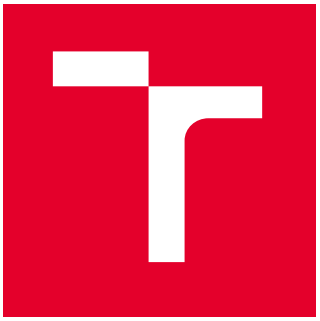


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

EXPERIMENTÁLNÍ SOFTWAROVÝ HUDEBNÍ NÁSTROJ, JEHOŽ ZVUK VZNIKÁ TRANSFORMACÍ VIDEO SOUBORU A JE ŘÍZEN KLAVIATUROU

EXPERIMENTAL SOFTWARE SYNTHESIZER WHOSE SOUND IS CREATED BY TRANSFORMING VIDEO FILE AND CONTROLLED BY A KEYBOARD

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **David Hromas**

AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. MgA. Mgr. Dan Dlouhý, Ph.D.**

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

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

Student: David Hromas ***ID:*** 231336
Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Experimentální softwarový hudební nástroj, jehož zvuk vzniká transformací video souboru a je řízen klaviaturou

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je realizovat plně funkční prototyp nástroje, navrženého v semestrální práci.

DOPORUČENÁ LITERATURA:

[1] PUCKETTE, M., Theory and Techniques of Electronic Music, 2006. 337 s. online: http://msp.ucsd.edu/techniques.htm

[2] FORRÓ, D., Svět MIDI. Grada, Praha, 1997. 375s. ISBN 80-7169-412-6.

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

Vedoucí práce: doc. Ing. MgA. Mgr. Dan Dlouhý, Ph.D.

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

Abstrakt

Práce se zabývá tvorbou experimentálního softwarového hudebního nástroje. Hlavním cílem je vytvoření hudebního nástroje, který bude schopen použít data z videa pro tvorbu zvuku. Projekt je tvořen ve vizuálním programovacím jazyku Max. V první části textu je probrána teorie, která je potřebná k vytvoření takového nástroje. Druhá část textu se věnuje návrhu jednotlivých částí nástroje, realizaci a hudebním možnostem nástroje.

Klíčová slova

aditivní syntéza, granulární syntéza, Max/MSP/Jitter, MIDI, sampler, sekvencér, syntezátor, video

Abstract

This work deals with the creation of an experimental software musical instrument. The main goal is to create a musical instrument that will be able to use video data to create sound. The project is created in the visual programming language Max. The first part of the text discusses the theory needed to create such an instrument. The second part of the text is devoted to the design of individual parts of the instrument, the realization, and the musical possibilities of the instrument.

Keywords

additive synthesis, granular synthesis, Max/MSP/Jitter, MIDI, sampler, sequencer, synthesizer, video

Bibliografická citace

HROMAS, David. *Experimentální softwarový hudební nástroj, jehož zvuk vzniká transformací video souboru a je řízen klaviaturou* [online]. Brno, 2023 [cit. 2023-12-05]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/151122. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. MgA. Mgr. Dan Dlouhý, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*David Hromas*
VUT ID studenta:	*231336*
Typ práce:	*Bakalářská práce*
Akademický rok:	*2022/23*
Téma závěrečné práce:	*Experimentální softwarový hudební nástroj, jehož zvuk vzniká transformací video souboru a je řízen klaviaturou*

Prohlašuji, že svou bakalářskou práci na téma „Experimentální softwarový hudební nástroj, jehož zvuk vzniká transformací video souboru a je řízen klaviaturou“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: …… ----------

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce doc. Ing. MgA. Mgr. Danu Dlouhému, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: …… ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

Úvod

Pokusy o tvorbu zvuku transformací vizuálních dat probíhají již přes sto let. Projektů, které se tímto typem syntézy zabývají, však není mnoho, v porovnání s projekty zaměřenými na jiné typy syntézy. Cílem této práce je navržení dvou softwarových hudebních nástrojů, jejichž vstupem je obrazový signál, konkrétně video, výstupem je zvuk. Bude probíhat snaha o vytvoření softwaru, jenž se svými kvalitami odlišuje od ostatních nástrojů, které pracují na stejném, nebo podobném principu.

Projekt bude navržen v prostředí pro vizuální programování *Max*. To umožňuje pokročilé možnosti práce s audiem i videem, pomocí objektů navržených pro zpracování těchto signálů. V teoretickém úvodu je probráno digitální zpracování zvukového signálu, syntéza zvuku, kapitola zaměřená speciálně na syntézu zvuku z obrazu, *MIDI* a program *Max*. Dále text obsahuje návrh konkrétních částí nástrojů, návrh kompletních nástrojů a zhodnocení jejich využití při kompozici hudby.

1. DIGITÁLNÍ ZPRACOVÁNÍ ZVUKOVÉHO SIGNÁLU

V kapitole jsou shrnuty obecné charakteristiky zvuku, jeho digitální reprezentace a převod z analogového signálu na digitální. Dále jsou popsány postupy digitálního procesování signálu, které budou použity v realizaci.

1.1 Zvuk

Jedná se o mechanické vlnění o frekvencích, které jsou postřehnutelné člověkem, tedy přibližně mezi 20 Hz a 20 kHz, nejvýrazněji mezi 2–4 kHz. Frekvence mimo slyšitelné pásmo se nazývají infrazvuk (pod 20 Hz) a ultrazvuk (nad 20 kHz). Zvuk se šíří ve vzduchu rychlostí přibližně 340 m/s. Rychlost nejvíce závisí na teplotě vzduchu. Zvuk může tvořit jakékoliv těleso, které je schopné buď periodicky kmitat, v tom případě vzniká tón, nebo vytvářet nahodilé mechanické vlnění, tak vzniká hluk. Technických parametrů zvuku je mnoho a dají se znázornit pomocí mnoha analogových (např. VU metr) a digitálních (např. spektrogram) nástrojů. Ve snaze o přiblížení parametrů zvuku lidskému vnímání vznikly jednotky jako *mel* (výška tónu), *fon* (hlasitost), nebo *son* (hlasitost). Věda zabývající se zvukem se nazývá akustika. [15]

1.2 Zvukový signál

Jedná se o reprezentaci zvuku ve spojité, nebo v diskrétní podobě. Ve spojité podobě jde o analogový signál, v diskrétní se jedná o digitální signál. Analogový signál je zastoupen změnou napětí, nebo proudu, v čase. U digitálního signálu jsou k popisu použity seřazené vzorky, zastoupené číslem.

Důležitým algoritmem pro zpracování digitálního signálu je Fourierova transformace, konkrétně FFT, což je algoritmus pro rychlý výpočet diskrétní Fourierovy transformace. Díky tomuto algoritmu se dá z časového průběhu signálu vypočítat jeho spektrum a naopak (IFFT). [1][3]

1.3 AD/DA převod

AD převod je převod signálu z analogového na digitální. Proces začíná vzorkováním signálu, kdy je na vstup převodníku přiveden analogový signál, z nějž se v dané frekvenci snímají vzorky. Tato frekvence se nazývá vzorkovací. Na vzorkovací frekvenci je závislá kvalita reprodukovaného signálu. Vybírá se podle Nyquistova teorému, který říká, že pro věrnou digitální reprodukci analogového signálu je třeba zvolit frekvenci alespoň dvakrát vyšší, než je nejvyšší frekvence daného signálu.

V případě, že není teorém dodržen, dochází k aliasingu, tedy poškození signálu. Aliasingu se dá zabránit použitím správné vzorkovací frekvence a antialiasingového

filtru typu dolní propust. Ten propouští pouze frekvence nižší, než je polovina vzorkovací frekvence.

Pro audiosignály se používají vzorkovací frekvence v rozsahu 44,1 kHz až 192 kHz. Při použití nižších frekvencí není možné věrně zreprodukovat slyšitelné pásmo frekvencí. Následně dochází ke kvantování, při kterém je navzorkovaná hodnota přiřazená k nejbližší možné úrovni, kterou je přistroj, do nějž je přiváděn signál, schopný vyjádřit. Počet těchto úrovní závisí na počtu bitů, které v přístroji reprezentují jeden vzorek signálu (bitové rozlišení). Standardně jde o 16–32 bitů, s pevnou řádovou čárkou (*fixed point*), nebo s pohyblivou řádovou čárkou (*floating point*). Tedy, například v případě 24bitového s pevnou řádovou čárkou jde o 2^{24} hodnot v rozmezí -1 až 1, ke kterým lze navzorkovanou hodnotu přiřadit. Při kvantování vzniká kvantizační šum, jehož odstup od signálu v decibelech, známy také jako *SNR*, přímo závisí na bitovém rozlišení, podle vzorce:

$$6{,}02 \ * \ N = SNR \ [dB], \tag{1.1}$$

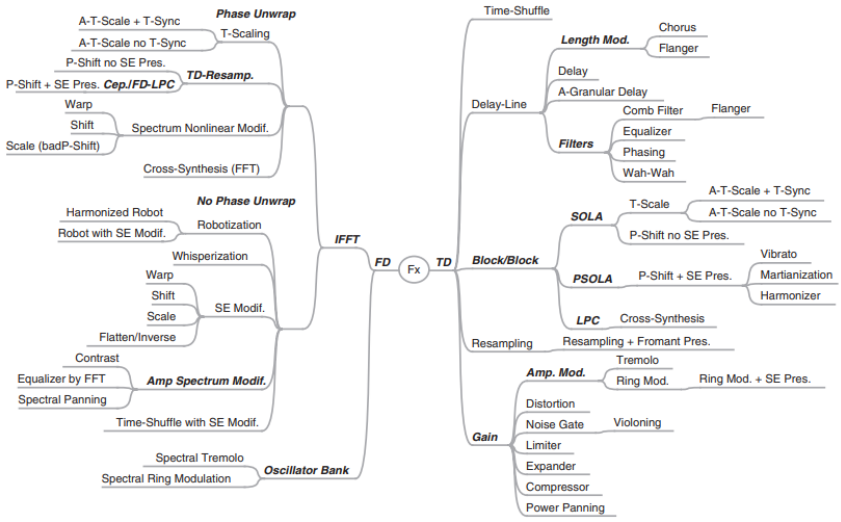
kde N je počet bitů.

DA převod má za cíl převést digitální signál na analogový. Při tvoření softwarového nástroje se setkáme prakticky pouze s DA převodem, protože je zvuk v počítači tvořen jako diskrétní signál. Základními parametry DA převodníku jsou vzorkovací frekvence, bitové rozlišení, dynamický rozsah a *THD*. [1][16]

1.4 Digitální procesování signálu

Audio procesorem se rozumí modul, který přijímá vstupní signál a na výstupu ho nějakým způsobem mění. Digitální zpracování signálů umožňuje efektivní práci jak s časovou, tak se spektrální doménou signálu, tedy jsou digitální efekty o něco všestrannější než analogové.

Zvukové efekty se dělí do několika kategorií. Ty mají za úkol popis způsobu, jakým je signál proměňován. Pro účely práce jsou v kapitole probrány pouze efekty, použité při realizaci.



Obr. č. 1.1: Jedno z možných rozdělení digitálních zvukových efektů [1]

Nástroj bude obsahovat několik, velmi často užívaných, efektových modulů, které snad přispějí k zajímavějšímu zabarvení zvuku. Těmito efekty budou Waveshaper, Frequency shifter, Reverb, Delay, Chorus a Filtr. Dále bude využit na výstupu efekt typu Limiter. [1][3]

1.4.1 Digitální filtr

Filtr je jedním ze základních prvků každého syntezátoru. Jeho cílem je signál filtrovat, tedy z něj ubrat, nebo zesílit některou část jeho spektra. Filtry se, podle toho, co vykonávají, dělí na dolní propust, horní propust, pásmovou propust, pásmovou zádrž a fázovací článek (ten má za úkol pouze měnit fázi signálu, nikoliv jeho spektrum). Hlavním parametrem horní a dolní propusti je odřezávací (cutoff) frekvence. Ta určuje, pod jakou, nebo nad jakou frekvencí bude signál zeslaben.

Dále je důležitým parametrem těchto filtrů rezonance, jenž určuje, jak moc bude zesílen odřezávací kmitočet. Posledním důležitým parametrem je strmost filtru. Ta popisuje, o kolik decibelů na dekádu (nebo na oktávu) bude signál po cutoff frekvenci zeslaben. Parametry pásmové propusti jsou dva postranní odřezávací kmitočty, mezi nimiž je u propusti zvuk propouštěn a mimo ně zeslabován. U zádrže je proces opačný.

Digitálně se filtry realizují pomocí zpožďovací linky a dělí se na *FIR* a *IIR* dle konečnosti impulzní odezvy filtru. *FIR* filtr neobsahuje zpožďovací linky, je stabilní, ale vyžaduje vyšší výpočetní výkon. [1][3][16]

1.4.2 Digitální delay

Delay je efekt, který (potom, co jím projde signál), přehraje signál znovu se zpožděním a opakovaně, v závislosti na nastavení. Digitálně se tento efekt realizuje jednou, nebo několika zpožďovacími linkami. Výstupní signál bývá přiveden zpět na vstup. Jeho

vynásobením číslem mezi 0 a 1 získáme zpětnou vazbu, čímž lze docílit opakovaného přehrání zpožděného signálu. [1][3]

1.4.3 Digitální reverb

Reverb dodává zvuku doznění. Skládá se z prvotních odrazů (zřetelně slyšitelné) a difuze (nekonkrétní část odezvy). V digitální podobě je reverb tvořen mnoha zpožďovacími linkami, jejichž úkolem je simulace různých částí, dle zadaného nastavení. Dále je možné efekt realizovat pomocí konvoluce, kdy je signál přiveden na impulsní odezvu, například konkrétního prostoru, a je provedena konvoluce těchto signálů.

Hlavními parametry reverbu jsou: nastavení dry/wet (potenciometr upravující poměr původního a procesovaného signálu), pre-delay (určuje dobu, za jakou lze slyšet prvotní odrazy), doba doznívání efektu a nastavení hlasitosti prvotních odrazů a difuze. [1][3][16]

1.4.4 Digitální chorus

U efektu typu chorus dochází ke zpoždění signálu a fázovým změnám, které se projevují jak v barvě signálu, tak ve změně stereo báze signálu. Efekt se realizuje pomocí zpožďovací linky. Ta signál zpozdí přibližně o 50–70 milisekund. Délka tohoto zpoždění je však ještě modulována nízkofrekvenčním oscilátorem. Poté je přímý signál smíchán s upraveným signálem, čímž vzniká barva zvuku, typická pro tento efekt. [1][3]

1.4.5 Digitální limiter

Limiter provádí kompresi signálu s kompresním poměrem ∞:1. To znamená, že jakýkoliv signál, který by překročil hranici specifikovanou pomocí parametru *ceiling*, je zeslaben tak, že tuto hranici nepřekročí. Digitální limitery se dělí na klasické, které pracují s hodnotou vzorků signálu, a *true peak* limitery.

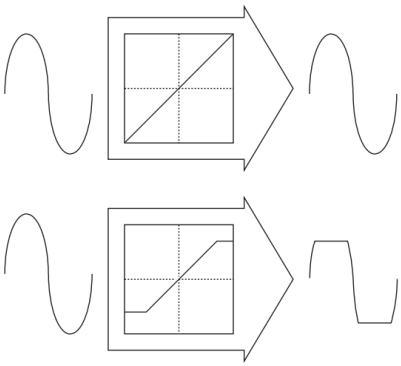
True peak limitery odhadují nechtěné překročení hodnoty *ceiling* i mezi vzorky signálu. Obvyklými nastavitelnými parametry limiteru jsou *ceiling*, a vstupní zesílení. U digitálních limiterů lze ještě ovlivnit parametr *lookahead*, který umožňuje sledování signálu těsně před tím, než bude procesován. [1][2]

1.4.6 Digitální zkreslení

Zkreslení vzniká při průchodu signálu nelineárním obvodem. Po průchodu takovým obvodem dochází ke změně tvaru časového průběhu signálu, tedy i k obohacení jeho spektra.

Digitálně je zkreslení realizováno převodní funkcí, což je nelineární rovnice, nebo soustava rovnic, z nichž každá ovlivňuje pouze část rozsahu vstupního signálu. Klasickým nastavitelným parametrem bývá *gain*, který upravuje zesílení vstupního

signálu. S rostoucí hodnotou prvku *gain* většinou roste zkreslení signálu na výstupu efektu. [1][3]



Obr. č. 1.2: Ukázka vlivu nelineární převodní funkce na výstupní signál [3]

1.4.7 Digitální frequency shifting

Jedná se o efekt, který posouvá všechny složky spektra o zadanou frekvenci, čímž je dosáhnuto změny v poměrech mezi složkami (v kontrastu s efektem nazývaným pitch shifter, který tyto poměry zachovává). Digitálně lze efekt realizovat v časové i ve spektrální doméně.

V časové doméně jde o kruhovou modulaci, u níž je využito pouze jedno postranní pásmo. Zvolené pásmo určuje, zdali jsou frekvence posouvány nahoru, nebo dolu od původní frekvence. [1][12]

Ve spektrální doméně jde o posun FFT binů o zadanou hodnotu. [12]

2. SYNTÉZA ZVUKU

V nástroji, který je konstruován, probíhá tvorba zvuku pomocí základních způsobů syntézy. Ty budou zmíněny v této kapitole. Dále se kapitola věnuje digitální reprezentaci prvků syntezátoru.

2.1 Syntéza

V obecném slova smyslu je pojem syntéza používán pro proces stavby celku koordinovanou kombinací jednodušších částí. V hudbě jde konkrétně o kreativní způsob elektroakustické tvorby barvy zvuku.

Existuje mnoho typů syntézy zvuku a několik podstatných rozdělení techniky, která je k tomuto způsobu tvorby zvuku používána. Základem je dělení na analogové a digitální nástroje, přičemž analogové nástroje používají pro syntézu zcela analogové obvody (zpracovává spojitý signál), zatímco digitální pracují s číslicovou reprezentací signálu. Dále je ještě možné oba postupy kombinovat a vytvořit nástroj, kombinující digitální a analogové obvody (např. digitální oscilátor v kombinaci s analogovým filtrem).

Nástroj, který tvoří zvuk pomocí syntézy, se nazývá syntezátor. Klasický syntezátor by měl obsahovat několik bloků (obvodů), sloužících k tvorbě a k úpravě zvuku. Jedná se o generační obvody (VCO), jejichž úkolem je zvuk generovat, kontrolní obvody (např. klaviatura, VCA, VCF), které slouží k ovládání hraného zvuku a modulační obvody (např. EG, LFO). Ty upravují signál generovaný oscilátorem. Základními typy syntézy jsou aditivní, subtraktivní a modulační syntézy (AM, RM, FM). Tyto typy syntézy je možné realizovat pomocí analogového i digitálního syntezátoru. Ostatní typy syntéz se dají analogově realizovat jen obtížně, nebo vůbec (granulární, vektorová a tvarová syntéza). Zbytek této kapitoly se bude zabývat digitální syntézou. [3]

2.2 Digitální syntéza a softwarové syntezátory

Digitální syntéza pracuje s diskrétní reprezentací signálu. Generování signálu probíhá na základě matematických operací se vzorky signálu. Například pro generování sinusového průběhu o dané frekvenci F s danou amplitudou A a daným vzorkovacím kmitočtem F_{vz} je potřeba tato čísla použít ve vzorci:

$$A * \sin(\acute{u}hel) = vzorek\ sign\acute{a}lu, \tag{2.1}$$

kde proměnná *úhel* závisí na proměnné *fázový posun*. *Fázový posun* se sčítá ze vzorce

$$(2 * \pi * F) / F_{vz} = f\acute{a}zov\acute{y}\ posun. \tag{2.2}$$

Ukázku těchto výpočtů v programovacím jazyku *C++* v programu *Visual Studio*:

```
Oscilator(float freq, float amp) : frequency(freq), amplitude(amp) {
    offset += 2 * PI * frequency / sampleRate;

}
```

Obr. č. 2.1: Vzorec pro fázový posun, vstupem jsou amplituda a frekvence

```
float sineProcess() {
    auto float sample = amplitude * sin(_Xx:angle);
    angle += offset;
    return sample;

}
```

Obr. č. 2.2: Vzorec pro navrácení hodnoty konkrétního vzorku sinusového průběhu v čase

Kvalita takto vypočítaných průběhů závisí na několika faktorech (viz kapitola *Digitální zpracování zvukového signálu*). Díky matematickým operacím se vzorky lze také signál procesovat pomocí filtrů a standardních zvukových efektů. Dají se tak simulovat i součásti analogového syntezátoru (např. LFO, EG).

Digitální syntezátory začaly vznikat v sedmdesátých letech minulého století jako samostatné nástroje. S rychlým vývojem výpočetní techniky však v devadesátých letech přišly softwarové syntezátory, které umožňovaly tvorbu zvuku pomocí osobního počítače. V té době také firma *Steinberg* představila standard VST, jenž měl za cíl tyto nástroje integrovat do DAW. Dnes existuje mnoho VST nástrojů pro tvorbu zvuku, které využívají jeden nebo více druhů syntéz. Některé se snaží věrně napodobovat starší analogové a digitální syntezátory (např. *Dexed*, simulace syntezátoru *Yamaha DX7*), jiné nabízí široké možnosti v oblasti syntézy (např. *Serum* a *Vital*). [3][16]

2.2.1 Aditivní syntéza

Cílem aditivní syntézy je poskládání požadované barvy zvuku pomocí součtu sinusových průběhů. K této syntéze je využíváno několik paralelně zapojených oscilátorů, z nichž každý generuje sinusový průběh. Tyto průběhy se dále sčítají a tvoří celkovou barvu. Signál je poté možno ovlivnit pomocí obvodů zmíněných v kapitole *Syntéza*.

Teoreticky je možné pomocí aditivní syntézy napodobit jakýkoliv zvuk, v praxi je to však těžší. Známým příkladem nástroje, který pracuje na tomto principu, jsou *Hammondovy varhany*. [3]

2.2.2 Subtraktivní syntéza

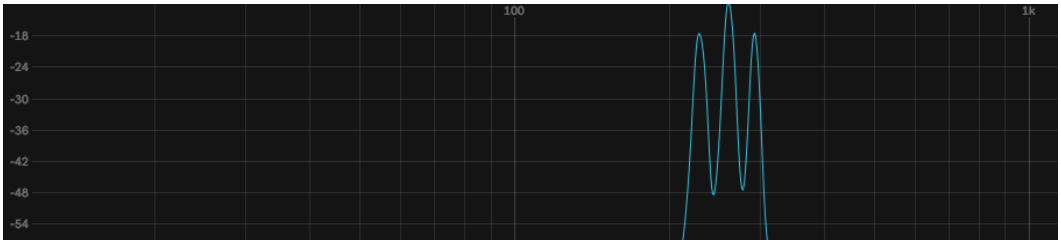
Tento typ syntézy používá signál, širokospektrální generátor, ze kterého jsou poté, pomocí různých typů filtrů, tvořeny průběhy (např. pila, obdélník) s různými poměry harmonických složek. Typů filtru je mnoho (viz kapitola *Digitální filtr*), použitím filtrů a nastavováním jejich parametrů lze dosáhnout změny průběhu signálu (např. šířka

střídy, frekvence) a tím i požadovaného zvuku.

S tímto principem syntézy přišly jako první nástroje značek *Moog* a *Buchla*. Dnes se jedná o jeden z nejrozšířenějších typů syntézy zvuku. [3]

2.2.3 AM syntéza

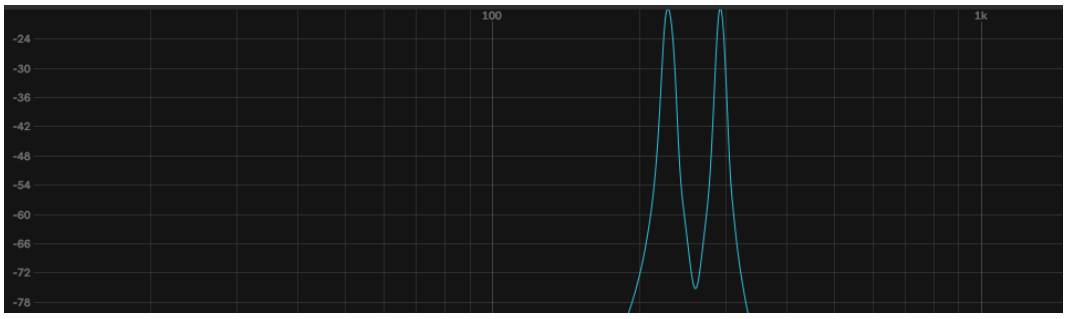
U AM syntézy dochází k amplitudové modulaci signálu. Dříve se princip amplitudové modulace používal pro přenos rádiového signálu. Jde o kombinaci dvou signálů, z nichž jeden je modulační a druhý nosný. Amplituda jednoho oscilátoru tedy moduluje amplitudu druhého. Při velmi nízkých frekvencích připomíná tato metoda modulace efekt tremolo (periodicky se opakující změna hlasitosti), při vyšších frekvencích si lze všimnout, že se kolem hlavní frekvence, na které je nosný signál, tvoří také dvě postranní pásma (součtové a rozdílové). Jejich frekvence je dána kmitočtem modulačního signálu. [1][3]



Obr. č. 2.3: Amplitudová modulace, nosná frekvence 261 Hz (sinus), modulační frekvence 32 Hz (sinus), postranní pásma 229 Hz a 293 Hz, plugin *Spectrum* v programu *Ableton*

2.2.4 RM syntéza

RM je druhem AM syntézy, v němž je potlačena nosná i modulační frekvence. Ve spektru jsou tím pádem přítomné pouze postranní frekvence. Tímto způsobem je možné vytvořit velmi specifické barvy zvuku. Syntéza se jmenuje podle analogového obvodu, používaného k její realizaci (Ring modulation/Kruhová modulace). [1][3]

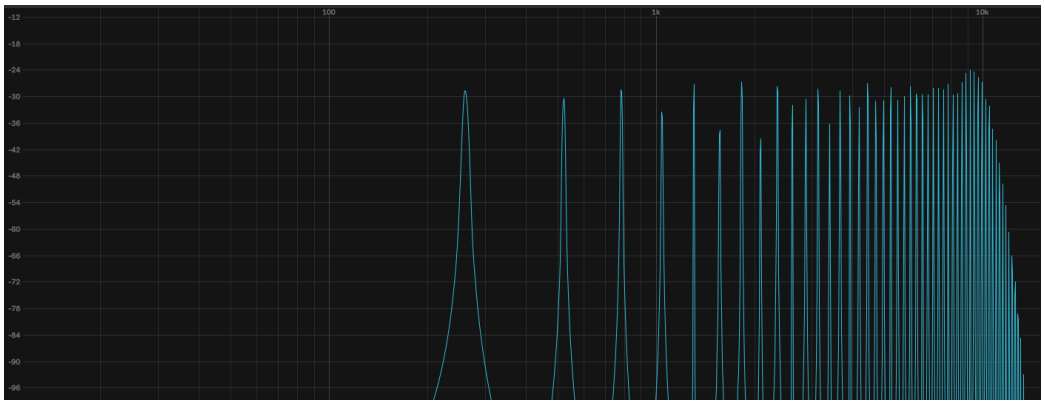


Obr. č. 2.4: Kruhová modulace, nosná frekvence 261 Hz (sinus), modulační frekvence 32 Hz (sinus), postranní pásma 229 Hz a 293 Hz, plugin *Spectrum* v programu *Ableton*

2.2.5 FM syntéza

FM syntéza zvuku byla matematicky popsána a patentována Johnem Chowningem v roce 1973. Dříve se frekvenční modulace používala pro přenos rádiového signálu. Jedná se o syntézu zvuku, v níž se kombinují dva a více oscilátorů, které se navzájem frekvenčně modulují, v nastaveném pořadí. To vytváří, při nižších frekvencích modulačního signálu, slyšitelný efekt typu vibrato (změna výšky tónu). Při vyšších frekvencích je produkován, dalo by se říct, kovový zvuk, typický pro tento typ syntézy. Takový zvuk má mnoho, převážně neharmonických, složek. Frekvenční modulace také produkuje postranní pásma, která se dají vypočítat. Na rozdíl od přechozích způsobů modulace je však těchto pásem mnoho, v závislosti na průbězích modulačního a nosného signálu.

Nejznámějším nástrojem, který využíval tento typ syntézy, je *Yamaha DX7*. Ačkoliv se tento způsob syntézy dá vytvořit i v rámci analogového obvodu, jeho digitální implementace je stabilnější, používá se častěji a nabízí více možností co se týče počtu oscilátorů a jejich uspořádání. [1][3]



Obr. č. 2.5: Frekvenční modulace, nosná frekvence 261 Hz (sinus), modulační frekvence 261 Hz (sinus), plugin *Spectrum* v programu *Ableton*

2.2.6 Vektorová syntéza

Vektorová syntéza spočívá v míchání, klasicky, čtyř zvuků na osách X a Y. Pohybem, který probíhá nejčastěji pomocí joysticku, dochází k zesilování a zeslabování čtyř vektorů hlasitosti, z nichž každý je umístěn v jednom krajním bodě os X a Y. Prvním syntezátorem, který nabízel tuto možnost kontroly zvuku, byl *Prophet VS*. [1][3]

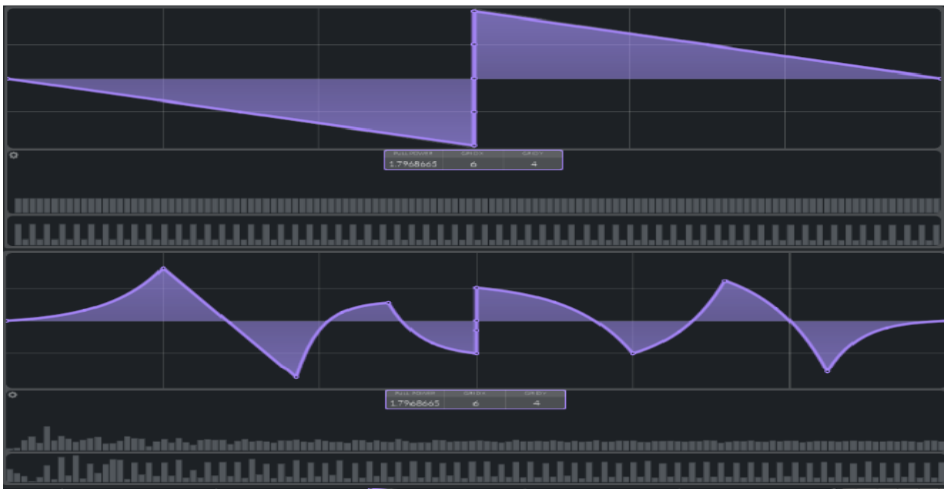
2.2.7 Granulární syntéza

Jedná se o typ syntézy, který bere jako základní jednotku tzv. zrno (*grain*). To je několik milisekund až několik stovek milisekund dlouhý zvukový vzorek, většinou odebíraný z delšího zvukového vzorku (samplu). Vzorky jsou poté umisťovány za sebe a do prostoru, s různými amplitudovými obálkami a délkami, čímž tvoří zvuk. Lze také

volit, z jaké části původního samplu jsou vzorky odebírány. Tento způsob syntézy je dnes realizován výhradně pomocí softwarových nástrojů nebo digitálních syntezátorů. [3]

2.2.8 Tvarová syntéza

Jedná se o techniky, které přetvářejí signál v časové doméně a ovládají tak tvar jeho průběhu. Mezi tvarovou syntézu se dá zařadit nelineární tvarování, fázové zkreslení a zadávání časového průběhu, což lze provést buď automatizací parametru již existujícího signálu nebo pomocí softwarových nástrojů, které umožňují kreslit hratelné periody zvukových průběhů. U nelineárního tvarování signál prostupuje nelineárním obvodem, nebo pluginem s nelineární charakteristikou (tedy dojde ke zkreslení signálu). Fázové zkreslení přehazuje vzorky v periodě signálu, čímž dochází ke vzniku nového průběhu. [3]



Obr. č. 2.6: Tvarová syntéza – perioda průběhu (dole) vytvarovaná z původní periody pilového průběhu (nahoře) v softwarovém syntezátoru *Vital*

3. OPTOELEKTRICKÁ SYNTÉZA ZVUKU

Optoelektrická syntéza je pojem, zaužívaný z první poloviny 20. století, pro oblast elektro–mechanických nástrojů. V kapitole je popsána historie optoelektrické syntézy, principy, na kterých tento typ syntézy funguje a současné příklady nástrojů, jenž používají tento typ syntézy.

3.1 Optoelektrická syntéza

Optickou syntézou by se dal nazvat souhrn technik, jejichž cílem je převést obraz na zvukový signál.

V digitální podobě se jedná o získávání informací z obrazového signálu, který je reprezentován 2D maticí o počtu horizontálních a vertikálních buněk, úměrném rozlišení obrazu, a převodu těchto informací na 1D zvukový signál. To je možné realizovat například nasměrováním dat z video signálu na ovládací prvky zvukových efektů nebo oscilátorů.

Pomocí tohoto typu syntézy lze tak získat zvuk reagující na vizuál, čehož se dá využít pro multimediální umělecké instalace nebo pro vytvoření neobvyklých barev zvuku. Obrazovým signálem, který může být takto použit, je buď statický obraz nebo video. Parametry, které se dají z obrazového signálu získat, jsou například poměry červené, zelené a modré barvy v obrazu.

Mnoho dosavadních digitálních optických syntezátorů také využívá skenování obrazu (např. z jedné strany na druhou) a tímto skenováním způsobené změny barev pro spouštění oscilátorů. [4][5][6]

3.2 Historie

První nástroje zabývající se tvorbou zvuku pomocí obrazu byly založené na optoelektrickém principu, který většinou spočíval v nanášení vlnových křivek na přenosové médium (velmi často rotující disk).

V roce 1916 přišel ruský futurista Vladimir Baranoff Rossiné s nástrojem, který se jmenoval *Optofonické piano* (ve francouzštině *Piano Optophonique*). Tento stroj generoval zvuk a promítal obraz pomocí světla procházejícího přes filtry, zrcadla a čočky. Klaviaturou byly ovládány kombinace světelných prvků a světlo bylo posíláno na fotosenzitivní elektrický prvek, který ovládal frekvenci oscilátoru.

V roce 1927 byl ve Francii vynalezen *Cellulophone*, což byl nástroj, který generoval tón pomocí zdroje světla a systému clon, přiřazenému ke klávesám. *Variophon* byl nástroj zkonstruovaný v Sovětském svazu v roce 1931. Umožňoval náhled na zvukovou vlnu, jenž byla přehrávána. Mnoho historických optoelektrických nástrojů (mezi nimi i *Variophon*) tvořilo zvuk pomocí rotačního kotouče s otvory, které periodicky měnily clonu světla. Dalším zajímavým nástrojem je *Saraga*. Tento nástroj je pojmenován po

jeho konstruktérovi. Bylo možné jej ovládat pomocí pohybu v prostoru. V tu dobu byl také vynalezen nástroj *Nivotone* (vynálezce Nikolai Voinov), který pro generování zvuku používal nastříhaný papír. V roce 1953 byl uveden *Composertron*. Jednalo se o první nástroj, na nějž se dala kreslit křivka, která poté měnila barvu zvuku. *ANS* byl nástroj uvedený v roce 1958. Obsahoval 720 sinusových tónů, které se zapínaly a vypínaly v závislosti na zadání uživatele (zápis na speciální povrch, jenž plnil funkci sekvenceru). Britská elektronická hudebnice Daphne Oram přišla v roce 1959 s nástrojem *Oramics*. Ten se dal ovládat pomocí kreslení na několik filmových pásů. V šedesátých a sedmdesátých letech proběhlo ještě několik projektů, které fungovaly na analogovém principu (např. nástroj *Optigan* od firmy *Mattel* nebo systémy pro fotosonickou syntézu od Jacquese Dudona). Na konci sedmdesátých let přišel skladatel a architekt Iannis Xenakis s nástrojem *UPIC*, který převáděl vizuální gesta na zvuk. Tento nástroj se skládal z počítače, programu a tabletu. Tvorba zvuku tedy probíhala digitálně. V osmdesátých letech začalo nastupovat digitální zpracování signálu a s ním i přesun optoelektrické syntézy do digitální roviny. [5][6]

3.3 Současná optoelektrická syntéza

Dnes většina nástrojů pracuje na principu digitálního zpracování obrazového i zvukového signálu. Nástroje, které z obrazu syntetizují zvuk, jsou většinou dostupné v softwarové podobě.

Jedním z mála hardwarových projektů, které se zabývají tímto typem tvorby zvuku, je syntezátor *Silhouette Eins*, jehož tvůrcem je Pit Przygodda. Tento syntezátor převádí obraz nebo jeho části na zvukovou vlnu. Ta je hratelná po stisknutí klávesy a její tvar se mění v reálném čase, s měnícím se obrazem na vstupu. [7]

Softwarových syntezátorů je několik. Začněme nástrojem *PIXELSYNTH*, který je nejdostupnější, vzhledem k tomu, že je zdarma, a dá se spustit v internetovém prohlížeči. V tomto syntezátoru je obraz postupně snímán ze strany na stranu. Tón je spuštěn, když snímač v obraze narazí na konkrétní barvu. Barvu zvuku konkrétního tónu měnit nelze, lze však měnit stupnici, ze které jsou přehrávané tóny a počet not, který je snímač schopen pochytit. Mixturou těchto samostatných tónů, které jsou hrány zároveň, vzniká zajímavá barva zvuku. [8]

Na principu snímání obrazu z jedné strany na druhou také funguje software *Photosounder*. Ten však nabízí větší funkčnost v rámci editování obrazu a zvuku a detailní práci se spektrogramem. Dalším podobným programem je *Metasynth*. [9]

Program *Usine* nabízí možnost manipulace s parametry zvuku pomocí obrazu nebo dotykových gest.

V softwaru *Blip* je možné sledovat a editovat sekvencer o velikosti 64x64. Lze do něj vložit až 64 zvuků. Spouštění každého zvuku se ovládá graficky (kreslením a další úpravou obrazu) na sekvencéru, který má 64 stop.

V DAW *FL Studio* můžeme najít plugin *Beep Map*. Ten skenuje obraz přes vertikální linku a generuje pomocí barev, nacházejících se v obrazu, zvukové parametry. Sken obrazu probíhá z levé do pravé strany.

Program *Coalgula 1.666* umožňuje kreslit obraz, který je poté možné upravovat pomocí vizuálních efektů. Zvuk je z obrazu tvořen mnoha sinusovými oscilátory a je následně převeden do souboru *WAV*. [4]

V softwaru *AudioPaint* lze importovat obraz, který je buď určený uživatelem, nebo náhodný. Je možné také vytvořit náhodně vygenerovaný grafický soubor přímo v aplikaci. *AudioPaint* nabízí i možnost vložení vlastního vlnového průběhu.

Software *The vOICe* převádí záběry z kamery v reálném čase na zvuk. Je primárně určen jako pomůcka pro orientaci v prostoru pro nevidomé. Obraz se v programu skenuje zleva doprava v pravidelném intervalu. Pro tvorbu zvuku je použito velké množství sinusových oscilátorů. Mezi tvary, snímanými v reálném čase, je rozlišeno pomocí délky tónu a rozmístění zvuku ve stereo bázi.

Transformovat obraz na zvuk lze i v programu *Adobe Audition*. Tento software umožňuje filtraci spektra, vytvořeného z jednoho obrazu, spektrem dalšího obrazu a mnoho dalších možností pro úpravu obrazu i zvuku. [4]

4. MIDI

Protokol *MIDI* je stále nejrozšířenějším protokolem pro přenos dat mezi audio zařízeními. *MIDI* zprávy budou v tomto projektu použity k ovládání nástroje. V textu je popsána základní funkčnost *MIDI* protokolu. Dále jsou uvedeny všechny *MIDI* zprávy, které budou v projektu využity. O fungování *MIDI* v programu *Max* pojednává odstavec v kapitole věnované tomuto programu.

4.1 O MIDI

Musical Instrument Digital Interface je komunikační standard, který umožnuje propojení elektronických hudebních nástrojů, počítačů a dalších, převážně audio, zařízení.

Verze *MIDI* 1.0 je dostupná od roku 1983. Od té doby se z *MIDI* stal hlavní standard pro komunikaci mezi audio technikou. Pro přenos dat mezi zařízeními se standardně používal pěti pinový DIN konektor. Pokud přenos dat probíhá z *MIDI* zařízení (např. kontroler) do počítače, používá se dnes již běžně USB konektor. Vzhledem k charakteristice projektu bude v tomto případě pro propojení ovládacího zařízení se syntezátorem využit USB kabel.

Významnými nástupci *MIDI* 1.0 jsou komunikační protokoly *OSC* a *MIDI* 2.0. *OSC* je poměrně často užívanou alternativou k protokolu *MIDI*. *MIDI* 2.0 je zpětně kompatibilní s verzí 1.0.

MIDI data se dělí na kanálová a systémová. Další odstavce pojednávají o kanálových datech. [11]

4.2 Kanálové MIDI zprávy

Data, která se přes *MIDI* přenáší, jsou informace o událostech. Tyto informace může přístroj vysílat, přijímat, nebo obojí. Informace je přenášena ve formě bajtů (8 bitů), přičemž většina zpráv obsahuje tři bajty (Stavový bajt a dva Datové bajty). Před každým bajtem je vyslán start bit (0) a po každém bajtu je vyslán stop bit (1). Tyto bity zajišťují, že je zařízení, které signál přijímá, na příjem připraveno. Informace jsou vysílány sériově, s rychlostí 31,25 kbit/s, do jednoho z šestnácti kanálů, případně do více kanálů najednou.

Stavový bajt se skládá z osmi bitů, z nichž MSB určuje, že se jedná o Stavový bajt (hodnota 1). Další tři bity určují typ posílané zprávy. Poslední čtyři bity jsou věnovány identifikátoru kanálu, pro nějž je daná zpráva určena (kanál 1–0000, kanál 16–1111).

Datový bajt, stejně jako stavový bajt, začíná identifikačním bitem (hodnota 0). Dále pokračuje sedmi bity, jejichž účel záleží na typu zprávy, kterou určuje stavový bajt.

Typy zpráv, které mohou být přenášeny, jsou: *Note On, Note Off, Polyphonic Key Pressure, Control Change, Program Change, Channel Pressure* a *Pitch Bend Change*. Projekt bude používat pouze některé z těchto typů zpráv. [11]

4.2.1 Note On

Kanálová zpráva *Note On* se přenáší při stisknutí noty. Skládá se ze stavového bajtu (ve tvaru 1001xxxx, kde xxxx reprezentuje číslo kanálu, do kterého je zpráva poslána) a dvou datových bajtů. První z nich přenáší informaci o čísle zahrané noty, druhá o rychlosti stlačení noty.

Vzhledem k maximálnímu objemu dat, který může být přenášen jednou zprávou, je počet not, hratelných přes *MIDI*, 128. Notám jsou v rozsahu od C−2 do G8 přiřazena čísla 0–127. Stejný rozsah platí pro rychlost stlačení noty. Ta určuje, jak nahlas je nota zahrána. Reakce na rychlost stisku lze využít při snaze o vytvoření realisticky znějícího softwarového nástroje. [10][11]

4.2.2 Note Off

Kanálová zpráva *Note Off* se přenáší při uvolnění noty. Identifikační část stavového bajtu má tvar 000 (tedy je identifikační číslo této zprávy 0, bajt ve tvaru 1000xxxx). Informace přenášené pomocí datových bajtů jsou stejné, jako u zprávy *Note On* (identifikační číslo uvolněné noty a rychlost s jakou byla nota uvolněna). V softwarových nástrojích se s využitím informace o rychlosti uvolnění noty nesetkáváme často, může však být využita. [10][11]

4.2.3 Control Change

Kanálová zpráva *Control Change* (často se používá zkratka CC) je přenášena při změně hodnoty ovládacích prvků přístroje, který vysílá *MIDI*. Stavový bajt je ve tvaru 1011xxxx (identifikační číslo 3). První datový bajt přenáší informaci o čísle kontroleru a druhý o hodnotě nastavovaného parametru. Rozsah hodnot těchto bajtů je 0–127. Čísla kontroleru od 121 do 127 jsou vyhrazena pro tzv. *Channel Mode Messages*.

Tyto zprávy ovlivňují chovaní nástroje po obdržení *MIDI* dat (například *Omni* mód pro přijímání zpráv ze všech kanálů nebo *Poly* mód pro polyfonické hraní not). Na číslech kontroleru 0–120 lze měnit libovolné parametry nástrojů (např. rozladění oscilátorů, hlasitost) nebo efektů. [10][11]

4.2.4 Pitch Bend Change

Kanálová zpráva *Pitch Bend Change* se používá k popisu zvýšení, nebo snížení výšky tónu. Stavový bajt je ve tvaru 1110xxxx (identifikační číslo 6). Dále zpráva obsahuje dva datové bity, z nichž v prvním se přenáší sedm LSB, v druhém sedm MSB. Celkem tedy tato zpráva má 16384 hodnot. Hodnoty 0–8191 slouží pro popis snížení výšky tónu, 8192 je střední hodnotou a hodnoty 8193–16383 popisují zvýšení tónu. [10][11]

5. PROGRAMOVACÍ PROSTŘEDÍ MAX/MSP/JITTER

5.1 Historie

Max (*Max/MSP/Jitter*) je vizuální programovací jazyk a software, využívaný k procesování audio signálu, k práci s daty, většinou nějak souvisejícími s hudbou (např. *MIDI*), a také k vytváření multimediálních aplikací. Práci na tomto jazyku začal Miller Puckette v roce 1985 na *Výzkumném a koordinačním ústavu pro akustiku a hudbu* (*IRCAM*) v Paříži.

První verze nesla název *Patcher* a pracovala pouze s *MIDI*, později (1989) byl stvořen *Max/FTS*, který jako první pracoval s audio signálem. V roce 1990 byla uvedena první komerční verze programu *Max* od společnosti *Opcode Systems*. V roce 1997 přešel *Max* pod společnost *Cycling´74*, pod kterou je komerčně vydáván dodnes. Rozšíření *Jitter*, které se používá pro práci s obrazem, bylo přidáno v roce 2001. Verze *Max for Live*, která přinesla *Max* do DAW *Ableton Live*, vyšla v roce 2009. [4][14]

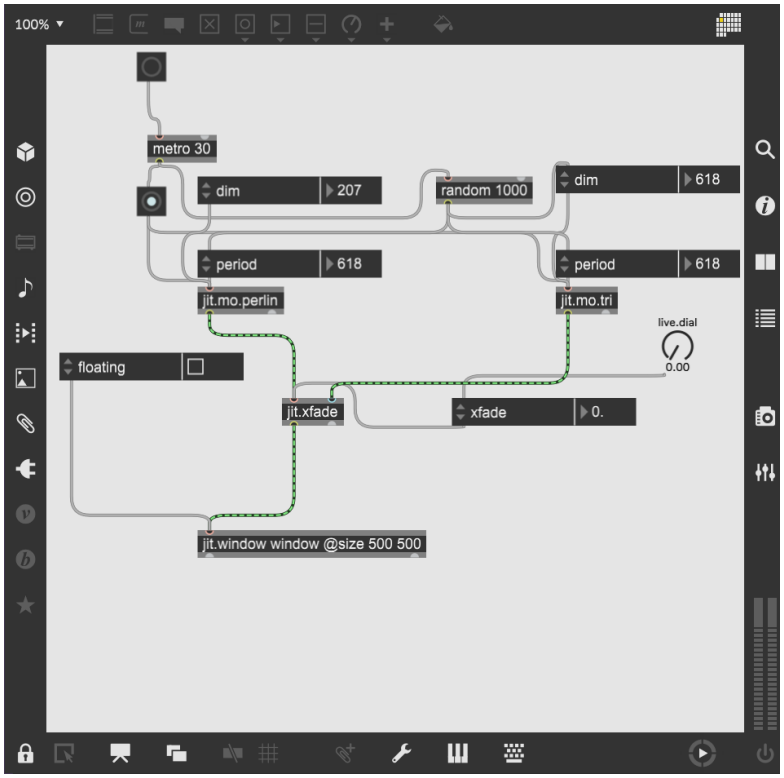
5.2 Popis programu

Max je podporován na operačních systémech *macOS* a *Microsoft Windows*. Program vytvořený v softwaru *Max* se nazývá patch. Patch je tvořen propojováním objektů, které plní jednoduché funkce. Objekty jsou programovány v jazycích *Java*, *C*, *C++* a *JavaScript*. [12]

Samotný *Max* je naprogramován v jazycích *C* a *C++*, konkrétně pomocí frameworku *JUCE*. Díky dokumentaci *Max API* je umožněno uživatelům vytváření objektů do programu *Max* v jazyku *C*. [12][13]

Obr. č. 5.1: Objekt v programu Max

Objekty obsahují libovolný počet vstupů a výstupů (záleží na konkrétním objektu). Vstupy se nachází v horní části objektu a dělí se na hot a cold (označeny červeně a modře), dle toho, který z nich spouští výstup funkce vykonávané objektem. Výstupy jsou umístěny dole. [12]



Obr. č. 5.2: Patch v programu Max

Na obrázku vidíme patch v programu *Max*, sloužící pro syntézu obrazu. Důležitým prvkem programu je ikona pro zamknutí/odemknutí projektu v levém dolním rohu, která umožňuje patch editovat (při odemčení), nebo používat (při uzamčení). Dále jsou na dolní liště prvky sloužící k ovládání celého patche (mapování klávesnice a *MIDI*, presentation view a debugging). Horní lišta obsahuje základní objekty (např. *comment*, *button*, *toggle*). Vpravo jsou důležitými ovládacími prvky hlavně *Inspector* (slouží pro úpravu parametrů v objektech) a *Max Console* (zobrazuje systémová hlášení, zejména chyby). Na levé straně se nachází průzkumník souborů, ve kterém lze najít objekty, pluginy (uvnitř patche může být použit VST3 plugin) nebo zvukové a obrazové soubory. [12]

5.3 Max

Software *Max* je rozdělen na tři části – *Max*, *MSP* a *Jitter*. Každá z těchto částí programu má za úkol práci s jinými typy dat a různé objekty pracují často s více datovými typy najednou. *Max* je využíván k práci s *MIDI*, číselnými a textovými zprávami.

Základní typy zpráv podporované v *Maxu* jsou *float*, *integer*, *bang*, *list* a *symbol*. Datový typ *integer* podporuje celá čísla v rozsahu -2^{63} až $2^{63}-1$, jedná se tedy o 64bitový signed integer (tento rozsah záleží na tom, jestli *Max* pracuje v 32 nebo

64bitovém módu, nové verze jsou pouze 64bitové). *Float* podporuje celá a desetinná čísla ve stejném rozsahu, jako *integer*. *Bang* dosahuje pouze hodnot jedna a nula, tedy buď probíhá, nebo neprobíhá. V datovém typu symbol se ukládají data jako *string* a vypisují se jako souvislý text. S tímto typem dat nelze provádět matematické operace. *List* je zpráva, která zapouzdřuje více zpráv dohromady (např. několik čísel).

Pomocí těchto datových typů se dají stavět části patche, které nevyžadují audio nebo video procesování (tedy např. moduly pro práci s *MIDI*, moduly pro matematické operace atd.). Propojovací kabely pro přenos *Max* dat mají šedou barvu. [12]

5.4 MSP

MSP je část *Maxu*, která se stará o procesování zvukového signálu. Zvukový signál je reprezentován pomocí vzorků, které mají 16–32 bitů. Vzorkovací frekvence je nastavitelná od 11025 Hz (věrně reprodukovány signály do cca. 5500 Hz) až 192 kHz (v dnešní době standard pro tzv. *high resolution audio*). Pro audio aplikace jsou nejčastěji používané vzorkovací frekvence 44,1 a 48 kHz.

MSP obsahuje kolem dvoustovky objektů, které jsou orientované na práci s audio signálem (např. filtry, zpožďovací linky, objekty pro vytvoření zkreslení signálu). Lze tak vytvářet zvukové efekty, syntezátory a jiné patche, které pracují se zvukem (např. je možné vytvořit plně funkční Digital Audio Workstation).

Propojovací kabely pro přenos *MSP* dat jsou přerušovaně světlezeleně šedé, objekty, které slouží k procesování audio signálu jsou doplněny příponou ~. [12]

5.5 Jitter

Jitter pracuje s obrazovým signálem. Obrazový signál je v digitální podobě reprezentován například pomocí barevného modelu RGBA, kdy je pro každý z kanálů (Red, Green, Blue a Alpha) použito 8 bitů (tedy lze pro každý R, G nebo B kanál nastavit 256 hodnot intenzity v rozsahu 0–255). Kanály RGB značí intenzitu červené, zelené a modré barvy, kanál Alpha reprezentuje průhlednost, ta je v rozsahu 0–1. *Jitter* pracuje s 2D maticí (počet řádků a sloupců reprezentuje rozlišení obrazu), která má v každé buňce čtyři hodnoty (reprezentující RGBA kanály).

Propojovací kabely pro přenos *Jitter* dat jsou přerušovaně tmavě-zeleně šedé a objekty, které patří pod *Jitter* jsou označeny předponou *jit*. *Jitter* obsahuje klasické objekty pro procesování obrazu (úprava jasu, kontrastu a saturace nebo efekt pro otáčení obrazu), ale i objekty pro generování obrazu či naplnění 2D matice pomocí 1D zvukového signálu. *Jitter* také obsahuje objekty pro práci s *OpenGL,* které umožňují vytváření komplexních reaktivních animací. [12]

5.6 Gen

Gen objekt využívá převážně výpočtů z proměnných a jednoduchých objektů k vytvoření nových objektů. Zdrojový kód funkce vytvořené pomocí objektu *gen* je pak možné exportovat jako *C++* kód. V *Maxu* se nachází několik *gen* objektů, *gen~* se používá pro práci s audiem (*MSP*), *gen* pro operace s *Max* daty a funkce *jit.gen*, *jit.pix* a *jit.gl.pix* se používají pro práci s obrazem (*Jitter*, *OpenGL*). [12]

5.7 MIDI v programu Max

Max obsahuje několik objektů, které umožňují práci s *MIDI* zprávami. Základní objekty jsou *midiin* a *midiout*, které přijímají a vysílají všechny *MIDI* informace. Dále jsou často používané objekty *notein, noteout, ctlin* a *ctlout*, které přijímají a odesílají specifické *MIDI* zprávy.

U objektů s předponou *note* jde o informace posílané zprávami *Note On* a *Note Off,* u předpony *ctl* se jedná o *Control Change* zprávy. Přípony *in* a *out* naznačují, jestli daný objekt signál přijímá z kontroleru (*in*), nebo odesílá do jiného kontroleru (*out*). [12]

5.8 Externí objekty a patche

Vzhledem k dostupnosti dokumentace API programu *Max* byl pro program vytvořen obsah, který nepochází přímo od vydavatele softwaru. Vzniklo tak mnoho objektů, v některých případech i celých patchů. Některé rozšiřují funkčnost programu, jiné se využívají k edukativním účelům.

Jedním ze známějších rozšíření je kolekce modulů *BEAP*, která byla vytvořena jako edukativní pomůcka a obsahuje mnoho patchů, které mají za cíl emulovat základní audio efekty. [12]

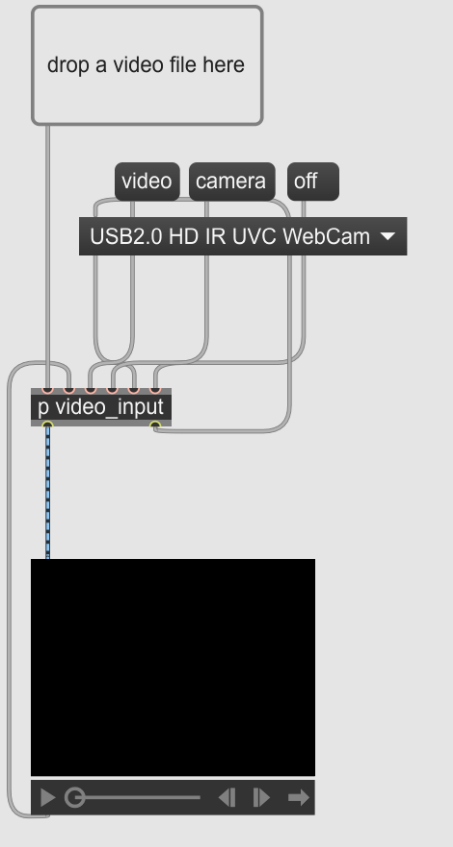
5.9 Projekty

Project soubor (označen příponou .maxproj) zahrnuje všechny patche a objekty (i externí) použité pro vytvoření konkrétního projektu (v tomto případě hudebního nástroje). Zapouzdření projektu do takového souboru značně usnadňuje přenos na jiná zařízení. V rámci *project* souboru je možné z existujících patchů vytvořit například samostatnou aplikaci (s příponou .exe) nebo zařízení pro MaxForLive (přípona .amxd). [12]

6. NÁVRH ČÁSTÍ NÁSTROJE

V rámci práce budou vytvořeny dva nástroje. První z nich kombinuje prvky aditivní a granulární syntézy, druhý funguje na principu sampleru a sekvencéru. V této kapitole je řešen návrh jednotlivých částí, ze kterých budou nástroje postaveny. Některé z modulů, popsaných v těchto kapitolách, jsou součástí obou nástrojů (video vstup, efektová jednotka, sekvencér a výstupní modul), jiné jsou použity pouze u jednoho z nástrojů.

6.1 Video vstup, display



Obr. č. 6.1: Patch pro načtení videa

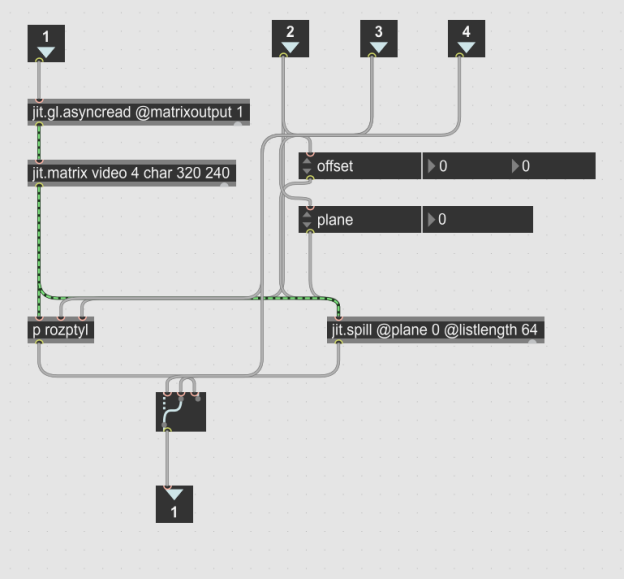
Obrazový signál je do patche možno přivézt dvěma způsoby. Vstup pro video je řešen pomocí objektu *dropfile*. Tento objekt slouží pro vložení souboru do patche v programu Max a podporuje několik formátů audio, video i textových souborů. V objektu je vložen komentář, který uživateli naznačuje, jaký typ souboru by měl být vložen (v tomto případě video). Soubor je veden do patche *video_input*. Tam následuje objekt *prepend,* který obsahuje zprávu *read*. Tato zpráva říká objektu *jit.movie*, že má objekt převzít video soubor, který byl importován pomocí objektu *dropfile*, do programu patche.

Objekt *jit.movie* slouží k přehrání video souboru. V tomto případě je na objektu ještě nastavena hlasitost videa na 0, aby z video souboru nešel zvuk. Dále je nastaven pomocí příkazu *output_texture* mód, ve kterém je výstup tohoto objektu nikoliv matice, která je procesovaná pomocí CPU, což vede k trochu horšímu výkonu v rámci přehrávání videa, ale textura, zpracovávaná pomocí GPU, což vede k plynulejšímu chodu patche. Textura pokračuje do objektu *jit.gl.pix*, ve kterém je naprogramován přepínač.

Přepínač pomocí ovládacích prvků (objekty *message* s texty *video, camera* a *off*) určuje, jaký zdroj video signálu je posílán na výstup patche. Textura dále pokračuje do objektu *jit.pworld*. Úkolem tohoto objektu je zobrazení daného videa. Pod tímto objektem se ještě nachází objekt *playbar*, který slouží k přehrávání videa a k posouvání času ve videu a je napojen na objekt *jit.movie*. Vstup pro video z externího zdroje je inicializován stiskem objektu *message* s nápisem *camera*. Tím je spuštěn objekt *jit.grab* a zároveň je tak na přepínači nastaven signál z tohoto objektu jako výstup. Pod přepínači se nachází ještě objekt *umenu*. Ten získává z *jit.grab* data o aktuálních zařízeních, které může objekt *jit.grab* použít pro zisk video signálu, a zároveň určuje, jaké z těchto zařízení je použito. [12]

6.2 Moduly pro získávání dat z videa

6.2.1 Metoda 1 – objekt jit.spill



Obr. č. 6.2: Patch pro zisk dat z videa – Metoda 1

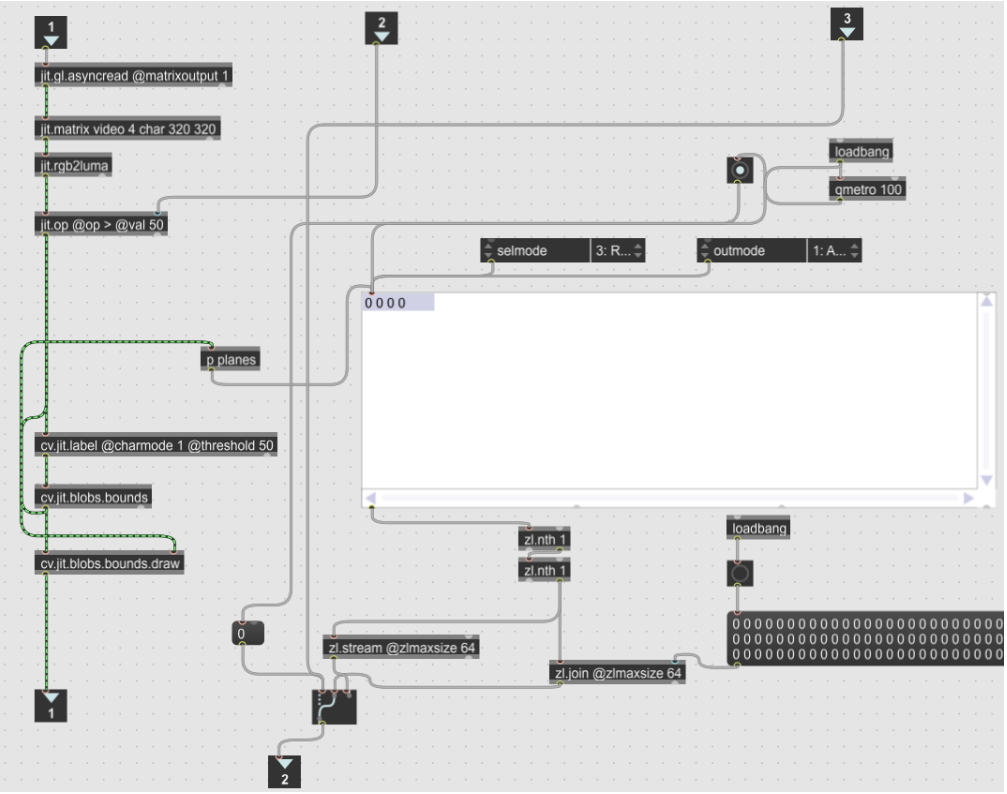
Modul pro zisk dat z videa vždy začíná objektem *jit.gl.asyncread*. Tento objekt asynchronně čte data z textury a jeho výstupem jsou data, která jsou poté objektem *jit.matrix* převedena do matice s názvem video o zadaném rozlišení (320x240), počtu

rovin (4 – ARGB) a typu uložených dat (char). Z *jit.matrix* jde matice do objektu *jit.spill*. Objekt *jit.spill* čte hodnoty z matice. Je možné u něj nastavit, z jakých souřadnic matice jsou data čtena, z kolika buněk matice jsou data čtena a z jaké roviny jsou data čtena.

Offset pozic x a y, ze kterých jsou data čtena, se dá ovládat pomocí jakéhokoliv objektu, ze kterého vystupují kladné celočíselné hodnoty. Byl vybrán objekt *pictslider*. Data je potřeba dostat k atributu *offset* jako list, tedy ačkoliv hodnoty do patche vstupují jednotlivě, jsou poté sdruženy do listu objektem *join*. Rovina, ze které jsou hodnoty čteny, lze také ovládat, číslem v rozsahu 0–3. Z objektu *jit.spill* vystupuje list.

V tomto modulu lze pomocí switche vybírat mezi dvěma možnostmi čtení dat. Pokud je zvolen první vstup, zadaný počet hodnot je čten pouze z jednoho *jit.spill*, na druhý vstup je vyvedeno více, vzájemně posunutých, objektů *jit.spill*. [12]

6.2.2 Metoda 2 – cv.jit.blobs.bounds



Obr. č. 6.3: Patch pro zisk dat z videa – Metoda 2

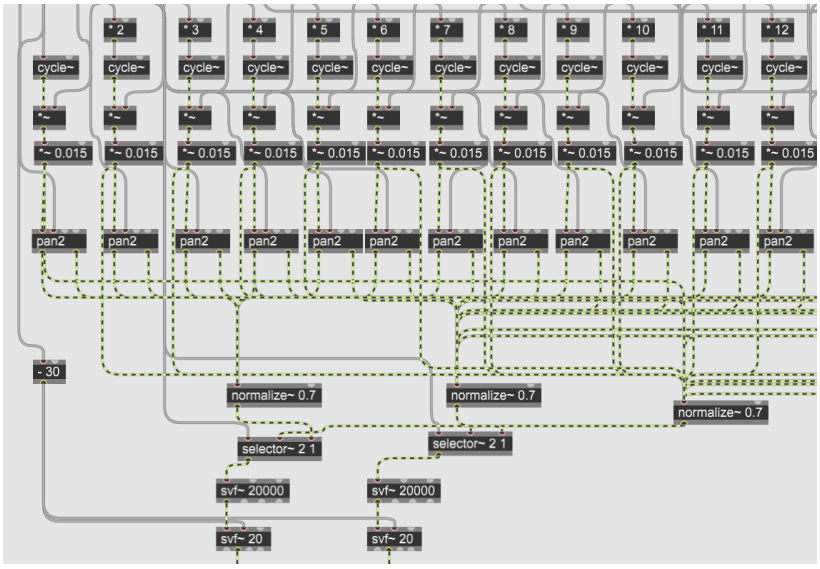
Objekt *cv.jit.blobs.bounds* z knihovny *cv.jit* (Computer Vision for Jitter), jejíž autorem je Jean-Marc Pelletier, má za úkol nacházet a ohraničovat objekty v obraze. Toho je docíleno pomocí objektů *jit.rgb2luma* (změna barevného obrazu na monochromatický), *jit.op* (nastavený tak, že jím prochází jako bílé pouze pixely, které mají hodnotu větší, než stanovená hranice) a *cv.jit.label* (také z knihovny *cv.jit*, jeho

úkolem je identifikace a označení propojených objektů – neohraničených shluků bílých pixelů).

Cv.jit.blobs.bounds poté tyto objekty ohraničí boxem, jehož hrany jsou vždy tam, kde je nejkrajnější pixel v objektu (výstupem z objektu je matice, která v každé buňce obsahuje souřadnice jednoho objektu v pořadí levá, horní, pravá a spodní hrana boxu). Z objektu jsou poté souřadnice vyvedeny na patch *cv.jit.blobs.bounds.draw* (z knihovny *cv.jit*, při správné konfiguraci vykresluje box, označující objekt, do původního obrazu) a na patch *planes*, který matici směřuje do objektu *jit.cellbock*, z nějž jsou odesílána data na výstup. Ještě před výstupem z patche jsou však skládána do jednoho listu pomocí funkcí *zl.stream* nebo *zl.join* (každý z těchto objektů skládá data do výsledného listu jinak, lze mezi nimi vybírat pomocí objektu *gswitch*). Také jsou z listu odebírány první dvě čísla, o což se stará objekt *zl.nth* (tato čísla jsou lokace konkrétní buňky v matici). [12]

6.3 Moduly pro přeměnu dat z videa na zvuk

6.3.1 Aditivní syntéza



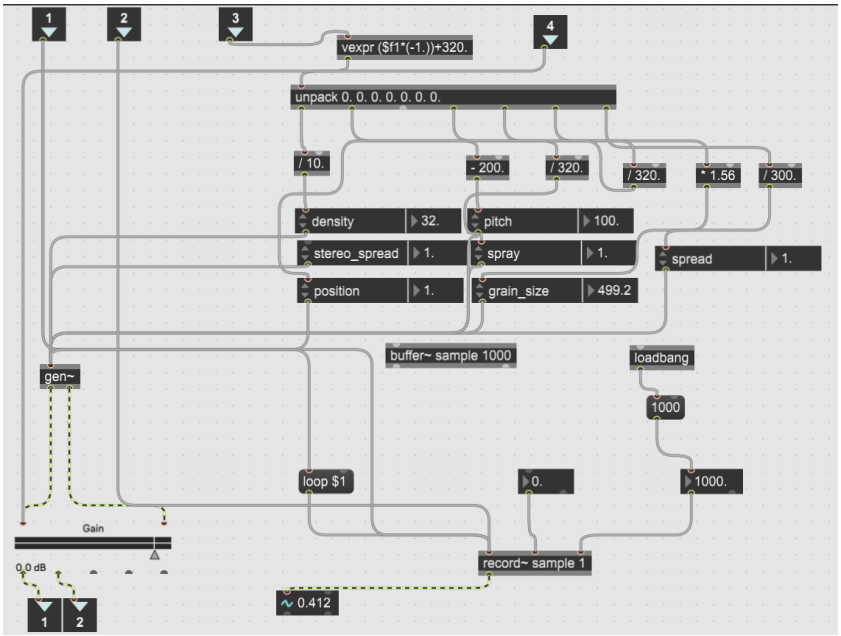
Obr. č. 6.4: Patch pro převod dat z videa na zvuk – Aditivní syntéza

Vstupem do modulu pro přeměnu dat z videa na zvuk je frekvence. Ta se dá získat jakkoliv, nezávisle či závisle na vstupním videu, v případě tohoto syntezátoru z *MIDI*. Získaná vstupní frekvence je poté přivedena do sinusového oscilátoru (v programu *Max* zastupován objektem *cycle~*). Vynásobením základní frekvence celými čísly v rozsahu 2–64 a přivedením těchto hodnot do adekvátního počtu objektů *cycle~* získáváme 64 harmonických složek. Tyto harmonické složky jsou poté přivedeny do objektu * ~,

který slouží pro násobení zvukového signálu, tedy jeho zesilování či zeslabování. Signál z oscilátoru přicházející do tohoto objektu je násoben daty z videa, která do objektu vstupují jako *list*. Ten je upraven tak, aby čísla, uložená jako datový typ *list,* dosahovala hodnot 0–1 a následně je rozdělen objektem *unpack* na 64 hodnot, které jsou vyvedeny do objektu *~. Každá hodnota se tedy násobí s jedním sinusovým oscilátorem. Zvuk z oscilátorů je poté vynásoben tak, aby při sečtení všech složek nepřekračoval hodnotu 1, a odeslán na patch *pan2*, který jednotlivé složky umisťuje v prostoru a je ovládán daty z videa. Výběr mezi automatizací a umístěním všech složek uprostřed stereo báze je prováděn objektem *selector~*.

Všechny harmonické složky jsou pak přivedeny na objekt *normalize~*, který signál upraví tak, aby jeho nejvyšší hodnota dosahovala zadaného čísla. Poté je signál proveden dvěma objekty *svf~*, nastavenými tak, aby tolik nedocházelo k aliasingu při hře ve vyšší poloze (dolní propust na 20 kHz, horní propust 30 Hz pod frekvencí první harmonické složky). Nakonec signál putuje na levý i na pravý výstup objektu. [12]

6.3.2 Granulátor

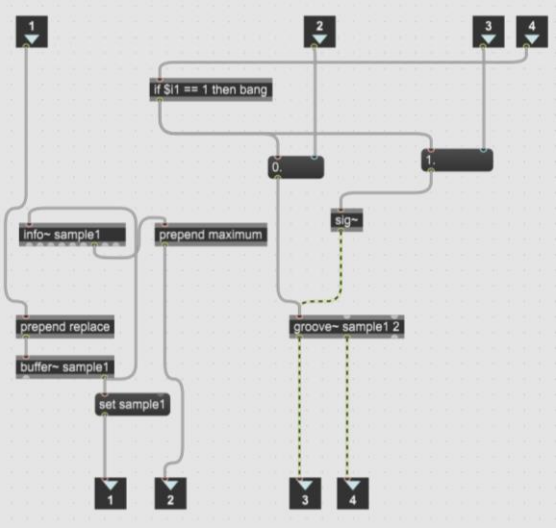


Obr. č. 6.5: Patch pro převod dat z videa na zvuk – Granulátor

Granulátor pouští malé úseky audio signálu, uloženého v objektu *buffer~*. Signál je vzorkován objektem *record~*. Ten je konfigurován tak, aby do bufferu opakovaně nahrával 1000 milisekund signálu, který vychází z jiného zdroje zvuku. K přehrání signálu je použit objekt *gen~*, v němž je, za pomoci objektu *codebox*, napsán kód, který upravuje parametry přehrávání krátkých vzorků signálu (hustota, délka vzorku, výška vzorku, náhodná výška vzorku, pozice v hlavním vzorku, odchylka od pozice v hlavním

vzorku a rozmístění do stereo báze). Parametry přehrávání jsou kontrolovány vhodně upravenými daty získanými z video signálu. Před výstupem je signál z granulátoru proveden objektem *live.gain~* (objekt z knihovny *Max for Live*), aby bylo možné upravit jeho výstupní hlasitost dle potřeby. Granulátor je převzat z edukačního videa, dostupného na platformě Youtube. [12][18]

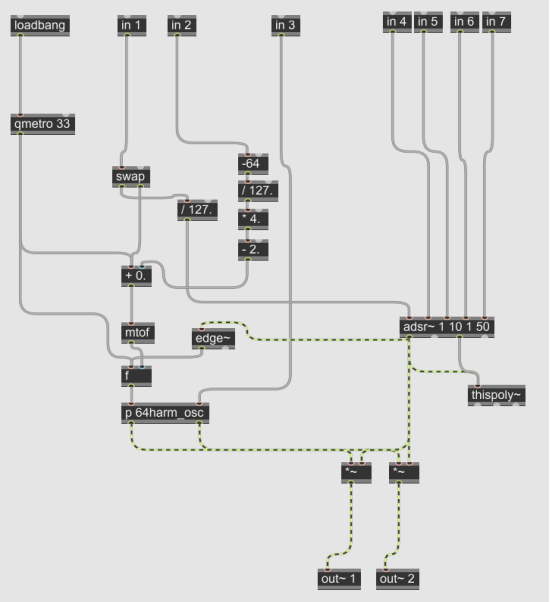
6.3.3 Sampler



Obr. č. 6.6: Patch pro převod dat z videa na zvuk – Sampler

Úkolem sampleru je uložení a následné přehrání zvukového vzorku. Zvukový vzorek je, stejně jako u granulátoru, uložen do objektu *buffer~*. Ukládání probíhá pomocí objektu *dropfile* a objektu *prepend* se zprávou *replace,* který směřuje do bufferu. Pro přehrání zvukového vzorku je použit objekt *groove~*, u nějž lze měnit rychlost a směr přehrávání vzorku a také počáteční bod, ze kterého bude vzorek přehráván. Objekt *info~* slouží k zjištění délky vzorku v milisekundách. [12]

6.4 Objekt *poly~*, MIDI vstup, ADSR



Obr. č. 6.7: Patch zapouzdřený uvnitř objektu *poly~*

Pro realizaci polyfonie (více instancí stejného nástroje hrající různé tóny) je třeba v programu Max využít objekt *poly~*. Do něj je vložen patch, který vytváří zvuk. Zároveň je v rámci tohoto objektu řešen vstup MIDI dat, jejich využití a generátor obálky ADSR. [12]

6.4.1 Objekt *poly~*

Poly~ slouží k vytváření vícehlasých (polyfonních) nástrojů. Obsahuje patch, ve kterém je zdroj zvuku a vytváří instance tohoto patche dle aktuální potřeby. Povinným argumentem objektu *poly~* je název patche, který je do něj vložen. Významným parametrem je parametr *voices*. Ten určuje maximální počet hlasů (a tedy kopií původního patche), které je možné vytvořit. Důležitou součástí každého patche, vloženého do *poly~*, je objekt *thispoly~*, jehož úkolem je vytváření nových a tlumení již nepoužívaných instancí zdroje zvuku. [12]

6.4.2 MIDI vstup

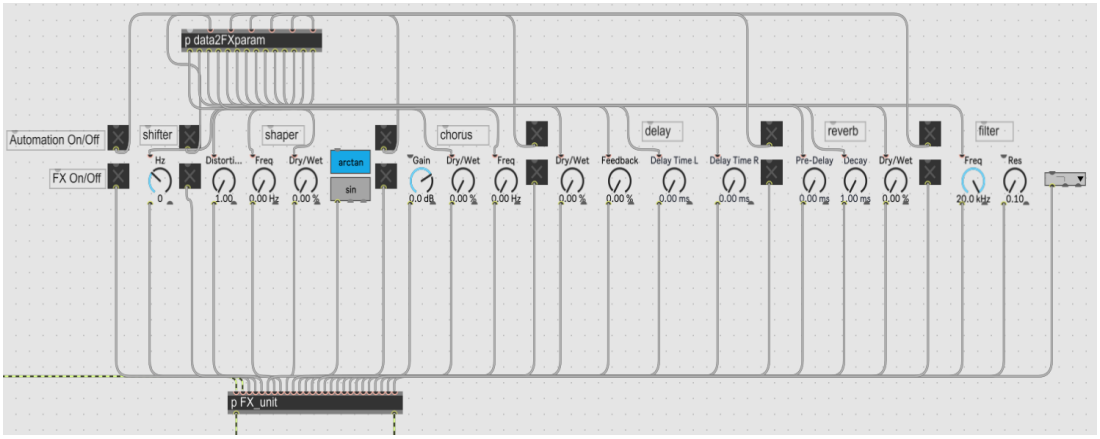
Vstup MIDI dat je realizován objekty *notein* a *bendin*. Data *notein*, jimiž jsou údaje o čísle zahrané noty v rozsahu 0–127 a údaje o rychlosti zahrání noty (*velocity*) v témž rozsahu (dále objekt obsahuje ještě informaci o čísle kanálu, ta však zde není využita), jsou sjednocena do listu objektem *pack* a odeslána na první vstup objektu *poly~*. Data *bendin*, v rozsahu 0–127 (pro 14ti bitovou reprezentaci pitch bendu je třeba využít objekt *xbendin*) jsou přivedena na druhý vstup tohoto objektu. Uvnitř *poly~* dochází

k přivedení hodnoty *velocity* na objekt *adsr~*. Údaj o čísle noty je sčítán s údajem o provedeném pitch bendu. Aby k tomuto sčítání docházelo pravidelně, je použit objekt *qmetro*, který vysílá signál typu *bang* v zadaném intervalu a je inicializován objektem *loadbang* (ten vysílá *bang* při spuštění programu). Součet zahrané noty a pitch bendu míří ještě na objekt *float* a poté na patch pro tvorbu zvuku. [12]

6.4.3 Generátor obálky ADSR

Objekt *adsr~* je zapouzdřen v *poly~*. Jeho kontrolní prvky jsou přivedeny na vstupy 4–7 tohoto objektu (jedná se o attack, decay, sustain a release). Na vstup *adsr~* je kromě kontrolních prvků přiveden signál *trigger*. Jedná se o údaj *velocity*, podělený číslem 127, aby byl v rozsahu 0–1. Obálka ADSR, která je na prvním výstupu objektu pokračuje do dvou objektů *~, v nichž je násobena s výstupním signálem zdroje zvuku, dále na objekt *edge~*, který detekuje přechody z nuly na nenulovou hodnotu a do objektu *thispoly~*. Do *thispoly~* je z *adsr~* odesílán ještě výstup *mute*. [12]

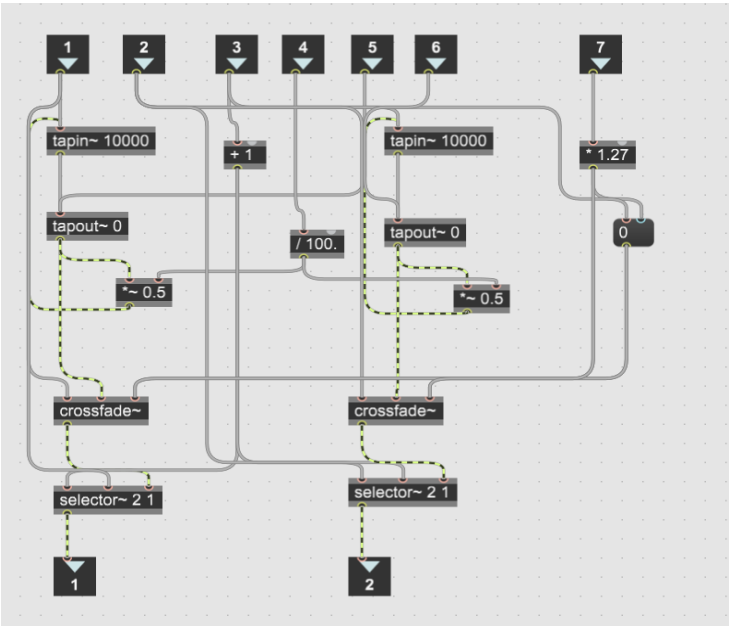
6.5 Efektová jednotka



Obr. č. 6.8: Efektová jednotka

Součástí nástroje je efektová jednotka. Ta obsahuje šest zvukových efektů (*frequency shifter*, *waveshaper*, *chorus*, *delay*, *reverb* a *filtr* v tomto pořadí). Na vstup je přiveden audio signál pro levý a pravý kanál a ovládací prvky pro parametry jednotlivých efektů. Efekty je možné zapínat a vypínat, dále je možné zapnout a vypnout automatizaci parametrů efektů pomocí dat získaných z videa. Výstupem je signál levého a pravého kanálu.

6.5.1 Delay



Obr. č. 6.9: Efekt delay

Efekt delay je tvořen dvěma zpožďovacími linkami (jednou pro levý, druhou pro pravý kanál signálu). Zpožďovací linky na obou kanálech jsou realizovány objekty *tapin~* a *tapout~*. Tyto objekty tvoří v programu Max jednoduchou zpožďovací linku, kde objekt *tapin~* přijímá vstupní signál a je na něm nastavováno maximální zpoždění linky (v tomto případě 10 sekund). Objekt *tapin~* musí být propojen s objektem *tapout~*, na němž je nastavováno aktuální zpoždění a ze kterého vychází výstupní signál zpožďovací linky. Pro nastavení tohoto zpoždění je v patchi připraven jeden vstup.

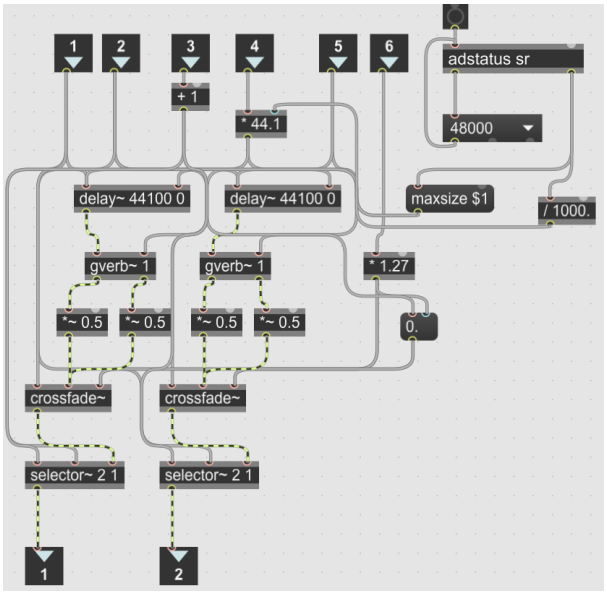
Přivedením výstupu objektu *tapout~* na vstup objektu *tapin~* lze vytvořit zpětnou vazbu zpožďovací linky, jejíž zesílení lze ovládat. Dále tento efekt obsahuje On/Off přepínač, realizovaný objektem *selector~* pro každý vstupní kanál, do kterého je přiveden neprocesovaný signál (pro pozici Off) a procesovaný signál (pro pozici On). Tento objekt je inicializován vstupem 1, do kterého jde neprocesovaný signál, tedy v pozici 1 je efekt vypnutý, v pozici 2 zapnutý. V pozici 0 nevychází z objektu žádný signál, tedy je třeba k *Toggle* objektu, který zajišťuje On/Off operaci a vysílá pouze hodnoty 1 a 0, přičíst jedničku.

Také je obsažen fader Dry/Wet, který pro oba kanály určuje poměr mezi procesovaným (Wet) a neprocesovaným (Dry) signálem, čehož je docíleno objektem *crossfade~*. Ten pro prolínání mezi dvěma signály používá 127 hodnot, Dry/Wet fader však pracuje s procenty, tedy bylo třeba vstup vynásobit číslem 1,27 pro optimální výsledek. Tento objekt také nepodporuje zadání počáteční hodnoty, bez níž nefunguje,

tedy bylo nutné tuto hodnotu nastavit na hodnotu 0, pomocí odeslání zprávy z objektu *message*, při zapnutí efektu. Objekt *crossfade~* pochází z externí knihovny *RTC-lib*.

Efekt delay tedy obsahuje (v tomto pořadí) vstupy pro levý a pravý kanál, On/Off přepínač, nastavení zpětné vazby, nastavení zpoždění pro levý a pravý kanál a Dry/Wet fader. Výstupy jsou signály levého a pravého kanálu. [12]

6.5.2 Reverb



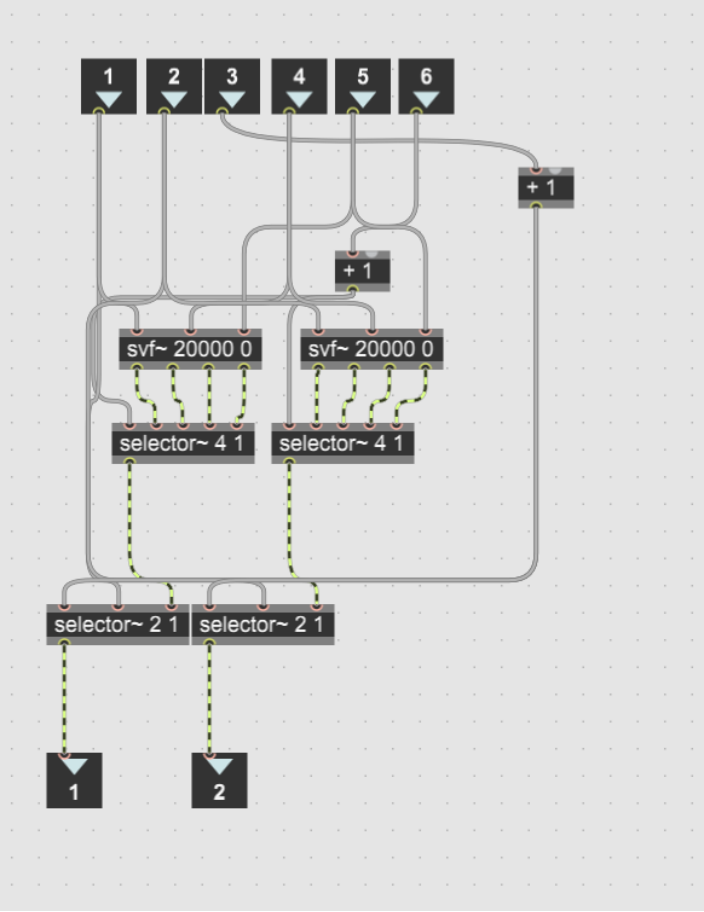
Obr. č. 6.10: Efekt reverb

Hlavním prvkem efektu reverb jsou dva stereo reverb objekty *nw.gverb~*, do kterých je na vstupu přiváděn mono signál (jeden pro levý a druhý pro pravý kanál). Objekt *nw.gverb~* pochází z externí knihovny *LowkeyNW*. Nastavit se dá v rámci objektu pouze parametr *decay*. Výstupy reverbu jsou násobeny polovinou, aby na výstupu nedošlo k přebuzení signálu. Vzhledem k tomu, že objekt *nw.gverb~* generuje pouze procesovaný signál, je třeba objektem *crossfade~* vytvořit Dry/Wet fader. On/Off přepínač je znovu tvořen pomocí objektu *selector~*. Fungování obou těchto objektů v efektových patchích je vysvětleno v kapitole *Delay*.

Dále efekt obsahuje nastavení parametru *pre-delay*. Tento parametr nelze nastavit v objektu *nw.gverb~*, je proto třeba použít objekt *delay~*, který signál pouští pozadu o zadaný počet vzorků. Pro převod počtu vzorků na milisekundy byl vstup vynásoben číslem 44,1. To je dáno tím, že program pracuje ve vzorkovací frekvenci 44,1 kHz, tedy pro posunutí signálu o 1 sekundu je potřeba tento signál posunout o 44100 vzorků. Dále je použit objekt *adstatus* s argumentem sr, který monitoruje momentální nastavenou vzorkovací frekvenci projektu a nastavuje ji jako parametr pro objekt *delay~*. Tím je zaručena přesnost nastavení zpoždění signálu.

Vstupy do efektu jsou: signál levého a pravého kanálu, On/Off přepínač, nastavení pre-delaye, nastavení parametru decay a Dry/Wet fader. Na výstup jsou přivedeny signály levého a pravého kanálu. [12]

6.5.3 Filtr

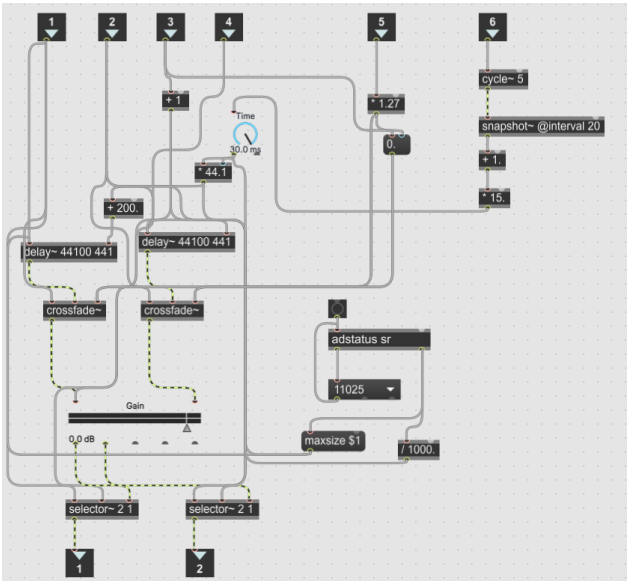


Obr. č. 6.11: Filtr

Filtr je tvořen dvěma objekty *svf~*, do kterých je přiváděn levý a pravý kanál (do každého jeden). Tento objekt obsahuje čtyři typy filtrů. Pro každý z nich má objekt výstup, tedy je třeba přepínání mezi typy filtrů řešit pomocí objektu *selector~*. Jedná se o filtry typu dolní propust, horní propust, pásmová propust a pásmová zádrž. V objektu *svf~* je třeba pro inicializaci objektu nastavit střední frekvenci, na které bude filtr pracovat a rezonanci filtru. Přepínání On/Off je znovu řešeno objektem *selector~*.

Do objektu vstupují levý a pravý signál, On/Off přepínač, potenciometr pro nastavení frekvence, potenciometr pro nastavení rezonance a přepínač, který určuje typ filtru. Objekt obsahuje výstupy pro signál levého a pravého kanálu. [12]

6.5.4 Chorus



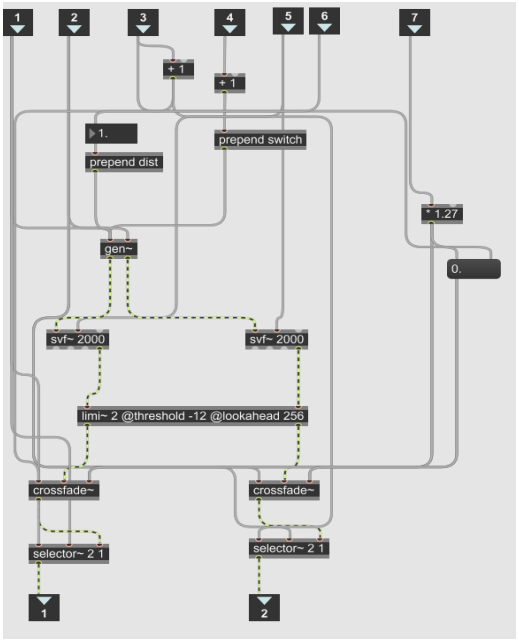
Obr. č. 6.12: Efekt chorus

Efekt chorus je znovu tvořen objekty *crossfade~* a *selector~* pro vytvoření faderu mezi Dry a Wet signálem a On/Off přepínač.

Procesování signálu probíhá pomocí LFO, které mění zpoždění signálu. LFO je tvořeno sinusovým generátorem, jehož hodnoty jsou pomocí objektu *snapshot~* převedeny na data. Data jsou poté matematicky upravena a přivedena na potenciometr, jenž ovládá zpoždění signálu. Zpoždění signálu je realizováno objektem *delay~* (o jeho využití je více v kapitole Reverb). Zpožďovaný signál v levém kanále je oproti signálu z pravého kanálu zpožděn ještě o danou konstantu. Objekt *live.gain~* je použit pro nastavení výstupního gainu tak, aby nedocházelo k přebuzení signálu.

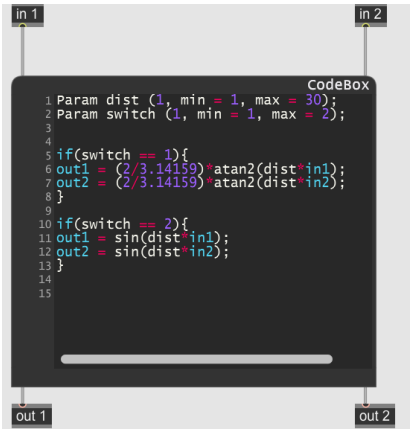
Na vstup objektu jsou přivedeny: signály levého a pravého kanálu, On/Off přepínač, *Gain* potenciometr, Dry/Wet fader a potenciometr pro nastavení frekvence LFO. V patchi jsou výstupy pro signál levého a pravého kanálu. [12]

6.5.5 Waveshaper



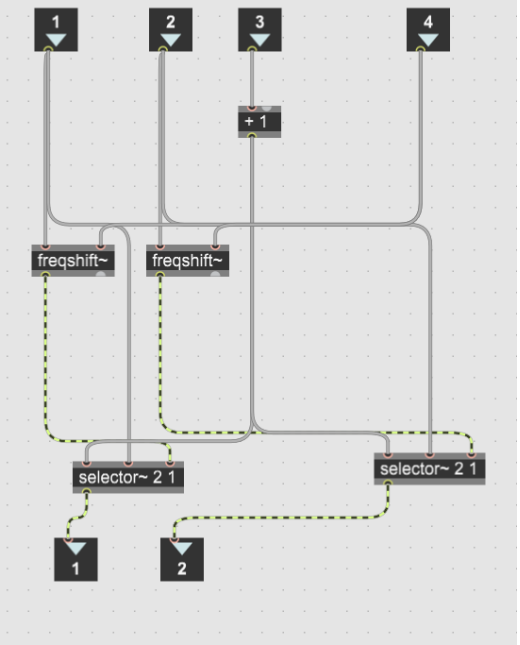
Obr. č. 6.13: Waveshaper

Waveshaper obsahuje, stejně jako další efekty, objekty *crossfade~* a *selector~*, které zde plní již uvedené funkce. Hlavní součástí tohoto efektu je objekt *gen~*, ve kterém je signál procesován. Pro tvarování signálu jsou použity funkce *arkus tangens* a *sinus*. Mezi nimi lze přepínat (což je realizováno objektem *live.tab* a parametrem *switch* uvnitř objektu *gen~*). Do argumentů těchto funkcí je vždy přiveden signál, který se násobí s parametrem *dist*. Tak je docíleno požadovaného zkreslení. Dále je signál procesován filtrem typu pásmová zádrž z objektu *svf~*, kterým lze signál ještě dále tvarovat. Před výstup je zařazen objekt *limi~*, nastavený tak, aby výstupní signál nebyl moc silný v porovnání s neprocesovaným signálem. [12][17]



Obr. č. 6.14: Waveshaper – použité vzorce

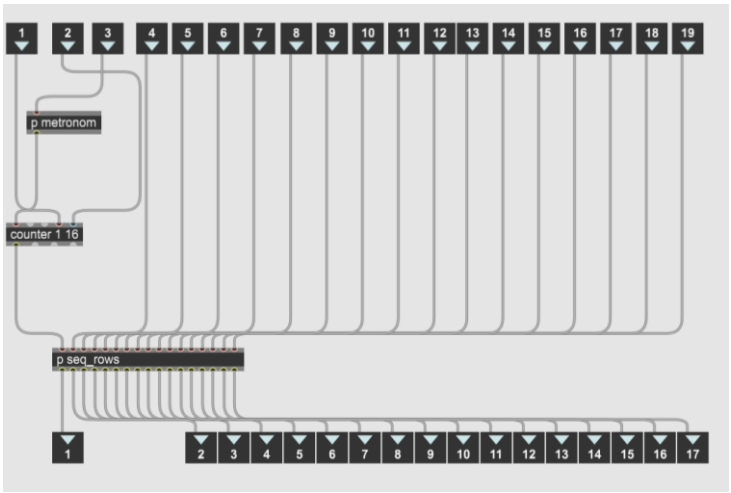
6.5.6 Frequency shifter



Obr. č. 6.15: Frequency shifter

Hlavním prvkem efektu frequency shifter je objekt *freqshift~*. Ten provádí posun frekvenční posun pomocí jednoho z postranních pásem kruhové modulace. Na vstup objektu je přiveden audio signál a frekvence, o kterou je třeba signál posunout. Na výstupy objektu jsou vyvedena postranní pásma (na každý výstup jedno). Dále se zde nachází objekt *selector~*, který slouží k zapnutí a vypnutí efektu. [12]

6.6 Sekvencér

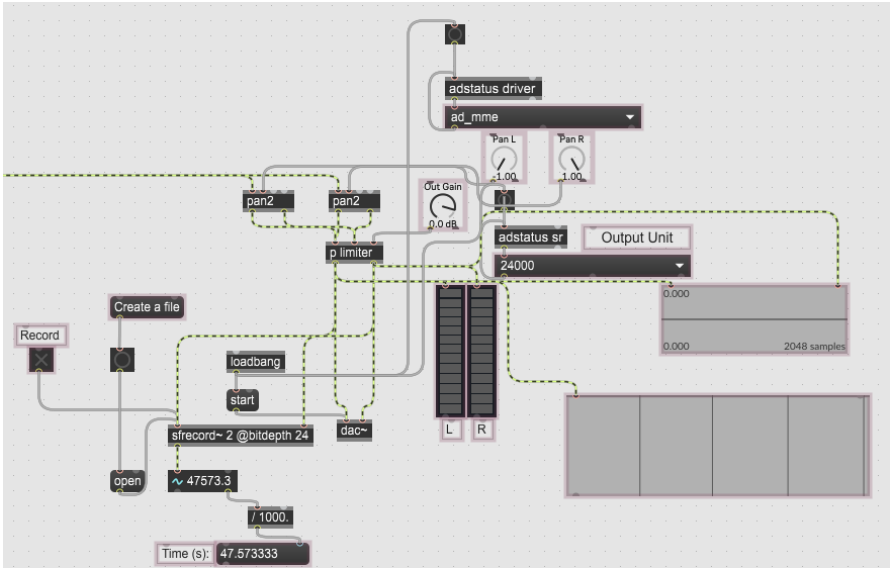


Obr. č. 6.16: Řada sekvencéru

Základním prvkem sekvencéru je metronom. Tempo metronomu je určeno jakýmkoliv objektem, na jehož výstupu jsou čísla, zavedeným do objektu *transport*, se zprávou *tempo*. V patchi *metronom* je přítomen objekt *metro*, nastavený tak, aby vysílal zprávu *bang* v určené subdivizi zadaného tempa. Poté signál putuje do objektu *counter*, na kterém je možné nastavit maximální počet kroků (1–16). Z něj jde signál do patche *seq_rows*, ve které je pomocí objektů *toggle* na vstupu a logických hradel *and* upravován tak, že když je na objektu *toggle*, který je určený pro konkrétní krok, hodnota 1, objeví se hodnota 1 i na výstupu patche. Sekvencér existuje ve třech verzích.

Verze ARP obsahuje arpegiátor, ovládaný daty z videa, jehož úkolem je generování čísla v rozsahu 48 hodnot (−24 až 24). To je následně přičteno k číslu aktuálně stisknuté noty. Verze MEMORY umožňuje nahrání sekvence až šestnácti not. Základní verzi, vysílající pouze hodnoty 0 a 1, lze použít pro spouštění samplů. [12]

6.7 Výstupní modul



Obr. č. 6.17: Patch pro výstup audio signálu

První v signálové cestě v tomto modulu jsou dva patche *pan2*, pro pravý a levý kanál, které umožňují rozhození kteréhokoliv ze signálů kamkoliv do stereo báze. Dále se výstupní modul skládá z patche, jenž obsahuje tři vstupy, dva pro levý a pravý kanál a třetí pro vstup potenciometru *gain*. Potenciometr *gain* ovládá objekt *live.gain~* , který je posledním stupněm, na němž lze ovládat zeslabení, či zesílení, signálu. *Live.gain~* je poté přiveden na objekt *omx.peaklim~*, který zde slouží k tomu, aby signál nikdy nepřekračoval hodnoty 1 a −1.

Použitými vstupy objektu jsou vstupy pro levý a pravý kanál, použité výstupy jsou také pouze ty pro levý a pravý kanál. Odtud jde signál na objekty *dac~*, *scope~*,

spectroscope~, *meter~* a *sfrecord~*. *Dac~* signál odvádí na audio hardware, nastavený v konfiguraci programu *Max* s názvem *Audio Status*. K té se lze dostat buď pomocí záložky *Options>Audio Status*, nebo dvojklikem na objekt *dac~*. Objekt *dac~* je při spuštění programu aktivován odesláním zprávy *Start* pomocí objektu *loadbang*.

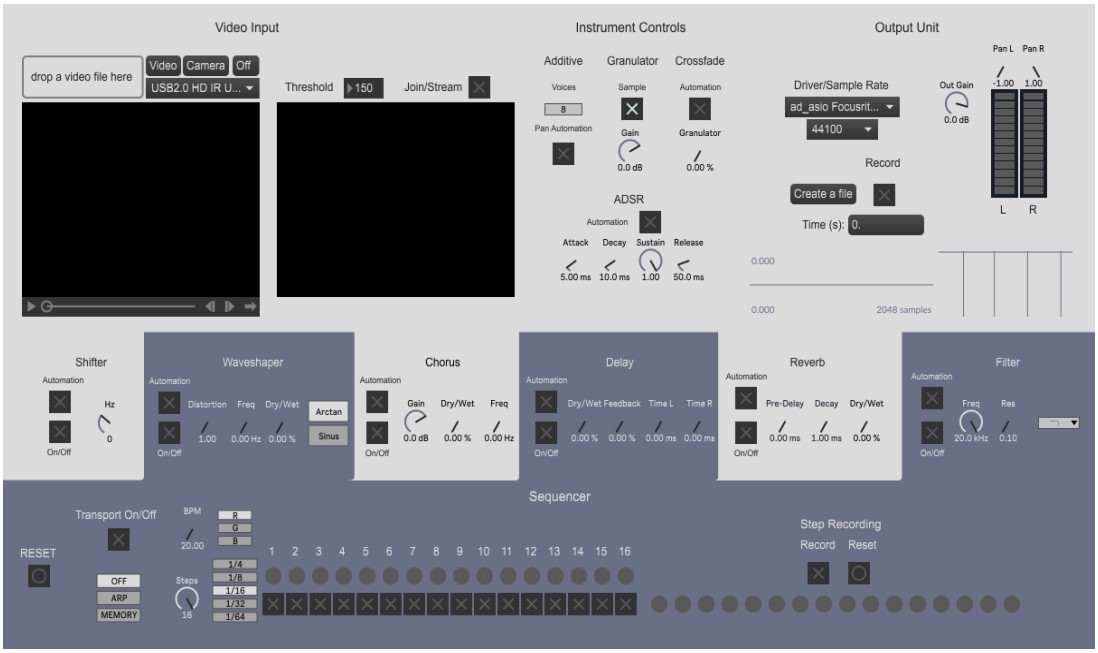
Scope~ monitoruje časový průběh výstupního signálu, *spectroscope~* zobrazuje jeho spektrum a dva objekty *meter~* vizualizují hlasitost signálu pro levý a pravý kanál. Objekt *sfrecord~*, který nahrává výstup nástroje, je konfigurován tak, aby po stisknutí ovládacího prvku *bang* odeslal zprávu *open*, čímž je otevřeno okno, v němž lze vybrat název a typ audio souboru. Dále umístění souboru v počítači, na který je nahráván. Nahrávání se poté spouští objektem *toggle* a je spuštěno, pokud tento objekt vysílá hodnotu 1. U nahraného audio souboru je dané bitové rozlišení (24 bitů). Jako vzorkovací frekvence nahrávání je použita hodnota, nastavená v audio driveru a typ souboru je možno zvolit z následujících: .wav, .aiff, .ogg, .mp3, .flac a .data.

Do výstupního modulu byly dále zařazeny dva objekt *adstatus* s argumenty *driver* a *sr*. Tyto objekty, společně s grafickým rozhraním objektu *umenu*, umožňují výběr audio ovladače a vzorkovací frekvence. [12]

7. REALIZACE

Cílem této kapitoly je obecný popis vytvořených hudebních nástrojů a základní popis jejich funkcí a ovládacích prvků. Pro detailní popis ovládacích prvků je připraven manuál.

7.1 Nástroj 1 – Kombinace aditivní a granulární syntézy



Obr. č. 7.1: Nástroj 1 – grafické rozhraní

První nástroj se skládá z video vstupu, ovládání nástrojů, sekvencéru, efektové jednotky a výstupní jednotky. Video vstup je realizován pomocí modulu, který je pro tento účel určený. Jako způsob zisku dat byla použita *Metoda 2*. Data získaná touto metodou bylo třeba řádně matematicky upravit, aby mohla sloužit k ovládání parametrů nástroje. Hodnoty dat z videa se totiž pohybují mezi 0 a 320, hodnoty parametrů nástroje nikoliv.

Pro zisk zvuku je použit aditivní syntezátor, jehož harmonické složky (jejich amplituda a rozmístění v prostoru) jsou modulovány daty z videa, a granulátor, který do bufferu ukládá vždy posledních 1000 ms, zahraných na aditivní syntezátor (pokud je jeho ovládací prvek nadepsaný *Sample* v pozici zapnuto).

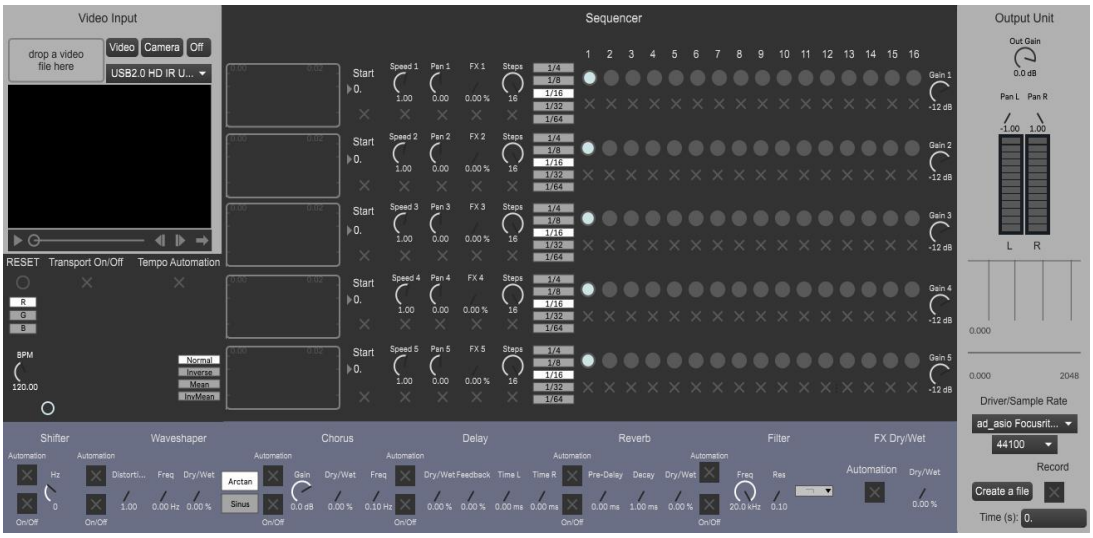
Ovládací prvky modulů pro tvorbu zvuku umožňují: zapnout a vypnout rozmístění harmonických složek v prostoru, podle dat získaných z videa (průměr levého a pravého okraje boxu), určit maximální počet hlasů, zapnout a vypnout funkci *Sample*, ovládat zesílení signálu z granulátoru, kontrolovat hodnotu prolínání mezi oběma nástroji (a její automatizaci daty z videa) a ovládat obálku ADSR aditivního syntezátoru (lze automatizovat daty z videa).

Sekvencér má na výběr ze tří režimů a je ovládán pomocí *Metody 1* (BPM je získáno jako průměr hodnot zvolené roviny videa). Režim *OFF* vypíná sekvencér a jako zdroj *MIDI* dat používá zprávy posílané kontrolérem. *ARP* zapíná sekvencér, čímž parametr *Voices* klesne na hodnotu 1 (tedy se syntezátor stane monofonním), aby nedocházelo k přetížení nástroje. Dále se zdroj *MIDI* přepne do konfigurace, ve které je k původní *MIDI* zprávě, typu Note On, přičítána proměnná, jejíž hodnota je ovládána daty z videa. Režim *MEMORY* umožňuje nahrání sekvence o délce až 16 stop. V rámci sekvencéru je možné ovládat: zapnutí a vypnutí konkrétního kroku, počet kroků, zapnutí a vypnutí sekvencéru (Transport On/Off), rovinu, ze které jsou získávána data z videa, délku jednoho kroku, *RESET* (návrat k prvnímu kroku) a parametry *Record* a *Reset* pro nahrávání kroků.

Efektová jednotka obsahuje ovládací prvky pro zapnutí a vypnutí efektů a pro jejich automatizaci pomocí dat z videa. Dále jsou také v rámci efektů k dispozici klasické ovládací prvky.

Výstupní jednotka nabízí možnost úpravy výstupní hlasitosti a možnost dodatečného rozmístění signálu z levého a pravého kanálu v prostoru. Signál, který z nástroje vychází, lze nahrát (nejdříve je třeba vytvořit soubor, poté spustit nahrávání, pomocí určených ovládacích prvků). [12]

7.2 Nástroj 2 – Sampler/Sekvencér



Obr. č. 7.2: Nástroj 2 – grafické rozhraní

Druhý nástroj se od prvního liší zdrojem dat z videa a zdrojem zvuku. Zdroj dat videa je v tomto případě *Metoda 1* (z níž získaná data také podstoupila vhodnou úpravu). Lze ovládat rovinu a konkrétní pozici ve videu, ze které jsou data získávána (až na parametr BPM – ten je ovládán průměrem hodnot zvolené roviny). Hodnoty získané touto metodou se pohybují mezi 0 a 255.

Základ sekvencéru je stejný, jako u prvního nástroje. Zde však sekvencér slouží pouze ke spouštění zvukových souborů. Ty jsou uloženy do bufferů pomocí příslušného ovládacího prvku (přetáhnutí a vložení do objektu *dropfile*, po úspěšném vložení vzorku se objeví časový průběh vloženého zvuku). Každá z pěti stop sekvencéru obsahuje několik parametrů, které lze kontrolovat. Jde o *Start*, *Speed*, *Pan*, *FX*, *Steps*, Délka kroku a *Gain*. *Start* určuje, od jaké ms se vzorek začne přehrávat, *Speed* určuje rychlost přehrávání vzorku (záporné hodnoty fungují pouze pokud *Start* není na hodnotě 0), *Pan* určuje rozmístění vzorku ve stereo bázi a *FX* ukazuje, kolik procent signálu je odesláno na efektovou jednotku. Tyto parametry lze automatizovat daty z videa. Parametry Délka kroku a *Gain* automatizovat nelze.

Výstupní jednotka a modul s efekty jsou stejné, jako u prvního nástroje, pouze jsou jinak rozmístěny v rámci grafického rozhraní. [12]

8. ZJIŠTĚNÉ HUDEBNÍ MOŽNOSTI NÁSTROJŮ

Oba vytvořené nástroje byly podrobeny zkouškám, jejichž cílem bylo nalezení a popsání hudebních možností nástrojů.

8.1 Kombinace aditivní a granulární syntézy

Zvuk, vycházející z tohoto nástroje je kombinací aditivní a granulární syntézy. Možnosti aditivního syntezátoru jsou plně závislé na videu, které je přehráváno. Počet objektů (a umístění jejich krajních bodů), nalezených pomocí *Metody 2*, přímo koresponduje s počtem harmonických složek, které jsou přehrávány a s jejich amplitudou a umístěním v prostoru. Lze ovládat parametry amplitudové obálky aditivního syntezátoru. Takto vzniklý výsledný zvuk však není moc zajímavý, proto je k dispozici ještě granulátor.

Ten obsahuje buffer, který vzorkuje vždy poslední vteřinu signálu z aditivního zdroje zvuku. Granulátor poté vezme signál z bufferu a přehraje ho určeným způsobem (parametry jsou ovládány daty z videa). Dále lze daty ovládat parametry efektové jednotky a prolínání mezi signálem z aditivního syntezátoru a z granulátoru. Nástroj obsahuje také sekvencér, který přispívá k zajímavějším zvukovým možnostem hlavně díky funkci *ARP*. Tato funkce ovlivňuje výšku tónu pomocí dat z videa. Výsledný zvuk je proměnlivý v závislosti na vloženém videu, po delší době poslechu však začne být předvídatelný.

Co se obrazu týče, nabízí se možnost vstupu jak pomocí vložení videa, tak pomocí kamery připojené k zařízení, na kterém je program spuštěn. Nastavením parametru *Threshold* dochází ke změně prahu detekce objektů. Lze tak přibližně kontrolovat počet detekovaných objektů (tedy i harmonických složek) a hlavně lépe přizpůsobit nástroj světlejšímu, nebo tmavšímu prostředí. Nástroj, vzhledem k použité metodě zisku dat, reaguje na pohyb. Tím pádem vstup s větším množstvím pohybu způsobí více změn v barvě výsledného zvuku nástroje než vstup, který je statický.

Nástroj může být, díky své schopnosti modulace parametrů zvuku v závislosti na videu v reálném čase, využit například při tvorbě doprovodné hudby k vizuálním dílům. Dále se dá použít k neobvyklému způsobu zvukového designu.

Tvorba zvuku je relativně podobná nástrojům, zmíněným v kapitole o optoelektrické syntéze. Rozdílem proti těmto nástrojům je však možnost ovládání pomocí MIDI, generování takto ovládaného signálu v reálném čase, použití harmonických složek (u dříve zmíněných nástrojů jsou použity i jiné složky signálu) a využití zvukových efektů. Dalším důležitým rozdílem je, v tomto případě, použití jiného způsobu zisku dat z videa než u zmíněného softwaru. Zatímco běžné optoelektrické nástroje berou data z informace o intenzitě barvy (buď v rovinách ARGB nebo v monochromatickém

zobrazení) a umístění pixelu, tento nástroj používá informaci o umístění objektu v obraze (získanou pomocí objektů z knihovny *cv.jit*). [4]

Efektová jednotka, přítomná v obou nástrojích, umožňující automatizaci pomocí dat z videa, přináší do nástrojů prvek velmi podobný náhodné automatizaci parametrů v DAW, nebo pluginům jako Glitch (efektová jednotka s možností ovládání parametrů efektů v grafickém rozhraní na bázi sekvencéru). V porovnání s těmito možnostmi má efektová jednotka jednodušší ovládání a méně efektů, dala by se však také využít jako samostatná aplikace.

Ačkoliv sekvencér v prvním nástroji nefunguje tak, jak by měl (občas neregistruje vypnutý krok jako podnět k vypnutí momentálně přehrávaného zvuku), jedná se stále o funkci, která rozšiřuje zvukové možnosti nástroje a odlišuje ho od jiných nástrojů, zabývajících se tímto typem syntézy.

8.2 Sampler/Sekvencér

Výsledný zvuk nástroje je závislý jak na vložených zvukových vzorcích, tak na vloženém videu. Nástroj je schopen nahrát vzorky a pouštět je v sekvenci (což nástroje tohoto typu zvládají běžně), nabízí ale i pokročilejší ovládání stop, ve kterých jsou vzorky umístěny (možnosti volby délky a počtu kroků pro každou stopu zvlášť). Dále je k dispozici efektová jednotka, která značně rozšiřuje zvukové možnosti nástroje.

Nejdůležitějším prvkem je však možnost ovládání většiny těchto parametrů pomocí dat z videa. Díky změně, buď samotného video signálu, nebo změnou souřadnic, ze kterých jsou data čtena, dochází ke změně parametrů přehrávání zvukového vzorku i efektů, aplikovaných na vzorek. Tak je dosaženo zvuku, který, i při použití poměrně jednoduché sekvence, dokáže být zajímavý a proměnlivý po delší časový úsek.

Data, získávaná z obrazového vstupu nástroje, jsou informace o hodnotě barvy v konkrétní rovině a pixelu obrazu (až na ovládání parametru BPM, které průměruje data z celé roviny). Pro rychlejší změnu dat, a tedy i výsledného zvuku nástroje, je potřeba použít buď obraz, na němž se rychle mění barvy, nebo ovládací prvky, schopné měnit pozici, ze které jsou data brána.

Nástroj lze využít jak pro účely scénické hudby, tak pro zvukový design. Pomocí ovládání vybraných prvků daty z videa lze také, při nastavení fixního BPM, tvořit zajímavé smyčky, použitelné k ozvláštnění kompozic s daným tempem, které by jinak musely být tvořeny zdlouhavě, například pomocí automatizace v příslušném softwaru. Nevýhodou pro tvorbu kompozice k vizuálnímu dílu v reálném čase může být poměrně velké množství ovládacích prvků.

Nejvíce podobný je nástroj softwaru *Blip* (jedná se také o sekvencér). Oproti tomuto nástroji však obsahuje méně stop a nabízí možnost zisku dat z videa výběrem roviny a souřadnic, ze kterých jsou data sbírána (v případě *Blip* jde o kreslení do matice o rozměrech 64x64). [4]

9. ZÁVĚR

Cílem práce bylo vytvoření experimentálního softwarového nástroje s neobvyklým způsobem tvorby zvuku. Jako způsob tvorby zvuku byla zvolena modulace parametrů zvukových zdrojů pomocí dat z videa.

Práce obsahuje teoretický úvod, který pojednává o digitálním zpracování audio signálu, o syntéze zvuku, o optoelektrické syntéze, o MIDI a o programu Max/MSP/Jitter. Praktická část práce obsahuje kapitoly o tvorbě jednotlivých částí nástrojů, o nástrojích a o možnostech jejich využití při tvorbě hudby.

Oproti semestrální práci byla bakalářská práce výrazně rozšířena. Syntezátor, vytvořený v rámci semestrální práce, byl obohacen o granulátor a o možnost automatizace rozmístění harmonických složek v prostoru. Zároveň byla změněna metoda, pomocí které získává nástroj data z videa. Efektová jednotka byla rozšířena o efekty frequency shifter a waveshaper. K výstupní jednotce přibyly možnosti volby audio driveru a vzorkovací frekvence, korekce rozmístění levého a pravého kanálu ve stereo bázi a možnost nahrání výstupního signálu. V rámci bakalářské práce byl také vytvořen zcela nový nástroj, fungující na principu sampleru a sekvencéru.

Cíl práce byl tedy splněn. Byly vytvořeny dva nástroje, schopné tvořit zvuk experimentálním způsobem, které se zároveň liší od současných dostupných možností tvorby zvuku z obrazu.

Pokud by byly nástroje i po ukončení této práce dále rozvíjeny, mohlo by dojít k propojení objektů, které by bylo méně náročné na procesor. Dále by bylo možné vylepšit grafické rozhraní. Rozhraní je pro účely této práce dostačující. Dalo by se však ještě jistě zbavit některých viditelných ovládacích prvků, popřípadě změnit jejich rozmístění a tím i celkový vzhled nástrojů.

LITERATURA

[1] ZÖLZER, Udo. *DAFX: digital audio effects*. 2nd ed. Chichester: Wiley, 2011. ISBN 978-0-470-66599-2.

[2] ZÖLZER, Udo. *Digital audio signal processing*. 2nd ed. Chichester: Wiley, 2008. ISBN 978-0-470-99785-7.

[3] RUSS, Martin. *Sound Synthesis and Sampling.* 3rd ed. Oxford: Focal Press, 2009. ISBN 978-0-240-52105-3.

[4] DLOUHÝ, Dan. *Počítačem podporovaná algoritmická kompozice.* V Brně: Janáčkova akademie múzických umění, 2018. ISBN 978-80-7460-141-5.

[5] FORRÓ, Daniel. *Musitronika: elektroakustické hudební nástroje.* Brno: Janáčkova akademie múzických umění v Brně, 2004. ISBN 80-854-2939-X.

[6] HOLZER, Derek. *A Brief History Of Optical Synthesis. UMATIC [online]*. Utrecht, 2007 [cit. 2022-12-06]. Dostupné z: http://www.umatic.nl/tonewheels_historical.html

[7] *S I L H O U E T T E – Synthesizer [online].* Embsen: SILHOUETTE, 2019 [cit. 2022-12-06]. Dostupné z: http://silhouette-synthesizer.de/

[8] JACK, Olivia. *PIXELSYNTH [online].* 2016 [cit. 2022-12-07]. Dostupné z: https://ojack.xyz/PIXELSYNTH/

[9] ROUZIC, Michel. *Photosounder [online].* 2008 [cit. 2022-12-07]. Dostupné z: https://photosounder.com/

[10] *Summary of MIDI 1.0 Messages. Midi.org [online*]. MIDI Association, 2019 [cit. 2022-12-06]. Dostupné z: https://www.midi.org/specifications-old/item/table-1-summary-of-midi-message

[11] SCHIMMEL, Jiří. *Komunikační rozhraní MIDI. Elektrorevue [online]*. Brno: ISES, 2002 [cit. 2022-12-06]. Dostupné z: http://www.elektrorevue.cz/clanky/02069/index.html

[12] *Max 8 Documentation [online].* San Francisco: Cycling '74, 2019 [cit. 2022-12-06]. Dostupné z: https://docs.cycling74.com/max8

[13] *Building Max with JUCE. JUCE [online]*. London: JUCE, 2018 [cit. 2022-12-07]. Dostupné z: https://juce.com/discover/stories/building-max-with-juce

[14] *About Cycling '74. Cycling '74 [online]*. San Francisco: Cycling '74, c2022 [cit. 2022-12-07]. Dostupné z: https://cycling74.com/company

[15] *Sound. In: Wikipedia: the free encyclopedia [online]*. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-12-07]. Dostupné z: https://en.wikipedia.org/wiki/Sound

[16] W. SMITH, Steven. *The Scientist and Engineer's Guide to Digital Signal Processing*. 2nd ed. San Diego: California Technical Publishing, 1999. ISBN 0-9660176-6-8.

[17] VIATOR, Landon. *Prototyping Distortion Effects In MaxMSP For JUCE*. In: *The Audio Programmer, Youtube* [online]. 12. 11. 2021 [cit. 2023-04-05]. Dostupné z: https://www.youtube.com/watch?v=uDSiVz0ZjFQ

[18] ROBERTS, Ed. *Max/MSP Tutorial | A granular synthesiser built with [codebox] in gen~.* In: *Toneparticle, Youtube* [online]. 7. 4. 2022 [cit. 2023-04-05]. Dostupné z: https://www.youtube.com/watch?v=VU2TQmxte9A&t=5s

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

FFT	*Fast Fourier transform* – Rychlá Fourierova transformace
IFFT	*Inverse Fast Fourier transform* – Zpětná rychlá Fourierova transformace
AD	*Analog to digital* – Analogový na číslicový
DA	*Digital to analog* – Číslicový na analogový
THD	*Total harmonic distortion* – Celkové harmonické zkreslení
FIR	*Finite impulse response* – Konečná impulsní odezva
IIR	*Infinite impulse response* – Nekonečná impulsní odezva
VCO	*Voltage controlled oscillator* – Napětím řízený oscilátor
VCA	*Voltage controlled amplifier* – Napětím řízený zesilovač
VCF	*Voltage controlled filter* – Napětím řízený filtr
EG	*Envelope generator* – Generátor obálky
LFO	*Low frequency oscillator* – Nízkofrekvenční oscilátor
AM	*Amplitude modulation* – Amplitudová modulace
RM	*Ring modulation* – Kruhová modulace
FM	*Frequency modulation* – Frekvenční modulace
DAW	*Digital Audio Workstation* – Digitální zvuková pracovní stanice
WAV	*Waveform audio file format* – Formát souboru pro uložení zvukové informace
DIN	*Deutsches Institut für Normung* – Německý institut pro normalizaci
MIDI	*Musical Instrument Digital Interface* – Digitální rozhraní hudebního nástroje
USB	*Universal Serial Bus* – Univerzální sériová sběrnice
MSB	*Most Significant Bit* – Nejvýznamější bit
LSB	*Least Significant Bit* – Nejméně významný bit
API	*Application Programming Interface* – Programovací rozhraní aplikace
VST	*Virtual Studio Technology* – Virtuální studiová technologie
OpenGL	*Open Graphics Library* – Otevřená grafická knihovna
GPU	*Graphics processing unit* – Grafický procesor
CPU	*Central processing unit* – Centrální procesorová jednotka
ADSR	*Attack, Decay, Sustain, Release* – Nástup, Útlum, Podržení, Uvolnění
SNR	*Signal to noise ratio* – Poměr signálu vůči šumu

RTC-lib	*Realtime Composition Library* – Knihovna pro kompozici v reálném čase
AU	*Audio unit* – Zvuková jednotka
OSC	*Open Sound Control* – Otevřená kontrola zvuku
BPM	*Beats per minute* – počet úderů za minutu

SEZNAM PŘÍLOH

Manuál.pdf

Příloha A – Manuál k nástrojům

Projekty - spustitelné v programu Max.zip

Příloha B – Projekty s nástroji