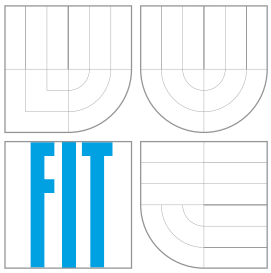
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SYNTÉZA HUDEBNÍCH SIGNÁLŮ NA ANDROIDU

MUSICAL SIGNAL SYNTHESIS ON ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN MIKULÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Dr. Ing. JAN ČERNOCKÝ,

BRNO 2016

Zadání bakalářské práce/18686/2015/xmikul53

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Mikulík Jan**
Obor: Informační technologie
Téma: **Syntéza hudebních signálů na Androidu**
Musical Signal Synthesis on Android
Kategorie: Zpracování řeči a přirozeného jazyka

Pokyny:

1. Shrňte state-of-the art v oboru syntézy hudebních signálů na mobilních zařízeních.
2. Vyberte vhodnou metodu syntézy, základy otestujte v Matlabu.
3. Vytvořte návrh syntezátoru pro systém Android - vlastní syntéza zvuku a sada efektů, včetně návrhu uživatelského rozhraní.
4. Implementujte vytvořený systém.
5. Navrhněte, proveďte a vyhodnoťte uživatelské testy.
6. Vytvořte krátké video dokumentující Vaši práci.

Literatura:

- ŘEZÁČ Martin. Parametrizovatelný hudební nástroj pro mobilní platformy, DP FIT 2013/14
- ŠVEC Michal. Tvorba zvuku v technologii VST, DP FIT 2013/14
- dále dle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3 zadání

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Černocký Jan, doc. Dr. Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cílem této práce bylo navrhnout a implementovat syntetizátor na mobilní platformě Android. Syntetizátor zvládá generovat 3 základní typy signálů - sinus, pila a obdélník. Dále obsahuje dva filtry typu dolní propust a horní propust, které lze ovládat pomocí mezní frekvence. Pro zvukovou interpretaci bylo využito AudioTrack API. Syntetizátor neobsahuje klaviaturu, ale ovládá se pomocí pomocí zón, kde každá zóna značí jeden tón. Zóny si lze libovolně vybrat. K syntetizátoru bylo vytvořeno jednoduché uživatelské rozhraní.

Abstract

The main task of this thesis is to design and implement new synthesizer on mobile platform under Android. The synthesizer generates three types of signals - sine, sawtooth and square. Two types of filters can be applied, low pass and high pass, and their cut-off frequency can be controlled. For the sound interpretation, an AudioTrack API was used. The synthesizer is controlled by zones, where one zone equals one tone. Zones can be selected by the user. For this synthesizer, a simple user interface was created.

Klíčová slova

Hudební nástroj, hudba, syntéza, syntetizátor, signál, filtry, Android

Keywords

Musical instrument, music, synthesis, synthesizer, signal, filters, Android

Citace

MIKULÍK, Jan. *Syntéza hudebních signálů na Androidu.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Černocký Jan.

Syntéza hudebních signálů na Androidu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Dr. Ing. Jana Černockého. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Mikulík
16. května 2016

Poděkování

Poděkování patří především panu Doc. Dr. Ing. Janu Černockému za odbornou pomoc a cenné rady při vývoji této práce. Dále bych chtěl poděkovat Ryanu Foo z New Yorkské univerzity za pomoc s druhým návrhem této práce. Poděkování patří také uživatelům, kteří výslednou aplikaci otestovali.

Obsah

Kapitola 1

Úvod

Tato práce byla vytvořena na Fakultě informačních technologií na Vysokém učením technickém v Brně. Cílem této práce bylo navrhnout a implementovat syntetizátor na mobilní platformě Android s použitím filtrů a efektů. V dnešní době se na trhu vyskytuje obrovské množství hudebních programů, jak na počítačích, tak i na mobilních platformách. Mezi mobilními platformami prozatím v hudbě vítězí mobilní platforma iOS, která je vyvíjena společností Apple Inc. Tato platforma nabízí široké spektrum hudebních nástrojů všeho druhu a lze ji najít také u většiny hudebníků. Proto také byla vybrána platforma Android.

Aplikace je implementována pouze v jazyce Java, existují také jiné možnosti a ty si probereme dále. Pro zvukovou interpretaci bylo využité AudioTrack API, uživatelské rozhraní bylo cíleno vytvořit jednoduché a přehledné, aby aplikace byla jednoduchá a použitelná i ve zhoršených podmínkách, tedy živě na koncertě.

1.1 Motivace

Mobilní platformy se čím dál více dostávají do popředí a pomalu se čím dál více umělců snaží tuto platformu zakomponovat i do živých vystoupení. Známý klávesista Jordan Rudess ze skupiny Dream Theater si již několik let vyvíjí na platformě iOS nové hudební aplikace, které nemají obdoby a jsou zcela jedinečné. Navíc jeho nejslavnější aplikace MorphWiz byla portována také na ostatní mobilní platformy. Tato aplikace je také inspirací k napsání této práce a vyzkoušení si, jak by se taková aplikace mohla vyvíjet.

Kapitola 2

Existující syntetizátory na platformě Android

Tato kapitola shrnuje současný výběr syntetizátorů na Google Play [1], většina z nich je dostupná zdarma.

2.1 MorphWiz

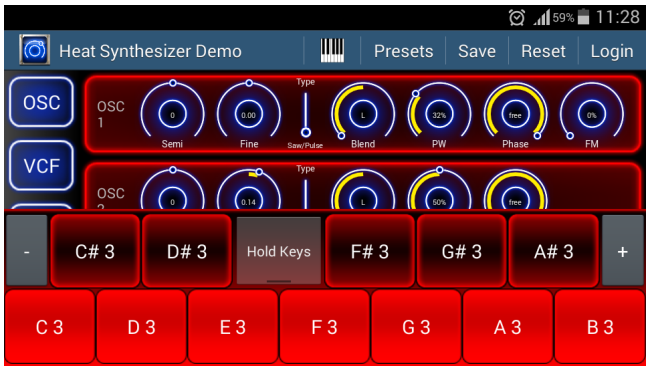
Tato aplikace je vytvořena známým klávesistou Jordanem Rudessem z kapely Dream Theater. Tuto aplikaci mimo jiné používá i živě na svých koncertech. Syntetizátor neobsahuje žádnou editaci zvuku, pouze nabízí již před vytvořené zvuky, kde každý zvuk má také jiné rozhraní. U každého zvuku jde nastavit ladění podle nabízených stupnic pro volnější hru. Klaviatura není tvořena klavírním rozhraním, ale pomocí strun, na které se hraje a u každého zvuku se liší efekt při posunutí výšky dotyku prstu po struně. Syntetizátor také obsahuje efekt glissanda, kdy přechod z jednoho tónu na druhý je plynulý. Některé typy zvuků však klaviaturu neobsahují, neboť jednotlivé tóny jsou irelevantní. Tento syntetizátor je na současném trhu jedním z nejlepších, sice bohužel bez editace zvuků (pravděpodobně z důvodu náročného generování samotných zvuků), ale stále je velice použitelný a latence je velice dobře řešena[1].

Obrázek 2.1: Morphwiz

[1] https://play.google.com/store/apps/details?id=com.wizdommusic.morphwiz

2.2 Heat Synthesizer

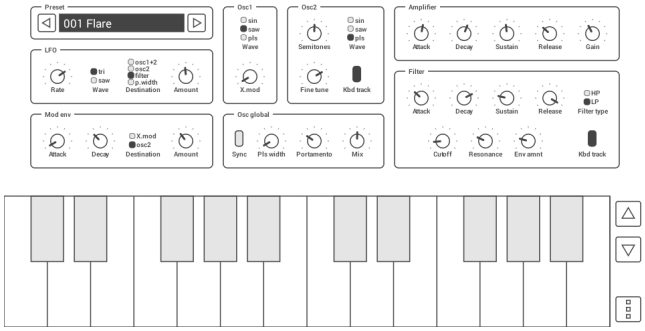
Komplexní syntetizátor obsahující obrovské množství parametrů k editaci. Jak jde vidět z obrázku 2.2, pro malé displeje je velmi nepřehledný a práce s ním je obtížná. Jde vidět, že je spíše dělaný pro tablety větší velikosti, trochu místa by se navíc ušetřilo skrytím systémového panelu. Existuje také jeho placená verze PRO s ještě větší funkčností, která je již ve volné verzi velmi bohatá. Zvukově je pravděpodobně současně nejlepší syntetizátor na Google Play, pokud nepočítáme Morphwiz, který nemá editační možnosti[2].



Obrázek 2.2: Heat Synthesizer

2.3 Common Analog Synthesizer

Tento syntetizátor využívá dva oscilátory, u každého z nich lze vybrat typ zvukového signálu (sinusový, pilový a pulzní). Syntetizátor dále obsahuje další moduly pro detailnější editaci zvuku, jako jsou filtry, amplitudu, modulaci obálky a LFO[3]. Bylo zde zvoleno jednoduché a přehledné rozhraní. Je zde možnost také ukládání vytvořených zvuků. Tento syntetizátor se nejvíce přibližuje běžným syntetizátorům na počítačích, což je jeho velká výhoda a také unikátnost[4].



Obrázek 2.3: Analog common synthesizer

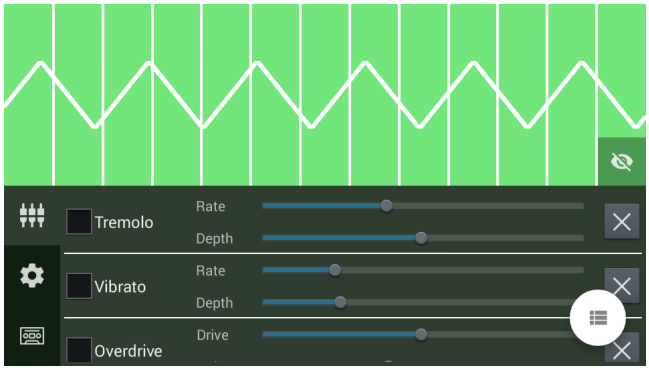
[2] https://play.google.com/store/apps/details?id=com.nilsschneider.heat.demo

[3] Low frequency oscillator (LFO) je nízko frekvenční oscilátor (obvykle pod 20Hz), který se používá k modulaci jiných komponent syntetizátoru.

[4] https://play.google.com/store/apps/details?id=org.oxxxide.vasynth

2.4 Synth

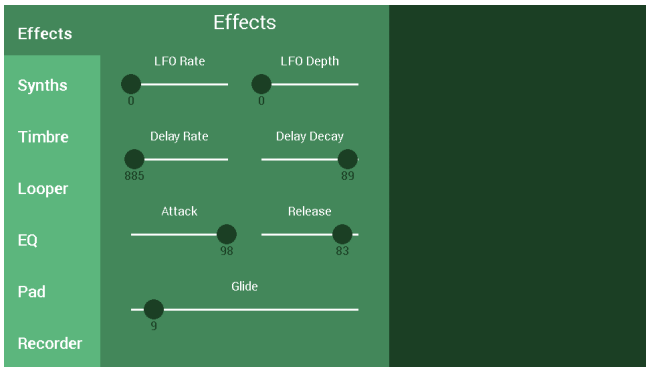
Tento syntetizátor obsahuje pouze jeden oscilátor a minimum editace zvuku, na rozdíl od ostatních avšak nabízí výběr základních efektů, jako například tremolo, vibrato, overdrive, reverb atd. Vzhled tohoto syntetizátoru je netradiční, klaviatura je rozdělená na zóny a podle pozice dotknutí prstu se rozlišuje i hlasitost. V pozadí se navíc zobrazuje tvar signálu. Chybí mi tam ale označení jednotlivých tónů, což by orientaci určitě zlepšilo. Syntetizátor také nabízí výběr tónů jako to je u aplikace MorphWiz (sekce 2.1)[5].



Obrázek 2.4: Synth

2.5 Saucillator

Syntetizátor na bázi thereminu[6], jako základ zvuku lze vybrat několik typů signálu a na něj aplikovat efekty jako LFO, Delay nebo Attack a Release. Také se nastavuje vlastnost Glide, která značí jak intenzivně probíhá změna tónu při posunutí prstu. Uživatelské rozhraní je přehledné, ale určitě by se zvládlo udělat lépe, nelíbí se mi jak jsou udělané otočné knoby a přepínání mezi typem signálu pomocí + a -, které by se dalo vyřešit jednoduchou nabídkou[7].



Obrázek 2.5: Saucillator for Android

[5] https://play.google.com/store/apps/details?id=it.psas.synth

[6] Theremin je elektronický hudební nástroj na který se hraje, aniž by se ho hráč jakkoli dotýkal. Je tvořen dřevěnou skříní se dvěma anténami a je ovládán pouze pohybem paží, dlaní a prstů.

[7] https://play.google.com/store/apps/details?id=com.mattfeury.saucillator.android

2.6 G-Stomper Studio

Plnohodnotná hudební pracovní stanice, obsahující také vlastní syntetizátor. Tato aplikace je zde zmíněna jen pro demonstraci jak komplexní se dá vytvořit hudební aplikace a jak se čím dál více snižuje rozdíl mezi konkurenční platformou iOS[8].

Obrázek 2.6: G-Stomper Studio

2.7 KORG i ONE

Tento syntetizátor by se dal označit jako rompler[9], jelikož je vytvořen podle slavné řady nástrojů PA[11] stejnojmenného výrobce. Neobsahuje totiž žádnou editaci zvuků, pouze modulační joystick, který je pro tohoto výrobce také charakteristický. Panel nástroje obsahuje množství ovládacích prvků pro takzvaného bubeníka, který hraje podle zvoleného stylu. Celkově je aplikace velmi dobře provedená. Tato aplikace však momentálně na Google Play není dostupná.

Obrázek 2.7: KORG i ONE

[8] https://play.google.com/store/apps/details?id=com.planeth.gstomper

[9] Rompler je označován za druh hudebního nástroje, který pouze přehrává zvuky, které má uložené v paměti.

2.8 Sonic synth

Syntetizátor obsahuje narozdíl od ostatních až 3 oscilátory, u každého lze nastavit typ zvukového signálu, hlasitost, modulace, transpozice a míra rozladění. Tímto však jeho možnosti končí a jeho zvuk není tak použitelný[10].

[10] https://play.google.com/store/apps/details?id=com.finestandroid.piano

Kapitola 3

Cíl práce

Výsledkem této práce by měl být syntetizátor schopný generování základního signálu a následně jeho filtraci.

3.1 Ovládání

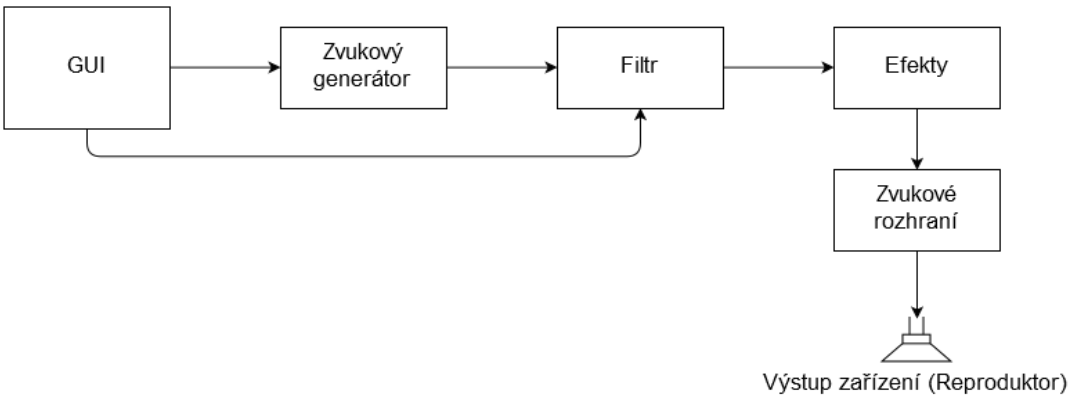
Ovládání by mělo být jednoduché a prosté pro rychlou potřebu změnit parametry hraní. Prvky uživatelského rozhraní musí být adaptivní vůči jakémukoliv rozlišení.
Hrací plocha nebude obsahovat jednotlivé klávesy, ale bude obsahovat zóny, kde každá zóna znamená jeden zón. Počet zón a jednotlivé tóny si uživatel zvolí sám.

3.2 Syntéza

Syntetizátor by měl generovat základní tři typy signálu - sinus, pila a obdelník. Dále jednoduchý filtr, u kterého lze nastavit mezní (cut-off) frekvence.
Změna tónu by měla být spojitá, tzv. efekt glissanda. Při dotyku horní části displeje by se měl aktivovat efekt vibráta.

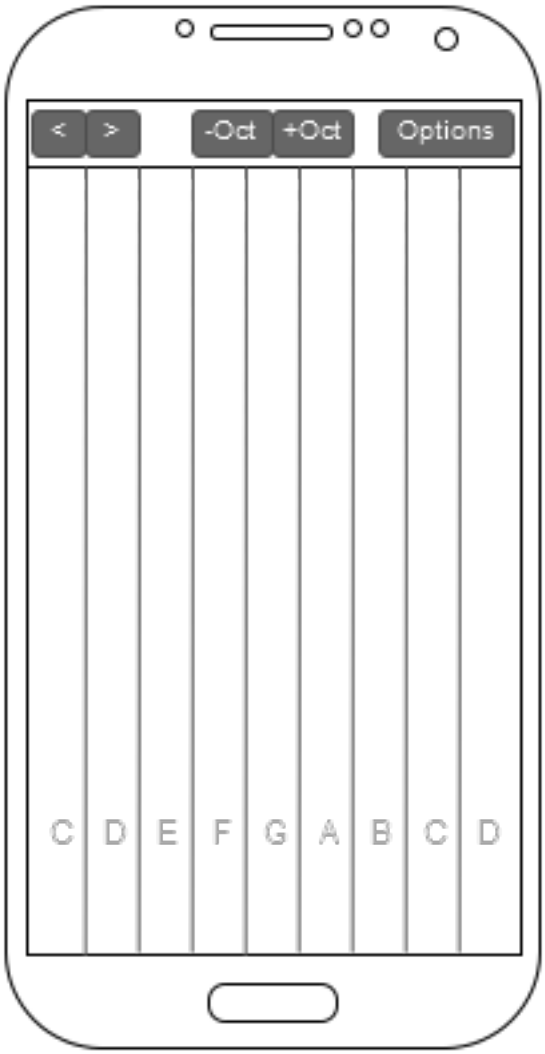
3.3 Návrh aplikace

Základní návrh aplikace je zobrazený na obrázku 3.1.



Obrázek 3.1: Blokové schéma aplikace

Již na počátku vývoje této aplikace byla představa o uživatelském rozhraní jasná. Mockup UI je zobrazen na obrázku 3.2, uživatelské rozhraní je velmi jednoduché, obsahuje pouze důležité kontrolky pro ovládání hry, velkou plochu pro hru a případné další nastavení je schované pod tlačítkem nastavení.



Obrázek 3.2: Mockup vzhledu aplikace

Kapitola 4

Zvuková syntéza

Zvuková syntéza je technika generování zvuku dle specifikovaných parametrů pomocí nějakého zařízení (syntetizátoru) nebo softwaru [7].
Existují dva druhy generování zvuku:

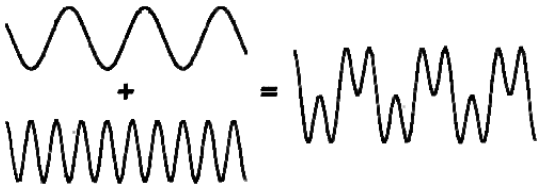
- Napodobení již existujícího zvuku v reálném světě (klavír, flétna atd.)
- Vytvoření zcela nového zvuku pomocí elektronického zařízení

4.1 Součtová syntéza

Základním stavebním kamenem této syntézy je sinusový signál. Jednotlivým vrstvením tohoto signálu vzniká postupně komplexnější signál. V praktickém případě se jedná o množství oscilátorů generující tuto funkci. Ve výsledku vzniká nový signál s určitou frekvencí a amplitudou.

$$y(t) = \sum_{k=1}^{K} A_k \sin(\omega_k t + \varphi_k), \tag{4.1}$$

kde K je počet sčítajících se signálů, A_k je amplituda signálu k, ω_k je normovaná kruhová frekvence signálu k a φ_k je fáze signálu k. Na obrázku 4.1 lze vidět, jak sečtením dvou sinusových signálů s jinou frekvencí se postupně tvaruje obdélníkový signál. Dalším takovým postupným sčítáním by se signál čím dál více tvaroval do obdélníkového signálu [7].



Obrázek 4.1: Výsledek sečtení dvou sinusových signálů. Převzato z [7].

4.2 Rozdílová syntéza

Rozdílová syntéza je přesný opak součtové syntézy. Základem je složitý a frekvenčně bohatý signál a postupně se z něj odstraňují složky, abychom docílili výsledného zvuku, který

potřebujeme. Základem je oscilátor generující signál (pilový, obdélníkový) a na něj se aplikuje filtr, například dolní propust nebo horní propust. Příkladem může být signál bohatý na frekvence, například bílý šum, obdélník nebo pila. Následný filtr tyto frekvence ořeže a vznikne výsledný zvuk [12].

4.3 Frekvenčně modulační syntéza (FM)

FM syntéza je forma syntézy, kde základní signál je modulován jiným signálem. V nejjednodušší formě tato syntéza obsahuje dva sinusové signály. První se nazývá modulační signál, druhý nosný signál a modulační signál mění frekvenci nosného signálu. Tato syntéza také umí jednoduše vytvořit efekt vibráta, pokud je frekvence modulačního signálu menší než 30 Hz.

$$y(t) = A\sin(\omega_c t + \beta\sin(\omega_m t)), \tag{4.2}$$

kde A je amplituda signálu, ω_c je frekvence nosného signálu, β je modulační index, ω_m je frekvence modulačního signálu a t je čas [7] [12].

4.4 Základní techniky zpracování signálu

4.4.1 Generování

Generování zvuku neprobíhá v reálném čase, tedy pracujeme s pamětí, ve které je uložen nějaký blok toho zvuku, který přehráváme a mezitím probíhá generování nového bloku. Při generování pracujeme s fází, která je na začátku nulová a postupně narůstá hodnotou, která je známa jako inkrementace fáze. Tato hodnota je k fázi připočítána při každé iteraci výpočtu nového vzorku. Vzorec pro výpočet n-tého vzorku sinusové vlny je

$$x[n] = sin(\omega n) \tag{4.3}$$

kde ω a n je fáze φ_n, potom n+1 vzorek je

$$x[n+1] = sin(\omega(n+1)) \tag{4.4}$$

kde ω a $(n+1)$ je fáze φ_{n+1}, pak tedy získáme rozdíl fází

$$\Delta\varphi = \varphi_{n+1} - \varphi_n = \omega_n + \omega - \omega_n = \omega \tag{4.5}$$

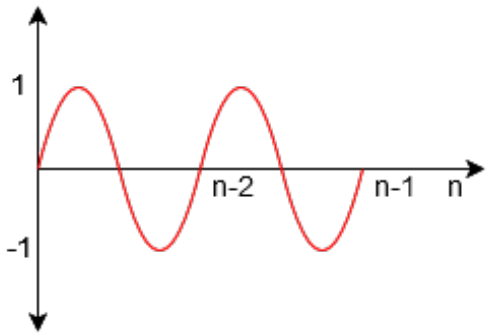
potom

$$\text{Inkrementace fáze} = \omega = \frac{2\pi\text{Frekvence tónu}}{\text{Vzorkovací frekvence}} \tag{4.6}$$

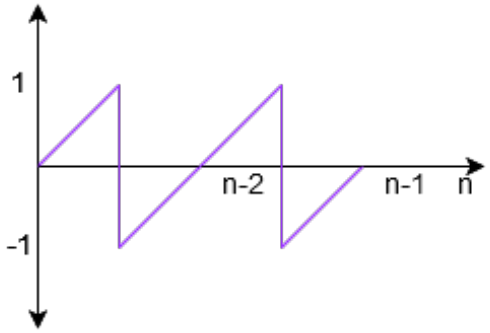
Při generování signálu je také nutné, aby začínal v bodě 0 a také tak končil.

Sinus

Sinusový signál se generuje velmi snadno, při výpočtu nového vzorku se použije hodnota výsledku sinusové funkce s použitím fáze. Pokud fáze přesahuje hodnotu 2π, tak je vhodné ji o tuto hodnotu také odečíst. Lépe se s ní pracuje a nedojde k přetečení proměnné.



Obrázek 4.2: Ukázka sinusového průběhu



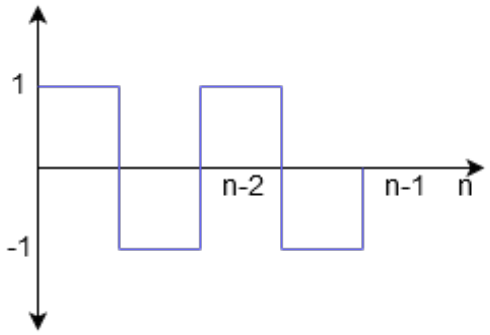
Obrázek 4.3: Ukázka pilovitého průběhu

Pila

U generování signálu s pilovitým průběhem je hodnota nového vzorku hodnota fáze, pokud fáze překročí hodnotu 1, odečítá se -2. Zde se ale inkrementace fáze dělí hodnotou π.

Obdélník

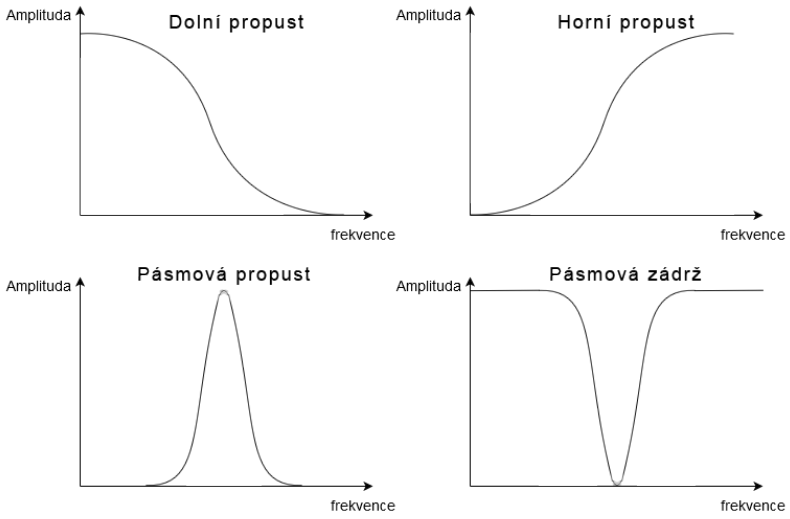
Generování obdélníkového signálu nebo také pulzního signálu je velmi jednoduché, každé π se střídají pouze hodnoty 1 a -1 a frekvence signálu udává pouze rychlost změny. Při přesažení hodnoty fáze 2π se také tato hodnota od ní odečítá.



Obrázek 4.4: Ukázka obdélníkového průběhu

4.4.2 Filtrování

Filtrování jak už z názvu vypovídá je proces, kde se část vstupního signálu odstraní a výsledkem vzniká transformovaný signál. Podle výsledků, jakých chceme dosáhnout, existují čtyři základní typy filtrů - dolní propust, horní propust, pásmová propust a pásmová zádrž.



Obrázek 4.5: Základní typy filtrů

Dolní a horní propust operují na základě mezní frekvence. V ideálním případě by všechny frekvence od této hodnoty byly potlačeny, ale to zkrátka není možné a nelze vyfiltrovat všechny frekvence v určitém měřítku. Pásmová zádrž a propust jsou kombinace dvou již zmíněných typů, jedna propouští frekvence mimo úzké pásmo okolo mezní frekvence a druhá dělá přesný opak, tedy propouští pouze úzké pásmo okolo mezní frekvence.

Filtry se dělí na dva typy: nekonečná impulsová odezva IIR (infinite impulse response) a konečná impulsová odezva FIR (finite impulse response). FIR jsou jednoduší, lépe se navrhují a jsou účinné v jednoduchých situacích. IIR jsou složitější, jelikož se k výpočtu přidává zpětná vazba, dodávají však lepší výsledky. Vzorec pro výpočet IIR filtru

$$y[n] = \sum_{i=0}^{Q} b_i x[n-i] - \sum_{j=1}^{P} a_j y[n-j] \tag{4.7}$$

kde $y[n]$ je výstupní signál, $x[n]$ je vstupní signál, a_j jsou zpětnovazební koeficienty filtru, P je přímý řád filtru, b_i jsou přímé koeficienty filtru a Q je zpětnovazební řád filtru.

Koeficienty musí být určeny tak, aby se po jejich vynásobení umístily uvnitř jednotkové kružnice. Pokud se vyskytují mimo jednotkovou kružnici, filtr se stává nestabilní [7] [13]. Výpočet koeficientů pro dolní propust

$$x = e^{-14.445 \frac{\text{mezní frekvence}}{\text{vzorkovací frekvence}}} \tag{4.8}$$

$$a = [(1-x)^4] \tag{4.9}$$

$$b = [4x, -6x^2, 4x^3, -x^4] \tag{4.10}$$

Výpočet koeficientů pro horní propust

$$x = e^{-2\pi \frac{\text{mezní frekvence}}{\text{vzorkovací frekvence}}} \tag{4.11}$$

$$a = [\frac{1+x}{2}, -\frac{1+x}{2}] \tag{4.12}$$

$$b = [x] \tag{4.13}$$

Kapitola 5

Programování na Androidu

5.1 Android OS

Mobilní operační systém založený na Linuxovém jádře. Je dostupný jako open source (otevřený software). Je vyvíjen konsorciem Open Handset Alliance[1] a je orientován převážně na chytré telefonní zařízení a tablety. Nově i na ostatní chytré zařízení. Již od začátku vývoje se Android vyvíjel jako platforma, kterou je možné zprovoznit na různých hardwarových zařízeních. Proto je také nejvyužívanějším operačním systémem na poli mobilních telefonů.

Jednotlivé verze operačního systému jsou číslovány a od verze 1.5 je každá verze také označována názvy různých sladkostí, například KitKat nebo Lollipop. Každá verze má také svoji API úroveň.
V případě vývoje aplikace je nutné stanovit minimální úroveň API aplikace, které značí od které verze systému bude aplikace podporována [2].

5.2 Architektura Androidu

Architektura systému je rozdělena do 5 vrstev:

Jádro (Linux Kernel) - Jádro je základem operačního systému a zajišťuje především komunikaci mezi hardwarem a softwarem. Stará se také o správu procesů, paměti, napájení nebo síťové spojení atd. Jádro je převzato z Linuxu, standardně ve verzi 2.6.

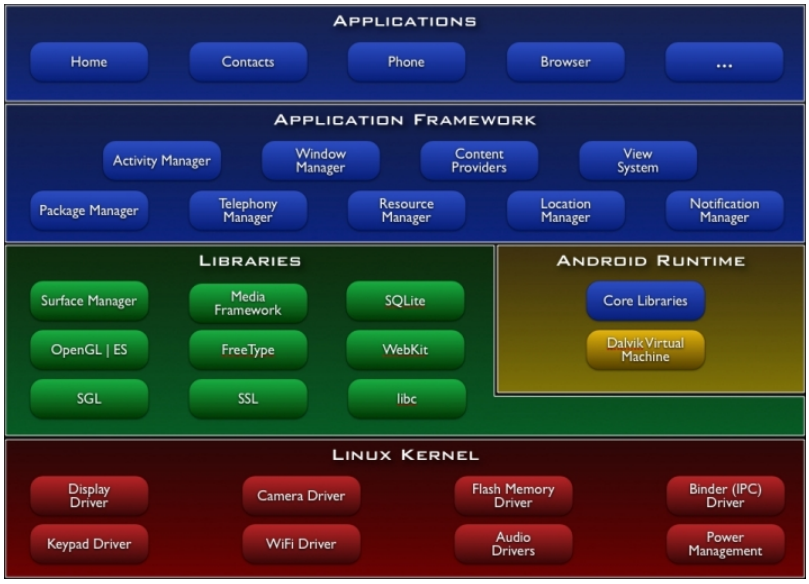
Knihovny - Nativní knihovny, napsané v C++ implementující základní funkce systému. Patří sem také knihovna OpenSL ES, která slouží pro práci se zvukem.

Android Runtime - Slouží primárně pro běh aplikací. Jelikož Android aplikace nejsou napsané v nativním jazyce, ale jazyce Java, tak pro překlad do nativního kódu slouží virtuální stroj Dalvik Virtual Machine.

Aplikační Framework - Obsahuje další knihovny, napsané v Javě, tvořící systémové API, což je soubor funkcí dostupných pro vývojáře umožňující práci s prvky operačního systému. Lze zde také najít AudioTrack API, které je základem této práce.

Aplikace - Samotné aplikace přístupné pro běžné uživatele, také výsledek této práce.

[1] Open Handset Alliance je uskupení výrobců mobilních telefonů, telekomunikačních operátorů a technologických firem [8].



Obrázek 5.1: Architektura operačního systému Android. (Zdroj: svetandroida.cz)

5.3 Zvukové API

Android OS nabízí několik možností jak pracovat se zvukem, ale pro pokročilejší využití jsou k dispozici dvě možnosti. První je využití AudioTrack API, které je napsané v jazyce Java a její použití je jednoduché, je také součástí vývojového balíku Android SDK. Naproti tomu druhá možnost je využití knihovny OpenSL ES[2], která je napsaná v nativní jazyce (C, C++) a k jednotlivým funkcím této knihovny je třeba přistupovat přes nativní kód. Knihovny napsané v nativním kódu se řadí do vývojového balíku Android NDK[3], jenž umožňuje programátorovi použít nativní kód v Android aplikaci napsané v jazyce Java, toto rozhraní se nazývá JNI (Java Native Interface). Tento balík také obsahuje několik dalších knihoven zmíněných výše na obrázku 5.1. Tedy využití nativního kódu nabízí vyšší výkon a nižší latenci [3] [9].

5.3.1 AudioTrack

Tato třída dokáže přehrávat jeden zvukový zdroj v čase pro Java aplikace. Umožňuje také streamování PCM[4] bufferů na zvukový výstup. Součástí třídy je metoda `write (short[] audioData, int offsetInShorts, int sizeInShorts)`, která odesílá obsah bufferu audioData na zvukový výstup. Je možné také použít buffer typu `byte` nebo `float`, avšak typ `float` byl zabudován až od Android API 21 a to zatím není tak rozšířené.
Instance této třídy může operovat ve dvou módech - Static nebo stream. Ve streamovacím módu aplikace neustále zapisuje data pomocí metody `write()`. Write je blokující operace, která následně vrací počet zapsaných dat.
Statický mód by se měl využívat pro krátké zvuky, které jsou malé velikosti a není problém je přehrávat s nízkou latencí [5].

[2]Knihovna pro zpracování zvuku. Dostupná z [9].
[3]Vývojový balík obsahující nástroje pro práci v nativním jazyce. Dostupný z [3].
[4]Pulzně kódová modulace je modulační metoda převodu analogového zvukového signálu na signál digitální, ekvivalentní prostému vzorkování a kvantování.

5.3.2 OpenSL ES

OpenSL ES nabízí pokročilé zvukové vlastnosti jako přehrávání PCM dat, MIDI[5] a zvuků uživatelského rozhraní. Dále nabízí základní ovládání zvuku jako hlasitost, tempo nebo výšku, nebo efekty jako ekvalizér, zesílení basových frekvencí, reverb a rozšíření sterea.
Nabízí podporu několika MIDI technologií a možnost editace MIDI zpráv. Také nabízí podporu audio nahrávání v PCM formátu nebo jiných formátech - může být z mikrofonu nebo line-in vstupu.
Tato technologie umožňuje vývoj aplikací jako hudební přehrávače, MIDI sekvencery, hlasové rekordéry. Také může mít uplatnění ve 2D nebo 3D hrách [9].

5.4 Práce se zvukem

5.4.1 Buffery

Pro práci se zvukem je potřeba rezervovat paměť pro blok zvukových dat, se kterým chceme nějakým způsobem manipulovat či jej přehrát. K tomu slouží buffer, který je technicky řečeno pole hodnot.

Hodnoty bufferu můžou být typu `byte`, `short` nebo `float`. Pokud bychom chtěli do bufferu generovat sinusový signál, kde rozsah hodnot je od -1 do 1, tak jediná možnost je použít typ `float`. Velikost bufferu může být různá a liší se od zařízení. U každého zařízení je hardwarově definováno, kolik by měla být optimální velikost. Navíc moderní výkonnější zařízení obsahují technologii Fast Track, která nabízí nižší latenci pro audio výstup.

Normalizace hodnot

U použití obou zvukových API se při inicializaci musí stanovit formát zakódování bufferu, těchto formátů existuje několik, ale pro nás nejdůležitější jsou následující tři:

- ENCODING_PCM_16BIT,
- ENCODING_PCM_8BIT,
- ENCODING_PCM_FLOAT

V důsledku to znamená, že výsledný buffer, který budeme zasílat na audio výstup musí být v nějakém z výše zmíněném formátu. Nejčastější volba je 16 bitový PCM formát, na který můžeme použít typ `short`, který má rozsah $(-32768; 32768)$. 16 bitů lze převést jako 2^{16} možností, a to je 65536 [5].

5.4.2 Vlákna a synchronizace

V aplikaci implicitně běží jedno hlavní vlákno, které se stará o uživatelské rozhraní. Reaguje na podněty uživatele a úkony odpovídající tomu, co dělá. V důsledku ale není možné vykonávat složitější výpočetní operace, které by mohly na dobu vykonávání zablokovat uživatelské rozhraní.
Na takové úkony je třeba vytvořit nové vlákno, které se o tyto operace bude starat. Android OS nabízí několik typů, které je možné využít, ale my si tady zmíníme jen 3 nejzajímavější, které by mohly mít u hudební aplikace využití [4].

[5] Musical Instrument Digital Interface (MIDI) je komunikační protokol pro hudební průmysl, umožňující komunikaci různých typů zařízení [6].

AsyncTask (Asynchronní vlákno) - Tato třída umožňuje vykonávat operace v pozadí aplikace a následně dodávat výsledky do UI vlákna bez nutnosti manipulace vláken. Také je jednoduchá na použití.

Service (Služba) - Aplikační komponenta, která slouží pro dlouhodobé operace na pozadí a neobsahuje uživatelské rozhraní. Služba se může starat o síťovou komunikaci, hraní hudby, práci s databází atd.

Thread (Vlákno) - Základní vlákno, které se dá vytvořit. Jinak ničím výjimečné.

V rámci této práce je použito vlákno AsyncTask kvůli snadnému použití.

5.5 Použité knihovny

5.5.1 TarsosDSP

Knihovna implementovaná v jazyce Java umožňující práci se zvukem[6]. Tuto komplexní knihovnu lze využít i na platformě Android, ale pro naše účely to není potřeba. Z této knihovny byla pouze převzata implementace IIR filtru a výpočet koeficientů jednotlivých typů filtrů.

[6] https://github.com/JorenSix/TarsosDSP

Kapitola 6

Návrh

V této kapitole se budou rozebírat návrhy aplikace a následně i uživatelské rozhraní. Dále také náročnost implementace a výhody a nevýhody jednotlivých návrhů.

6.1 Návrh syntetizátoru

6.1.1 První návrh

Tento návrh využívá třídu AudioTrack pro práci se zvukem Aplikace je rozdělena na výpočetní část a uživatelské rozhraní. Výpočetní část zahrnovalo vlákno generující sinusový signál a ten následně zapisovalo na audio výstup.
Uživatelské rozhraní obsahuje pouze základní operace ovládání syntetizátoru, jako posunutí rozsahu tónů a navýšení tohoto rozsahu. Tento návrh byl pouze základ pro další postup, který následuje v druhém návrhu.

6.1.2 Druhý návrh

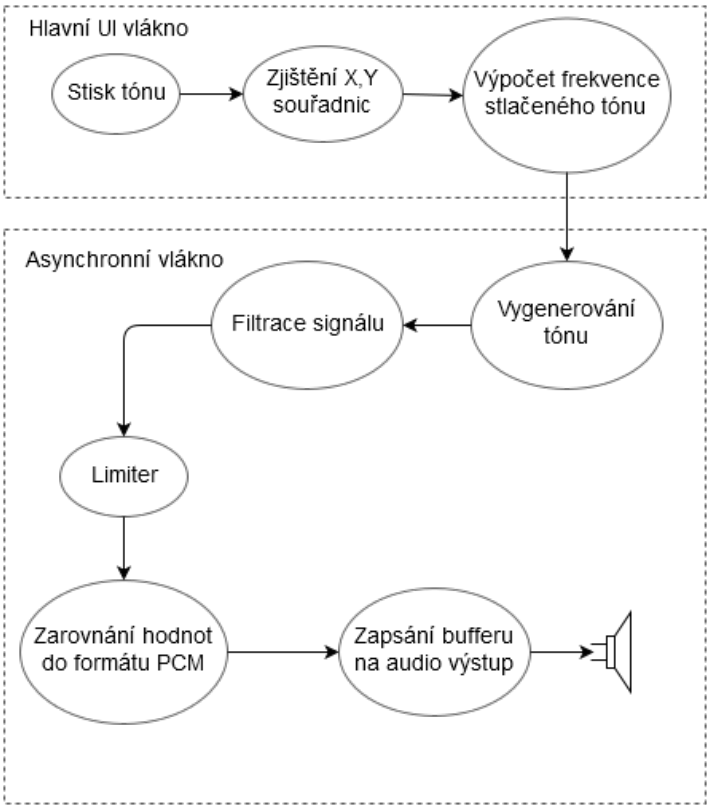
Zde je využita technologie OpenSL ES, která pouze modifikuje přístup ke zvukovému rozhraní, jinak si aplikace zachovává stejné rozložení jako u prvního návrhu. Výpočetní část již obstarává vlákno, které kontroluje indikaci stlačení displeje a následně po dobu stlačení neustále naplňuje buffer a zasílá jej na audio výstup. Zde je hlavní rozdíl, že plnění bufferu a zasílání na výstup je řešeno v nativní kódu, tedy jazyce C. Aplikace umí generovat oproti prvnímu návrhu tři základní typy signálu - sinus, pila a obdélník. Také latence je zde o mnoho lépe řešena.

6.1.3 Výsledná architektura

Vývoj aplikace začal rovnou prvním návrhem, neboť to bylo ještě před základním průzkumem trhu. Postupně po zjištění dalších informací bylo doporučeno využít technologii OpenSL ES z druhého návrhu.
Druhý návrh nabízí nižší latenci a pokročilejší manipulaci se zvukem, bohužel však kvůli neustálým problémům se zprovozněním této knihovny a také z nedostatku dokumentace a velké náročnosti se od tohoto návrhu upustilo. Přešlo se tedy zpět k prvnímu návrhu, který byl modifikován o pokročilejší implementaci z druhého návrhu a také optimalizaci, díky které je latence skoro totožná jak s použitím nativního kódu.

Dále byla přidána implementace dvou IIR filtrů - Dolní propust a horní propust, které se aplikují po vygenerování signálu. Hodnotu mezní frekvence je možné libovolně nastavit.

Pro případ přesáhnutí amplitudy kontroluje následný limiter (omezovač). Aplikace také podporuje multi-touch ovládání.



Obrázek 6.1: Výsledný návrh aplikace

6.2 Uživatelské rozhraní

Vzhled aplikace převážně tvoří samotná hrací plocha. Zbytek tvoří rozhraní pro nastavení hrací plochy. Po dotknutí tlačítka nastavení zakryje polovinu hrací plochy nastavení obsahující podrobnější konfiguraci různých parametrů ke hraní, tedy uživatel má stále možnost hrát a přitom měnit konfiguraci celé aplikace.

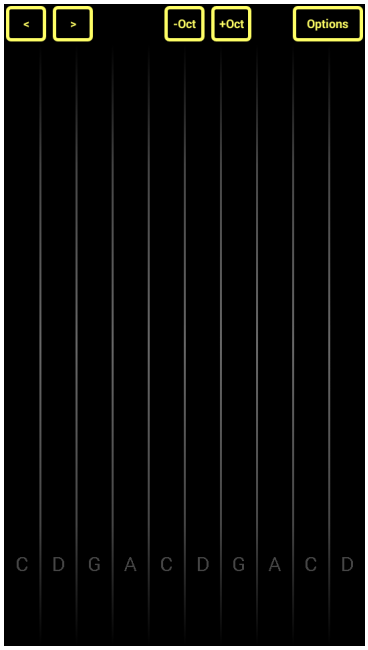
Vzhled je také navržen tak, aby bylo možné jej přetočit i na šířku. Při ukončení aplikace je uživatel dotázán, zda si přeje uložit nastavení a ukončit aplikaci nebo ukončit aplikaci bez uložení.

6.2.1 Horní panel

Horní panel obsahuje pouze jednoduché základní úkony pro ovládání aplikace, tedy posouvání rozsahu klaviatury, buď o zónu nebo o oktávu, a také tlačítko pro podrobnější nastavení aplikace.

6.2.2 Hrací plocha

Hrací plocha je rozdělena na zóny, kde při zvyšování výšky dotyku se aktivuje efekt vibráta. Uživatel může libovolně táhnout po displeji nebo přerušovaně hrát.



Obrázek 6.2: Hrací plocha

6.2.3 Nastavení

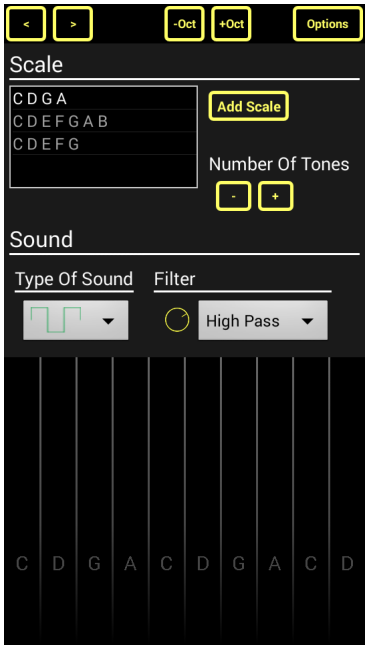
Tato část obsahuje důležité prvky pro ovládání syntetizátoru, je zobrazeno na obrázku 6.3. První má uživatel možnost vybrat stupnici, kterou bude chtít použít. Dále si může nastavit, kolik zón má být zobrazeno na displeji. Vzhledem k neustále rostoucí velikosti displejů mobilních telefonů a využití tabletů je toto velmi užitečná schopnost. V případě, že si bude chtít zobrazit pouze tóny svého výběru, může tak učinit po dotyku na tlačítko přidání stupnice. Dále má možnost vybrat si typ signálu, ze kterého se má zvuk odvíjet a nakonec si může vybrat jeden ze dvou typů filtru a také jeho mezní frekvenci.

Výběr stupnice

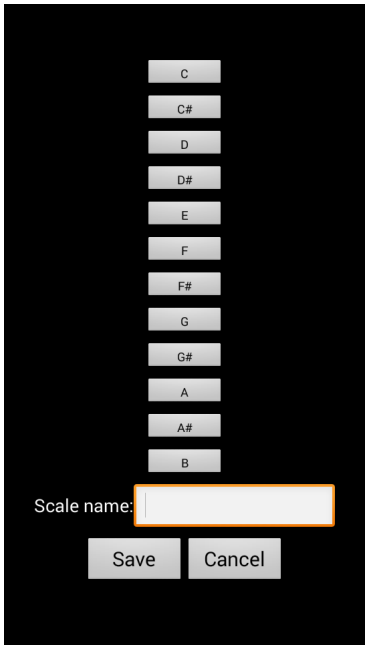
Stupnice je zde brána jako posloupnost tónů, kterou lze nakonfigurovat. Je tedy možná selekce libovolných tónů, které uživatel chce hrát. Není to tedy vázáno na durové nebo mollové stupnice.

6.3 Přidání stupnice

Po stisknutí tlačítka přidání stupnice aplikace uživatele přesměruje na nové okno, kde mu nabízí všechny dostupné tóny. Přidání nové stupnice je vidět na obrázku 6.4. Po vybrání tónu se tlačítko vybraného tónu viditelně zablokuje pro indikaci, že je vybrán. Zvolenou stupnici lze také pojmenovat, například jménem písně, pokud toto pole bude prázdné, pojmenuje se jako seznam vybraných tónů. To lze vidět na obrázku 6.3.



Obrázek 6.3: Uživatelské rozhraní pro nastavení syntetizátoru



Obrázek 6.4: Uživatelské rozhraní pro přidání stupnice

Kapitola 7

Implementace

Tato kapitola popisuje konkrétní implementaci již zmíněného návrhu. Je zde také zmíněno jaké nástroje byly pro vývoj této práce použity.

7.1 Implementační prostředí

Aplikace je vyvíjena v jazyce Java, přesněji Android Java ve vývojovém prostředí Android Studio, což je pracovní prostředí vytvořené přesně pro vývoj Android aplikací. Aplikace byla vyvíjena na chytrém telefonu Samsung Galaxy S4 Mini ve verzi Androidu 4.4.2 (KitKat), avšak aplikace je podporována již od verze Androidu 2.3 (GingerBread).

Poznámka: K datu 4. 4. 2016 byl proveden průzkum verzí operačního systému a podíl nejnižší verze 2.2 na trhu je 0.1% [10].

7.2 Start aplikace

Při startu aplikace se provádí inicializace proměnných a to z nastavení aplikace, které je uloženo v databázi, nebo se nastaví defaultní. Dle zvolené stupnice se nahrají z databáze jednotlivé frekvence tónů. V databázi jsou uložené 4 oktávy, od C2 až po C7 (malé C až pětičárkované C). Poté následuje vytvoření instance knihovny AudioTrack a spuštění asynchronního vlákna AsyncTask.

Asynchronní vlákno kontroluje, zda byl stlačen displej a poté následuje samotné generování zvuku do bufferu. Dále se na samotný buffer aplikuje filtr a následně se výsledek zapíše na audio výstup. Jelikož samotný vygenerovaný signál je v rozsahu $< -1; 1 >$, tak se ještě před zápisem musí hodnoty vynásobit maximální hodnotou typu `short`.

7.3 Generování signálu

Samotné generování signálu ovládá asynchronní vlákno AsyncTask, které je v nekonečné smyčce a neustále kontroluje, zda je stlačena zóna, aby se zvuk začal generovat. Odchytávání dotykových událostí je realizováno přes funkci `OnTouch()`, zde se také počítá frekvence stlačeného tónu.
Vzorec pro výpočet indexu stlačeného tónu (podle indexu se pak určí frekvence)

$$\text{Index} = \text{Hodnota počátečního tónu} + \frac{\text{Osa X}}{\frac{\text{Šířka displeje}}{\text{Rozsah tónů}}} \tag{7.1}$$

kde hodnota počátečního tónu znamená index prvního zobrazeného tónu, osa X je pozice prstu na X-ové souřadnici.
Počítá se frekvence pouze posledního stlačeného tónu, to uživateli umožňuje plynule střídat vícero tónů.

Generování je na bázi neustálého naplňování bufferu, u kterého jsem velikost zvolil 256, je důležité, aby v momentě opuštění displeje zvuk nedoznával a zároveň, aby se buffer stíhal naplňovat. Proto celý algoritmus vygenerování hodnot musí být co nejlépe optimalizovaný. Výpočty, které jsou konstantní je třeba dát mimo smyčku, aby se zbytečně znovu neprováděly.

```
double phase = 0;
...
double phaseIncrement = (2 * Math.PI) / 44100;
for (int i = 0; i < bufferSize; i++) {
    if (phase < Math.PI)
        floatSamples[i] = 1;
    else
        floatSamples[i] = -1;
    phase += phaseIncrement * freqOfTone;
    if (phase > 2 * Math.PI)
        phase -= 2 * Math.PI;
}
```

Kód 7.1: Implementace generování obdélníkového signálu

Zde je vidět, že fáze je částečně předpočítána dle vzorce ze sekce o generování (sekce 4.4.1), poté se pouze násobí frekvencí aktuálně přehrávaného tónu. Samozřejmě by to šlo ještě více zefektivnit přidáním frekvence již při inicializaci, ale to by zpozdilo reakci na vstup a každý buffer by znamenal samotný tón, což není správný postup.

Další optimalizací by mohla být vyhledávací tabulka obsahující předpočítané vzorky pro každý tón, ale v tomto případě není nutná. Byla by vhodná v nahrazení například funkce sinus, která je relativně pomalá na výpočet.
V ukázce kódu je inicializace fáze na hodnotu 0, toto se však provádí pouze při začátku nového generování, aby další buffer začínal od toho bodu, kde minulý buffer skončil.

7.4 Filtr

Po vygenerování signálu následuje aplikace filtru. Samotná třída filtru obsahuje dvě důležité metody. První počítá koeficienty A a B, a druhá již zpracovává buffer a operuje s vypočítanými koeficienty.
Výpočet koeficientů je pro každý filtr odlišný, výpočet pro koeficienty dolní propusti je v sekci o filtrování 4.8. Koeficienty se přepočítávají při každé změně mezní frekvence. Samotné zpracování bufferu je dle rovnice 4.7.

7.5 Limiter

Limiter je komponenta, která kontroluje hlasitost signálu a pokud překračuje danou mez, tak je signál zeslaben [14]. Projde tedy celý buffer a zaznamenává maximální naměřenou hodnotu, pokud překročí povolenou mez, tak je celý signál zeslaben (vynásoben) následující hodnotou

$$\frac{\text{Mezní hodnota}}{\text{Maximum nameřené hodnoty}} \tag{7.2}$$

kde mezní hodnota je v našem případě 1.

Kapitola 8

Testování

Tato kapitola popisuje jak probíhalo testování aplikace na uživatelích a následně shrnuje výsledky tohoto testování.

8.1 Uživatelské testování

Jelikož je nutné více pohledů na aplikaci, proběhlo testování na uživatelích. Testování probíhalo tím způsobem, že jim byla zaslána aplikace, kterou si měli nainstalovat na svůj chytrý telefon nebo tablet. Následně jim byly pokládány otázky ohledně aplikace. Před začátkem testování žádný z uživatelů nedostal bližší informace o aplikaci krom faktu, že se jedná o hudební aplikaci.

Byly kladeny následující otázky

- Jak dlouho vám trvalo se v aplikaci zorientovat?
- Jak hodnotíte uživatelské rozhraní a jeho přehlednost?
- Myslíte si, že by se tento nástroj s použitím dalších efektů mohl použít například na koncertě?
- Jak byste nástroj vylepšili?

8.1.1 Odpovědi jednotlivých uživatelů

Celkově se k testování přihlásilo 7 lidí, zde jsou vybrány nejlepší 4 odpovědi.

Jaroslav Hýbner, hudebník

Zorientování v aplikaci trvalo méně než minutu.
Přehlednost 10/10, Prostředí 6/10.
Pro: rozsáhlá aktivní plocha, nástroje a nastavení ukrytá v menu, mimo nebezpečí nechtěného zmáčknutí. Šikovná možnost nastavení rozsahu a klíče (i když zatím jen jednoho).
Proti: Maličký potenciometr filtru (ale dobře použitelný). Jen jeden nastavitelný parametr. Použití na veřejnosti si představit dokážu, ale mimo koncertní podmínky. Nabízí asi tolik zábavy jako Korg Monotron.
Doporučení na vylepšení: Chce to využít osu Y pro změnu zvoleného parametru. Jeden efekt by úplně stačil. Jako řadový XY midi kontroler. Chválím možnost otočení aplikace na

šířku i na výšku a kontrastní, tmavé prostředí. Jiní by sáhli po nepraktické bílé, ve snaze napodobit klaviaturu. Je to slibný projekt do budoucna s velkým prostorem pro nové možnosti a efekty. V současné verzi je jeho největší výhodou přívětivé a praktické uživatelské rozhraní.

Václav Slanina, teoretik

V aplikaci jsem se zorientoval velice rychle, je jednoduchá, takže bych neváhal dobu odhadnout na jednotky sekund.
Rozhraní je jednoduše řešeno, zdá se mi dostatečně přehledným.
Použít jako nástroj mixování efektů na koncertu určitě dá.
Určitě bych přidal možnost vytváření vlastních patternů možná i možnost jejich exportu.
Možná by bylo i užitečné přidat možnost syntézy vlastního spektra signálu.

Anonym, hudebník

Orientace v aplikaci je v celku jednoduchá, a proto jsem se zorientovala během několika minut, asi 3 minut.
Aplikace je přehledná a srozumitelná, lehce pochopitelná. Uživatel má dobré uživatelské rozhraní a má dostatečné množství funkcí.
Určitě by bylo možné jej použít na koncertě, například s každým tónem by se mohla „pojit“ určitá barva.
Jako vylepšení bych doporučila přidání dalších tónů různých nástrojů nebo utváření akordů.

Vojtěch Tomanec, hudebník

Orientace trvala se všemi funkcemi zhruba 5 minut.
Uživatelské rozhraní je jednoduché a přehledné. Ovládání oktáv a posun tónů v dobrém dosahu.
Při vytváření nové stupnice mě to ale nepustí dál než k výběru jednotlivých tónů. Můžu je maximálně vybrat, ale už tam nevidím žádné jiné tlačítko na uložení nebo vrácení zpět. V případě stlačení fyzického tlačítka zpět se zavře celou aplikace. V případě vybrání jiné stupnice se aplikace také nečekaně ukončí.
Líbí se mi rozvržení tlačítek a na koncert by to určitě použitelné bylo.
Jako doporučení na vylepšení bych uvítal automaticky generátor melodie i nějaký rytmický doprovod. Dále přidání nahrávání do smyčky.

8.1.2 Vyhodnocení odpovědí uživatelů

Orientace v aplikaci se hodnotila kladně, maximum doby je 5 minut, což je velmi uspokojivý čas na seznámení s tímto druhem aplikací. Přehlednost uživatelského rozhraní byla hodnocena jako velmi dobrá. Je to také díky menšímu množství editovatelných parametrů a velké ploše pro hru samotnou. Samotné uživatelské rozhraní však má své mouchy, zde by pomohla výpomoc designéra, který by aplikaci graficky zpříjemnil. Použitelnost aplikace byla vyhodnocena jako velmi dobrá.

Během testování se u jednoho uživatele vyskytly problémy s funkčností aplikace. Tyto problémy se již nadále neprojevily a po přezkoumání zmíněných problémů to zřejmě ani nebyla chyba aplikace.

Návrhy na vylepšení

Využití osy Y při hře - S touto možností se počítalo již od začátku vývoje, osa Y měla aktivovat efekt vibráta, bohužel z nedostatku času tato vlastnost nebyla implementována.

Možnost vytvoření nového tvaru signálu - Tato možnost je již nějakou dobu v plánech na budoucí vývoj.

Přidání možnosti hraní více tónů zaráz - Určitě bude zakomponováno do budoucího vývoje

Automatický generátor melodie - Taková vlastnost je dle mého názoru spíše zbytečná pro tento typ nástroje

Nahrávání smyčky - Zajímavý návrh, tuto možnost již obsahuje Saucillator a určitě je to návrh, který by se dal v budoucnu zvážit.

Kapitola 9

Závěr

Cílem této práce bylo navrhnout a implementovat syntetizátor na mobilní platformě Android s použitím filtrů a efektů. Aplikace obsahuje přehledné rozhraní, které disponuje ovládacími prvky pro úpravu hry, jako počet zobrazených tónů, selekce vlastních tónů a standardní posun o tón nebo o oktávu. Syntetizátor zvládá generovat tři typy signálu - sinus, pila a obdélník. K dispozici jsou dva filtry - dolní propust a horní propust. Tyto filtry je možné ovládat pomocí knobu, který mění hodnotu mezní frekvence. Jako efekt byl použit Limiter pro případné výkyvy amplitudy signálu.

Vývoj probíhal již od začátku září, za tu dobu se vystřídaly 2 návrhy, ze kterých nakonec vyšel finální. Druhému návrhu jsem věnoval nejvíce času, kdy byly neustálé problémy s knihovnou OpenSL ES a z toho důvodu jsem se nakonec vrátil zpět k prvnímu jednoduššímu návrhu. Bohužel jsem však nestihl implementovat funkci vibráta.

Aplikace byla také podrobena testu na uživatelích pro získání zpětné vazby a případných návrhů na vylepšení, kde aplikace dopadla velmi pozitivně.

9.1 Budoucí možnosti vývoje aplikace

- V první řadě by se měl dopracovat efekt vibráta, který byl naplánovaný v původním návrhu aplikace.
- Možnost spojité změny tónu, takzvaný glissando efekt.
- Přepracování zvukového rozhraní na oscilátory, tím by se umožnilo zahrát více tónů v jednom čase. Následně přidání základních vlastností zvuku jako attack a release.
- Pro lepší využití tohoto nástroje by se mohlo implementovat více efektů, například zkreslení.
- Možnost vytvoření vlastního tvaru signálu - Tento nápad je velmi zajímavý, tvar signálu by se mohl přímo nakreslit, poté by se musel zapsat do vyhledávací tabulky z důvodu velké náročnosti na vygenerování. A následně by se mohla uživateli zpřístupnit možnost signál přehrát a dále jej vyexportovat.
- Možnost změny barev uživatelského rozhraní, například mít dosavadní motiv a světlý motiv.

Literatura

[1] Google: *Google Play*. [Online; navštíveno 10.5.2016].
URL https://play.google.com/

[2] Kolektiv autorů: *Android Internals: Fragment of a course detailing the architecture of Android and interaction of its components*. [Online; navštíveno 11.5.2016].
URL http://technologeeks.com/Courses/Android-Excerpt.pdf

[3] Kolektiv autorů: *Android NDK*. [Online; navštíveno 9.5.2016].
URL http://developer.android.com/ndk/guides/index.html

[4] Kolektiv autorů: *Android: Processes and Threads*. [Online; navštíveno 11.5.2016].
URL http://developer.android.com/guide/components/processes-and-threads.html

[5] Kolektiv autorů: *AudioTrack API*. [Online; navštíveno 11.5.2016].
URL
http://developer.android.com/reference/android/media/AudioTrack.html

[6] Kolektiv autorů: *MIDI Association*. [Online; navštíveno 10.5.2016].
URL https://www.midi.org/

[7] Kolektiv autorů: *Music and Computers*. [Online; navštíveno 9.5.2016].
URL http://music.columbia.edu/cmc/MusicAndComputers

[8] Kolektiv autorů: *Open Handset Alliance*. [Online; navštíveno 10.5.2016].
URL http://www.openhandsetalliance.com/

[9] Kolektiv autorů: *OpenSL ES - The Standard for Embedded Audio Acceleration*. [Online; navštíveno 9.5.2016].
URL https://www.khronos.org/opensles/

[10] Kolektiv autorů: *Průzkum trhu na platformě Android*. [Online; navštíveno 10.5.2016].
URL http://developer.android.com/about/dashboards/index.html

[11] KORG: *Korg PA Arrangers*. [Online; navštíveno 10.5.2016].
URL http://www.korg.com/us/products/synthesizers/#subcate-73

[12] Miranda, E. R.: *Computer Sound Design: Synthesis Techniques and Programming*. Focal Press, 2002, ISBN 978-0240516936.

[13] Thede, L.: *Practical Analog and Digital Filter Design*. Artech House, 2004, ISBN 978-1580539159.

[14] Zölzer, U.: *DAFX: Digital Audio Effects*. John Wiley & Sons, Ltd, 2002, ISBN 0-471-49078-4.

Přílohy

Příloha A

Obsah CD

Na CD se nachází adresáře:

- /src - zdrojové kódy aplikace
- /bin - spustitelná aplikace ve formátu APK
- /pdf - text této práce ve formátu pdf
- /text - text této práce ve zdrojovém formátu latexu
- /vid - krátké video prezentující tuto práci

Příloha B

Manuál

Spustitelná aplikace je umístěna na přiloženém CD. Po nainstalování se zobrazí hrací plocha aplikace, kde si již uživatel může libovolně hrát. Na horním panelu se nacházejí tlačítka pro posunutí rozsahu tónů buď o jeden tón nebo oktávu. Po stisknutí tlačítka nastavení polovinu hrací plochy zakryje nabídka s konfigurací aplikace. Zde má uživatel možnost vybrat jinou stupnici, rozšířit si rozsah tónů nebo stupnici přidat. Také má možnost výběru jaký zvuk se má generovat a typ filtru a jeho ořezávací frekvenci pomocí otočného knobu.