VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 Marek Bezůšek

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 NA PLATFORMĚ ANDROID, KOMBINUJÍCÍ SAMPLER, SYNTETIZÉR, EFEKTOVÝ PROCESOR A SEKVENCER

EXPERIMENTAL SOFTWARE MUSICAL INSTRUMENT ON ANDROID PLATFORM COMBINING SAMPLER, SYNTHESIZER, EFFECTS PROCESSOR AND SEQUENCER

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Marek Bezůšek

AUTHOR

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

SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Audio inženýrství**
Ústav telekomunikací

Student: Marek Bezůšek ***ID:*** 195797
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Experimentální softwarový hudební nástroj na platformě Android, kombinující sampler, syntetizér, efektový procesor a sekvencer

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je kompletně naprogramovat mobilní aplikaci, kterou je experimentální hudební nástroj pro platformu Android. Experimentálnost spočívá v kombinaci jednoduchého sampleru, syntetizéru, efektového procesoru a sekvenceru v rámci mobilní platformy.

DOPORUČENÁ LITERATURA:

[1] Electronic abd experimental music: technology, music and culture. Editace Thom Holmes. 3. vydání, Routledge, New York, 2008, 462 s. ISBN 978-0-415-95782-3.

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

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 27.5.2019

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

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Cílem práce je navrhnout a realizovat experimentální softwarový hudební nástroj na platformě Android, kombinující sampler, syntetizér, efektový procesor a sekvencer. Aplikace je navržena pro mobilní telefony. Experimentálnost aplikace spočívá v použití granulární syntézy na nahrané zvuky a s kombinací syntetických zvuků wavetable syntetizéru a efektů.

KLÍČOVÁ SLOVA

Softwarový hudební nástroj, sampler, syntetizér, efektový procesor, sekvencer, Pure Data, Android

ABSTRACT

The aim of this thesis is to design and realize experimental software musical instrument on Android platform, combining sampler, synthesizer, effects processors and sequencer. The application is designed for smartphones. The application's experimental approach is in granular synthesis applied on recorded sounds with a combination of synthetic sounds produced in wavetable synthesizer and with applied effects.

KEYWORDS

Software musical instrument, sampler, synthesizer, effect processor, sequencer, Pure Data, Android

BEZŮŠEK, Marek. *Experimentální softwarový hudební nástroj na platformě Android, kombinující sampler, syntetizér, efektový procesor a sekvencer*. Brno, 2019, 48 s. 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

Prohlašuji, že svou bakalářskou práci na téma „Experimentální softwarový hudební nástroj na platformě Android, kombinující sampler, syntetizér, efektový procesor a sekvencer“ 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/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. MgA. Mgr. Danu Dlouhému, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Obsah

Seznam obrázků

Úvod

Tato bakalářská práce se zabývá strukturou a metodami konstrukce digitálního experimentálního hudebního nástroje pro mobilní operační systém Android. Princip experimentálnosti tohoto nástroje spočívá ve využití okolních běžných zvuků, které lze obohatit a přetvořit ve složitější hudebně zajímavé struktury. V historii lidé experimentovali se zvukem a hudbou různými způsoby. Inovace v jedné oblasti lidské činnosti otevírají nové možnosti pro ostatní oblasti. S dokonalejší technologii vznikají možnosti ji využít hudebně. Téměř každý v dnešní době u sebe nosí mobilní telefon. Tyto zařízení dnes díky vývoji technologií přesahují jejich původní účel a lze je využít hudebně ke kreativní interakci se zvukem v našem okolí.

První část práce teoreticky popisuje procesy, které byly při realizaci aplikace použity. Druhá část práce je zaměřena na návrh jednotlivých částí experimentálního hudebního nástroje a řešení grafického rozhraní v aplikaci. Ve třetí části je vysvětlen postup programování aplikace.

V první části teorie je vysvětlen princip některých druhů syntéz zvukových signálů, které budou aplikovány při konstrukci tohoto hudebního nástroje. Druhá teoretická část zkoumá zpracování digitálních signálů. Témata v této části jsou seřazena podle průběhu zpracování digitálního signálu, od vstupu signálu do digitálního systému po jeho výstup. Jsou zde popsány problémy, které při tomto procesu vznikají a jejich řešení. Protože je tento hudební nástroj konstruován jako aplikace pro operační systém Android, je tomuto systému věnována další část. Ve čtvrté části je vysvětlena funkce programovacího jazyka Pure Data, ve kterém je programována hlavní část aplikace. Protože mezi sebou Pure Data a operační systém Android nemohou přímo komunikovat, bude v poslední teoretické části představena platforma MobMuPlat (Mobile Music Platform), prostřednictvím které jsou systémy propojeny. Pomocí této platformy je také vytvořeno grafické uživatelské rozhraní (GUI) aplikace.

Druhá a třetí část postupně popisuje sekce aplikace rekordér, sampler, granulární syntetizér, wavetable syntetizér a efekty. V druhé části jsou popsány funkce aplikace se zaměřením na uživatelské rozhraní a ve třetí části je popsána funkce "audio enginu".

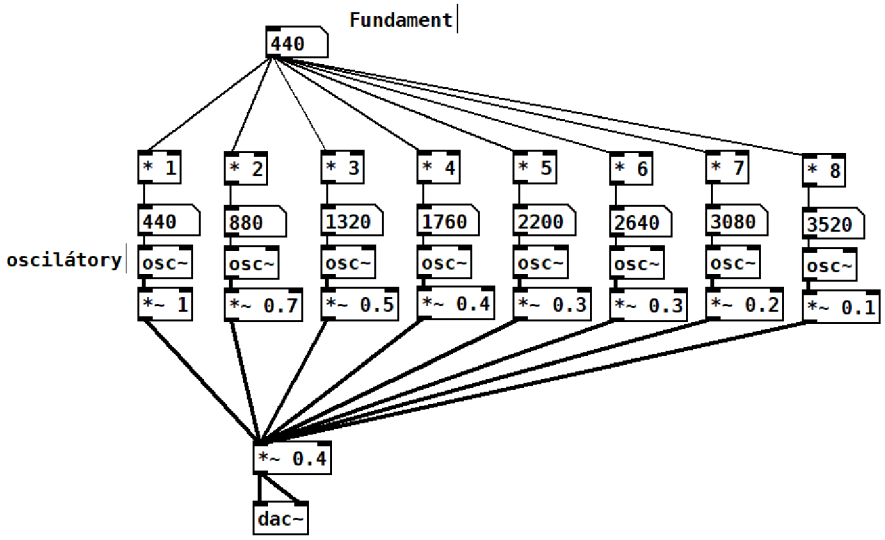
1 Teorie

1.1 Druhy syntéz signálů

Obsah této části je zaměřen na princip fungování druhů syntéz, které budou využity v tomto hudebním nástroji.

1.1.1 Aditivní syntéza

Aditivní syntéza vychází z principu Fourierovy řady, komplexní signál lze vyjádřit jako součet funkcí sinus. Barva zvuku je tvořena přidáváním spektrálních složek s různou amplitudou a frekvencí. Aditivní syntetizéry se skládají z několika oscilátorů s různým zesílením jejichž výstup se sčítá. Výsledná barva pak vzniká jejich upravením amplitudovou obálkou. V případě analogových syntetizérů se používá napětím řízený oscilátor VCO, napětím řízený zesilovač VCA a generátor obálky EG.



Obr. 1.1: Harmonická aditivní syntéza v Pure Data

Nástroje, které využívaly nebo využívají aditivní syntézu jsou například varhany, Telharmonium, Hammondovy varhany, Synclavier.

1.1.2 Subtraktivní syntéza

Při subtraktivní syntéze se používají filtry pro omezení harmonicky bohatého spektra, vzniká tak nová spektrální obálka, která se od původní liší frekvenční odezvou filtru. Střední, nebo mezní, kmitočet tohoto filtru se může v čase měnit pomocí nízkofrekvenčního oscilátoru (LFO). Pro kontrolu středního, nebo mezního, kmitočtu

filtru se často využívá signál kontrolující amplitudovou obálku. Díky tomu se barva zvuku mění zároveň s hlasitostí. Vzniká tak zvuk, který se podobá časovému průběhu zvuku hudebních nástrojů. Pro generování signálu s vyššími harmonickými složkami se používají oscilátory s tvarem signálu pila, čtverec, puls, trojúhelník.

Mezi první syntetizéry využívající tuto metodu patří Minimoog.

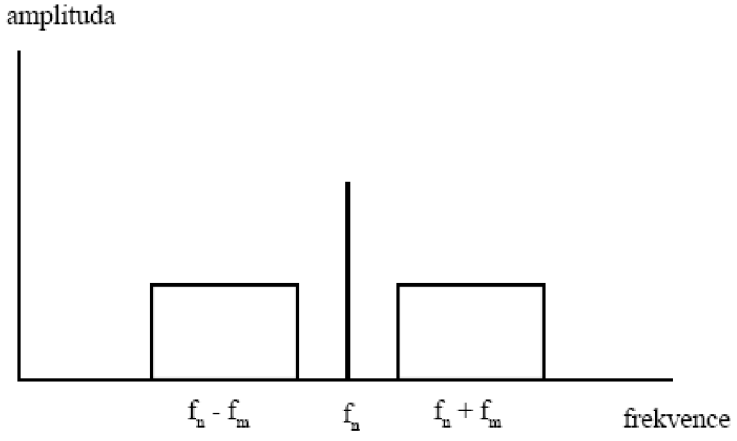
1.1.3 Modulační syntéza

Modulace je typ syntézy, při které je jeden signál (nosný) změněn signálem druhým (modulačním). Při modulaci vznikají nové (postranní) spektrální složky, jejichž frekvence a fáze je dána součtem a rozdílem frekvencí a fází nosného signálu se signálem modulujícím. Amplituda těchto nových složek klesá.

Modulace nosného signálu sinus:

$$\cos(\alpha n + \phi)\cos(\beta n + \xi) = 1/2[\cos((\alpha + \beta)n + (\phi + \xi)) + \cos((\alpha - \beta)n + (\phi - \xi))]$$

U komplexních signálů vznikají ještě kombinační složky, dané kombinací vyšších harmonických složek nosného signálu se signálem modulačním. Na obou stranách spektra nosného signálu tak vzniknou postranní pásma (angl. sidebands) těchto kombinačních složek.



Obr. 1.2: Modulace komplexního signálu f_n signálem f_m

Modulační index určuje míru modulace nosného signálu.

Změnou modulačních parametrů v čase vzniká spektrum (barva zvuku), které se mění v čase.

Modulační syntézy lze rozdělit na:

- Frekvenční modulace
- Amplitudová modulace
- Kruhová modulace (RM)

- Modulace šířky pulsu (Puls-Width modulation)
- Tvarovací (Waveshaping) modulace

Frekvenční modulace

V porovnání s aditivní a subtraktivní syntézou lze pomocí frekvenční modulace dosáhnout podobných zvuků ale s menším nárokem na výpočetní systém. Frekvence nosného signálu je při modulaci řízena modulačním signálem. Frekvence modulačního signálu určuje rychlost změny kmitočtu nosného signálu (angl. rate) a amplituda určuje míru modulace (rozsah změny kmitočtu nosného signálu tzv. hloubka modulace, angl. depth). Při nízkých frekvencích modulačního signálů vzniká efekt vibráto, při vyšších frekvencích se barva zvuku obohacuje o další spektrální složky. Množství generovaných spektrálních složek určuje míra modulace. Při modulaci může nastat aliasing, pokud se postranní pásma dostanou za hraniční Nyquistovu frekvenci.

Frekvenční modulací lze generovat harmonické spektra pokud poměr frekvencí oscilátorů je celé číslo, v druhém případě vzniká spektrum inharmonické.

Použitím podobné obálky průběhu pro nosný a modulační signál se amplituda a šířka pásma modulace mění zároveň. Tím lze dosáhnout změny hlasitosti a barvy zvuku ve stejný čas, podobně jako u jinných hudebních nástrojů.

Frekvenční syntetizéry používají i více oscilátorů, které jsou modulované jedním oscilátorem. Vzniká tak ve spektru více formantových oblastí. Formantové oblasti mohou mít jinou amplitudovou obálku ADSR (attack,decay, sustain, release). Lze tak simulovat průběh frekvenčního spektra jiných hudebních nástrojů.

Další možností je modulovat jeden oscilátor s nosným signálem, více modulačními oscilátory. Oscilátory mohou být v sériovém nebo paralelním zapojení.

FM syntézou lze vytvářet velké množství zvuků. Jeden z nejpopulárnějších FM syntetizérů byl DX7 od společnosti Yamaha.

Amplitudová modulace

Amplitudová modulace je jedna z nejstarších modulačních metod používaných v elektronické hudbě. Při amplitudové modulaci je okamžitá hodnota amplitudy nosného signálu měněna modulačním signálem. Pokud je frekvence modulačního signálu v pásmu frekvencí lidského slyšení (f_m>20Hz) vznikají ve slyšitelném spektru postranní pásma spektrálních složek. Pomalé změny amplitudy nosného signálu (f_m<20Hz) se projevují jako efekt tremolo. Narozdíl od kruhové modulace (RM) je u amplitudové modulace frekvence nosného signálu zachována. Další rozdíl oproti kruhové modulaci je, že modulační signál je unipolární.

Kruhová modulace (RM)

Kruhová modulace vzniká při násobení dvou bipolárních signálů. Je velmi podobná s amplitudovou modulací. Tvarem modulačního signálu je většinou sinus. Pokud je frekvence modulačního signálu v rozsahu frekvencí lidského slyšení vznikají slyšitelná postranní pásma modulace a nosná frekvence se ztrácí. Její název vychází z analogového zapojení kruhového diodového modulátoru.

Tvarovací (Waveshaping) modulace

Tvarovací modulace mění vstupní signál podle nelineární funkce. Vznikne tak více spektrálních složek kvůli nelineárnímu zkreslení vstupního signálu. Míra zkreslení je závislá na amplitudě vstupního signálu. Tím lze simulovat hudební nástroje, jejichž barva zvuku se mění s amplitudou. Nelineární zkreslení lze vytvořit digitálně, nebo analogově použitím nelineárních prvků v obvodu např. dioda, elektronka.

Modulace šířky pulzu (Puls-Width modulation)

Při této modulaci dochází ke změně šířky obdélníkového signálu. Střída obdélníkového signálu mám vliv na výskyt vyšších harmonických ve spektru. Modulací se tak mění spektrum v čase.

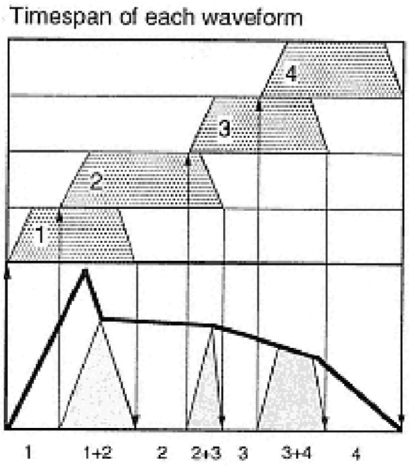
1.1.4 Wavetable syntéza

Wavetable je navzorkovaný signál uložen v paměti zařízení. Tento signál je periodicky přehráván metodou table-lookup. Wavetable look-up syntéza je základem pro digitální oscilátory. Při této syntéze vzniká statické spektrum, protože tvar signálu se nemění.

Wavetable syntézu lze rozdělit na dvě metody syntézy: wavetable crossfading a wavestacking. Tyto metody lze kombinovat.

Wavetable crossfading

Při této metodě digitální oscilátor přehrává několik signálů za sebou. Přechod z jednoho signálu na druhý je plynulý (crossfade). Vznikají tak zvuky které se prolínají v čase. Přechod mezi čtyřmi signály může být také řízený manuálně joystickem, tento typ syntézy se pak nazývá vektorová. První syntetizér se čtyřmi oscilátory, využívající vektorovou syntézu, byl Prophet VS.



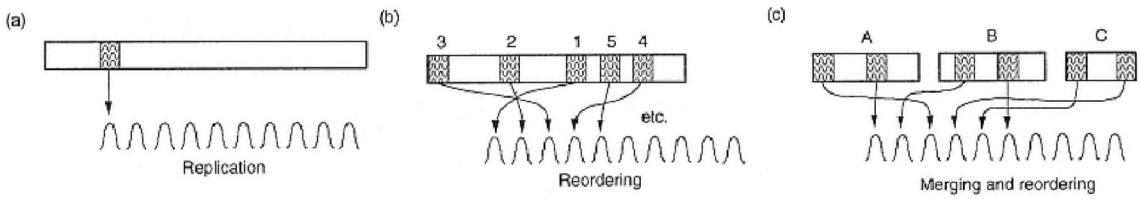
Obr. 1.3: Prolínání vzorků (Zdroj: [1])

Wavestacking (Wavetable stacking)

Tento typ wavetable syntézy pracuje na principu aditivní syntézy. Kromě jednoduchých signálů sinus lze ale použít složitější samplované zvuky.

1.1.5 Granulární syntéza

Tato syntéza pracuje se vzorky dlouhými 1 až 100ms, které jsou přehrávány ze zvukového souboru, nebo to mohou být syntetické signály, kombinace signálů sinus. Tyto vzorky se nazývají tzv. grainy. Každý vzorek má amplitudovou obálku, mohou se překrývat, přehrávat jinou rychlostí a s jinou frekvencí, vzorky mohou být jinak dlouhé a mít odlišnou startovní pozici. Vzorky mohou mít různé druhy amplitudových obálek, od jednoduché ASD (attack,sustain,decay) až po složitější např. pulsní. Při pomalém přehrávání vzorků vznikají komplexní atmosférické zvuky, při rychlejším přehrávání vzorky splývají a vznikají tóny.



Obr. 1.4: Příklad druhů extrakce vzorků ze zvukového souboru (Zdroj: [1])

1.2 Digitální zpracování signálů

V této části je popsán proces převodu analogového signálu na digitální, jeho zpracování v digitální oblasti a převod zpět na analogový signál. Při tomto procesu se původní signál degraduje. Jsou zde popsány metody jak tuto degradaci signálu snížit.

1.2.1 AD převod

Při převodu analogových signálů na číslicové vzniká zkreslení signálu, které je možné vhodným zpracováním signálu snížit tak, aby nebylo lidským sluchem rozpoznané.

Převod spojitého signálu na číslicový lze rozdělit do tří částí: vzorkování, kvantování, kódování.

1.2.2 Vzorkování

Vzorkováním je analogový spojitý signál převeden na posloupnost okamžitých hodnot signálu (diskrétní signál). Rychlost vzorkování je dána vzorkovací frekvencí. Nízká vzorkovací frekvence způsobuje zkreslení navzorkovaného signálu od původního spojitého signálu (tzv. aliasing). Signál musí být vzorkován aspoň dvěma vzorky na periodu signálu, aby se zamezilo aliasingu. Vzorkovací frekvence tedy musí být dva-krát větší než maximální frekvence signálu ($f_{vz} > 2f_{max}$). Toto pravidlo je stanoveno vzorkovacím teorémem (Nyquistův teorém). U komplexních signálů s bohatým spektrem vznikají vyšší harmonické složky, které snadno můžou tuto frekvenci přesáhnout. Dochází tak k aliasingu a ve spektru se objeví nové nežádoucí složky. Proto se používá před AD převodníkem anti-aliasingový filtr typu dolní propust pro odfiltrování složek překračující Nyquistovu mezní frekvenci ($f_{vz}/2$).

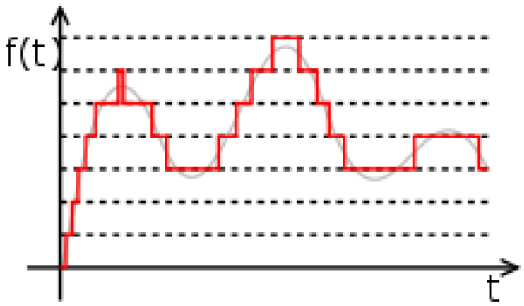
Rozsah lidského sluchu je 20 Hz až 20 kHz, proto pro splnění Nyquistova teorému musí být vzorkovací frekvence aspoň 40 kHz. Pro digitální záznam zvuku se používají vzorkovací frekvence 44,1 kHz (CD), 48 kHz (standardní vzorkovací frekvence používaná u profesionálních video zařízeních).

Pro reprodukci některých zvukových signálů není vždy potřeba celý rozsah frekvencí lidského sluchu. Proto je výhodné použít nižší vzorkovací frekvenci pro menší velikost dat. Nižších vzorkovacích frekvencí se využívá u telefonního přenosu ($f_{vz} =$ 8000 Hz). Srozumitelná řeč je ve frekvenčním rozsahu 300 Hz - 3,4 kHz.

V určitých případech je pro efektivnější zpracování dat vhodnější použít vzorkovací frekvenci vyšší než je dvojnásobek maximální frekvence. Například pro vytváření CD ($f_{vz} = 88.2$ kHz), DVD a Blu-ray Disc ($f_{vz} = 96$ kHz) . Výhody jsou například zlepšení efektivity filtrů, lepší poměr šumu a požadovaného signálu, snížení kvantizačního šumu, snížení aliasingu.

1.2.3 Kvantizace

Při kvantování jsou navzorkované hodnoty signálu zaokrouhleny na kvantizační úrovně. Počet kvantizačních úrovní je omezený a závisí na bitové hloubce. Pokud amplituda vstupních vzorků přesáhne maximální kvantizační hladinu dojde k omezení amplitudy signálu a tím ke zkreslení. Při kvantizaci dochází zaokrouhlením hodnot mezi hladinami k odchylce od původního signálu, která je dána počtem kvantizačních hladin a polohou rozhodovacích úrovní mezi kvantizačními hladinami. Tato odchylka se nazývá kvantizační šum.



Obr. 1.5: Kvantizace signálu

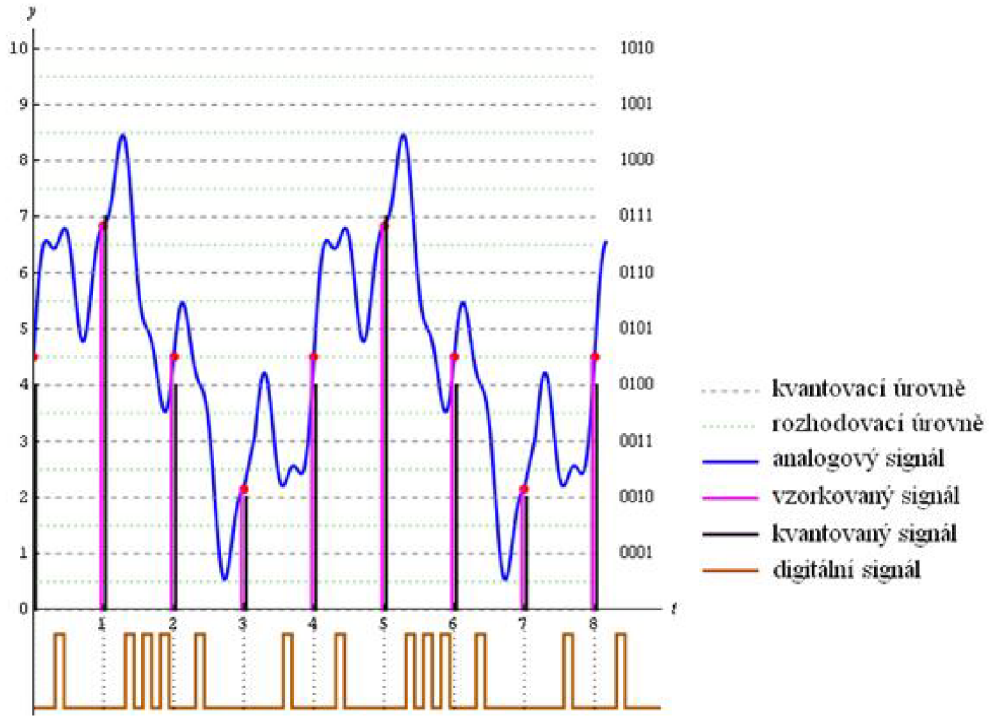
Kvantizační šum lze potlačit zvýšením počtu kvantizačních hladin, převzorkováním nebo přidaným náhodným šumem -dither (kompenzace pro zkreslení při malé úrovni signálu). Kvantizační šum je maskován užitečným signálem, a proto je při dostatečné bitové hloubce pro člověka nepostřehnutelný. Zvýšením rozlišení o 1 bit se se zvýší odstup signálu od kvantovacího šumu o 6 dB. Bitová hloubka stanovuje dynamický rozsah signálu.

1.2.4 Kódování

Během kódování dochází k přiřazení binárního čísla ke kvantizačním hladinám. Počet kvantizačních hladin v jednom vzorku lze vyjádřit bitovým binárním kódem, tzv. kódové slovo. Při kódování dochází ke ztrátě informace. Nejméně ztrátová metoda kódování používaná u audio zařízení je pulsní kódová modulace (angl. PCM-Pulse Code Modulation).

1.2.5 Sigma - Delta převodník

Nejvíce používaným A/D převodníkem pro audio zařízení je sigma - delta převodník. Převodník pracuje jako lineární filtr. Převodník generuje pulzy, jejichž střední hodnota odpovídá vstupnímu napětí.



Obr. 1.6: Převod analogového signálu na digitální (Zdroj: [7])

Převodník využívá převzorkování pro snížení kvantovacího šumu. Signál je nadvzorkován, převeden na digitální signál (vzniká šum) a potom převzorkován (decimován) zpět. Výkon kvantovacího šumu je rozprostřen v celém spektru. Při decimaci se tak sníží úroveň kvantizačního šumu v požadované části spektra. Šum bude snížen N-krát pokud je signál N-krát nadvzorkován. Převodník pracuje se stupněm převzorkování 64 nebo 128.

1.2.6 Číslicové zpracování zvukových signálu

Interní číslicové zpracování signálů u digitálních audio systému musí mít vyšší rozlišení než vstupní číslicový signál. Je to kvůli zabránění nelineárnímu zkreslení po překročení hladiny 0 dBFS u systémů, které pracují s celočíselnými vzorky a také kvůli zaokrouhlovacím chybám. Interní rozlišení digitálních systému proto používá rozlišení 24 nebo 32 bitů pro zpracování v pevné řadové čárce a 32 nebo 64 bitů pro zpracování v plovoucí řadové čárce.

Při zesílení signálu v číslicových systémech dochází k zesílení kvantovacího šumu. Je to proto, že zesílení se provádí násobení konstantou, takže se násobí užitečný signál zároveň s kvantizačním šumem. Při násobení může také dojít ke zmenšení odstupu signálu od šumu.

1.2.7 Převzorkování

Převzorkování je proces používaný při přenosu signálu mezi systémy s odlišným vzorkovacím kmitočtem.

Při podvzorkování je výsledný signál tvořen posloupností každého M-tého vzorku původního signálu, kde M je poměr podvzorkování. Před podvzorkováním je potřeba signál omezit antialiasingovým filtrem.

Při nadvzorkování v poměru celého čísla L se výsledný signál doplní L-1 vzorky. Po nadvzorkování je nutné odstranit nežádoucí složky ve spektru antialiasingovým filtrem.

Při převzorkování (v poměru racionálního čísla L/M) se signál nadvzorkuje a po té podvzorkuje. Převzorkování má vyšší výpočetní náročnost. Nevýhoda je také dvojí filtrace signálu antialiasingovými filtry.

1.2.8 DA převod

DA převodník převádí digitální signál zpět na analogový. Při převodu z interního bitového rozlišení na rozlišení DA převodníku dochází ke snížení bitového rozlišení. Poté se binární čísla zpět dekódují obvodem sample-and-hold na spojitý signál v čase ale diskrétní v úrovni. Nakonec je filtrací dolní propustí převeden na spojitý signál.

1.2.9 Dithering

Při DA převodu se sníží bitové rozlišení a vzniká kvantovací chyba. Kvantovací chyba vzniká useknutím nebo zaokrouhlením vzorků. Tuto chybu lze odstranit přičtením náhodného čísla k nejnižším bitům datového slova, ještě před redukcí bitového rozlišení. Tento proces se nazývá dithering a přičtený náhodný signál se označuje dither. Dithering mírně degraduje výsledný zvuk.

Filtrací dithru lze zvýšit kvantovací šum v kmitočtových pásmech kde je lidský sluch méně citlivý.

1.2.10 Jitter

Jitter, tzv. fázová nestabilita, vzniká při rozdílných vzorkovacích okamžicích AD a DA převodu, které jsou způsobeny vzájemnou nestabilitou vzorkovacího kmitočtu. Jitter způsobuje kmitočtově závislé zkreslení průběhu signálu, tzv. fázový šum. Pro stabilizaci vzorkovacího kmitočtu se používá obvod nazvaný fázový závěs.

1.3 Operační systém Android

Operační systém Android je nejpoužívanějším operačním systémem pro tzv. "chytré"mobilní telefony. Android je open source software, který je založený na jádře Linux, které umožňuje přenositelnost systému. Systém byl v začátku určen pro mobilní telefony, dnes se však používá také pro tablety, chytré televize, hodinky, náramky a auta. Android byl v roce 2005 odkoupen firmou Google. V roce 2007 vzniklo konsorcium Open Handset Alliance, jehož součástí je dnes Google a dalších 83 firem. Účelem tohoto konsorcia je vývoj multiplatformního operačního systému. Momentálně nejnovější verzí systému Android je verze 9.0 (Pie), která byla uvedena 6. srpna 2018. Přesto, že spousta zařízení používá systém Android, jejich vzhled se může lišit. Většina z nich používá grafické nadstavby od výrobce, místo klasického vzhledu Androidu.

1.3.1 Architektura systému

Systém Android je rozdělen do 5 vrstev.

Nejnižší vrstvou je jádro operačního systému (modifikovaný Linux), které umožňuje komunikaci mezi hardwarem a softwarem. Jádro systému využívá vlastnosti Linuxu např. správa paměti, správa sítí, správa procesů, souběžný chod aplikací.

Další vrstvou jsou knihovny, které obsahují základní příkazy a vlastnosti operačního systému. Knihovny jsou využívány různými komponenty systému a vývojářům jsou zpřístupněny prostřednictvím Android Application Framework. Knihovny jsou využívány např. při práci s multimédii, webovým prohlížečem, databázemi, renderu písma, grafiky a dalších procesech.

Třetí vrstvou je Android Runtime (dříve Dalvik), jehož součástí je virtuální stroj. V této vrstvě jsou aplikace překládány ze zdrojového kódu Javy do byte kódu.

Další vrstva Application framework, umožňuje vývojářům přístup ke službám, které mohou být v aplikaci použity. Tyto služby zpřístupňují například: data v jiných aplikacích, prvky uživatelského rozhraní, upozorňovací stavový řádek, aplikace běžící na pozadí, hardware používaného zařízení a jiné.

Poslední vrstvu tvoří základní aplikace, které jsou využívány uživateli. Mohou to být aplikace předinstalované nebo stažené aplikace třetích stran.

1.4 Pure Data

Pure Data (zkráceně PD) je vizuální programovací jazyk, který je open source. Vytvořil jej americký softwarový inženýr Miller Puckette. Pure Data předcházel vizuální programovací nástroj Max, který Puckette začal vytvářet v roce 1985 pod institutem IRCAM. Ze softwaru Max se postupně stával komerční produkt. V roce

1990 byla vydána komerční verze Max. Protože hrozilo zrušení softwaru Max, pod společností Opcode Systems, David Zicarelli získál publikační práva k softwaru Max a v roce 1997 založil společnost Cycling '74, pod kterou Max komerčně vydával. Poté, co se z Maxe začal stávat komerční produkt, vytvořil Puckette freewarový program Pure Data, který byl založený na stejném principu jako Max. Puckettovi a Zicarellimu se díky těmto softwarům podařilo z počítače udělat kompletní hudební nástroj použitelný při živém vystoupení.

PD se používá zejména ke zpracování a generování zvuku, videa, 2D/3D grafiky, nastavení senzorů, MIDI a jiných zařízení. PD lze také použít pro komunikaci po síti, s tzv. wearable technologií (hodinky, náramky), nebo pro řízení osvětlovacích a motorových systému. Tento programovací jazyk lze využít pro zpracování multimédií ale také pro složitější systémy a projekty.

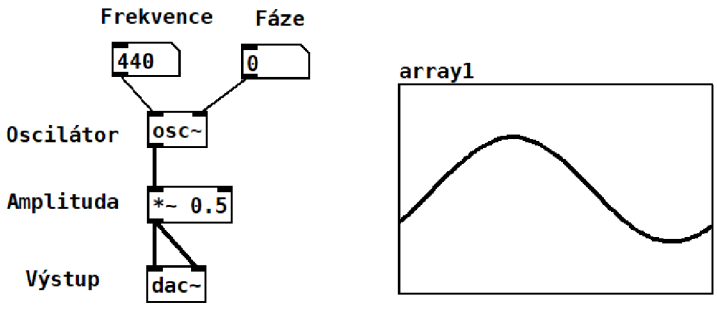
Díky tomu, že je Pure Data open source a bezplatný produkt vznikla velká komunita, která se ho snaží vylepšovat. Se základní verzí jazyka Pure Data vanilla je tak možné využívat další komunitou vytvořené externí knihovny, které usnadňují programování. Vzniklo několik distribucí, které v sobě obsahují několik externích knihoven. Mezi nejznámější patří PD extended, který dnes již není podporován, nebo Pd-L2ork/Purr-Data, který vychází z PD extended. PD vanilla je naprogramována v jazyce C a C++, zatímco Purr-Data je napsáno v Javascriptu pomocí webových technologií: nw.js.

Vzniklo také několik komunitou vytvořených rozhraní, které umožňují PD pracovat na jiných zařízeních jako "embedded library"(vestavěné knihovny). Mezi tyto rozhraní patři např. libpd, DroidParty (Android), a PdParty (iOS). PD je tak multiplatformní "sound engine". Na libpd je také postaven software MobMuPlat (Mobile Music Platform), který vytváří uživatelské rozhraní (GUI) pro systém Android a iOS.

1.4.1 Struktura a GUI Pure Data

Algoritmy jsou v jazyce Pure Data reprezentovány pomocí vizuálních boxů tzv. objektů. Data jsou mezi objekty přemisťovány přes tzv. kabely (patch cords). Objekt má určitou funkci, která je definovaná jeho názvem. Tyto funkce mohou být např. matematické operace, audio operace, video funkce, FFT transformace, dekódování videa nebo objekty umožňující komunikaci mezi objekty.

Příklad objektu s funkcí oscilátoru: [osc~]. Objekty mají své vstupy a výstupy. Jejich počet závisí na tom s kolika parametry objekt pracuje. Oscilátor má 2 vstupy (frekvence a fáze) a jeden výstup (audio signál). Amplituda lze regulovat násobícím objektem za výstupem oscilátoru např. [*~ 0.5]. Znak vlnky v objektech značí, že objekt zpracovává audio signál.



Obr. 1.7: Jednoduchý oscilátor v Pure Data

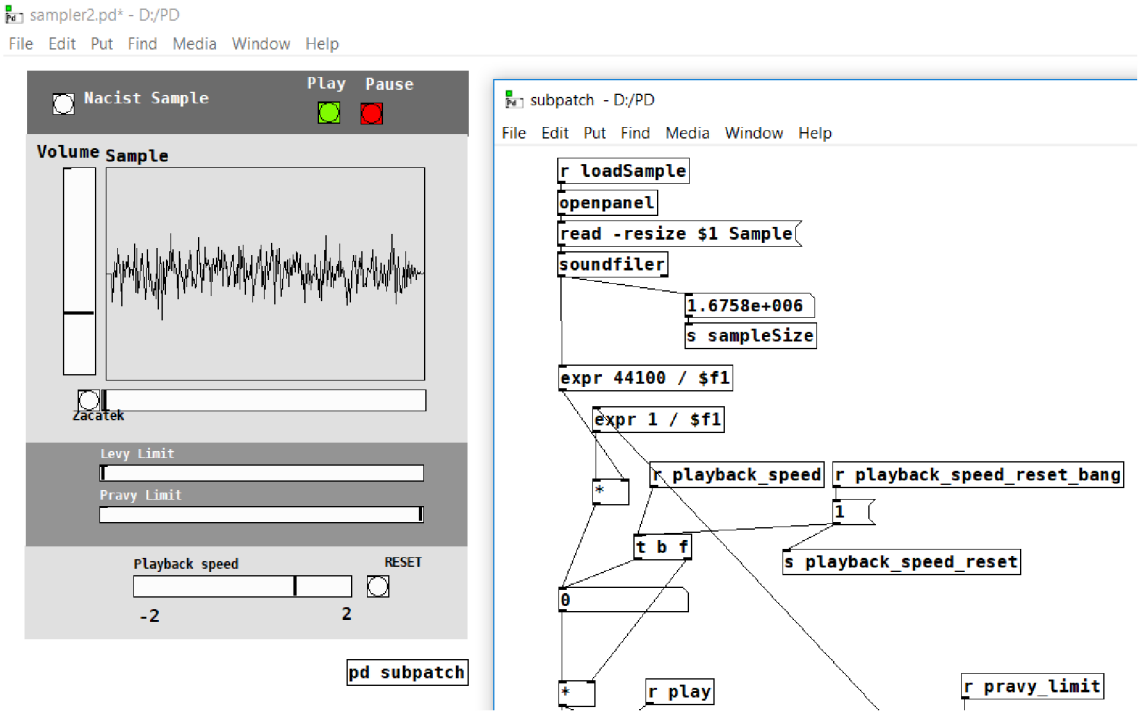
Jak je vidět na obrázku Pure Data má další prvky, které předávají informace objektům nebo ukládají informace předané z objektu. Příkladem na obrázku jsou prvky obsahující číslo (number box), které objektu předávají informace o frekvenci a fázi. Dalším příkladem je pole, ve kterém je zaznamenán průběh výstupního signálu, a který přijímá informace z násobícího objektu. Mezi další základní prvky patří zpráva (Message), symbol určený pro zápis textu a komentář sloužící k popisu kódu pro snadnější orientaci. Další prvky jsou bang, který je určen jako spouštěč jiných procesů, toggle je přepínač (odešle hodnotu 0 nebo 1), Slidery -posuvné potenciometry, canvas (pouze zbarvuje pozadí oblasti) -lze ho použít pro lepší orientaci v kódu a při vytváření GUI.

Kód sestavený z těchto prvků se nazývá patch. V patchi lze vytvářet i tzv. subpatche, což je objekt, který v sobě skrývá další patch. Pokud chceme takto vytvořený subpatch používat na více místech v kódu je lepší ho uložit do paměti na disku, aby nedocházelo k duplikaci kódu v patchi.

V Pure Data můžeme vytvářet jednoduché GUI. V hlavním patchi vytvoříme subpatch, který bude obsahovat všechen kód. Ovládací prvky v subpatchi nahradíme objektem, který přijímá data ([receive]) nebo odesílá data ([send]) na určenou adresu. V hlavním patchi vytvoříme GUI s objekty, kterým nastavíme správné adresy pro přijímání nebo odesílání dat do subpatche.

1.5 GUI pro Pure Data a Android

Pro operační systém Android je několik platforem, které propojují Android s Pure Data a umožňují vytvořit grafická uživatelská rozhraní. Já jsem si vybral MobMuPlat (Mobile Music Platform), zejména kvůli její snadnosti ovládání a celému procesu propojení. Grafické ovládací prvky MobMuPlat jsou podobné s prvky Pure Data. Ovládání a vytváření GUI při přechodu z Pure Data do MobMuPlat tak není složité.



Obr. 1.8: GUI v hlavním patchi a kód v subpatchi

MobMuPlat umožňuje chod PD na mobilních operačních systémech Android a iOS. MobMuPlat funguje na rozhraní libpd, které z Pure Data vytvoří embedded library (vestavěnou knihovnu) a umožní komunikaci se systémem Android. MobMuPlat používá další knihovny např. PGMidi, OSC a GUI objekty z platformy PdParty.

Platforma zvládá syntézu, samplování, MIDI, Open Sound Control (OSC), komunikaci prostřednictvím wifi sítě, dokáže přijímat data hardwarových součásti zařízení jako mikrofon, gyroskop, kompas, gps, kamera, blesk nebo externí ovladače (joystick).

Proces propojení Pure Data s Androidem lze rozdělit do několika částí. První se PD propojí s aplikací MobMuPlat editor v počítači(Windows, iOS) pomocí spleciálního PD patche (PdWrapper.pd). Po té lze vytvořit GUI pomocí editoru. Následně se potřebné soubory spolu s PD patchem přenesou na mobilní zařízení a spustí v aplikaci MobMuPlat na zařízení.

2 Návrh a grafické rozhraní aplikace

Aplikace se skládá z následujících částí: sampleru, granularního syntetizéru, wavetable syntetizéru se sekvencerem, filtrem, efekty a amplitudovou obálkou. V první části je zvukový soubor načten z paměti zařízení nebo nahrán mikrofonem do sampleru, kde je zvukový záznam zpracován pro další použití v granulárním syntetizéru. Druhou částí je granulární syntetizér, který pracuje s nahraným zvukovým záznamem. Wavetable syntetizér má dva oscilátory, které přehrávájí signály se zvoleným průběhem a ty dále upravuje filtrem a amplitudovou obálkou. Součástí wavetable syntetizéru je sekvencer, který slouží k vytváření melodií a rytmů. V další části jsou efekty pro wavetable syntetizér: amplitudová modulace, přechod mezi průběhy signálů oscilátorem (LFO). V poslední části je efekt reverb.

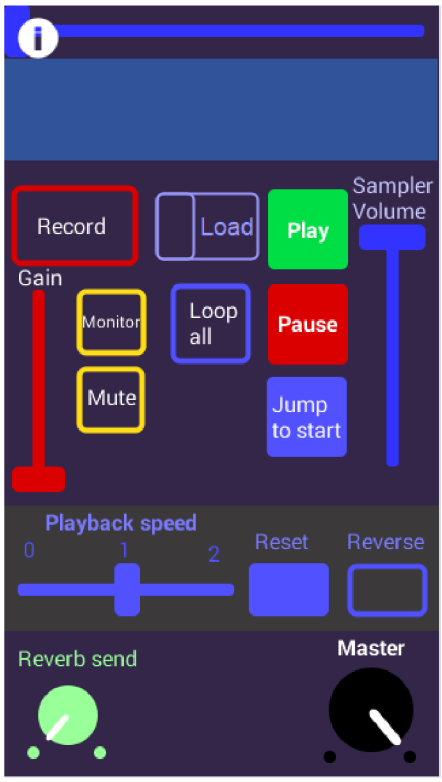
Aplikace je naprogramovaná jazykem Pure Data. Pure Data komunikuje se zařízením Android prostřednictvím aplikace MobMuPlat. Pomocí editoru MobMuPlat je na počítači vytvořeno grafické uživatelské rozhraní (GUI).

2.1 Rekordér, sampler

Tato část aplikace je uvedena na obrázku 2.1. V levé části rekordéru a sampleru lze nahrávat zvuk prostřednictvím mikrofonu zařízení tlačítkem Record a nastavením vstupní úroveň signálu do rekordéru z mikrofonu sliderem Gain. Tlačítko Monitor umožňuje poslouchat výstup z mikrofonu a mute ztlumí výstupní signál z mikrofonu. Po ukončení nahrávání se nahrávka uloží jako audio soubor a následně nahraje do sampleru. V pravé části lze načíst sample uložený v paměti zařízení tlačítkem Load. Dále jsou zde ovládací prvky pro přehrávání a zastavení samplu a hlasitost sampleru. V horní části je pole pro výběr délky smyčky pro přehrávání samplu a ukazatel pozice přehrávání. V další části sampleru je nastavení rychlosti přehrávání a přehrávání pozpátku. Rychlost přehrávání ovlivňuje výšku tónu. Ve spodní části sampleru je pontenciometr "Reverb send"ovlivňující množství odeslaného signálu do efektu reverb. Potenciometr "Master"nastavuju výstupní úroveň hlasitosti celé aplikace. Tato sekce slouží ke zpracování nahraného zvuku a jeho dalšího použití v granulárním syntetizéru.

2.2 Granulární syntetizér

Granulární syntetizér zpracovává vybranou část samplu. Z této vybrané části jsou náhodně přehrávány jednotlivé grainy. Granulaci lze nastavit několika parametry délka grainu, stereo rozmístění, tónová transpozice a množství grainů. Tyto parametry se generují náhodně v rozsahu zadaném uživatelem. Pro výběr rozsahu jsem



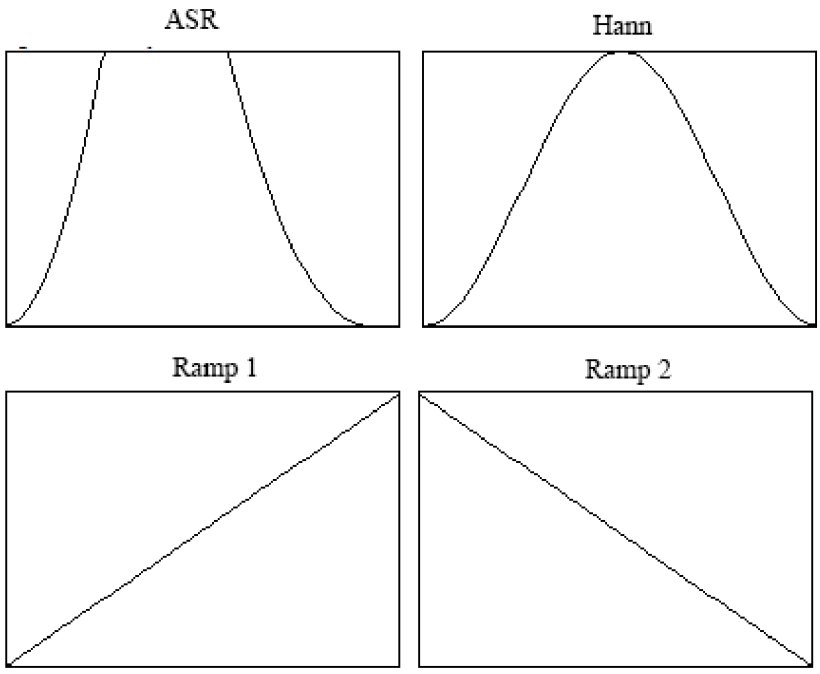
Obr. 2.1: Grafické rozhraní sampleru

vybral GUI prvek "table"s módem "select". Na obrázku 2.3 je GUI granulárního syntetizéru. Dalším parametrem je amplitudová obálka.

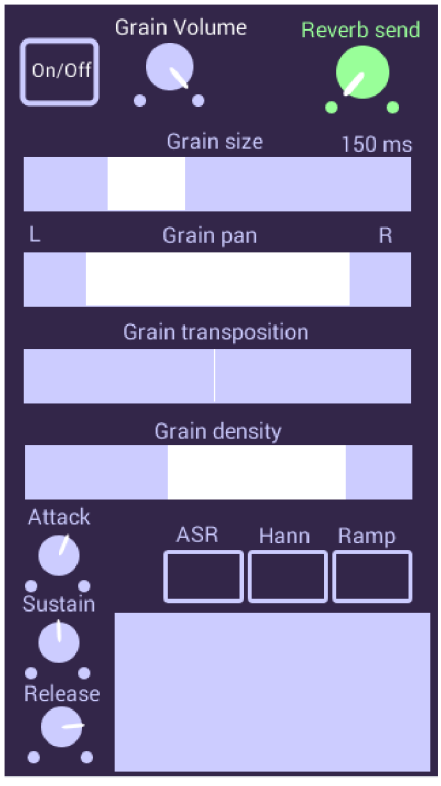
Na obrázku 2.2 jsou druhy amplitudových obálek .K dispozici je celkem 5 možností jak obálku vygenerovat. Prvním je nastavení parametry Attack, Sustain, Release (ASR), dalšími tvary obálky jsou Hannovo okno, Ramp 1 a Ramp 2. Mezi obálkami Ramp 1 a Ramp 2 se přepíná jedním tlačítkem Ramp. Průběh obálky se zobrazuje v prvku "table"ve spodní části obrazovky. Tento ovládací prvek je v módu "draw", který umožňuje přepisovat obsah pole v Pure Data patchi. Poslední možností je tedy kreslení tvaru obálky do pole. V horní části syntetizéru jsou ovládací prvky pro zapnutí generování grainů, nastavení hlasitosti a odeslání signálu do efektu reverb.

2.3 Wavetable syntetizér

Wavetable syntetizér se skládá ze dvou oscilátorů, které přehrávají signál načtený z pevné paměti. Tento signál je složen z několika tvarů vln, mezi kterými lze přecházet potenciometrem "Wt-position"(wavetable position). Oscilátory přehrají wavetable s velikostí framu (danou v samplech) 515, 64, 128, 256, 512, 1024, 2048. Na obrázku

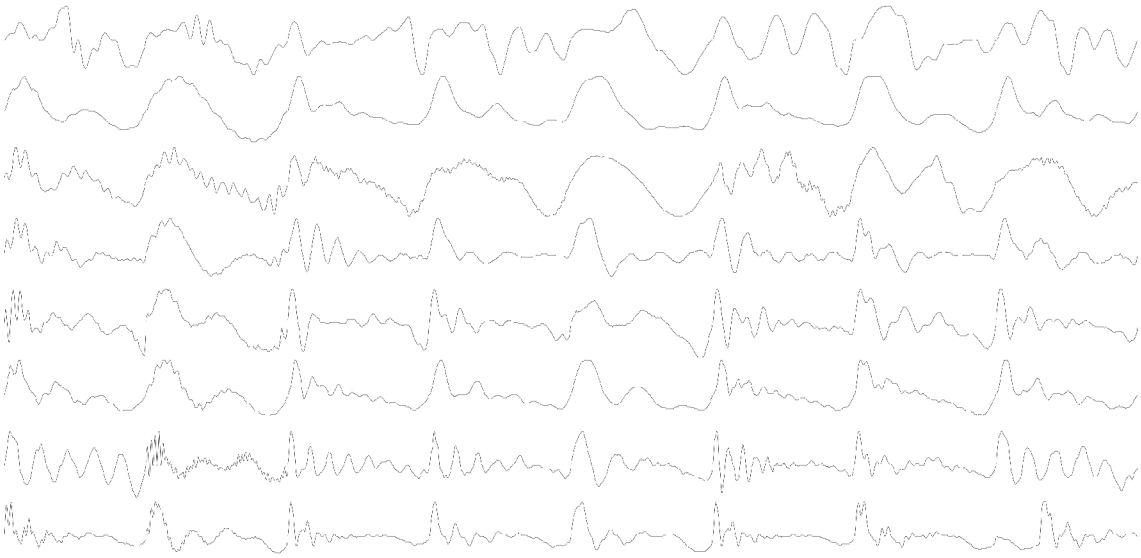


Obr. 2.2: Amplitudové obálky grainu



Obr. 2.3: Grafické rozhraní granulárního syntetizéru

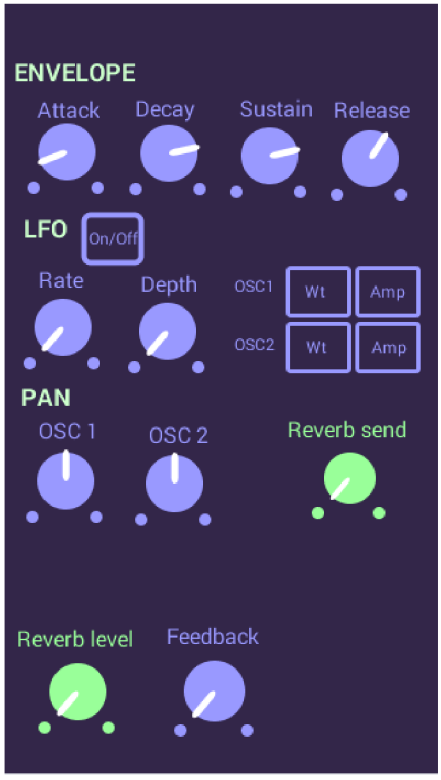
2.4 jsou zobrazeny příklady signálů používaných pro wavetable syntézu. Dalšími parametry lze oscilátory rozladit a změnit zesílení. Syntetizér obsahuje jednoduchý pásmový filter ovládaný LFO. Melodii lze vytvořit pomocí osmi krokého sekvenceru s nastavitelným tempem v rozsahu 40-300 BPM. Tempo lze synchronizovat s granulárním syntetizérem a kombinovat tak oba syntetizéry. Na další straně aplikace je nastavení amplitudové obálky ADSR pro oba wavetable oscilátory.



Obr. 2.4: Wavetable, druhy signálu

Oscilátor LFO může být použit pro modulaci amplitudy wavetable oscilátorů nebo pozice přehrávání wavetable (Wt-position). Změnou pozice přehrávání ve wavetable se změní tvar vlny. Přehráváním wavetable s modulací pozice přehrávání lze replikovat jiné druhy syntéz a efektů, které by jinak mohly být pro mobilní zařízení příliš výpočetně náročné. Druh syntézy je přímo zaznamenaný ve wavetable a nahráním např. wavetable s frekvenční modulací do oscilátoru lze tento efekt simulovat. V poslední části aplikace je nastavení panorama pro oscilátory a nastavení parametrů efektu reverb.

Obr. 2.5: Wavetable syntetizér GUI



Obr. 2.6: Wavetable envelope, LFO, pan, reverb

3 Programování aplikace

Aplikaci jsem programoval v jazyce Pure Data. V následující části se budu věnovat jednotlivým sekcím aplikace na programovací úrovni jazyka Pure Data. Popíšu také způsob propojení s platformou MobMuPlat, protože aplikace je určena pro ovládání přes ovládací prvky MobMuPlat.

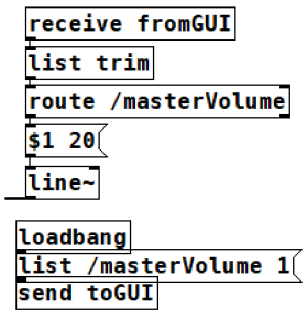
3.1 Propojení Pure Data s MobMuPlat

V platformě MobMuPlat se vytváří GUI s ovládacími prvky. Každému prvku je přiřazena adresa ve formě /názevPrvku. S touto adresou jsou dále posílána data, která určují stav prvku. Data jsou posílána přes Pure Data patch PdWrapper.pd a lze je přijímat v patchi objektem [receive fromGUI]. Data jsou poslána jako list hodnot kde první prvek je adresa ovládacího prvku (/adresaPrvku) a další je jeho hodnota.

Například při nastavení potenciometru Master v sampleru na maximum je přijat list /masterVolume 1 (rozsah potenciometru je 0-1).

Přijímaná data lze filtrovat objektem [route] s parametrem "list"nebo objektem [list] s parametrem "trim".Dále jsou data filtrována objektem [route] s parametrem adresy prvku jehož data potřebuji přijmout. Data do platformy se odesílají ve stejné formě jako přijímají tzn. ve formě zprávy (message) s parametrem list, adresa prvku, hodnota. Pro odeslání se však použije objekt [send toGUI].

Pro komunikaci s operačním systémem se používá objekt [receive fromSystem] a [send toSystem]. Při vytváření GUI lze sledovat vstup a výstup dat v editoru MobMuPlat, záložka Lock.

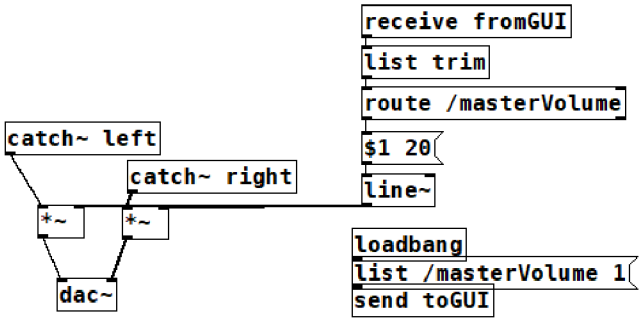


Obr. 3.1: Přijímání a odesílání dat do MobMuPlat

3.2 Struktura Pure Data projektu

Pure Data projekt je rozdělen do několika subpatchů a abstrakcí, na které odkazuje hlavní patch main.pd. Hlavní patch obsahuje sampler s rekordérem a hlavní audio výstup kam jsou směrovány ostatní výstupy subpatchů a abstrakcí. Výstup jednotlivých sekcí je propojen s hlavním výstupem bezdrátově prostřednictvím objektů [throw~] a [catch~].

Pro odstranění rušivých kliků při změně potenciometru "Master"je použita interpolace mezi hodnotami prostřednictvím objektu [line~]. Tento signál je násoben výstupem ostatních sekcí a přiveden na hlavní audio výstup [dac~].



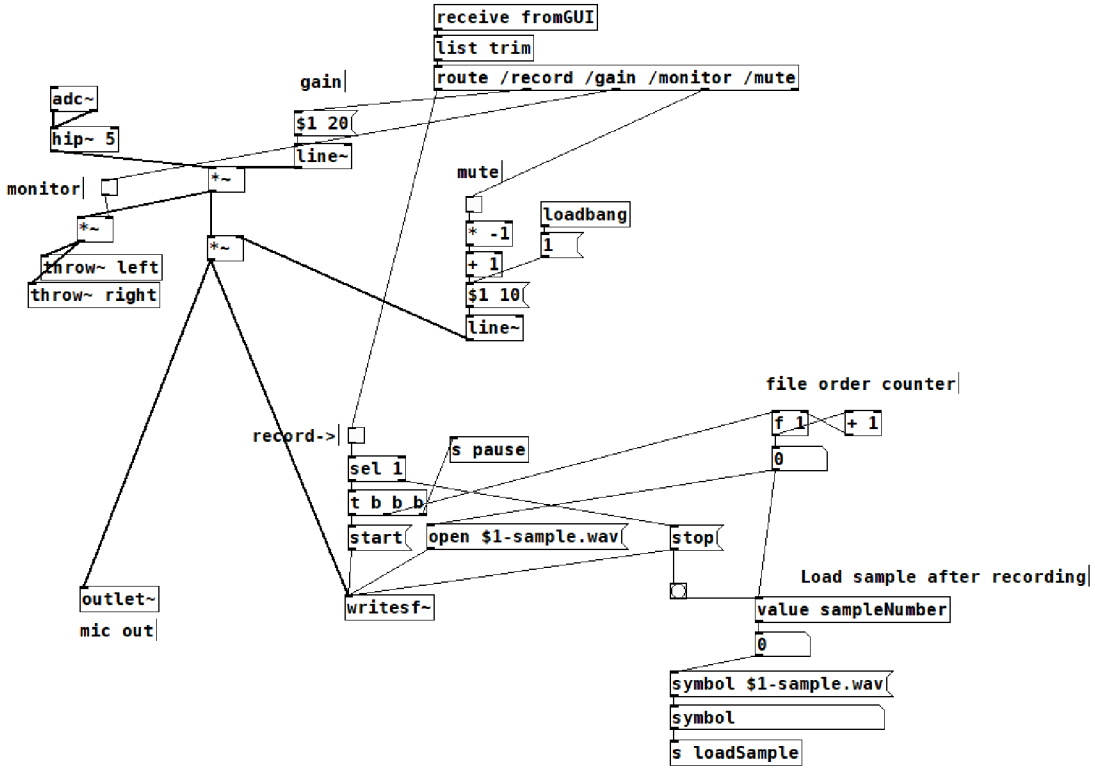
Obr. 3.2: Hlavní audio výstup

Main.pd dále odkazuje na abstrakce granular2.pd (granulární syntetizér), wavetable.synth.wrap.pd (wavetable syntetizér), efekty2.pd. Podrobnější strukturu patchů popíšu v následujících sekcích.

3.3 Rekordér

Kód pro nahrávání se nachází v subpatchi "pd record", který je v hlavním patchi main.pd. Výstup z mikrofonu zařízení je v rekordéru zpracován po částech následovně. Nejprve je signál filtrován horní propustí pro odstranění stejnosměrné složky (objekt [hip~]) dále je zesílen sliderem Gain z GUI prostřednictvím násobícího objektu [*~]. Signál je odeslán do monitorovací části, ze které může uživatel poslouchat výstup mikrofonu, a do části pro záznam zvuku a uložení zvukového souboru. Před záznamovou částí je do cesty signálu vložen tlumící článek mute. Mute funguje jako toggle (hodnoty 0,1) ale posílá opačné hodnoty (1,0) tak aby když uživatel stiskne v GUI Mute tak se výstupní signál mikrofonu násobí nulou a utlumí. Při záznamu zvuku je vždy nejdříve vytvořen název souboru (ve tvaru čísloZáznamusample.wav) do kterého bude zvuk zaznamenán. Název souboru je tvořen dynamicky,

inkrementací hodnoty 1 a přidáním před název souboru při každém nahrávání (např. 1-sample.wav). V dalším kroku je objektu pro zápis zvuku do souboru [writesf~] předána zpráva s názvem souboru (open $1-sample.wav). Hodnota $1 je přepsána proměnnou na vstupu zprávy. Zápis do soubroru je spuštěn po odeslání zprávy "start"a zastaven zprávou "stop". Po dokončení zápisu je soubor načten do sampleru. Data do sampleru jsou odeslána bezdrátově objektem [send loadSample]. Subpatch "pd record"má pouze jeden signálový výstup tj. signál z mikrofonu.



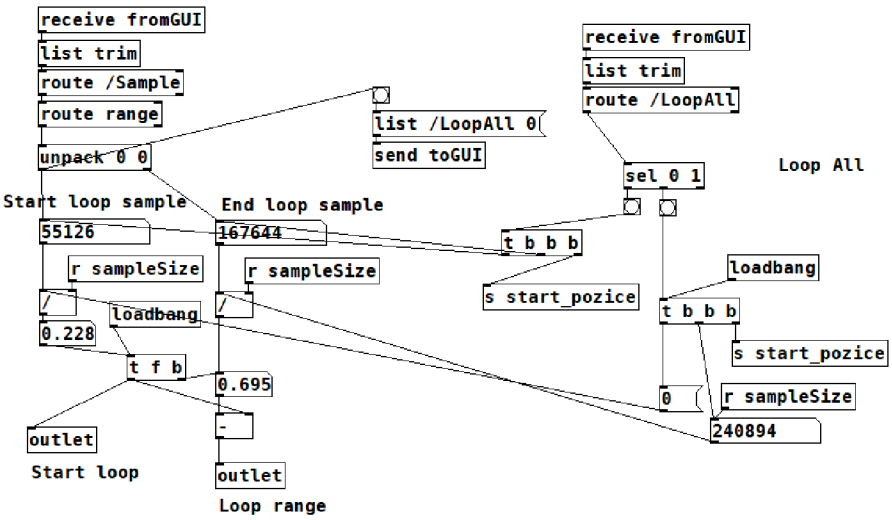
Obr. 3.3: Rekordér (subpatch: pd record)

3.4 Sampler

Sampler se skládá z několika částí: načtení samplu, nastavení rychlosti přehrávání, přehrávání samplu z pole, zesílení signálu před výstupem ze sampleru, odeslání signálu na hlavní výstup. Patch je na obrázku 3.5.

Sampler načítá data ze souboru uloženého na pevném disku do paměti RAM a na ní je odkázáno v poli /Sample v Pure Data patchi. Objekt [soundfiler] analyzuje soubor a nahraje ho do pole /Sample. Výstupem objektu [soundfiler] je počet samplů v souboru. Po nahrání do pole se aktualizuje waveform v panelu GUI a resetuje se smyčka.

Sampl je čten z pole objektem [tabread4~] a oscilátorem [phasor~], který určuje rychlost přehrávání pole. Phasor umožňuje přehrávat sample ve smyčce a generuje pilový signál nabývající hodnot 0-1, je proto vhodný pro čtení dat z pole. Pro správné přehrání samplu je potřeba přizpůsobit frekvenci phasoru pro vzorkovací frekvenci a počet vzorků v souboru. Uživatel má možnost vybrat pouze část samplu, kterou chce přehrát. Při tom dochází ke snížení počtu přehrávaných samplů podle zvoleného rozsahu, proto je nutné přepočítat frekvenci phasoru tak aby rychlost přehrávání samplu ve zvoleném rozsahu zůstala stejná jako při přehrávání celého samplu. Subpatch "pd LoopRange"(obr. 3.4) přijímá z GUI dvojici vzorků, na kterých smyčka začíná a končí. Z těchto hodnot vypočítá rozsah smyčky ve vzorcích. Výstupem subpatche je počáteční vzorek, na kterém [tabread4~] začne číst data, a délka smyčky ve vzorcích pro nastavení správné rychlosti přehrávání.



Obr. 3.4: Subpatch LoopRange

Frekvenci oscilátoru phasor lze vypočítat podle vztahu:

$$F_{phasor} = \frac{F_{vz}}{N}[\text{Hz}]$$

Kde F_{phasor} je frekvence oscilátoru phasor, F_{vz} vzorkovací frekvence, N počet vzorků v souboru. Pokud se změní rozsah smyčky lze frekvenci oscilátoru phasor získat podle vztahu:

$$F_{phasor} = \frac{F_{vz}}{N \cdot N_s}[\text{Hz}]$$

Kde N_s je počet vzorků ve smyčce.

V aplikaci lze měnit rychlost přehrávání slidrem v rozsahu 0 až 2. Po zapnutí režimu reverse playback se tento rozsah násobí konstantou -1. Při změně rychlosti přehrávání je ovlivněna také frekvence zvuku, nejedná se tedy o tzv. time stretching. Frekvence oscilátoru se získá vynásobením hodnotou, která určuje změnu rychlosti přehrávání (v celkovém rozsahu -2 až 2). Vztah pro výpočet frekvence oscilátoru phasor, který je aplikován v patchi je:

$$F_{phasor} = \frac{F_{vz}}{N \cdot N_s} \cdot R[\text{Hz}]$$

Kde R je změna rychlosti přehrávání zvolená uživatelem.

Přehrávání lze spustit tlačítkem "Play"v GUI. V patchi se přehrávání spustí po odeslání frekvence oscilátoru phasor. Tato hodnota je uložena v objektu [f] (float). Přehrávání se zastaví po odeslání hodnoty 0 do oscilátoru. Pozici přehrávání lze nastavit fází oscilátoru tj. druhý vstup objektu [phasor~] (tzv. cold inlet).

Za objektem [phasor~] je signál násoben délkou smyčky ve vzorcích a pozice přehrávání je posunuta objektem [+~] o počet vzorků, který se rovná čísle vzorku, na kterém smyčka začíná. Signál je dále zpět vynásoben počtem vzorků v souboru. Tento signál řídí čtení hodnot objektem [tabread4~ /Sample] z pole /Sample.

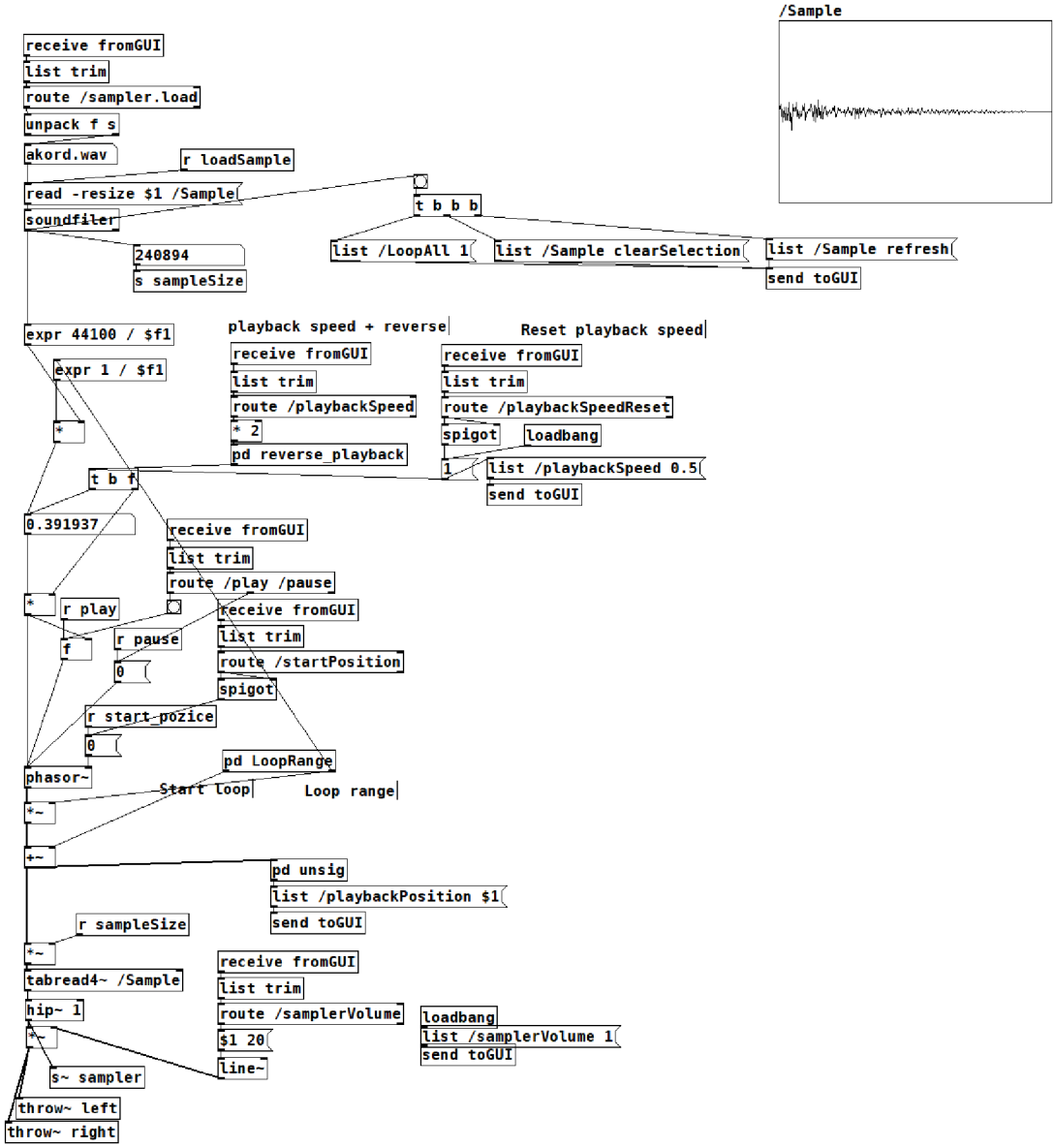
Poslední části sampleru je zesilovač, který určuje výstupní úroveň signálu ze sampleru a je ovládán slidrem "Sampler Volume"v GUI. Signál je do efektu reverb odeslán před tímto zesilovačem.

3.5 Granulární syntetizér

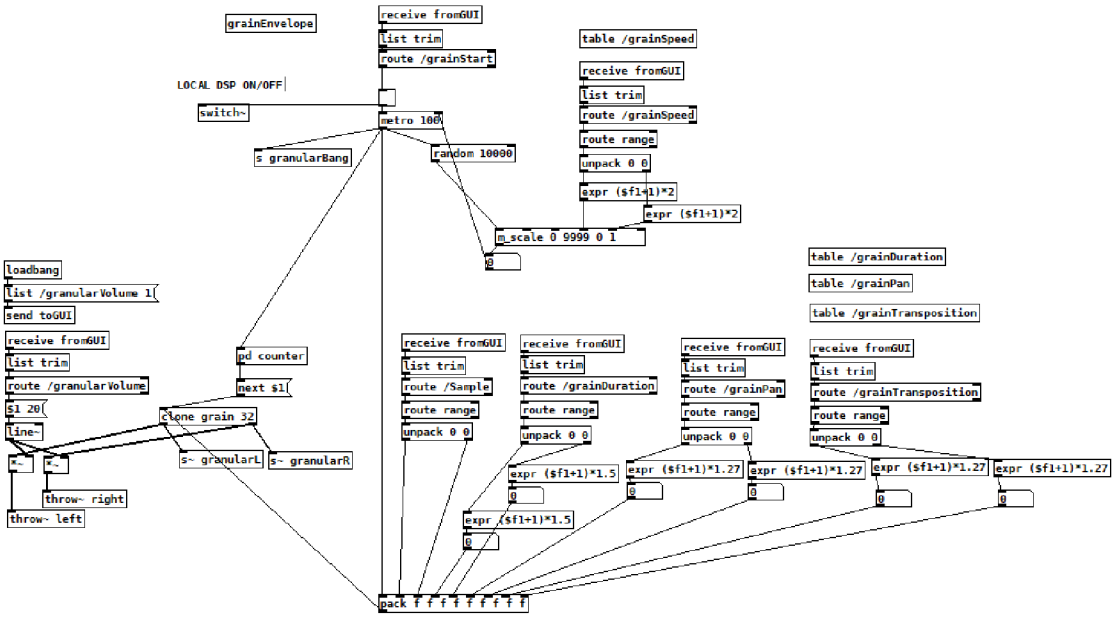
Granulární syntetizér má několik parametrů: pozice grainu v samplu, délka grainu, rozmístění ve stereu, transpozice a množství grainů. Grainům lze nastavit amplitudová obálka. Tyto parametry jsou poslány abstrakci "grain.pd", která přehraje malou část samplu (maximální délka grainu je 150 ms). Abstrakcí grain je v syntetizéru celkem 32. Pro snadnější práci s abstrakcemi grain jsem použil objekt clone, který je nakopíruje a umožňuje snadné adresování jednotlivých kopií.

Generování grainů začíná objektem [metro], který časuje generaci grainů. Objekt [metro] pošle "bang"objektu [pack], který seřadí parametry získané z GUI do listu a ten pošle přes objekt [clone] do abstrakce grain. V každé abstrakci je pak z parametrů s rozsahem hodnot zvolené uživatelem vybrané náhodně číslo, které je použito pro vytvoření grainu. Způsob vytvoření grainu v abstrakci grain popíšu v poslední části.

Rozsah hodnot je získan z GUI prvku "table", který je dán rozsahem pole v patchi. GUI prvek table odkazuje na pole. Standardní velikost pole v Pure Data je 100, proto jsem výstup hodnot z GUI musel přizpůsobit každému parametru.



Obr. 3.5: Sampler Pure Data patch



Obr. 3.6: Granulární syntetizér

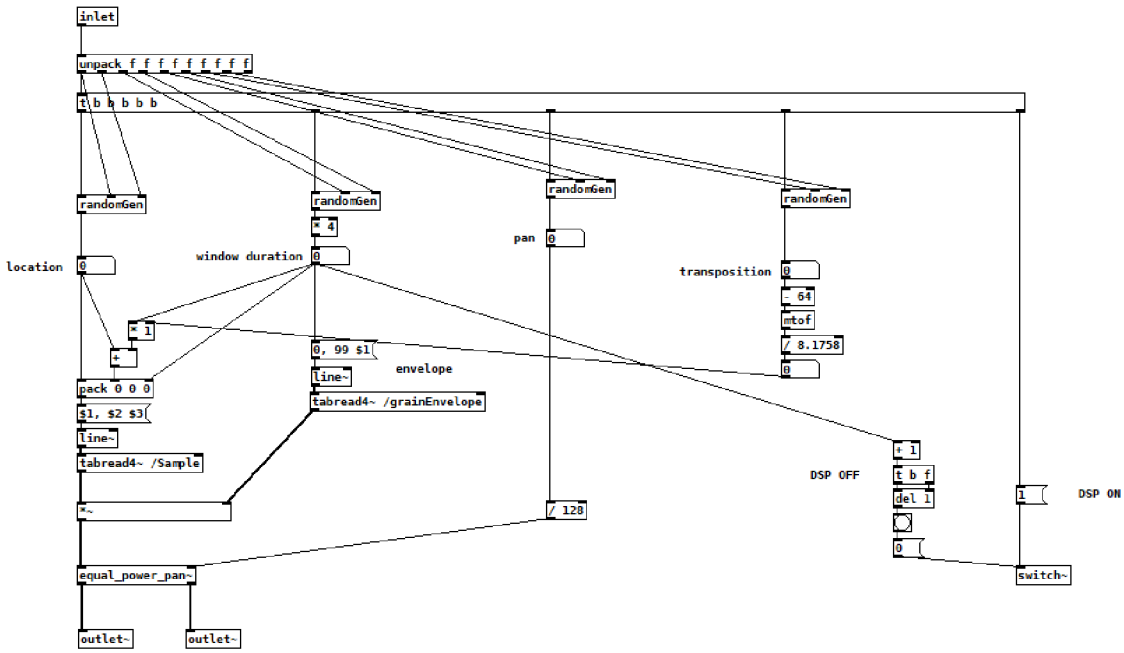
Parametr objektu [metro] určuje rychlost generace grainů. Tento parametr je vybrán náhodně v rozsahu hodnot vybrané uživatelem. "Bang"generovaný metrem je poslán objektem [send granularBang] do wavetable syntetizéru pro synchronizaci tempa. Při každém "bangu"je vybrán interval do dalšího "bangu", poté jsou parametry získané z GUI odeslány do kopie abstrakce v pořadí a v subpatchi "pd counter"je vygenerováno pořadí další kopie abstrakce grain. Jednotlivé abstrakce grain jsou adresované zprávou "next $1", kde $1 je nahrazeno pořadním číslem odeslaným subpatchem counter při každém "bangu".

Stereo výstup abstrakce je odeslán objektem [send∼] do efektu reverb. Zároveň je odeslán do poslední části syntetizéru, kde je nastavena výstupní úroveň granulárního syntetizéru (Grain Volume v GUI). Signál je dále poslán objekty [throw∼] na hlavní výstup v patchi main.pd. Pro snížení výpočetní náročnosti se DSP v patchi granular2.pd zapíná a vypíná spolu se spínačem pro generaci grainů.

3.5.1 Grain

V následující části vysvětlím princip vytvoření grainu v abstrakci grain.pd. Abstrakce má jeden vstup, kterým je předán list parametrů a ten je roztříděn objektem [unpack]. První parametr je použit také pro poslání "bangu"do objektu [trigger], který rozešle "bang"ve správném pořadí do sekcí patche. Sekce patche jsou spuštěné v následujícím pořadí. První se zapne DSP pro konkrétní abstrakci grain a dále jsou spuštěny sekce následovně: transpozice, stereo umístění (pan), délka grainu (window

duration), pozice přehrávání grainu v samplu (location). DSP v patchi se vypne po zpožění, které je rovno délce grainu (window duration).

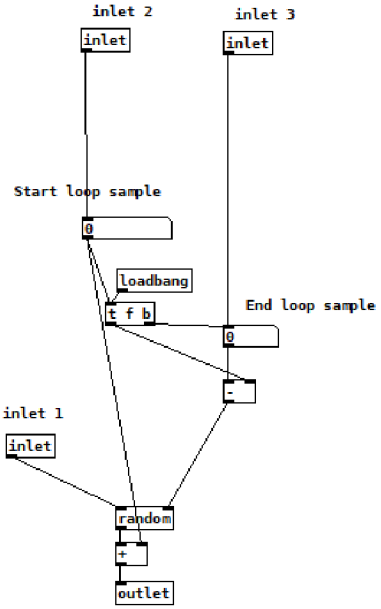


Obr. 3.7: Grain

V každé sekci je abstrakce "randomGen", která generuje náhodné číslo v rozsahu hodnot. Abstrakce má tři vstupy. První vstup je určen pro "bang", který spustí objekt [random]. Druhý vstup je pro nejmenší hodnotu v rozsahu a třetí pro největší. Z těchto hodnot je vypočítán rozsah, z kterého objekt [random] vybere náhodně jednu hodnotu.

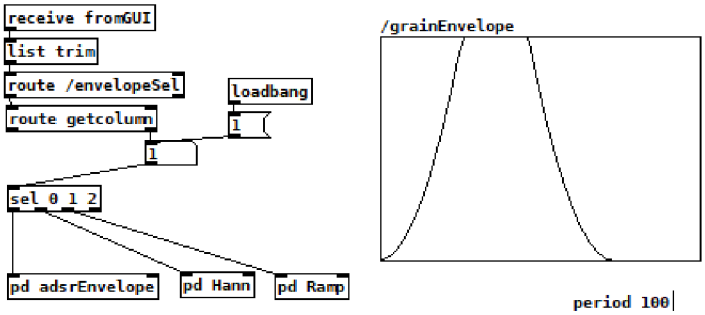
Grain je přehráván objektem [line∼]. Objekt [line∼] generuje pilový signál, narozdíl od oscilátoru [phasor∼] ale vygeneruje pouze jedenu periodu. Je tak vhodný pro přehrání grainu jedenkrát. Objekt [line∼] genureje pilový signál od počáteční do koncové hodnoty za určitý čas daný v milisekundách. Tyto parametry jsou předány objektu zprávou. Pilový signál je zpracován objektem [tabread4∼ /Sample], který přečte vzorky samplu. Parametr počáteční hodnoty pilového signálu je dán vygenerovanou pozicí přehrávání v samplu, parametr konečné hodnoty je dán součtem lokace a délky grainu a poslední parametr je dán délkou grainu. Změnou konečné hodnoty lze transponovat frekvenci grainu.

V další části je signál násoben amplitudovou obálkou. Amplitudová obálka je generována v abstrakci grainEnvelope.pd, která se nachází v patchi granulárního syntetizéru. Průběh obálky je zapisován do pole /grainEnvelope, které je čteno v grainu (grain.pd). Abstrakce obsahuje celkem tři generátory obálek: Hannovo okno, pilový signál, inverzní pilový signál a křivku určenou náběžnou hranou (attack), sestupnou



Obr. 3.8: Generátor náhodného čísla z rozsahu hodnot

hranou (release) a dobou mezi těmito parametry (sustain). Poslední typ obálky se

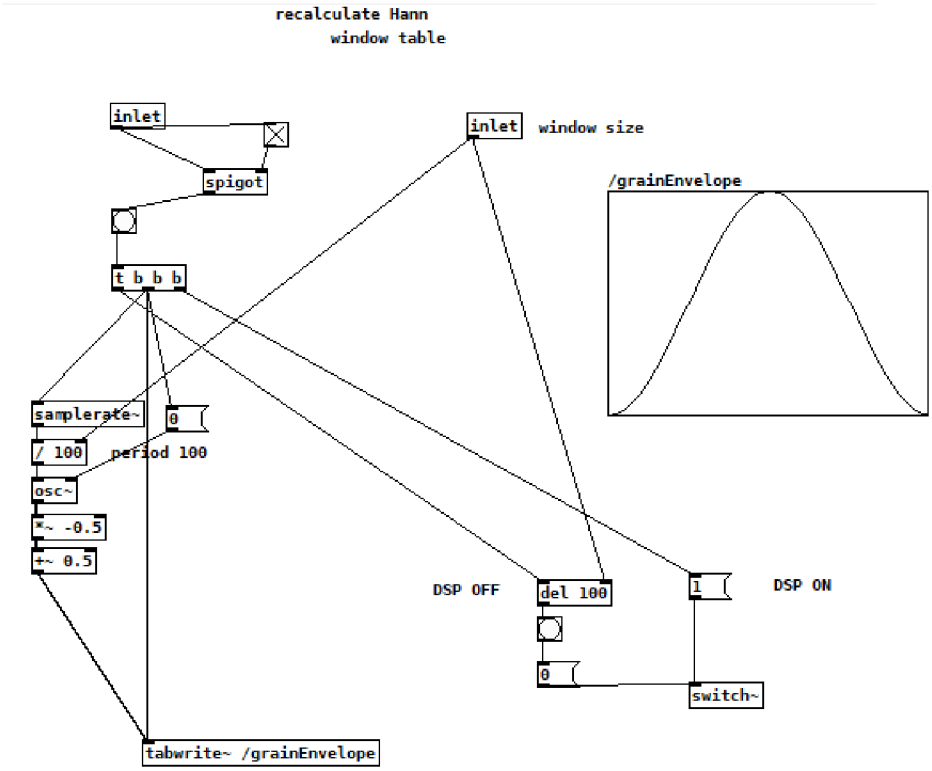


Obr. 3.9: Generátory amplitudových obálek

také nazývá three-segment-line. Tato obálka je vytvořena v abstrakci [exp_adsr~], která vytváří křivku pomocí objektu [vline~]. DSP v patchi generátorů obálek je zapnuté pouze po dobu zapisování jedné periody signálu do pole /grainEnvelope.

Pro generaci Hannova okna jsem použil oscilátor s kosinusovým průběhem [osc~]. Signál je převrácen kolem osy x a jeho amplituda je zmenšena na polovinu, poté je posunut o polovinu amplitudy ve směru kladné osy y.

Obálka s pilovým průběhem je generována oscilátorem [phasor~], v případě inverzního pilového signálu je frekvence oscilátoru násobena hodnotou -1.



Obr. 3.10: Generátor Hannova okna

V poslední části patche grain.pd je abstrakce pro umístění grainu ve stereu. Využil jsem abstrakci od Hans-Christoph Steiner nazvanou "equal_power_pan~". Abstrakce má dva vstupy, první pro signál a druhý pro řízení panoramy. Panorama je dosažena v abstrakci použitím sinusových a kosinusových funkcí.

3.6 Wavetable syntetizér

Princip funkce syntetizéru je založen na přehrávání pole se specifikým vlnovým průběhem signálu. Přehráváním těchto signálů lze napodobovat jiné druhy syntéz, jejichž průběh je přímo zaznamenán v signálu.

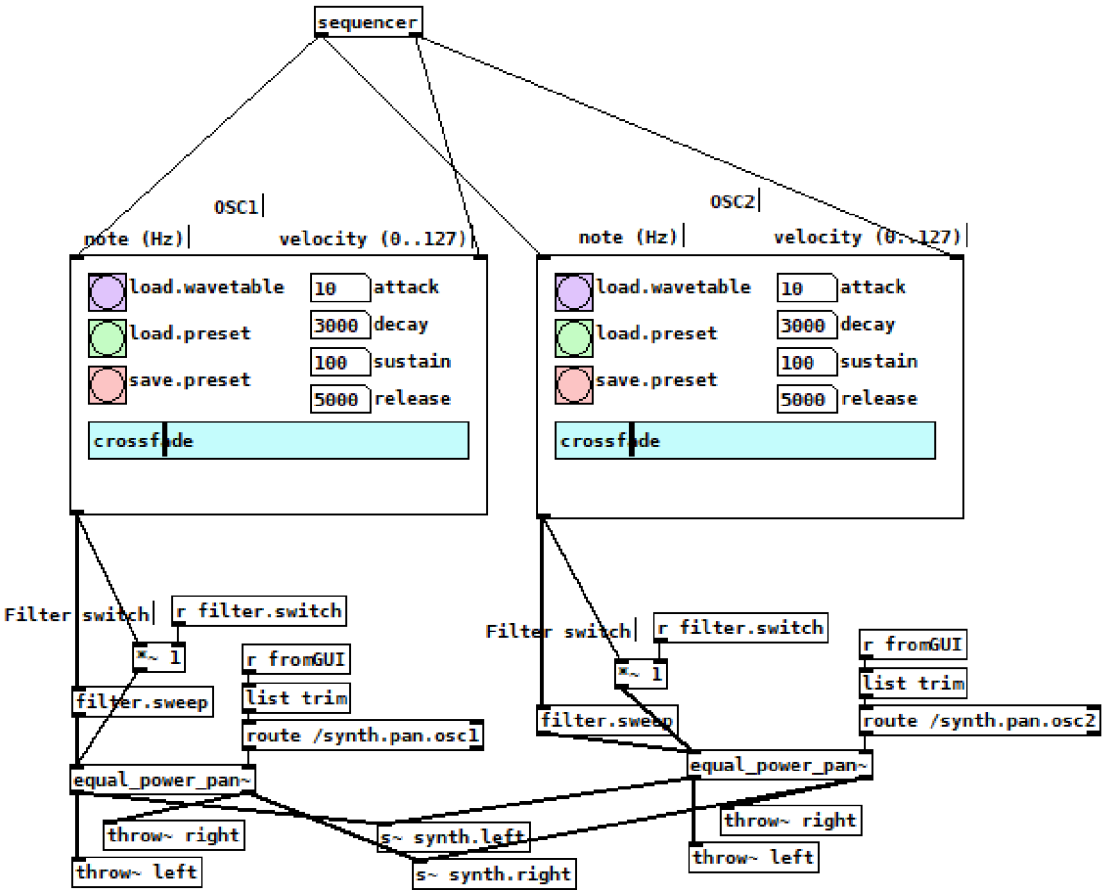
Signály jsou uložené v tzv. wavetables, které lze načíst z pevné paměti zařízení do pole v Pure Data patchi. Wavetables mohou obsahovat pouze jednu periodu signálu (tzv. single-cycle wavetable), nebo set signálů, mezi kterými lze interpolovat potenciometrem "Wt-position"(wavetable position). Procházení wavetable lze také automatizovat nízko frekvenčním oscilátorem LFO. Interpolace mezi signály je realizována lineárním crossfadem.

Jedna perioda signálu ve wavetable se označuje také jako "frame". Přehrávání wavetable je závislé na délce framu, proto je potřeba při načítání wavetable změnit nastevení framu. Velikost framu se udává v samplech. Nejčastěji se lze setkat s

velikostí framu 64, 128, 256, 512, 1024, 2048. Při programování syntetizéru jsem pro testování používal wavetable s velikostí framu 515 samplů, proto jsem tuto velikost zahrnul do aplikace s ostatními běžnými velikostmi.

Pure Data patch se skládá z několika subpatchů a abstrakcí. Stručně tedy popíšu strukturu patche.

Hlavním patchem wavetable syntetizéru je "wavetable.synth.wrap.pd". Ten obsahuje dva wavetable oscilátory, sekvencer, subpatch LFO, abstrakci filter.sweep, equal_power_pan~, audio výstup. Stejný signál, který je odeslán na výstup, je také odeslán do efektu reverb.

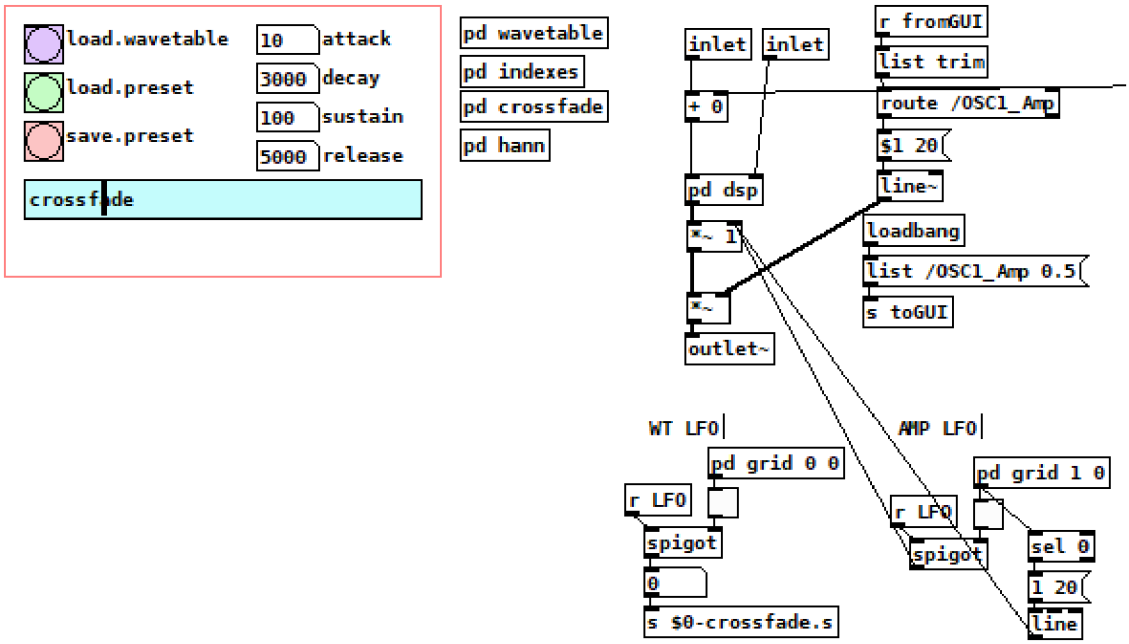


Obr. 3.11: Hlavní patch wavetable syntetizéru

3.6.1 Wavetable oscilátor

Oscilátor obsahuje subpatch pro nahrání wavetable do pole (subpatch wavetable), dále subpatch "crossfade", který určuje pozici přehrávání ve wavetable a interpoluje mezi framy ve wavetable. Subpatch "indexes"vypočítává indexy pro dva framy a zapisuje je do dvou polí index1 a index2, které jsou čteny z patche "crossfade". Subpatch

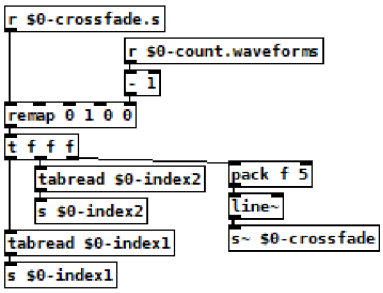
"dsp", jehož vstupy a výstup jsou shodné se vstupy a výstupy oscilátoru, umožňuje vícehlasou reprodukci abstrakce "wavetable.synth.voice.pd". Abstrakce "wavetable.synth.voice"přijímá data z ostatních subpatchů a přehrává zvolenou část wavetable.



Obr. 3.12: Wavetable oscilátor

Většina komunikace mezi patchi probíhá bezdrátově prostřednictvím objektů "send", "recieve", "throw"a "catch"s lokálními andresami aby nedocházelo ke shodě adres při vícehlasé reprodukci patche.

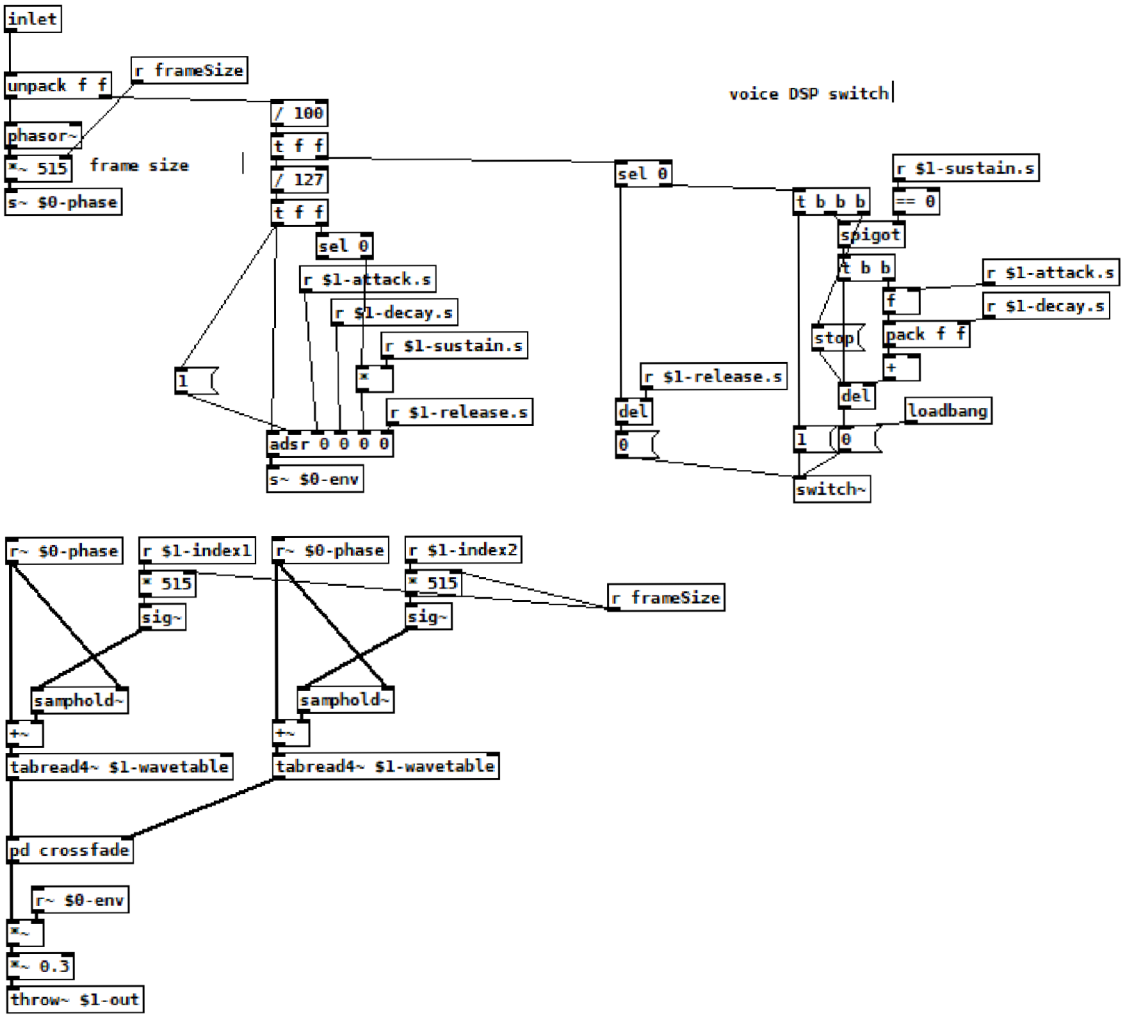
Patch crossfade má tři vstupy: pozici potenciometru "Wt-postion"a počet framů ve wavetable. Výstupy jsou hodnoty čtené z pole framů a signál, který interpoluje mezi pozicemi wavetable. Výstup patche "crossfade"je přijímán v abstrakci "wavetable.synth.voice.pd".



Obr. 3.13: Subpatch crossfade

Wavetable.synth.voice je základem celého oscilátoru. Patch přijímá frekvenci tónu a hlasitost. Frekvencí tónu je řízen oscilátor [phasor~], jehož signál násobený velikostí framu je použit pro vzorkování dvou polí framu. Tímto signálem jsou čteny hodnoty z wavetable. Signál je dále zpracován subpatchem crossfade, který na signály aplikuje Hannovo okno a interpoluje mezi nimi. V posledním kroku je signál násoben amplitudovou obálkou ADSR. Pro generaci obálky jsem použil abstrakci z návodu pro Pure Data od Millera Pucketta (abstrakce se nachází v Pure Data prohlížeči v souboru "audio examples"- adsr.pd).

Abstrakce wavetable.synth.voice má DSP switch, který po doznění noty vypne DSP. Lze tak snížit výpočetní náročnost pro polyfoní patche.



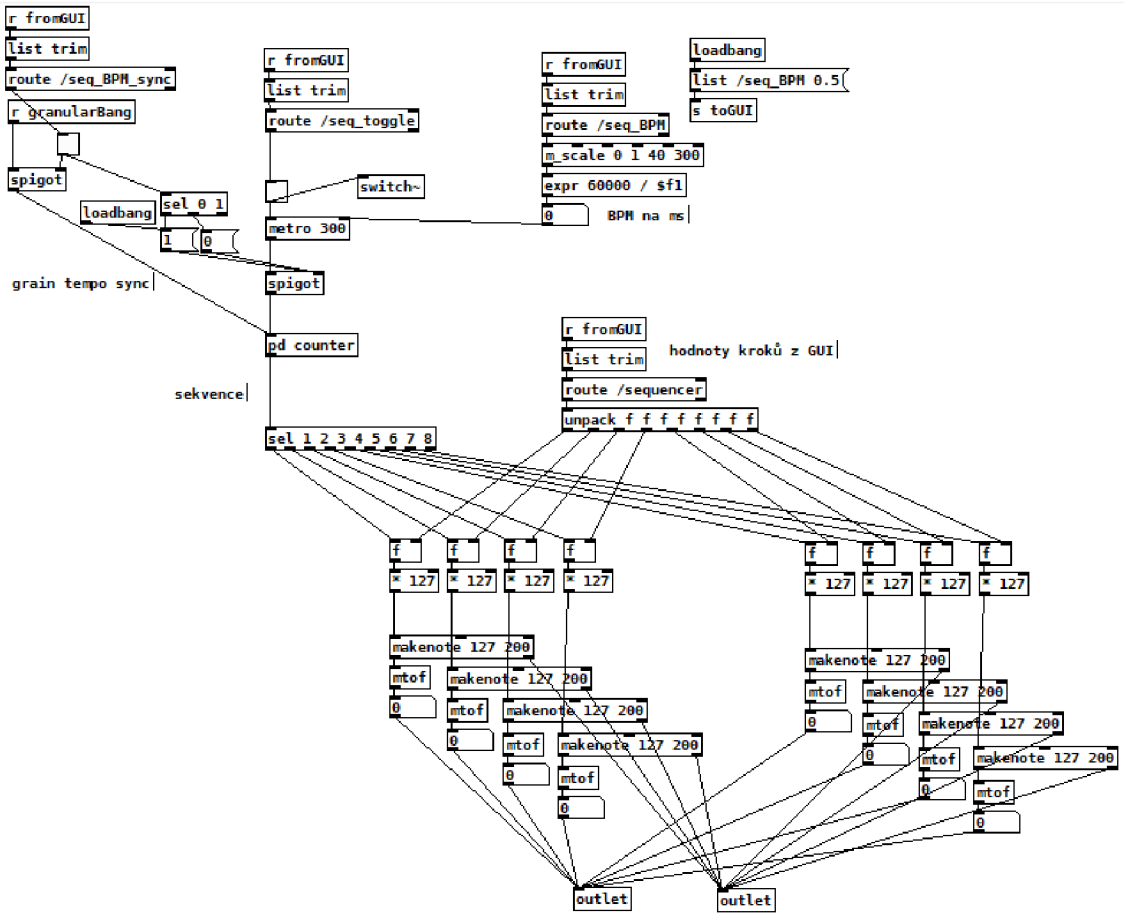
Obr. 3.14: Abstrakce wavetable synth voice

3.6.2 Sekvencer

Funkce sekvenceru se dá rozdělit do tří hlavních částí tj. časování iterace, iterace sekvence, odeslání hodnoty na výstup při každé iteraci.

Časování iterace je řízeno objektem [metro] s volitelnou rychlostí danou v BPM (beats per minute). Iterace probíhá v subpatchi counter. Při každém kroku je objektem [select] směrován bang na příslušný výstup. Dále je hodnota zíkaná z GUI prvku "multislider", která přísluší danému kroku, odeslána do objektu [makenote]. Tento objekt vytvoří časované MIDI zprávy nota zapnuta/vypnuta. Jeho parametry jsou výška tónu noty, hlasitost noty a délka noty. V mém sekvenceru se mění pouze výška tónu noty. Časované MIDI zprávy jsou převedeny na frekvenci funkcí [mtof] (MIDI to frequency) a poslány na výstup sekvenceru.

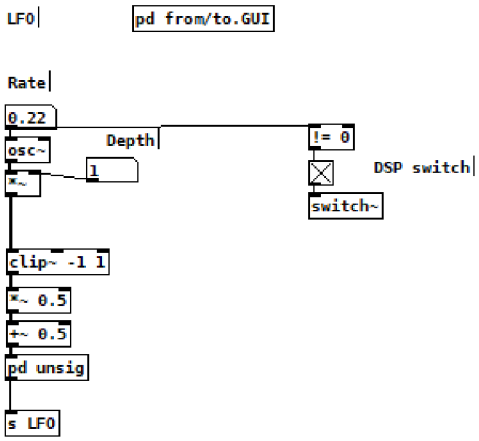
Součástí sekvenceru je možnost synchronizace BPM s granulárním syntetizérem. DSP patche se opět zapíná/vypíná spolu se zapnutím/vypnutím sekvenceru.



Obr. 3.15: Sekvencer

3.6.3 LFO

Základem nízko frekvenčního oscilátoru je oscilátor s kosinusovým průběhem, který je upraven tak, aby osciloval kolem hodnoty 0.5 v rozmezí od 0 do 1. Frekvence oscilátoru určuje frekvenci modulace tzv. "Rate"a amplituda hloubku modulace tzv. "Depth"(realizovaná násobícím objektem za oscilátorem). Hodnoty LFO jsou odesílány do oscilátorů kde modulují amplitudu a pozici přehrávání ve wavetable.



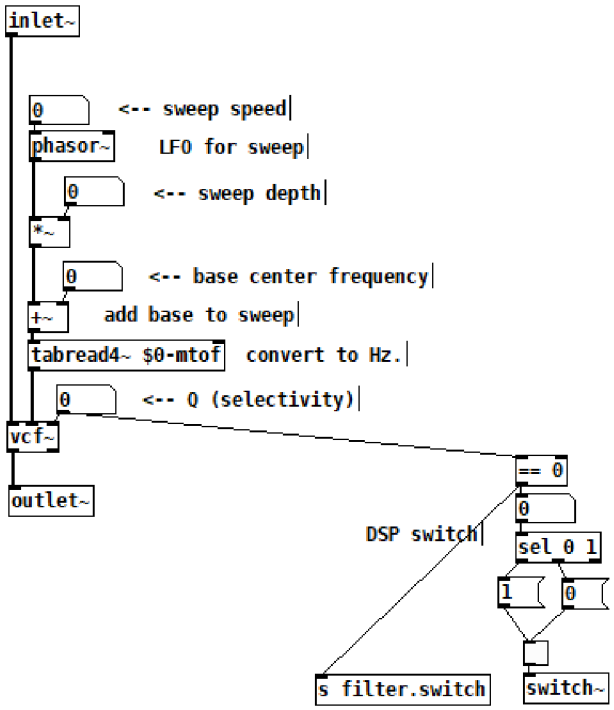
Obr. 3.16: Subpatch LFO

3.6.4 Filter

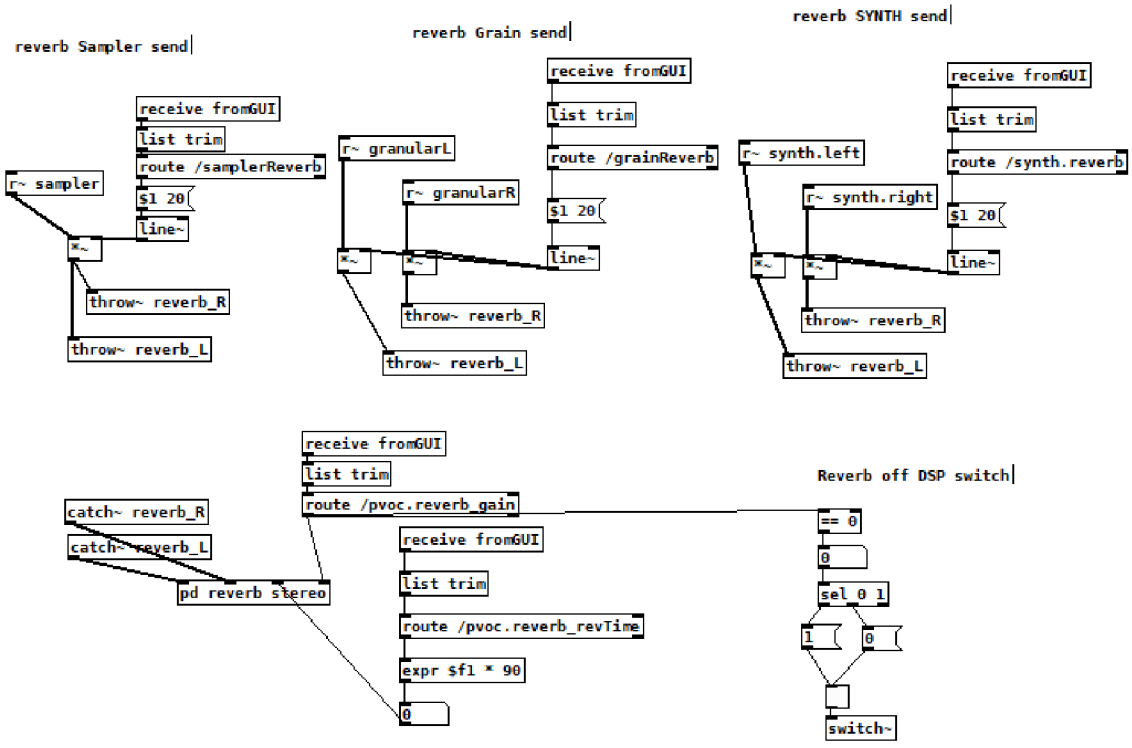
Filter je generován objektem [vcf∼] (voltage-controlled-filter), který má tři vstupy: signál, střední kmitočet filtru, resonance filtru Q. Řízením středního kmitočtu oscilátorem lze dosáhnout filtru řízeného LFO. Tato abstrakce je z návodu pro Pure Data od Millera Pucketta (abstrakce se nachází v Pure Data prohlížeči v souboru "audio examples"- filter.sweep.pd). K abstrakci jsem přidal DSP switch, který reaguje na resonanci filtru Q. Pokud je Q=0 DSP se vypne.

3.7 Efekt reverb

Efekt reverb funguje na principu spožďování signálu několika objekty [delwrite∼] a [delread∼]. Tato abstrakce je z návodu pro Pure Data od Millera Pucketta (abstrakce se nachází v Pure Data prohlížeči v souboru "audio examples"- reverb.pd). Abstrakce má jeden parametr "Feedback". Protože některé sekce aplikace mají stereo výstup, použil jsem abstrakci dvakrát, pro levý a pravý audio kanál. Abstrakci jsem přidal další parametr "Gain". Pro sekce sampler, granulární syntetizér a wavetable syntetizér jsem přidal regulátor odesílaného signálu z těchto sekcí. Efekt reverb je umístěn v abstrakci efekty2.pd.



Obr. 3.17: Filter



Obr. 3.18: Efekt reverb

4 Závěr

V rámci bakalářské práce byly teoreticky popsány metody vytváření experimentálního hudebního nástroje pro systém Android. Znalosti z teoretické části byly využity k návrhu aplikace v druhé části práce a její realizaci v části třetí.

V první části jsou vysvětleny principy syntéz zvukových signálů, které lze reprodukovat samplerem, syntetizéry a efekty v aplikaci. Dále byla popsána problematika zpracování digitálních signálu, která se uplatňuje ve všech částech aplikace. V poslední teoretické části jsou představeny prostředky pro vytváření a implementaci audio aplikace na operační systém Android. Těmito prostředky jsou programovací jazyk Pure Data, operační systém Android a platforma potřebná pro jejich vzájemnou komunikaci (MobMuPlat).

Uživatelské rozhraní aplikace vychází z návrhu popsaném v semestrální práci. Při realizaci se ale vyskytlo několik nedostatků platformy MobMuPlat, musel jsem tedy přizpůsobit uživatelské rozhraní a některé funkce platformě. Příkladem jsou omezené možnosti importu souborů přes platformu do Pure Data patche. Tato část aplikace by určitě potřebovala jiné řešení nebo alternativní platformu.

Uživatelské rozhraní aplikace by bylo vhodné ještě zlepšit, především zavedením menu nebo jiného způsobu orientace v aplikaci. Při realizaci uživatelského rozhraní jsem řešil problém velkého množství ovládacích prvků na malém displeji. Přestože, jsou dnes displeje mobilních zařízení a tabletů dostatečně velké, snažil jsem se, aby aplikce byla snadno ovladatelná i na menších zařízeních. Proto jsem využíval ovládací prvek "table"pro výběr rozsahu hodnot, "knob"místo "slideru"nebo maticový prvek "grid", který umožnil snadné směrování signálu z oscilátoru LFO.

Přestože, aplikace příliš nezatěžuje mobilní zařízení, snažil jsem se snížit zatížení dynamickým vypínání DSP v částech aplikace, které nejsou momentálně využívány.

Literatura

[1] PUCKETTE, M. *Theory and Techniques of Electronic Music*, 2006. 337 s. Dostupné z URL: <http://msp.ucsd.edu/techniques.htm>

[2] ROADS, Curtis. *The Computer Music Tutorial*, 1996. 1256 s. MIT Press. ISBN 0-262-68082-3

[3] ROADS, Curtis. *Microsound*, 2002. 424 s. MIT Press. ISBN: 9780262182157

[4] SCHIMMEL, Jiří. *Studiová a hudební elektronika*, druhé vydání. Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav telekomunikací, Brno 2015. 197 s. ISBN 978-80-214-4452-2

[5] SMÉKAL, Zdeněk. *Analýza signálů a soustav - BASS*, Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav komunikací, Brno 2012. ISBN 978-80-214-4453-9

[6] BRINKMANN, Peter. *Making Musical Apps – Real-time audio synthesis on Android and iOS*, 2012. 124 s. O'Reilly Media. ISBN 978-1449314903

[7] REICHL Jaroslav, Martin Všetička. *Encyklopedie fyziky: Digitalizace analogového signálu* [online], Dostupné z URL: <http://fyzika.jreichl.com/main.article/view/1355-digitalizace-analogoveho-signalu>

Seznam příloh

5 Obsah přiloženého CD

Na přiloženém disku je složka *main*, ve které se nachází všechny Pure Data soubory potřebné ke spuštění aplikace a soubor platformy MobMuPlat: *main.mmp*, který lze spustit v editoru MobMuPlat nebo v aplikaci MobMuPlat pro operační systém Android/iOS. V této složce jsou také zvukové soubory ve formátu *wav*, které se využijí při práci s aplikací.

Ve složce *MobMuPlatDistribution_1.82iOS_0.34Android\Editor\CrossPlatformJava* je editor MobMuPlat, potřebný pro spuštění spuštění aplikace na PC. Na disku se dále nachází *Příručka pro zprovoznění aplikace* a PDF soubor bakalářské práce.

6 Příručka pro zprovoznění aplikace

6.1 Postup pro zprovoznění na PC

1. Spustit MobMuPlat editor, který se nachází v adresáři: *MobMuPlatDistribution_1.82iOS_0.34Android\Editor\CrossPlatformJava*
2. V editoru načíst soubor s projektem: main.mmp
3. Spustit Pure Data Vanilla a načíst soubor*PDWrapper.pd*, který je v adresáři *MobMuPlatDistribution_1.82iOS_0.34Android*
4. V Pure Data Vanilla otevřít hlavní soubor *main.pd*, který se nachází v adresáři *\main*

6.2 Postup pro zprovoznění na operačním systému Android

1. Stažení aplikace MobMuPlat z Google Play url:<MobMuPlat Google Play>
2. Kopírovat soubory ze složky (main) do adresáře MobMuPlat na uložišti zařízení.
3. Spustit aplikaci MobMuPlat a otevřít soubor main.mmp

Více informací o platformě MobMuPlat: `http://danieliglesia.com/mobmuplat/`