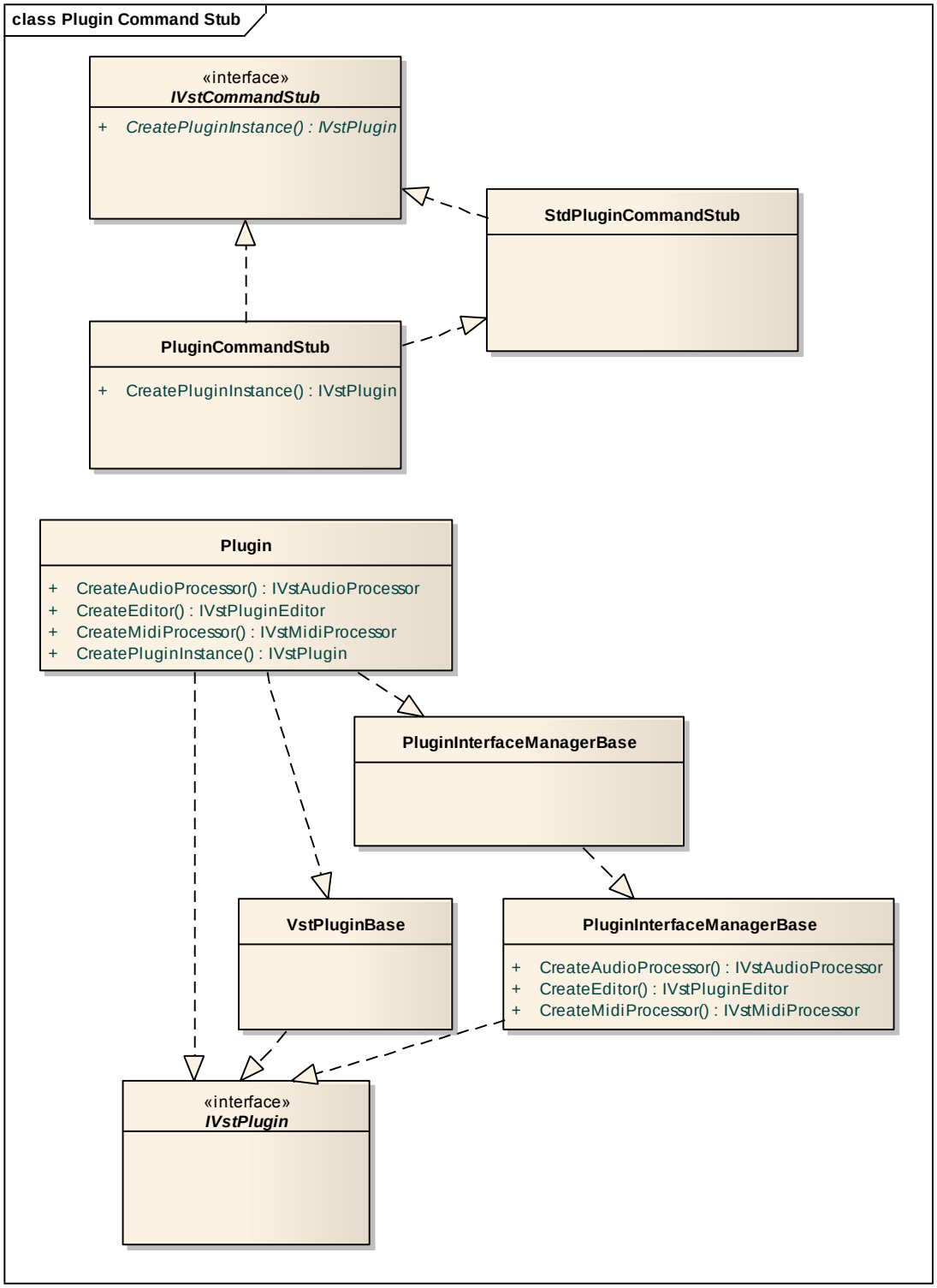
Poslední knihovna, Framework, je volitelná a poskytuje programátorovi možnost strukturované implementace VST pluginu, neboli poskytuje celý VST .NET Framework. Tato knihovna implementuje rozhraní z knihovny Core a tím poskytuje třídy pro přímo práci s VST .NET.

4.2. Práce s VST .NET

Pro rozeznání a načtení pluginu hostující aplikací musí plugin obsahovat třídy, které implementují určitá rozhraní. Nezbytná je třída Plugin Command Stub. Další třídy jsou pouze volitelné, ale jejich implementace je nutná pro požadovanou funkčnost pluginu (zpracování MIDI zpráv, zvukový výstup, grafické rozhraní).

4.2.1. Plugin Command Stub a Plugin

Třídy Plugin Command Stub a Plugin jsou důležité k identifikaci a načtení pluginu. Na tyto třídy se přímo odkazuje rozhraní VST.



Obrázek 9: Různé způsoby implementace třídy Plugin Command Stub

Knihovna Interop se přímo odkazuje na metody třídy PluginCommandStub. Tato třída se dá vytvořit dvěma způsoby. Na úrovni knihovny Core může být tato třída vytvořena implementací rozhraní IVstCommandStub. Na úrovni knihovny Framework může být třída PluginCommandStub odvozená od třídy StdPluginCommandStub. V tom případě je nutné předefinovat metodu

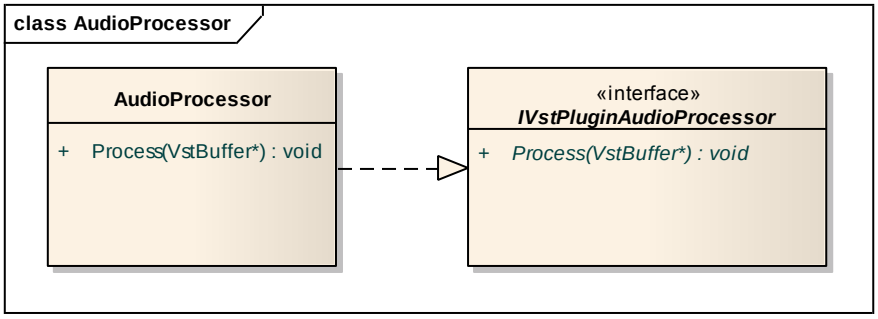
CreatePluginInstance, která vrací instanci pluginu. Instance pluginu je třída implementující rozhraní IVstPlugin.

Implementace rozhraní IVstPlugin je povinná pro všechny VST .NET pluginy. V pluginu je nezbytná přítomnost třídy Plugin implementující toto rozhraní. Opět je na výběr z několika možností, jak toto rozhraní implementovat.

První možnost je implementace vlastního rozhraní.

Druhá možnost je odvození od některých ze tříd VstPluginBase nebo VstPluginWithInterfaceManagerBase. Třída VstPluginBase implementuje základní metody a hodnoty nezbytné pro chod pluginu. Třída VstPluginWithInterfaceManagerBase je odvozená od třídy PluginInterfaceManagerBase, která implementuje rozšířené metody pro pokročilejší práci s VST. Především to jsou metody CreateMidiProcessor (která vrací instanci třídy MidiProcessor), CreateAudioProcessor (která vrací instanci třídy AudioProcessor) a CreateEditor (která vrací instanci třídy PluginEditor). Tyto metody jsou postupně volány hostující aplikací, a tím poskytují pokročilejší funkčnost.

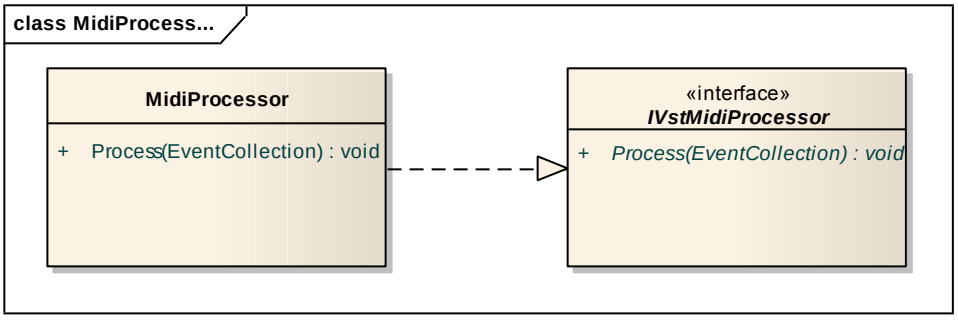
4.2.2. Audio Processor



Obrázek 10: Implementace třídy AudioProcessor

Třída AudioProcessor je z principu nutná, aby plugin mohl předat výstupní data hostující aplikaci. Pro napsání VST pluginu typu efekt i instrument je nutné implementovat tuto třídu. To je možné provést implementací rozhraní IVstPluginAudioProcessor. Zvláštní pozornost je třeba věnovat metodě Process, která je volaná pokaždé, když hostující aplikace skrze knihovnu Interop žádá a vygenerování výstupu pluginem. Plugin předá výstup aplikaci nakopírováním do bufferů, jejichž reference se nachází v parametrech metody.

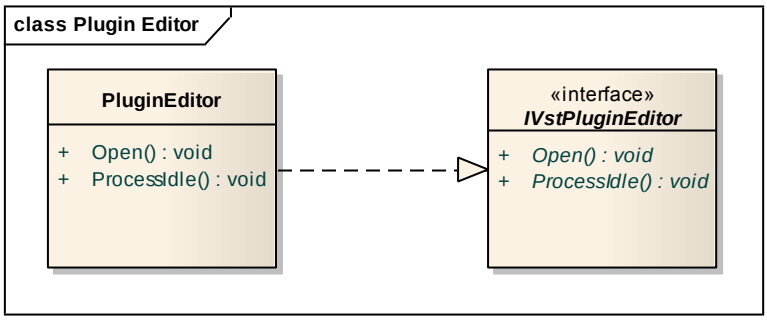
4.2.3. Midi Processor



Obrázek 11: Implementace třídy MidiProcessor

Implementací třídy MidiProcessor plugin získá možnost obsluhovat příchozí MIDI zprávy. Tato třída musí implementovat rozhraní IVstMidiProcessor, kde je klíčová metoda Process. Její parametr obsahuje kolekci MIDI zpráv (protože v jednom čase může přijít více zpráv, je nutné je předávat v kolekci).

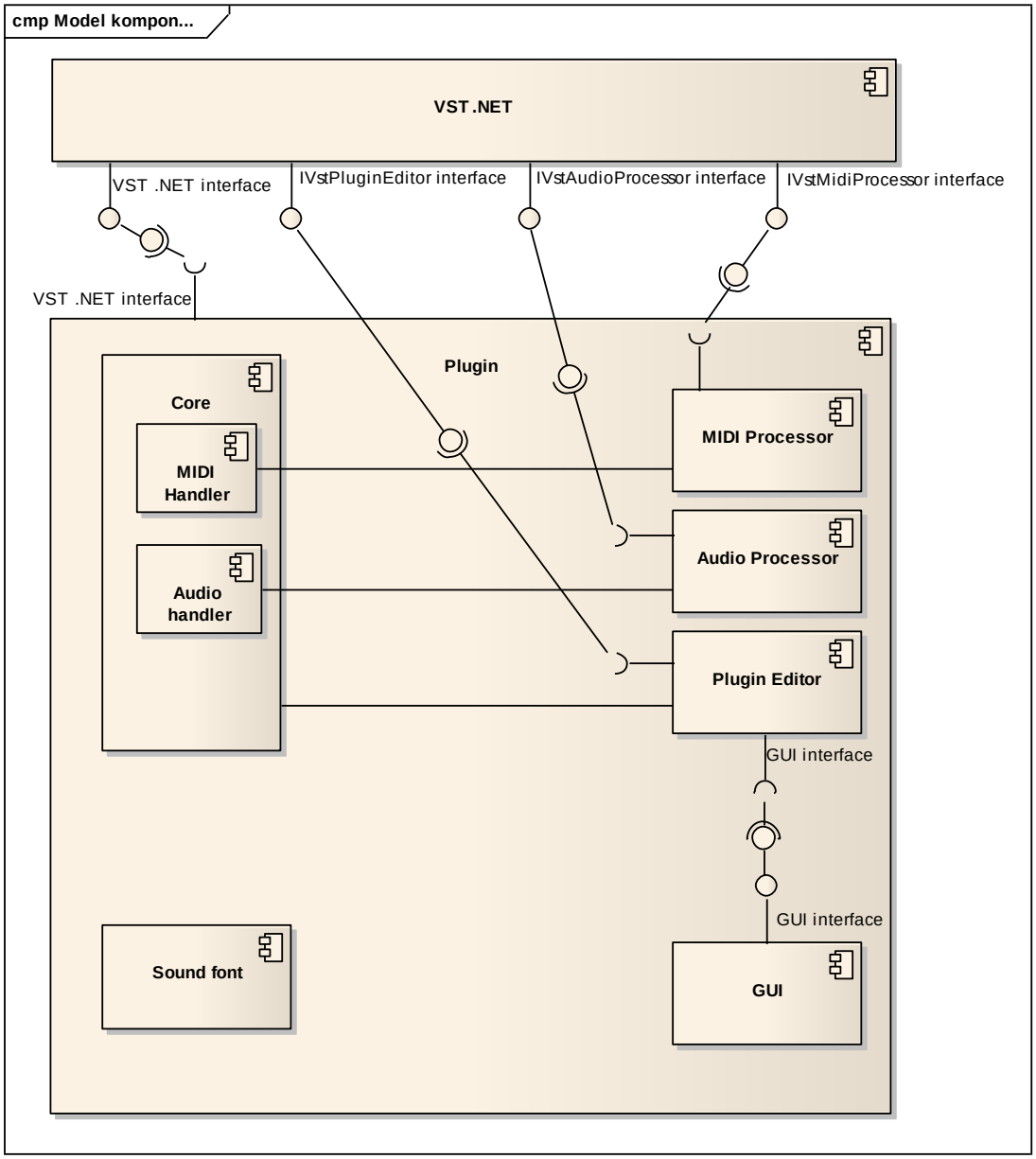
4.2.4. Plugin Editor



Obrázek 12: Implementace třídy PluginEditor

Implemetací třídy PluginEditor pomocí rozhraní IvstPluginEditor bude plugin schopný zobrazit grafické prostředí. VST .NET pracuje s objekty WinForms. Při implementaci třídy je zapotřebí vytvořit objekt typu WinFormsControlWrapper, což je interní datová struktura VST .NET, která zpracovává grafické rozhraní WinForms. Je to generický typ, který je třeba instanciovat na typ odvozený od UserControl. Ve třídě PluginEditor jsou podstatné dvě metody. První metoda, Open, inicializuje grafický prvek. Druhá metoda, ProcessIdle, je prostředník pro komunikaci mezi pluginem a grafickým prvkem.

4.3. Model komponent



Obrázek 13: Model komponent

Plugin bude dekomponován na následující vzájemně komunikující komponenty.

Komponenta VST .NET se bude skládat pouze z knihoven VST .NET. Bude poskytovat rozhraní a datové struktury pro implementaci tříd potřebných pro správný chod pluginu.

Komponenta Plugin bude zapouzdřovat třídy obsluhující příchozí události (ať už MIDI, interakci uživatele nebo žádost o vygenerování výstupu) a jádro

aplikace. V této komponentě se bude nacházet šest pod-komponent, a to Core, MIDI Processor, Audio Processor, Plugin Editor, Sound Font a GUI.

Na komponentu MIDI Processor se bude odkazovat hostující aplikace vždy, když rozhraní VST bude chtít odeslat MIDI zprávy pluginu. Tato komponenta bude pro obsloužení těchto zpráv využívat rozhraní komponenty Core.

Podobně je na tom komponenta Audio Processor, na kterou se bude odkazovat hostující aplikace pokaždé, když rozhraní VST bude od pluginu požadovat nějaká výstupní data.

Komponenta Plugin Editor bude komunikovat s grafickým rozhraním. Slouží jako prostředník pro komponentu GUI, které implementuje vlastní grafické rozhraní pluginu. Na komponentu Plugin Editor se bude přímo odkazovat VST přes hostující aplikaci. Komponenta Plugin Edtior bude využívat API nabízející komponenta GUI, která poskytuje uživatelské rozhraní a zajišťuje interakci s uživatelem.

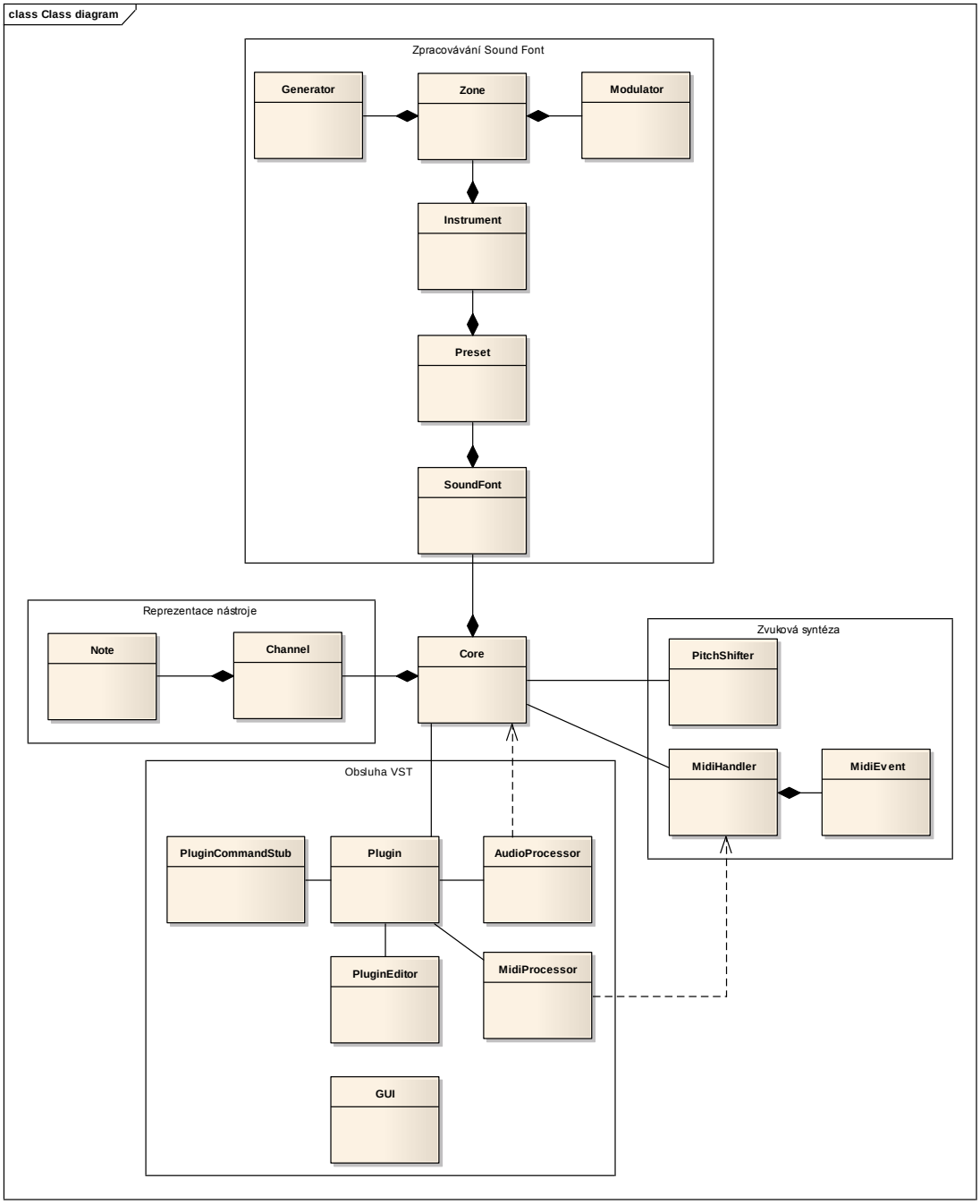
Komponenta Sound Font bude mít za úkol načítání souborů formátu Sound Font 2 a jejich parsování. Bude obsahovat pomocné metody pro práci s tímto formátem a odpovídající datovou strukturou.

Komponenta Core bude v pluginu nejdůležitější a bude vykonávat hlavní funkčnost. Bude zodpovědná za komunikaci mezi interními komponentami pluginu. Bude obsahovat třídy a datové struktury pro syntézu. Její funkčnost bude rozdělena do dalších pod-komponent a to MIDI Handler a Audio handler.

Komponenta MIDI Handler bude obsluhovat všechny příchozí MIDI zprávy. Při jejich zpracování se bude odkazovat na objekty z komponenty Core.

Komponenta Audio Handler bude obsluhovat a řídit generování výstupu. Při generování bude využívat objekty z komponenty Core.

4.4. Model tříd



Obrázek 14: Model tříd

Horní část modelu popisuje strukturu datové struktury reprezentující soubor formátu Sound Font 2. Třída reprezentující tento formát se nazývá SoundFont. Tato třída obsahuje, kromě vzorku, kolekci instancí tříd Preset. Hierarchicky pak každá třída Preset obsahuje kolekci instancí tříd Instrument, každá třída instrument se skládá z instancí tříd Zone a tyto třídy se skládají z kolekcí tříd Generator a Modulator, dále obsahují další potřebné datové struktury. Kolekce instancí tříd

SoundFont je vložená ve třídě Core, protože ta bude moci zpracovávat více souborů formátu Sound Font 2 (proto více instancí třídy SoundFont).

Ústřední třída Core se logicky dělí do čtyř částí, a to, již zmíněná, část zpracovávající soubory formátu Sound Font 2, obsluha VST, realizace zvukové syntézy a reprezentace nástroje.

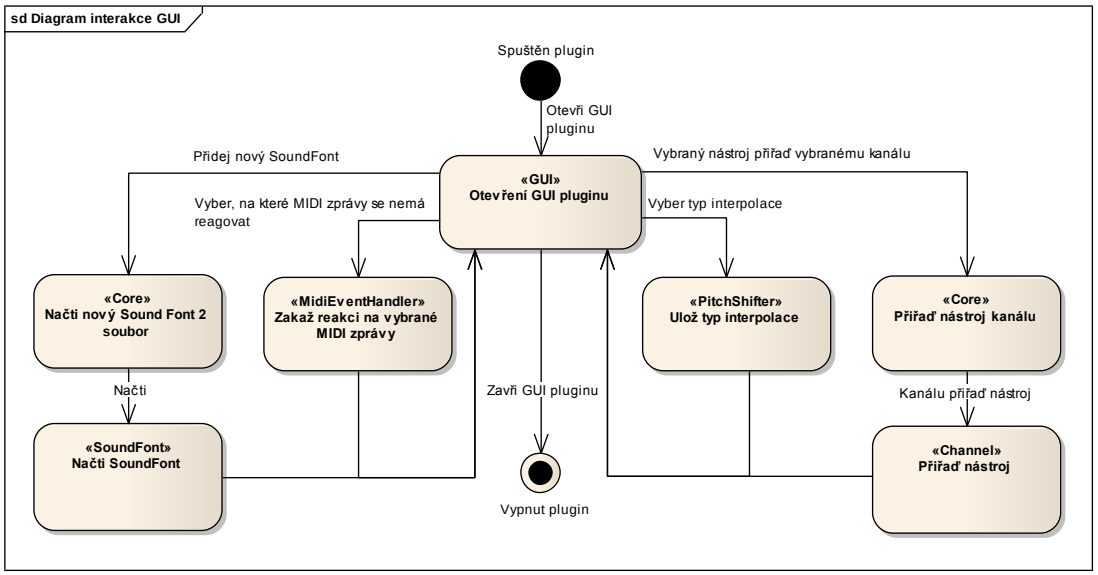
Obsluha VST zahrnuje implementaci tříd popsaných v kapitole 4.2.

Třídy realizující zvukovou syntézu jsou čtyři, a to PitchShifter, MidiHandler, MidiEvent a AudioProcessor. Přitom MidiEvent slouží jako abstraktní třída pro odvození konkrétních MIDI zpráv. Třída MidiHandler má za úkol obsluhu příchozích MIDI zpráv reprezentovanými instancemi tříd odvozených od třídy MidiEvent. Třída PitchShifter má za úkol změnu výšky tónu vzorku. Třída AudioProcessor nacházející se třídě Plugin řídí přehrávání vzorků podle parametrů získaných z MIDI zpráv a metadat SoundFontu.

Třída Core využívá pro reprezentaci nástroje třídy Channel (která reprezentuje MIDI kanál) a Note (která reprezentuje klávesu na stupnici). A to tak, že obsahuje maximálně 16 instancí třídy Channel (instanciované jsou pouze využívané MIDI kanály) a třída Channel obsahuje 128 instancí tříd Note (protože každý kanál má právě 128 různých tónů).

4.5. Diagramy interakcí

Popsané třídy mezi sebou budou komunikovat následujícím způsobem.



Obrázek 15: Diagram interakce grafického rozhraní pluginu

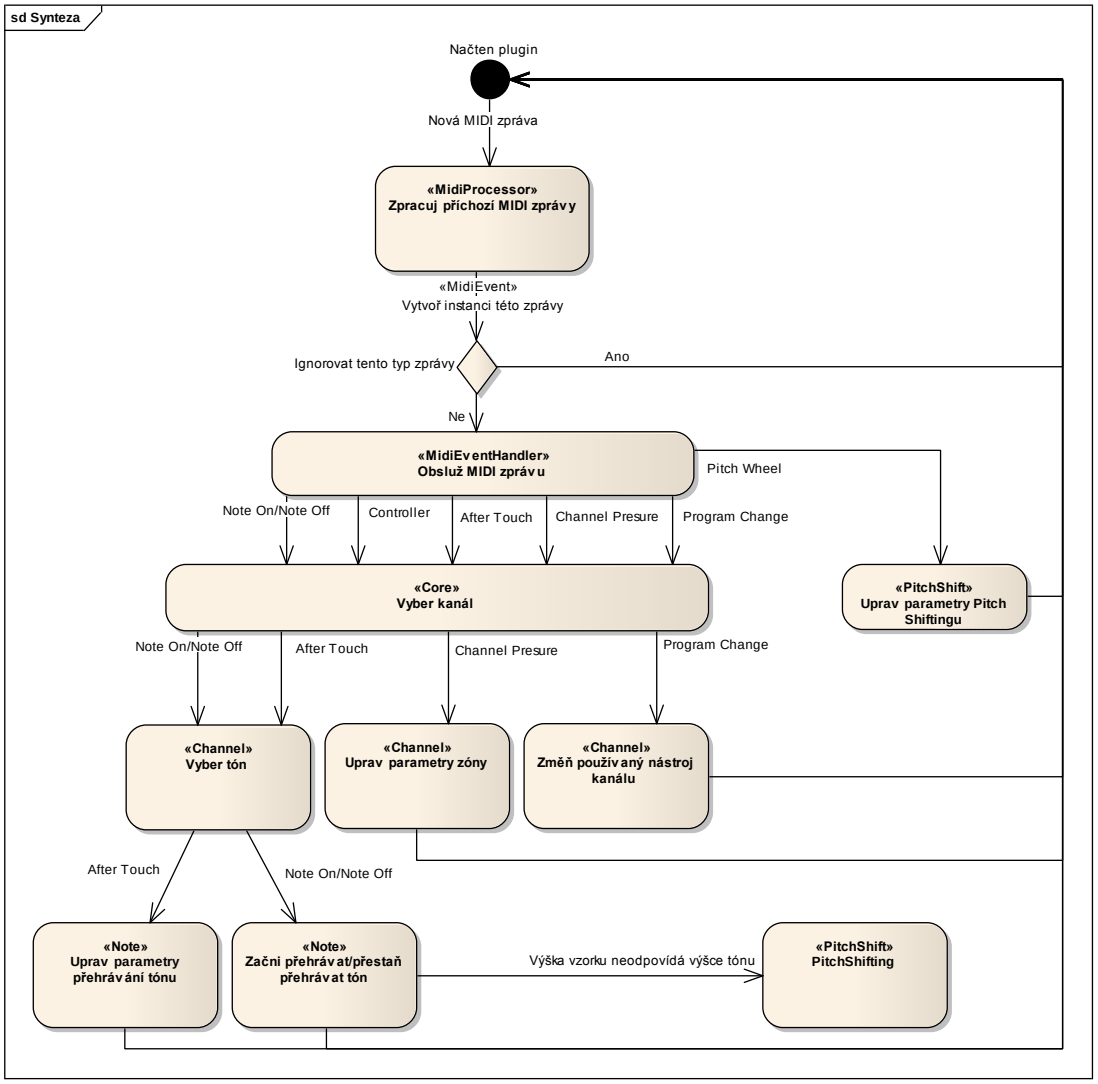
Grafické rozhraní pluginu bude nabízet čtyři funkce:

- načtení nového Sound Font 2 souboru
- přiřazení nástroje z tohoto souboru vybranému kanálu
- výběr typu interpolace v syntéze a zakázání reakce na vybrané MIDI zprávy
- Přidání nového Sound Font 2 souboru se uskuteční zavoláním příslušné metody ve třídě Core, která vytvoří novou instanci třídy SoundFont.

Přiřazení nástroje ze zvoleného Sound Font 2 souboru ke zvolenému kanálu se rovněž uskuteční zavoláním příslušné metody ve třídě Core. Tato metoda vybere zvolený nástroj z odpovídající instance třídy SoundFont a zavolá odpovídající metodu instance třídy Channel (výběr instance závisí na výběru kanálu v grafickém rozhraní) a jako parametr jí předá vybraný nástroj.

Informace o zakázání reakcí na některé MIDI zprávy se uloží do instance třídy MidiEventHandler, která poté bude tyto vybrané typy zpráv ignorovat.

Výběrem typu interpolace se nastaví typ zvolené interpolace v instanci třídy PitchShifter, která bude dále pracovat pouze s vybraným typem interpolace.



Obrázek 16: Diagram interakce zvukové syntézy

Tok dat začíná příchozí MIDI zprávou (ve skutečnosti kolekcí MIDI zpráv, protože v jednu chvíli může přijít více než jedna MIDI zpráva). MIDI zprávu zachytí instance třídy MidiProcessor. Ta zkontroluje, jestli je povolena reakce na tento typ zprávy. Jestli ano, pak vytvoří instanci třídy odvozené od třídy MidiEvent odpovídající typu zprávy. Instance MIDI zprávy je předána na zpracování třídě MidiEventController.

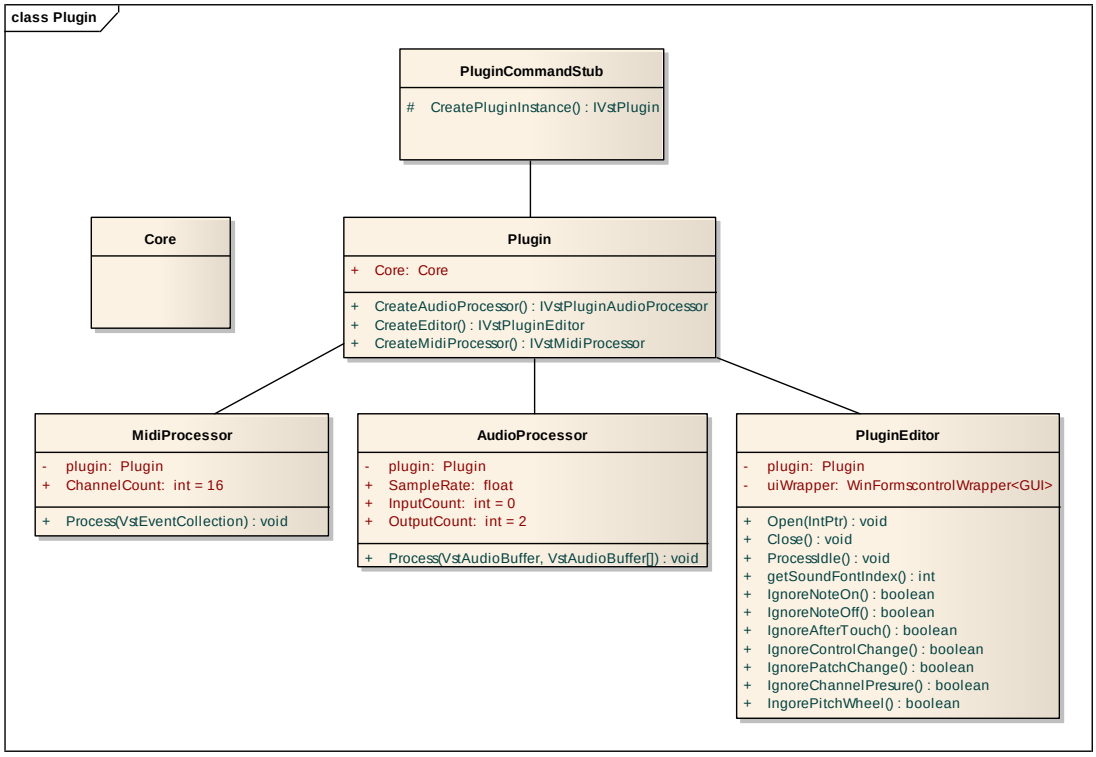
Hlavní typy MIDI zpráv jsou Note On a Note Off, které ovládají přehrávaní vzorků, a proto reakce na tyto typy zpráv hrají hlavní roli ve zvukové syntéze. Třída MidiEventHandler přes třídu Core vybere kanál a tón popsaný v MIDI zprávě (tedy vybere konkrétní instanci třídy Channel a v ní konkrétní instanci třídy Note). V této instanci třídy Note zavolá odpovídající metodu, která začne nebo ukončí přehrávání vzorku odpovídající tomuto tónu. Přitom se může stát, že výška tónu vzorku

neodpovídá výšce přehrávaného tónu. V tom případě se zavolá obslužná metoda v instanci třídy PitchShifter, která vykoná změnu výšky tónu vzorku (Pitch Shifting).

Další MIDI zprávy už pouze mění parametry syntézy. Reakce na ostatní typy zpráv spočívá ve změně některých z datových struktur vyskytujících se v hierarchii instanci třídy Core (především instance tříd PitchShifter, Channel a Note).

5. Programátorská dokumentace

5.1. Třída Plugin



Obrázek 17: Struktura třídy Plugin

Na následujících pět tříd se přímo odkazuje rozhraní VST.

První třída použitá v pluginu se nazývá PluginCommandStub. Má jedinou metodu CreatePluginInstance, jejímž cílem je vrátit instanci třídy Plugin.

Třída Plugin obsahuje instanci třídy Core, která je nejdůležitější částí programu, a odehrávají se v ní nejdůležitější děje v pluginu (tato třída bude detailněji popsána v kapitole 5.3). Třída Plugin dále obsahuje tři metody, přičemž každá z nich vrací instanci odpovídající třídy:

- CreateAudioProcessor
- CreateMidiProcessor
- PluginEditor

Třída MidiProcessor obsahuje privátní referenci na instanci třídu Plugin (pro snadnější práci), dále obsahuje proměnnou ChannelCount, která je inicializovaná na hodnotu 16, což je počet MIDI kanálů. Obsahuje také metodu Process, která je volaná pokaždé, když VST předává MIDI zprávy pluginu. Tyto zprávy předává

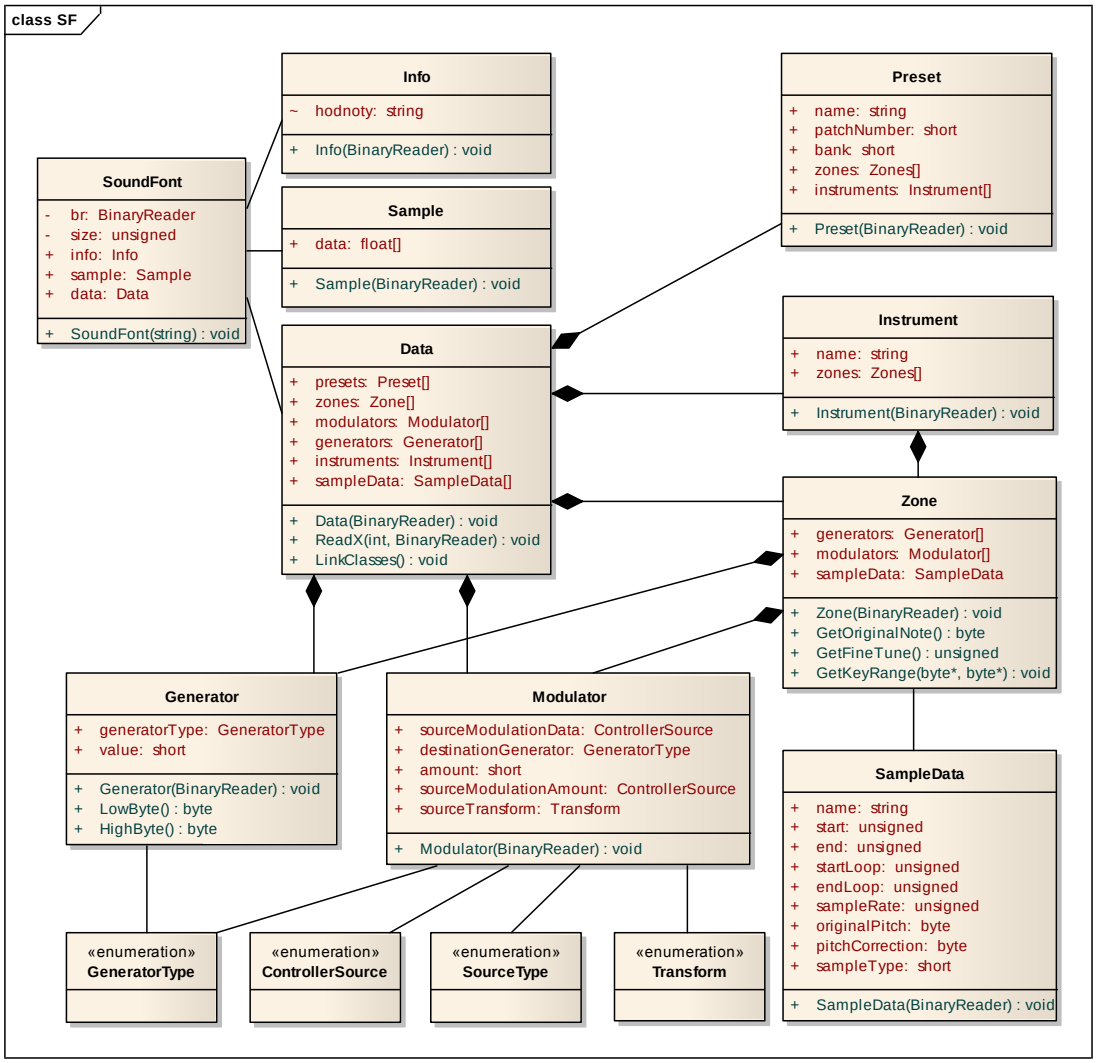
v argumentu typu VstEventCollection, což je kolekce událostí (VstEventCollection je interní třída VST .NET), ve kterých se nacházejí jednotlivé příchozí MIDI zprávy.

Třída AudioProcessor rovněž obsahuje privátní referenci na třídu Plugin, a dále proměnnou SampleRate, ve které je uložena velikost vzorkovací frekvence při práci s bufferem (typicky 44100). V proměnné InputCount je uložen počet vstupních kanálů (plugin nepracuje se vstupním bufferem, proto je hodnota nulová) a v proměnné OutputCount je uložen počet výstupních kanálů. Plugin pracuje se dvěma výstupními kanály (levý a pravý). Metoda Process řídí kopírovaní výstupu zvukové syntézy do výstupních bufferů.

Třída PluginEditor rovněž obsahuje privátní referenci na třídu Plugin. Podstatná datová struktura v této třídě se nazývá uiWrapper, což je reprezentace grafického rozhraní. Důležité metody třídy PluginEditor jsou:

- Open, která otevírá uživatelské rozhraní
- Close, která zavírá uživatelské rozhraní
- getSoundFontIndex, která vrací index uživatelem zvoleného načteného Sound Font 2 souboru
- třídy s předponou Ignore, které vrací hodnotu True, jestliže uživatel zakázal reakci na příslušný typ zprávy.

5.2. Třída SoundFont



Obrázek 18: Struktura třídy SoundFont

Úkolem třídy SoundFont je načíst soubor formátu Sound Font 2 do paměti a poskytnout pluginu odpovídající datovou sktrukturu pro snadnou práci s tímto formátem.

Načítání souboru do paměti je řešeno v konstruktorech jednotlivých tříd. Ve třídě SoundFont se nachází objekt BinaryReader, který postupně čte zvolený soubor. Instance tohoto objektu je postupně předávaná konstruktorům tříd Info, Sample a Data, které vždy přečtou požadovanou část souboru a předají instanci tohoto objektu dál. Přitom konstruktor třídy Data čtení souboru dále rozděluje do konstruktorů podřazených tříd (Preset, Zone Generator, Modulator, Instrument, SampleData).

Třída SoundFont se logicky dělí na tři části reprezentované třídami Info, Sample a Data.

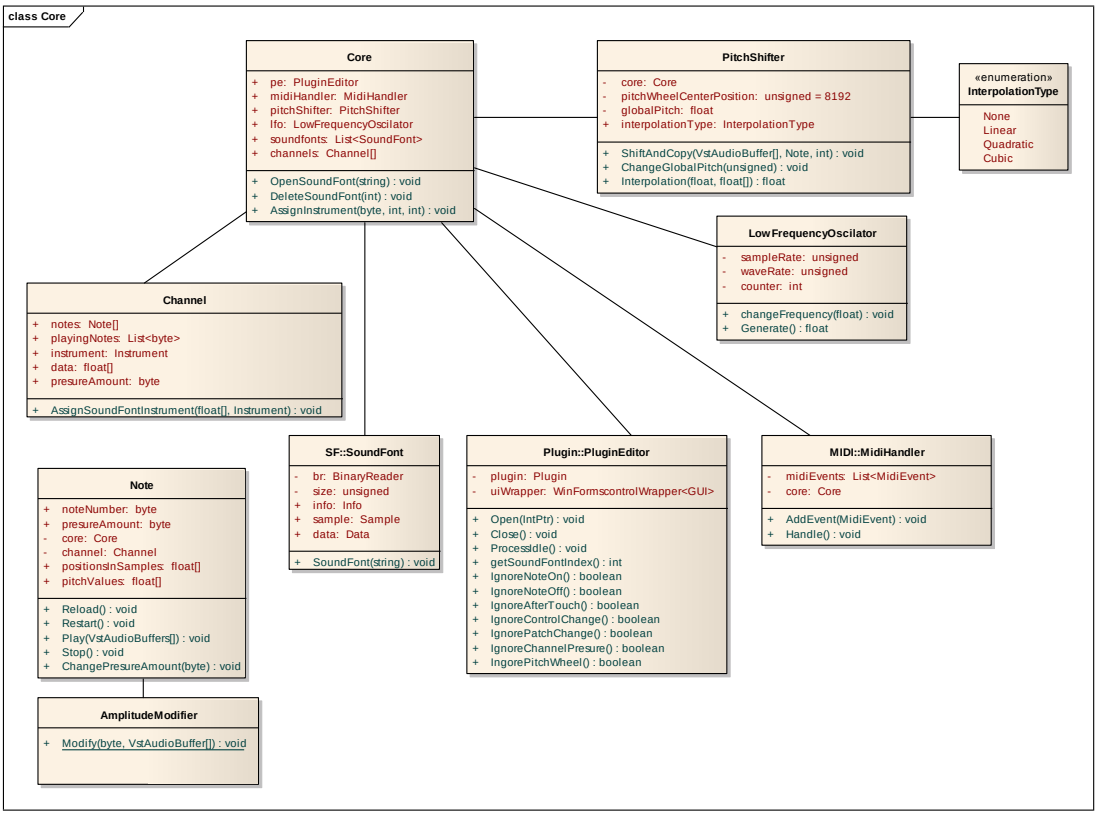
Třída Info obsahuje pouze několik proměnných typu řetězec, ve kterých jsou uloženy obecné informace o souboru (název, autor, použitý software, datum vytvoření, atd.)

Třída Sample obsahuje samotný vzorek. Tento vzorek je uložen v poli hodnot typu Float. Třída Sample je klíčová, protože obsahuje data, ze kterých bude čerpat zvuková syntéza.

Třída Data obsahuje informace popisující vzorek. Tato třída je rovněž velmi důležitá, protože dává datům ve vzorku hlubší význam. Protože tato třída čte nejsložitější část souboru, rozděluje čtení souboru do metod s předponou Read. Tyto metody zajišťují postupné volání konstruktorů tříd Preset, Zone, Modulator, Generator, Instrument a SampleData. Sémantika těchto tříd odpovídá rozložení formátu Sound Font 2 z kapitoly 3.6. Zajímává metoda v této třídě je metoda LinkClasses, které propojí všechny jmenované třídy dohromady. Ve formátu Sound Font 2 je toto propojení popsáno pomocí indexů. Metoda LinkClasses pomocí těchto indexů propojí třídy hierarchicky pro snadnější a rychlejší práci.

Jednotlivé třídy obsahují mnoho objektů s viditelností Public. To není z programátorského hlediska ideální, ale volil jsem tuto možnost kvůli rychlosti. Při testování jsem zjistil, že kdyby se k těmto objektům přistupovalo přes metody nebo vlastnosti (Properties), znatelně by vzrostla odezva programu, protože k těmto objektům se přistupuje velmi často.

5.3. Třída Core



Obrázek 19: Struktura třídy Core

Třída Core je nejdůležitější třída v pluginu. Tato třída obsahuje následující instance tříd:

- instanci třídy PluginEditor, která reprezentuje grafické rozhraní. Na tuto konkrétní instanci se odkazuje metoda CreateEditor ve třídě Plugin.
- instanci třídy MidiHandler, která zpracovává příchozí MIDI zprávy
- instanci třídy PitchShifter, která provádí posun výšky tónu vzorku
- instanci třídy LowFrequencyOscilator, což je pomocná třída využívaná při zvukové syntéze
- pole instancí tříd Channel (toto pole bude mít vždy velikost 16), které reprezentuje MIDI kanály
- seznam instancí tříd SoundFont, což jsou datové struktury načtených Sound Font 2 souborů

Kromě zmíněných objektů třída Core obsahuje tři metody, které slouží ke správě instancí tříd SoundFont.

- Metoda OpenSoundFont vytvoří novou instanci třídy SoundFont (načtením souboru, jehož název se nachází v argumentu metody) a přidá ji do seznamu soundFonts.
- Metoda DeleteSoundFont ze seznamu vymaže zvolenou instanci třídy SoundFont.
- Metoda AssignInstrument přiřadí zvolenému kanálu (instanci třídy Channel) nástroj (instance třídy Instrument) ze zvolené instance třídy SoundFont.

Třída Channel obsahuje pole 128 instancí třídy Note. Ty reprezentují 128 tónů na stupnici v MIDI kanálu. Třída Channel dále obsahuje seznam playingNotes, ve kterém je uložen seznam aktuálně přehrávaných tónů (tento seznam se uchovává kvůli větší rychlosti, i když by se přehrávané tóny daly zjistit jiným způsobem, např. průchodem všech 128 tónů a u každého zjistit, jestli je zrovna přehráván). Ve třídě se dále nachází objekt Instrument a související pole data. V objektu Instrument je přiřazený nástroj pro daný kanál a v poli data je uložen vzorek odpovídající nástroji (jde pouze o referenci do datové struktury SoundFont, takže tato položka nezabírá zbytečné místo navíc). Přiřazení nástroje se dá změnit metodou AssignSoundFontInstrument, která nastaví proměnné Instrument i data. Poslední objekt obsahující třída Channel je proměnná presureAmount, která ovlivňuje hlasitost přehrávání vzorků pro daný kanál (obdobná proměnná ještě existuje pro každý tón, protože nástroj může měnit sílu stisku jak pro klávesu, tak pro celý kanál).

Třída Note reprezentuje klávesu na stupnici. Nacházejí se v ní proměnné noteNumber (značící číslo tónu), presureAmount (v ní je uložena informace o síle stisku klávesy, obdoba proměnná presureAmount ve třídě Channel), core a channel jsou privátní reference na rodiče, kvůli snadnější práci se třídou Note. Dále obsahuje pole positionsInSamples, ve kterém jsou uložené pozice ve vzorcích. Protože přes každý tón může ležet více zón, a každá zóna má vlastní vzorek, při přehrávání vzorku je třeba uchovávat informace o pozici v každém z těchto vzorků. Ze stejného důvodu je v poli pitchValues uchovávaná výška tónu všech vzorků. Zatímco pole positionsInSamples je nezbytné pro správné přehrávání vzorku (souvisí s ním i realizace smyček), pole pitchValues se ve třídě nachází pouze kvůli větší rychlosti. Tato třída obsahuje čtyři zajímavé metody. Metoda Reload je zavolaná třídou Channel pokaždé, když se v ní změnil nástroj, a zajistí správné inicializování hodnot do svých proměnných.

Metoda Start začne přehrávat vzorek kopírovním dat do výstupního bufferu. Je v ní řešena i implementace smyček. Metoda Stop zajišťuje zastavení přehrávání vzorku a následným voláním metody Restart uvedením třídy do výchozího stavu.

Třida Note využívá třídu AmplitudeModifier, ve které se nachází jediná statická třída Modify, která má za úkol upravit amplitudu (přeneseně hlasitost) vzorku v závislosti na dalších MIDI zprávách (síla stisku klávesy, hlasitost, atd.)

Třída PitchShifter je skrze třídu Core (na kterou má uloženou privátní referenci) volána třídou Note při přehrávání vzorků. Její úkol je převést vzorek o definované výšce tónu na vzorek s výškou tónu požadovanou stisknutou klávesou. Je v ní uložen typ interpolace, který se používá při přepočtu vzorku, v proměnné globalPitch je uložena změna výšky tónu, která nesouvisí s klávesou (tato změna mohla nastat různými MIDI kontrolery, např. Pitch Wheel, modulation Wheel nebo vibrato) a při přepočtu výšky tónu je tato hodnota zohledněna. S kontrolerem Pitch Wheel souvisí proměnná pitchWheelCenterPosition, ve které je uložena hodnota reprezentující střední pozici tohoto kontroleru. Tento kontroler totiž může logicky nabývat i záporných hodnot (značící posun výšky směrem dolů), ale v MIDI zprávě se přenáší pouze kladná hodnota. Hodnota uložená v proměnné pitchWheelCenterPosition znamená nulovou polohu (tzn. žádný posun výšky) tohoto kontroleru. Kromě těchto objektů třída PitchShifter obsahuje:

- Přímočarou metodu ChangeGlobalPitch, která mění hodnotu proměnné globalPitch).
- Metodu Interpolation, která jako argumenty přijímá pole hodnot a pozici v tomto poli. Protože pozice v poli většinou není celočíselná (jedná se o posun v závislosti na frekvenci tónu, která ve většině případu není celočíselná), provede tato metoda interpolaci, jejíž typ závisí na hodnotě uložené v porměnné InterpolationType.
- Metoda ShiftAndCopy provádí překopírování dat do výstupního bufferu. Využívá přitom proces Pitch Shifting a metodu Interpolation.

5.4. Princip Pitch Shiftingu

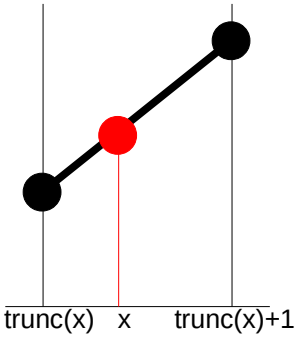
Tento proces se dá realizovat mnoha způsoby (phase vocoder, time domain harmonic scaling, atd.), ale přímo v definici formátu Sound Font 2 je princip Pitch Shiftingu popsán pomocí smyček a interpolace. Zjednodušeně se jedná o přehrávaní

vzorku různou rychlostí. Detailněji, používá se faktor, ve kterém je uložena změna frekvence (změna mezi frekvencí vzorku a fekvencí tónu klávesy). Vztah popisující výpočet faktoru je podobný, jako v kapitole 3.3.5 (přepočet čísla tónu na frekvenci tónu). V následujícím vzorci je y číslo tónu přehrávané klávesy a x je číslo tónu vzorku.

$$faktor = 2^{\frac{y-x}{12}}$$

Tento faktor je využívaný jako index při průchodu polem. Jestliže je číslo tónu klávasy větší, než číslo tónu vzorku, pak ja faktor větší než jedna. Naopak, jestli je číslo tónu klávesy menší, pak je faktor menší než jedna. Jesliže jsou obě čísla stejná, pak je faktor roven jedné.

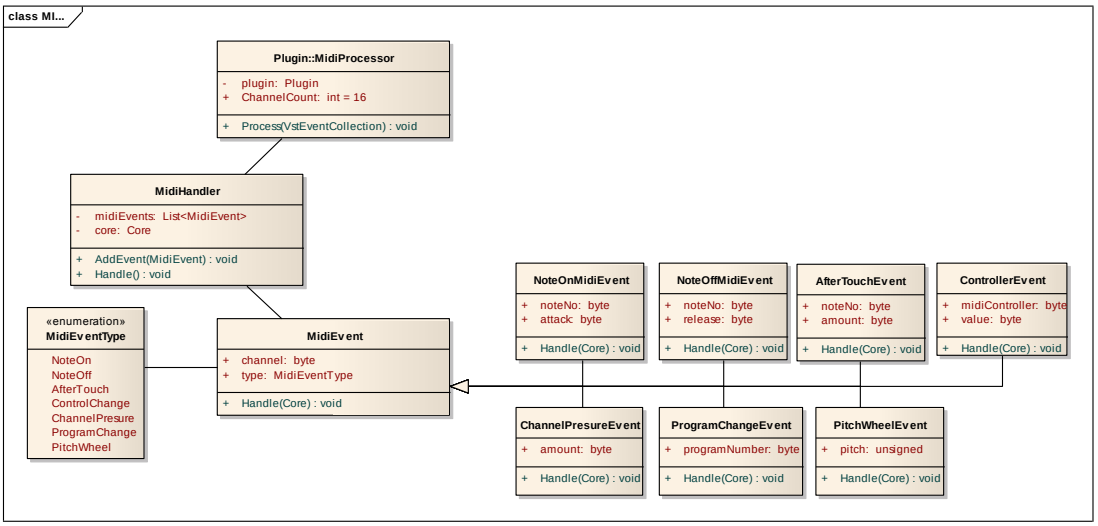
Při indexaci pole se samozřejmě musí indexovat celočíselnými hodnotami. K tomu se využívá interpolace (libovolného typu, nejčastěji lineární), která vypočte z pole hodnotu definovanou neceločíselným indexem.



Obrázek 20: Příklad lineární interpolace

Tím ale může nastat problém, že výsledný vzorek bude kratší. Tento problém řeší smyčky definované formátem Sound Font 2. Jestliže index přesáhne hodnotu definovanou, jako konec smyčky, pokračuje se dále od hodnoty indexu definované, jako začátek smyčky.

5.5. Zpracování MIDI



Obrázek 21: Struktura tříd zpracovávající MIDI zprávy

Zpracování MIDI zprávy začíná ve třídě MidiProcessor. Ta obdrží MIDI zprávu reprezentovanou objektem typu VstEvent. Třída MidiProcessor musí ověřit, jestli se skutečně jedná o MIDI zprávu a jestli ano, pak podle typu zprávy vytvoří instanci třídy reprezentující tento typ zprávy (tzn. třídy odvozené od MidiEvent). Tuto instanci přidá do třídy MidiHandler pomocí metody AddEvent.

Třída MidiHandler má v pluginu pouze jednu instanci a ta se nachází ve třídě Core. MidiProcessor se vždy odkazuje na tuto jednu konkrétní instanci. Třída MidiHandler má jediný úkol, a to zpracovat MIDI zprávy, které jí předala třída MidiProcessor. Jedná se o lineární průchod seznamem zpráv, při kterém je na každé zprávě (reprezentované instancí třídy odvozené od MidiEvent) zavolána metoda Handle, která obslouží daný typ zprávy a po provedení této metody instance třídy odvozené od MidiEvent zanikne.

Plugin umí obsloužit sedm typů MIDI zpráv, které jsou reprezentovány třídami NoteOnMidiEvent, NoteOffMidiEvent, AfterTouchEvent, ControllerEvent, ChannelPresureEvenet, ProgramChangeEvent a PithWheelEvent. Obsluha spočívá ve volání metody Handle na jednotlivých třídách. Tyto metody jsou definované následujícími způsoby.

5.5.1. NoteOnMidiEvent a NoteOffMidiEvent

Tyto zprávy se týkají konkrétní klávesy, proto budou pracovat s instancí třídy Note nacházející se v instanci třídy Channel (kanál určuje proměnná channel, klávesu určuje proměnná noteNo).

U zprávy NoteOnMidiEvent na odpovídající instanci třídy Note zavolá metodu Restart (uvedení třídy do výchozího stavu) a přidá číslo tónu do seznamu přehrávaných tónu ve třídě Channel. Vyvolání samotného kopírování vzorku do bufferu se provede až při volání funkce Process ve třídě AudioProcessor, které započne proces zvukové syntézy.

U zprávy NoteOffMidiEvent na odpovídající instanci třídy Note zavolá metodu Stop (zastavení přehrávání a následné uvedení do výchozího stavu třídy).

5.5.2. AfterTouchEvent a ChannelPresureEvent

Zpráva AfterTouchEvent představuje sílu úhozu na jednotlivou klávesu. Hodnota reprezentující sílu úhozu je uložena v proměnné amount. Reakce na tuto zprávu spočívá v zavolání metody ChangePresureAmount v odpovídající instanci třídy Note. Tato hodnota bude zohledněna při kopírování dat do bufferu ve zvukové syntéze.

Obdobná je zpráva ChannelPresureEvent, která se ale vztahuje na celý kanál a ne pouze na jednu klávesu. Při této zprávě se mění hodnota presureAmount v odpovídající instanci třídy Channel.

5.5.3. PitchWheelEvent

Tato zpráva má, narozdíl od ostatních typů zpráv, pouze jeden parametr. Reakce na tento typ zprávy spočívá v zavolání metody ChangeGlobalPitch v instanci třídy PitchShifter, které se jako argument předá hodnota této zprávy. Zavolaná metoda nastaví proměnnou v instanci třídy PitchShifter a tato hodnota bude ovlivňovat výslednou výšku tónu při pitch shiftingu.

5.5.4. ProgramChange

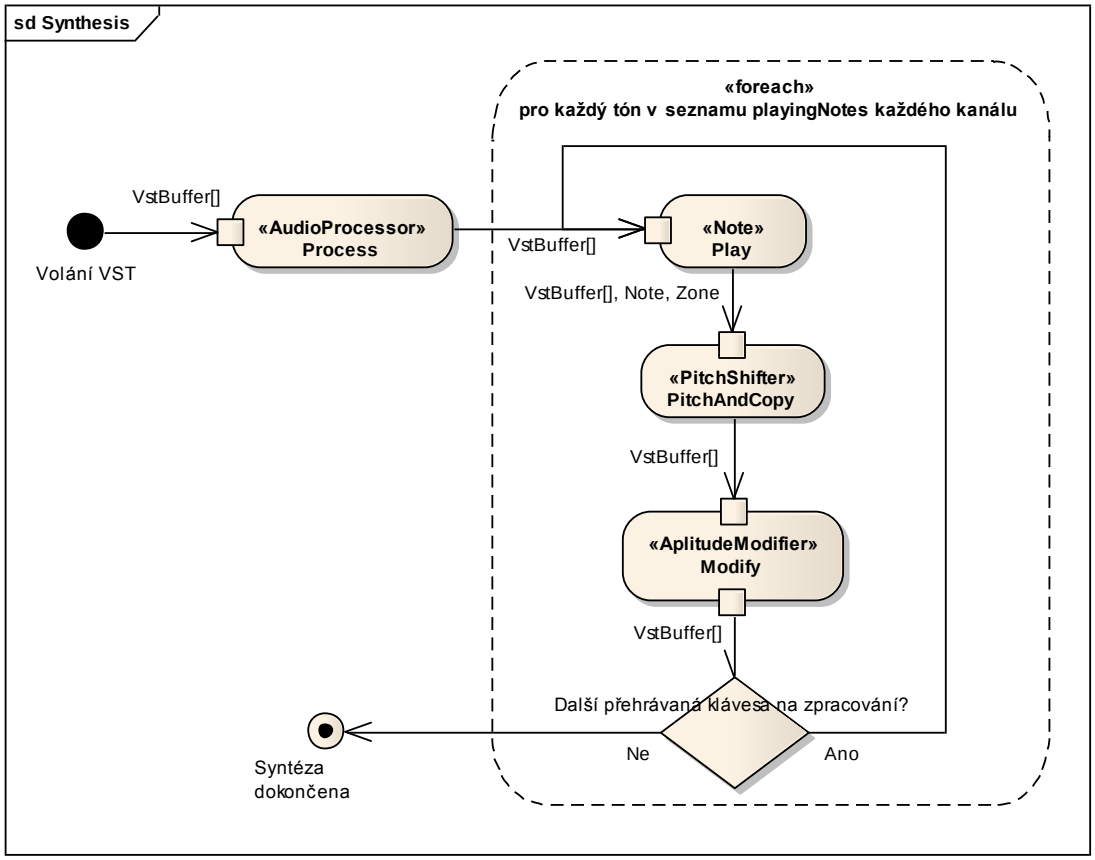
Tento typ zprávy danému kanálu mění zvolený nástroj. Reakce na tento typ zprávy spočívá ve volání metody AssignInstrument v odpovídající instanci třídy Channel. Zpráva je ignorována, jestliže je číslo programu (určené hodnotou

programNumber) větší, než je počet nástrojů ve zvolené třídě SoundFont (reaguje tedy pouze na čísla programů odpovídající indexům nástrojů ve načteném SoundFontu).

5.5.5. ControllerEvent

Plugin zatím reaguje pouze na kontroler s číslem 1, což je modulation Wheel. Tento kontroler je implementován pomocí LFO oscilátoru (Low-Frequency-Oscillator), který generuje signál o nízké frekvenci ovlivňující hodnotu globalPitch ve třídě PitchShifter. Reakce tedy spočívá ve volání metody changeFrequency na instanci třídy LowFrequencyOscillator nacházející se ve třídě Core. Velikost frekvence, určující parametr zprávy, není ve specifikaci MIDI určena jednoznačně. Má implementace je taková, že nulová hodnota (původní) znamená žádnou oscilaci (generování konstantního signálu), naopak maximální hodnota 127 znamená generování signálu oscilátorem o frekvenci 4 Hz.

5.6. Implementace zvukové syntézy



Obrázek 22: Schéma zvukové syntézy

Zvuková syntéza začíná voláním metody Process ve třídě AudioProcessor (toto volání uskutečňuje VST). Její argument je pole bufferů (v případě pluginu pole obsahuje dvě instance třídy VstBuffer, reprezentující levý a pravý výstupní kanál).

Metoda Process obsahuje dva vnořené cykly, přičemž vnější cyklus iteruje přes všechny kanály (v každém kanálu mohou být jiné seznamy přehrávaných tónů). Vnitřní cyklus iteruje přes přehrávané tóny v kanálu (seznam přehrávaných tónů se nachází v seznamu bajtů playingNotes nacházející se ve třídě Channel), kde je v každé iteraci volaná metoda Play v odpovídající instanci třídy Note.

Při provádění metody Play je prováděna iterace přes všechny zóny překrývající přehrávaný tón (tím se mísí více vzorků dohromady). Samotné nakopírování vzorku do bufferu provede metoda ShiftAndCopy ve třídě PitchShifter, která zároveň s kopírováním provádí Pitch Shifting (při průchodu vzorkem využívá již zmíněný faktor). Pitch Shifting závisí na dvou parametrech:

- rozdíl čísla tónu vzorku a čísla přehrávaného tónu
- celkový posun výšky na kanálu - globalPitch (změna výšky pomocí Pitch Wheel), aktuální hodnota LFO oscilátoru (změna výšky pomocí Modulation Wheel)

Tím je zvládnutá první část syntézy.

Druhá část syntézy zajišťuje metoda Modify nacházející se ve třídě AmplitudeModifier. Ta provede úpravu amplitudy signálu uloženého v bufferu, závisí na parametrech získaných z MIDI zpráv After Touch a Channel Presure.

Při každém průchodu jednoho kroku syntézy (tj. jedno zavolání metody Process) se vzorky ze všech zón a kanálů nahrávají do jednoho bufferu postupným přičítáním hodnot.

Tím je dokončeno předání dat do bufferu a proces zvukové syntézy.

5.7. Kompilace

Při kompilaci musí být vždy přítomny knihovny Jacobi.Vst.Interop.dll a Jacobi.Vst.Core.dll. V případě této práce ještě musí být navíc přítomna knihovna Jacobi.Vst.Framework.dll, protože práce využívá třídy uložené v této knihovně. Součástí pluginu musí být všechny čtyři uvedené knihovny.

Práce s VST .NET vyžaduje, aby knihovna Jacobi.Vst.Interop.dll byla přejmenovaná na název pluginu (tzn. tato knihovna musí být primární) a výstupní knihovna práce musí být uložena se stejným názvem s příponou .net.dll.

Příklad: Výstup z kompilace bude knihovna M-lizer.dll. Tato knihovna se musí přejmenovat na M-lizer.dll a knihovna Jacobi.Vst.Interop.dll se následně musí přejmenovat na M-lizer.dll. Toto je vhodné řešit přidáním post-build události do projektu.

6. Závěr

6.1. Přínos práce

Plugin využijí zejména uživatelé pracující s MIDI nástroji, především klávesami (existují například i MIDI snímače pro kytary, ale použití kláves je nejběžnější). Po jejich připojení k počítači mohou používat tento plugin jako zvukovou banku nástrojů. Možností ignorovat některé MIDI zprávy může uživatel dosáhnout zajímavých efektů a přizpůsobit se chování nástroje. Protože plugin podporuje technologii VST, může být použit spolu s dalšími pluginy (například zvukové efekty, ekvalizéry, a podobně) pro doladění a vylepšení výsledného zvuku.

6.2. Výhody

Největší výhodou je podpora technologie VST. Plugin mohou rozpoznat všechny aplikace podporující tuto technologii, takže se dá použít v mnoha neplacených i komerčních zvukových softwarech (včetně velkých studiových stanic). Důsledek podpory VST je možnost zapojení pluginu do většího řetězce dalších pluginů (ať už dalších nástrojů nebo efektů), které mohou dále zpracovávat a upravovat výstup tohoto pluginu.

Další výhodou je používání souborů formátu Sound Font 2, které velmi detailně a věrohodně popisují skutečně nástroje. Jestliže uživatel získá kvalitní soubor Sound Font 2, pak plugin dokáže vygenerovat velice kvalitní výstup.

Díky možnosti přiřazení více souborů různým MIDI kanálům získá uživatel dobrou kontrolu nad výsledným zvukem.

Poslední výhodou je možnost zakázat reakci na některé druhy MIDI zpráv. Zakázáním zprávy Note Off by uživatel mohl docílit zajímavého efektu například u zvuku klavíru. Zakázáním dalších typů zpráv může uživatel ovlivnit nepohodlné chování jeho nástroje. Jestli například nechce používat kontroler Pitch Wheel nebo nechce měnit program, který se předtím pracně navolil, může toto vypnout v nastavení pluginu.

6.3. Nevýhody

První mírnou nevýhodou je koncepce práce do podoby pluginu, protože uživatel potřebuje hostující aplikace, aby plugin načetl a spustil, ale ostatní výhody uvedené v práci převyšují tuto nevýhodu.

Druhou nevýhodou je občasná delší odezva při přehrávání tónů. Toto je pravděpodobně způsobeno prací Garbage collectoru. Může se stát, že plugin určitou dobu přehrává tóny správně, ale po nějaké době se začíná opožďovat. Kvůli tomu klesne použitelnost pluginu v praxi, protože krátká odezva je nezbytná.

6.4. Podobný software

K nalezení je početná skupina programů, tzv. SoundFont přehrávačů (SoundFont players), které dokáží otevřít a zpracovat soubory formátu Sound Font 2. Vyznačují se přítomností grafického virtuálního nástroje (většinou klávesy) v jejich uživatelském rozhraní. Tyto programy simulují nástroj s použitím načteného Sound Font 2 souboru. Některé aplikace mají samozřejmě funkce navíc (např. úprava souborů Sound Font 2, převod na jiný formát, atd.). Některé známé nebo zajímavé SoundFont přehrávače jsou např. Extreme Sample Converter, SoundFont Player, Terry West nebo Font!SF2.

Pro operační systém OS X je k dispozici plugin do zvukových aplikací s názvem SoundFont Synth. Funkčností je podobný mému pluginu, tedy slouží jako programový zvukový syntetizér využívající soubory formátu Sound Font 2, ale nepodporuje technologii VST.

Pro operační systém Windows a technologii VST jsem nalezl několik pluginů, které funkčností odpovídají mému pluginu. Tedy fungují jako programový zvukový syntetizér, který zpracovává MIDI vstup a při tom využívá zvukové vzorky uložené v souboru formátu Sound Font 2. Některé známé a zajímavé jsou: Font!, SF Synth 2, Safwan Soundfont Player, SafFron Soundfont Player, a další.

Implementací softwarového syntetizéru s použitím souborů Sound Font 2 je mnoho a uživatel má na výběr z mnoha alternativ zdarma. Jediná funkce, kterou jsem neviděl u žádných dalších implementací, je možnost ignorovat vybrané MIDI zprávy. Uživateli může vyhovovat, jestliže syntetizér bude ignorovat některé typy zpráv a

lépe si tak může nastavit jeho chování. U ostatních aplikací také není běžné přiřazení souboru Sound Font 2 k MIDI kanálu.

6.5. Návrhy na vylepšení

Důležité je plugin vylepšit tak, aby při práci nevznikaly žádné prodlevy při generování výstupu.

U ostatních pluginů je běžná přítomnost ekvalizéru, pomocí kterého může uživatel nastavit hlasitost různých frekvenčních pásem a výsledného zvuku. Plugin by mohl implementovat ekvalizér, i když nezbytné to není, protože se dá simulovat nahráním dalšího pluginu do aplikace, který ekvalizér zajistí.

Zajímavým vylepšením by byla možnost přiřadit více souborů Sound Font 2 jednomu kanálu. Výsledkem by byla kombinace více nástrojů pro zajímavější zvuk.

Za zmínku by stál návrh opustit vývoj pluginu pod platformou .NET a pokračovat ve vývoji pod nativním jazykem C++ za použití VST SDK API. Výhodou by byla větší rychlost pluginu způsobená přímou komunikací s VST voláním nativních funkcí. Jazyk C++ se navíc v současné době používá pro vývoj aplikací nebo pluginů podporujících technologii VST nejčastěji.

7. Reference

7.1. Seznam zdrojů a citací

1. http://cs.wikipedia.org/wiki/MIDI. [Online]

2. http://cs.wikipedia.org/wiki/Resource_Interchange_File_Format. [Online]

3. http://cs.wikipedia.org/wiki/SoundFont. [Online]

4. http://en.wikipedia.org/wiki/Sound_synthesis#Types_of_synthesis. [Online]

5. http://en.wikipedia.org/wiki/Virtual_Studio_Technology#Software. [Online]

6. http://en.wikipedia.org/wiki/Windows_Driver_Model#Criticism. [Online]

7. http://www.root.cz/clanky/rozhrani-midi-na-osobnich-pocitacich. [Online]

8. http://www.root.cz/clanky/rozhrani-midi-na-osobnich-pocitacich-ii. [Online]

9. http://home.roadrunner.com/~jgglatt/tech/midispec.htm. [Online]

10. http://en.wikibooks.org/wiki/Sound_Synthesis_Theory. [Online]

11. http://www.techno.cz/clanek/21526/pocitacova-hudba-3-zvukova-synteza-a-tvorba-samply-a-samplery. [Online]

7.2. Seznam obrázků

7.3. Seznam tabulek

8. Seznam použitých zkratek

A-D ..... Audio-digitální (například A-D převodník)

ADSR ..... Attack/Decay/Sustain/Release, různé částí v obálce ADSR

API ..... Application Programming Interface, rozhraní sloužící ke snadnějšímu programování a prací s nějakou platformou nebo aplikací

ASIO ..... Audio Stream Input/Output, speciální ovladače zvukových karet

GAC ..... Global Assembly Cache

GUI ..... Graphic User Interface, grafické uživatelské rozhraní

MIDI ..... Musical Instrument Digital Interface

RIFF ..... Resource Interchange File Format, souborový formát

SDK..... Software Development Kit

SF2 ..... Sound Font 2, souborový formát

VST ..... Virtual Studio Technology

VSTi ..... Virtual Studio Technology Instrument

9. Přílohy

9.1. Uživatelská dokumentace

9.1.1. Instalace

Instalace pluginu je jednoduchá a odpovídá klasické instalaci jiných VST pluginů. Provede se umístěním knihoven pluginu do adresáře hostující aplikace obsahující VST pluginy. Plugin se skládá ze čtyř knihoven:

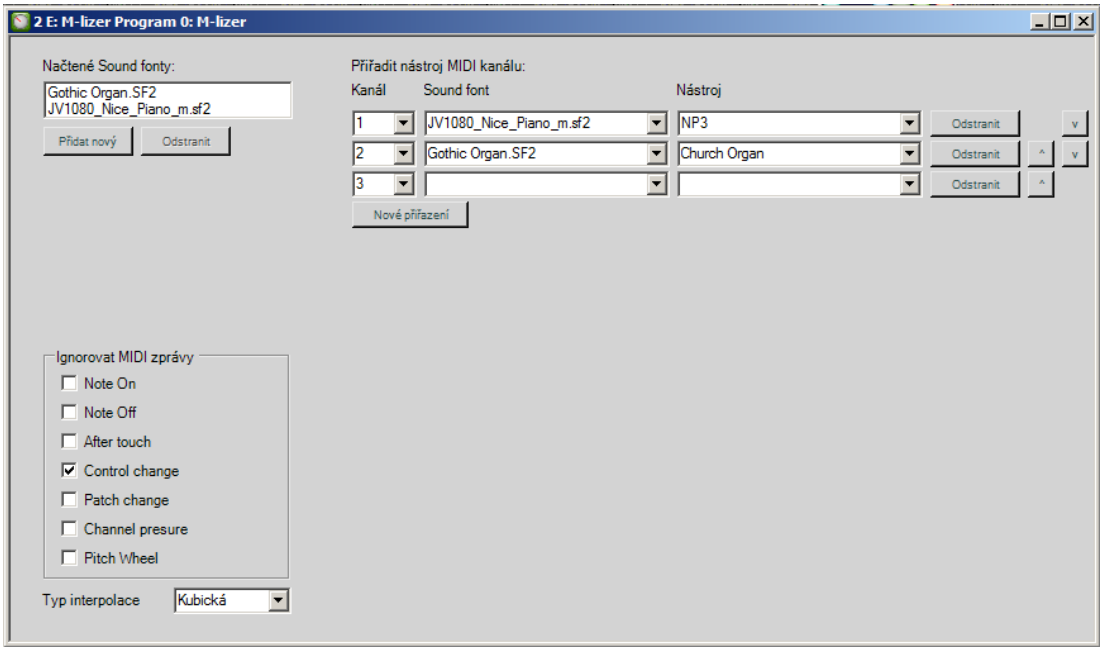
- M-lizer.dll
- M-lizer.net.dll
- Jacobi.Vst.Core.dll
- Jacobi.Vst.Framework.dll

Knihovny pluginu mohou, ale nemusejí být ve zvláštním adresáři. Důležité přitom je, aby se všechny knihovny pluginu nacházely ve stejném adresáři. Některé aplikace mají zvláštní adresář pro VST instrumenty, typicky se nazývá „Synth“, a v tom případě je nezbytné plugin umístit do tohoto adresáře.

9.1.2. Spuštění

Uživatel spustí hostující aplikaci, ve které načte plugin M-lizer (hierarchicky se nachází ve skupině VST Pluginy, podskupině „Synths“). V případě nutnosti je zapotřebí pluginu nastavit připojení MIDI vstupu a dvou výstupních audio kanálů, které využívají protokol ASIO. V posledním kroku uživatel otevře grafické rozhraní pluginu (liší se od použité aplikace).

9.1.3. Práce s pluginem



Obrázek 23: Grafické rozhraní pluginu

V levé horní části grafického rozhraní uživatel může do pluginu nahrát nový soubor formátu Sound Font 2 stiskem tlačítku „Přidat nový“ a následným vybráním souboru v otevřeném dialogu. Po načtení se zvolený soubor zobrazí v seznamu načtených souborů. Uživatel jej může z pluginu vymazat stiskem tlačítka „Odstranit“.

V pravé části grafické rozhraní uživatel může zvolenému kanálu přiřadit nástroj. Pomocí rozbalovacích nabídek uživatel vybere číslo kanálu, ke kterému chce přiřadit nástroj, poté vybere, z které načteného souboru chce vybírat nástroj, nakonec vybere samotný nástroj nacházející se ve zvoleném souboru. Po zvolení nástroje se ihned provede přiřazení nástroje kanálu. Toto přiřazení lze odstranit stiskem tlačítka „Odstranit“. Pro větší přehlednost může uživatel přeskupit svá přiřazení pomocí tlačítek „↑“ a „↓“. Pro vložení nového připojení uživatel musí stisknout tlačítko „Nové přiřazení“.

Po přiřazení prvního nástroje MIDI kanálu je plugin schopen pracovat.

Jestliže chce uživatel zakázet reakci na některý typ MIDI zprávy, může to uskutečnit zaškrtnutím příslušného políčka v části „Ignorovat MIDI zprávy“. Po zaškrnutí políčka se změny ihned projeví. Pro zrušení zákazu ignorace stačí políčka odškrnout.

Pomocí rozbalovací nabídky „Typ interpolace“ může uživatel zvolit typ interpolace, které bude použitá při přepočtu hodnot ve vzorcích.

9.2. Obsah přílohového CD

BP/BP.doc - Bakalářská práce ve formátu Microsoft Office 1997 - 2003

BP/BP.pdf - Bakalářská práce ve formátu PDF

Soundfont/* - několik volně šiřitelných souborů ve formátu Sound Font 2

VSTHost/ - hostující volně šiřitelná aplikace (licence GNU GPL) podporující VST

M-lizer/ - 4 knihovny pluginu nutné pro jeho spuštění

M-lizer (sources)/ - zdrojové soubory a projekt pluginu