TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Infotehnoloogia eriala

Andres Nirk

Adaptiivne Metronoom Androidile

Bakalaureusetöö (6 EAP)

Juhendaja: M. Niitsoo

Autor: …… „ ……“ mai 2013
Juhendaja: …… „ ……“ mai 2013

Lubada kaitsmisele
Professor: …… „ ……“ mai 2013

TARTU 2013

Sisukord

Sissejuhatus

Metronoom on füüsiline või digitaalne seade, mis aitab muusikutel tempot ühtlasena hoida. Tema tööpõhimõte seisneb selles, et muusik kuuleb heli (tavaliselt lühikest ning „teravat“, edaspidi klikk) mingi fikseeritud perioodi tagant, mida saab ise seadistada. Kahjuks aga segab metronoom mõnikord muusikut ennast, seda eriti algajate muusikute puhul. Selle vältimiseks on autor teinud digitaalse metronoomi rakenduse, mis kuulab, kuidas muusik instrumenti mängib. Kui muusik mängib korralikult tempos, siis metronoom on vait, et mitte muusikut segada. Kui aga muusik hakkab tempost ette minema või maha jääma, siis hakkab metronoom kaasa tiksuma. Muidugi tuleb arvestada, et keegi ei suuda mängida täpselt löögi peale, mistõttu tuleb määrata mingi vahemik, mille piires võib muusik eksida enne kui metronoom tiksuma hakkab. Lisaks sellele on võimalus ka valida selline režiim, kus metronoom läheb mängija tempoga kaasa kui ta väga mööda hakkab mängima. Seda režiimi on tarvis nendel, kes alles alustavad metronoomiga mängimist, kuna neid segab metronoom rohkem. See režiim on seega metronoomiga harjumiseks. Hetkel on antud rakendus mõeldud kasutamiseks trummidega.

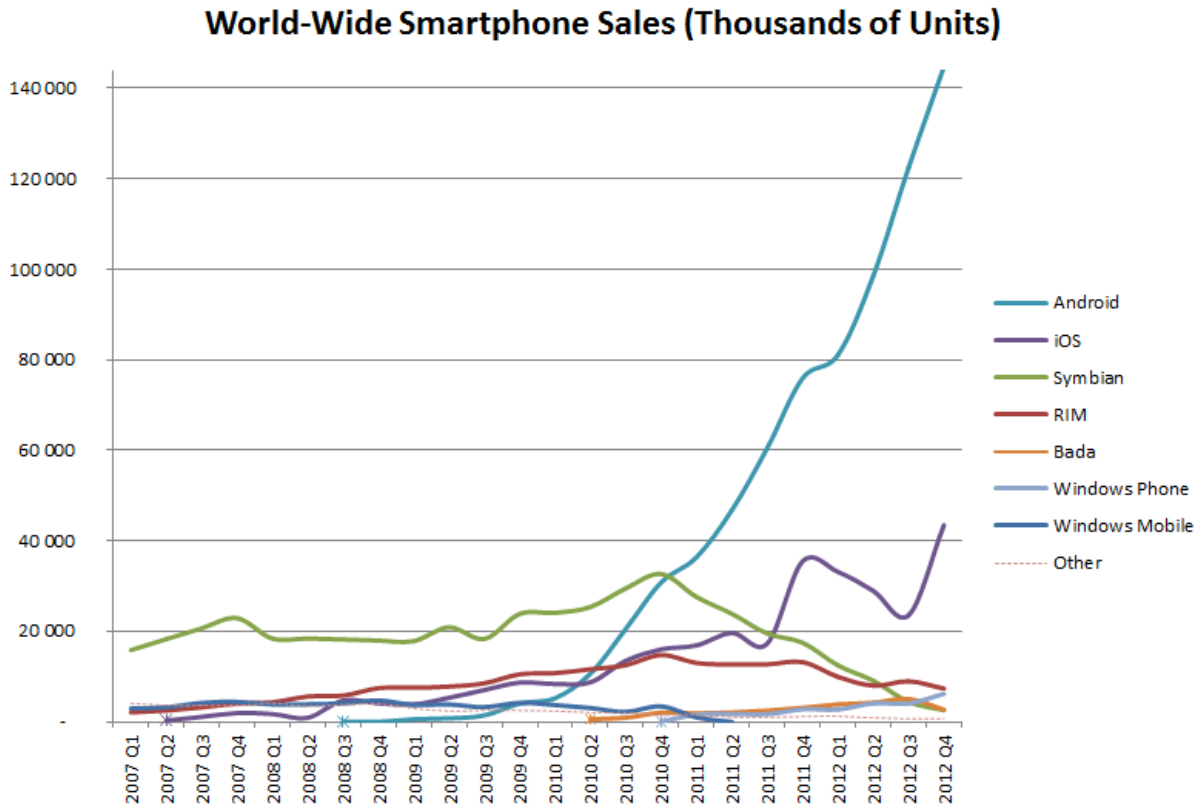
Nutiseadmete osakaal mobiilseadmete turul on kasvamas üha suuremaks ning seejuures on praegu kindel turuliider operatsioonisüsteemide vallas Android. Seejuures on suurenemas ka rakenduste arv ning kuigi Androidile mõeldud rakenduste pood Google Play loodi palju hiljem, kui Apple’i App Store, kus on praegu ligi 800000 rakendust, siis nüüdseks on Android jõudnud rakenduste arvu poolest järele, kuna Google Play’s on samuti ligi 800000 rakendust[1]. Kuna kasutajaid on Android platvormil tekkimas järjest rohkem, siis on ka käesolev rakendus tehtud just Android platvormile. Samas aga on ka muusikaga seotud rakendusi turul väga palju ning seetõttu tuleb luua midagi, mida ei oleks veel tehtud. Seetõttu ongi valitud teemaks adaptiivse metronoomi rakendus, luues algajatele muusikutele võimaluse metronoomiga harjumiseks.

1 Probleemi püstitus

Kuna Android platvormil puudub rakendus, mis aitaks algajal muusikul metronoomiga harjuda, siis on autori eesmärk luua selline rakendus. Et aga rakenduse või ka mõne muu tarkvara tegemine võimalik oleks, on tarvis teada, milline see rakendus peab täpselt olema ning milliseid tehnoloogiaid selleks on võimalik kasutada. Järgneva dokumendi eesmärk ongi selgitada, kuidas antud rakendus peaks töötama.

1.1 Platvormi valik

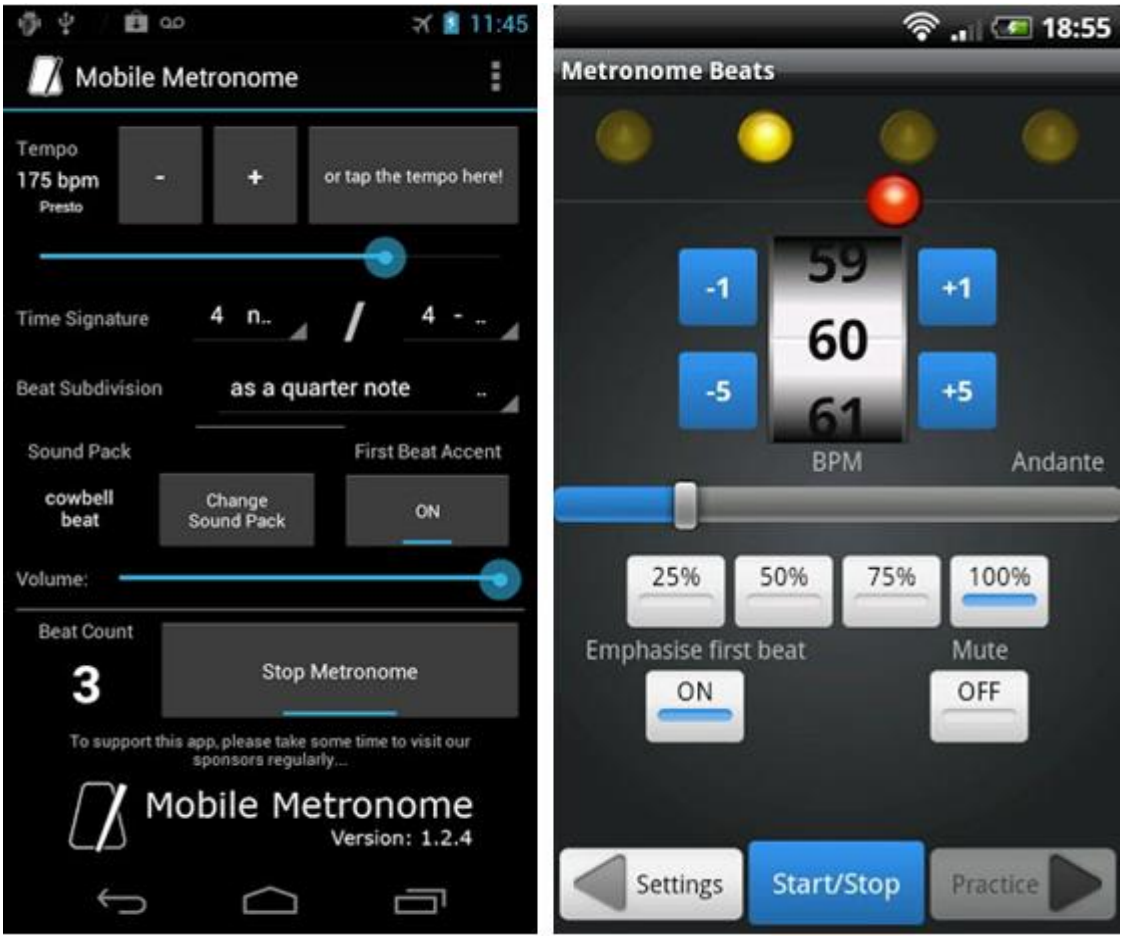
Rakenduse platvormiks on valitud Android, kuna Android on hetkel kõige populaarsem nutiseadmete operatsioonisüsteem (joonis 1) ning nutiseadmeid kasutatakse järjest rohkem. Rakenduse programmeerimiseks on kasutatud Androidi arendajatele mõeldud tööriistade komplekti SDK (*Software Development Kit*) ning seda kasutades tuleb enamus rakendus kirjutada Javas. Tuleb arvestada, et kuigi nutiseadmed saavad järjest rohkem arvutusvõimsust, on tegu selle rakenduse näol siiski reaalajasüsteemiga, saades korraga nii sisendit (instrumendi heli), kui ka andes sisendi põhjal väljundit (metronoomi heli). Kuigi antud versioonis on kasutatud ainult SDK-d, siis tulevastes versioonides võib optimeerimise mõttes vajalikuks osutuda Androidi NDK (*Native Development Kit*), mis laseb programmeerijal saada ligi Androidi osadele, millele SDK-ga ligipääs puudub. NDK abil loodud rakendustes on kasutada programmeerimiskeelena lisaks Javale ka C ja C++. Javas tuleb seejuures kindlasti osa programmeerida, C ja C++ aga ei tule tingimata kasutada.



Joonis 1. Nutitelefonide müük operatsioonisüsteemide järgi [4]

1.2 Sarnased realisatsioonid

Google Play's on kaks kõige populaarsemat metronoomi rakendust Mobile Metronome ja Metronome Beats (joonis 2). Mõlemad on küllalt lihtsad ning standardsed metronoomi lahendused. Nendel rakendustel on olemas kõik funktsioonid, mis käesoleval rakenduselgi, välja arvatud tempo tuvastus. Lisaks on olemas ka mitmeid rakendusi, mis jälgivad loo tempot ning isegi üks selline rakendus (Sandberg Sound'i poolt tehtud Metronome & Drum Machine [5]), mis ühendab endas metronoomi ning tempotuvastuse, kuid selle tempotuvastuse mõte on algse tempo saamine kasutajalt, et too ei peaks kasutama tempo koputamise funktsiooni. Käesolev rakendus aga ühendab metronoomi ja tempo tuvastuse rohkem, kuna annab kogu mängimise ajal kasutajale tagasisidet tempo kohta (kas klikk mängib või ei) ning isegi parandab tempot mängimise ajal algaja režiimis. Seega on tegelikult nii metronoomi osa kui ka tempo tuvastuse osa teistes rakendustes realiseeritud, kuid sellises kombinatsioonis ei ole neid veel kokku pandud.



Joonis 2. Rakendused Mobile Metronome [6] ning Metronome Beats [7]

2 Rakenduse ülesehitus

2.1 Metronoomi põhifunktsioonid

- Tempo määramine – Kasutaja saab määrata tempo, kasutades kas liugurit, + ja – nuppe, „Koputa tempo“ nuppu või „Kuula tempo“ nuppu.
- Taktimõõdu määramine – Kasutaja saab määrata taktimõõdu, mille ülemise osa järgi mängitakse (kasutaja eelistusel) aktsenti ning alumise osa järgi muudetakse löögipikkust.
- Heli valjuse määramine – Kasutaja saab määrata liuguri abil heli tagasi mängimise valjuse.
- Laulu seadete salvestamine – Kasutaja saab salvestada antud loo seaded ning neid lugude seadeid saab hiljem lihtsalt taastada ning seadete listidesse lisada.
- Seadete listide koostamine – Kasutaja saab salvestada ühte listi erinevad salvestatud lood, mida saab siis järjest valida.

2.2 Režiimid

Rakendusel on kolm režiimi: tavarežiim, algaja režiim ning treeningurežiim. Tavarežiim töötab nagu tavaline metronoom, kus saab valida tempo, taktimõõdu, aktsendi ning helitugevuse. Selles režiimis on olemas ka tempo tuvastuseks mõeldud „Koputa tempo“ funktsioon, mille abil saab nupul korduvalt vajutades määrata tempo. Erinevalt aga enamus olemasolevatest metronoomidest on võimalik aga ka määrata tempo, vajutades nuppu „Kuula tempo“, mille peale pannakse käima löögituvastus ning seejärel mängida järjestikku lööke, mille järgi tempo määratakse. Lisaks sellele on selles režiimis võimalik ka salvestada metronoomi seadeid, et neid pärast lihtsam leida oleks ning ka kuvada ning laadida salvestatud seadeid. Selles režiimis peab tempo jääma vahemikku 40 kuni 240 lööki minutis.

Algaja režiimi põhiline erinevus seisneb selles, et selles režiimis pannakse tempo tuvastamine käima nupule vajutusega ning metronoom hakkab ise mängima, kui on tuvastatud algne tempo. Algse tempo tuvastamiseks on vaja mängida vähemalt neli ühtlase tempoga lööki. Tulenevalt ajalistest piirangutest on selle režiimi tempo vahemikuks 40 kuni 130 lööki minutis. Lisaks ei ole selles režiimis vaja metronoomi seisma panna. Nimelt saab määrata kasutaja aja, mille möödudes kui pole tuvastatud ühtegi lööki, siis pannakse metronoom seisma. Salvestamine aga jätkub ning kui on antud uus algne tempo, siis pannakse ka

metronoom jälle käima. See funktsioon on loodud kasutaja mugavust silmas pidades. Nimelt kui kasutajal läheb sassi või soovib ta uut lugu mängima hakata, ei pea ta seadet puutuma vaid ootab kuni määratud aeg möödub. Selles režiimis ei ole võimalik ka seadeid salvestada, kuna tempo on pidevas muutumises.

Treeningurežiim on peaaegu nagu tavarežiim, selle erinevusega, et koos metronoomiga pannakse tööle ka löögituvastus ning vaadatakse, kas mängija mängib tempost mööda või püsib tempos. Kui kasutaja paneb metronoomi käima hakkab see mängima, 5 sekundi möödumisel aga vaigistatakse metronoom eeldusel, et mängija suudab tempos püsida. Kui mängija hakkab tempost mööda mängima, hakkab heli jälle mängima. Selle režiimi mõte on mängija rütmitaju arendamine, kuna sel hetkel kui metronoom vaigistatakse, tuleb mängijal ilma metronoomi tiksuta hakkama saada.

2.3 Seadete listid

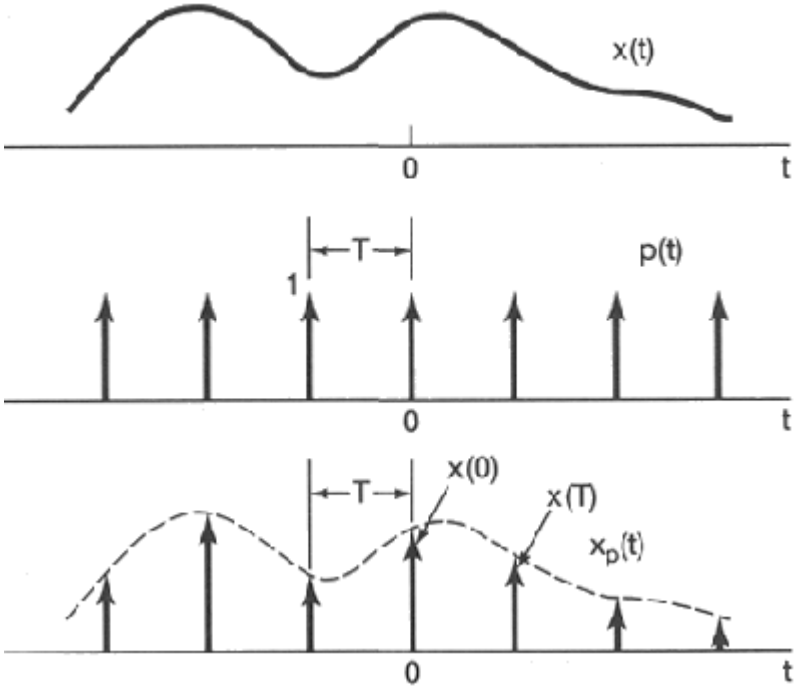
Selleks, et kasutaja ei peaks iga loo alguses panema paika tempo, taktimõõdu ning muud seaded, on neid võimalik ka salvestada. Selleks on kasutatud Androidi rakendustele mõeldud salvestusruumi, mis on mõeldud väikestele andmemahtudele. Suuremate mahtude korral tuleks kasutada kas faili salvestamist või andmebaasi. Seega on võimalik salvestada metronoomi seadeid mingi kindla laulu jaoks, kuid muusikud, kes mängivad proovis või esinemisel, soovivad tihti, et neil oleks list, kust oleks lihtsam laule üles leida. Seetõttu ongi lisatud nö. seadete listide (inglise k. *setlist*) salvestamise võimalus, kus igas listis on järjestatud kasutaja salvestatud laulud. Valides mingi listi mängimiseks, tekivad lisaks tavarežiimi nuppudele ka kaks lisanuppu, eelmine ning järgmine, mille abil saab kiiresti liikuda laulude vahel.

3 Heli töötlemine

3.1 Digitaalne heli

Selle peatüki kirjutamisel on kasutatud Dan Lavry koostatud materjale [8].

Kui analoogheli saab vaadata kui pidevat lainet, mis levib õhus tänu õhuosakeste võnkumisele, siis digitaalset heli tuleb vaadelda, kui diskreetset lainet. See tähendab, et digitaalne heli on kujutatud täisarvudena, mis on saadud analoogheli lainest näidiseid ehk sämpleid võttes . Joonisel 3 on kujutatud helilainest sämplite võtmist, kus tähis T vastab sämplimissageduse pöördväärtusele ning x(t) vastab ühe sämpli väärtusele. Need näidised näitavad analoogsignaali amplituudi mingil konkreetsel hetkel.



Joonis 3. Signaalist sämplite võtmine [9]

Iga näidis on aga mingis kindlas vahemikus ning selle vahemiku määrab ära bitisügavus. Näiteks 16 bitiste näidiste korral on võimalike väärtuste arv 2^{16}, ehk selleks skaalaks on -32768 kuni 32768. Mida väiksem on bitisügavus, seda suurem on kvantisatsioonimüra, ehk seda moonutatum on heli. Seetõttu kasutatakse madalama bitisügavusega mõningates väheste ressurssidega seadmetes, kus heli kvaliteet ei ole oluline. Näiteks telefonikõne puhul. Muusika salvestamiseks kasutatakse enamasti 16 bitiseid

sämpleid, kuna 16 biti puhul on kvantisatsioonimüra väike, võrreldes müraga, mida salvestusseadmed ise põhjustavad.

Teine tähtis parameeter on sämplimissagedus, mis näitab mitu korda sekundis sämpleid võetakse ning mida väljendatakse hertsides (Hz). Tavalised sämplimissagedused on 8000 Hz, 44100 Hz ning 96000 Hz. Antud projektis on kasutatud sämplimissagedust 44100 Hz, kuna see on ainuke sagedus, mille puhul on garanteeritud, et heli salvestamine selle sagedusega on võimalik kõikides Android seadmetes. 44100 Hz on standardne ka näiteks CD (*Compact Disk*) plaadile heli kirjutades. 44100 Hz seetõttu, et Nyquist-Shannoni sämplimisteoreemi järgi tuleb helisignaali perfektseks rekonstrueerimiseks kasutada sämplimissagedust, mis on 2 korda suurem, kui algne signaal. Kuna inimene kuuleb sagedusi vahemikus 20 Hz kuni 20000 Hz, siis peabki helisignaali rekonstrueerimiseks salvestama sämplimissagedusega, vähemalt 40000 Hz.

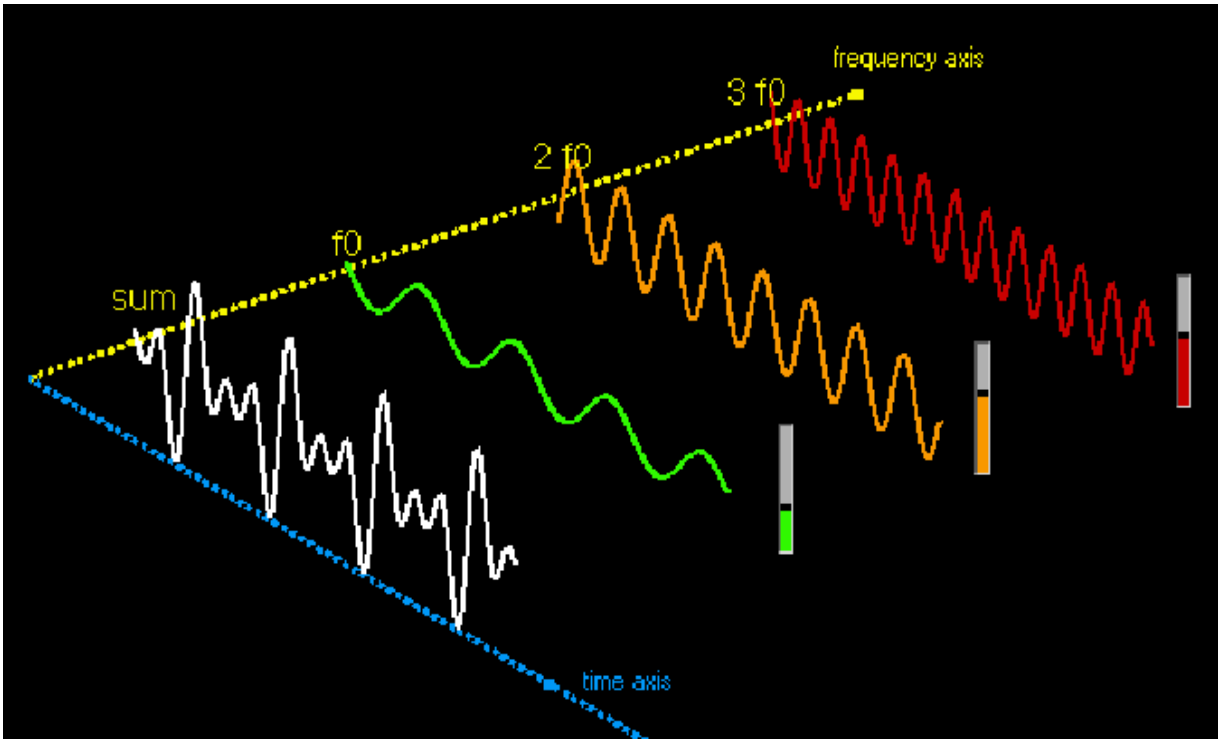
Kolmas, Androidi spetsiifiline parameeter – minimaalne puhvri suurus. Nimelt on pandud paika, kui suur peab sämplite puhver minimaalselt olema, et salvestamine korrektselt toimiks. See jääb aga üldiselt umbes 30 millisekundi juurde.

3.2 Löögituvastus

Löögituvastuse algoritmi idee on saadud Frederic Patin'i uurimustööst [10].

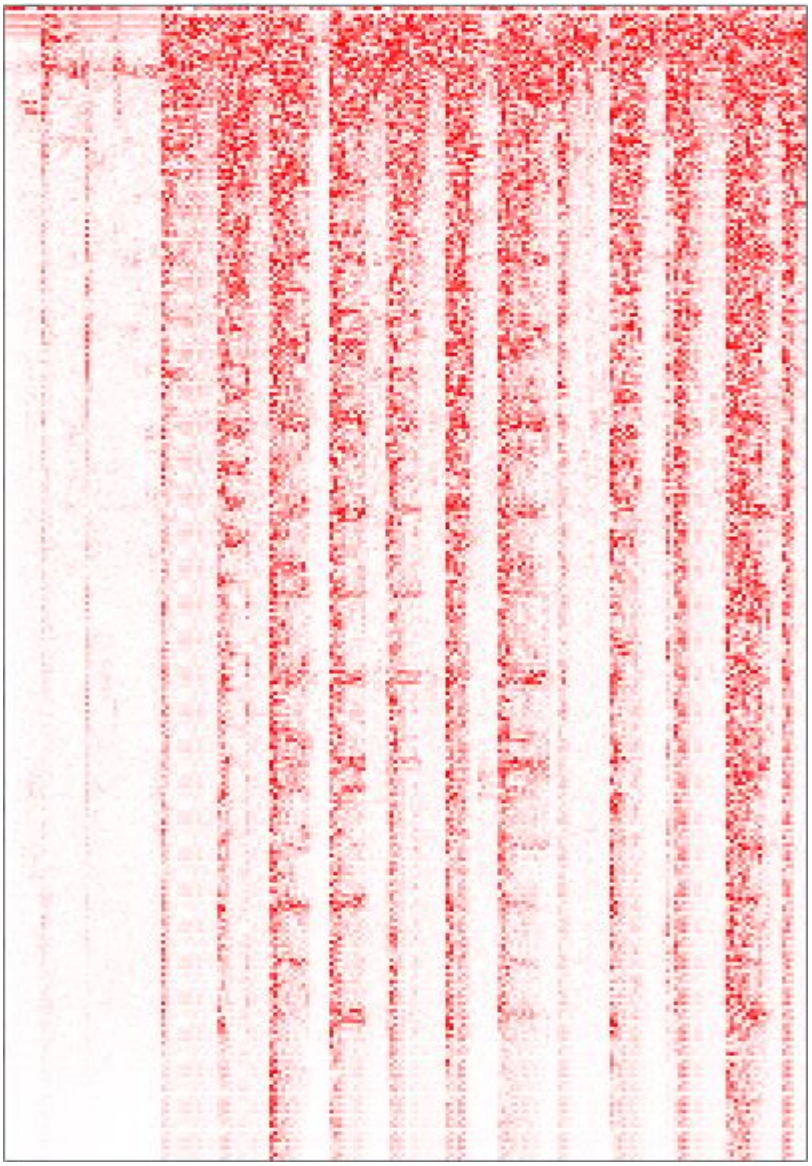
3.2.1 *Fourier*' teisendus

Enne kui saab selgitada konkreetset algoritmi, tuleks teha selgeks, mis on *Fourier'* teisendus [11]. Et digitaalsel kujul ei ole võimalik salvestada pidevat signaali, siis tuleb vaadelda seda kui diskreetset signaali. See tähendab, et signaal moodustub kindlatest üksikväärtustest aegruumis. Sellisel kujul signaali on aga raske analüüsida, kuna võib tekkida olukord, kus müra tõttu mingis sagedusalas jääb märkamata löök mingis teises sagedusalas. Selleks, et seda olukorda vältida, tuleks väärtused teisendada sagedusruumi. Selleks kasutataksegi *Fourier'* teisendust. Joonis 4 selgitab *Fourier'* teisendust, sellel kujutatud kollane telg on sageduse telg, sinine on aja telg ning värvilised lained vastavad igaüks ühe sageduse signaalile, mis saadakse *Fourier'* teisendusest ja valge on nende kompositsioon ehk algne laine.



Joonis 4. Fourier' teisendus [11]

Teisendades signaali sagedusruumi, saab leida iga sagedusvahemiku signaali tugevuse mingil ajaperioodil ning kui see periood on piisavalt lühike, siis saab koostada iga sagedusvahemiku energia muutumise graafiku ehk heli spektraalesituse. Joonisel 5 on horisontaalsel teljel aeg, kus igale pikslile vastab ühe puhvri salvestamiseks kuluv aeg ning vertikaalne telg on sageduse telg, kus igale pikslile vastab üks sagedusvahemik, piksli värv näitab ühe sagedusvahemiku energia väärtust konkreetsel ajavahemikul ning väikseimale väärtusele vastab valge ning suurimale punane. Joonise ülaosas on madalsagedused ning all osas on kõrgsagedused. Kui võtta puhvri suuruseks näiteks 1024 sämplit, siis peale *Fourier'* pööret on meil 1024/2=512 sagedusvahemikku.

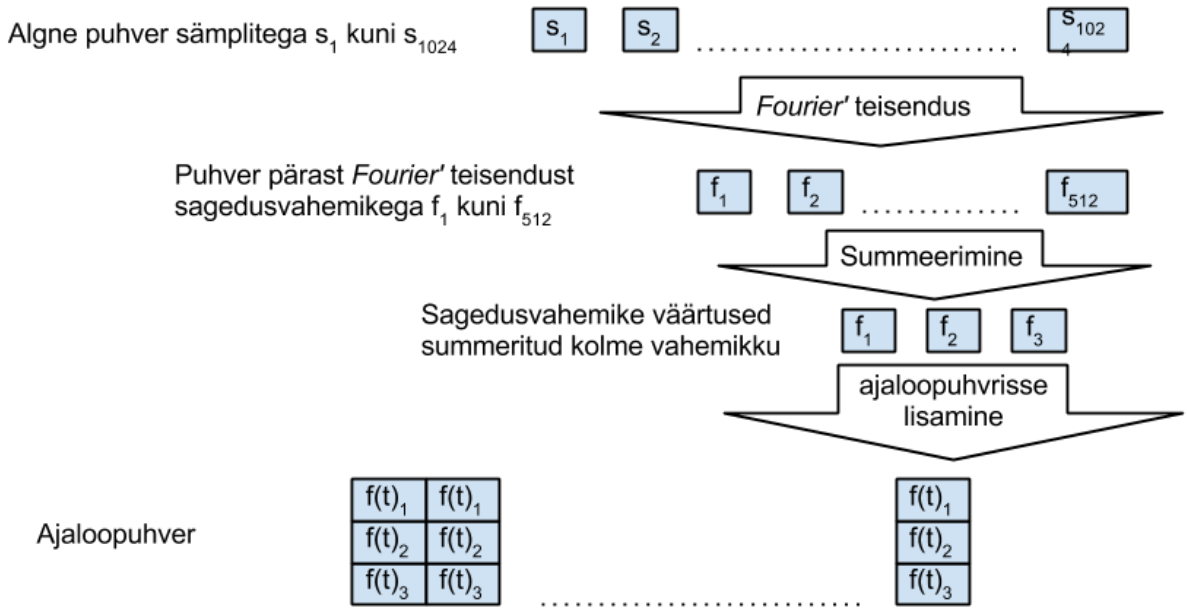


Joonis 5. Heli spektraalesitus.

3.2.2 Löökide leidmine

Iga kord kui saadakse uued 512 sagedusvahemikku, võrreldakse neid eelmisest puhvrist saadud sagedusvahemikega ning leitakse igas sagedusvahemikus toimunud positiivne muutus. Kuna meil ei ole vaja 512 sagedusvahemiku muutu eraldi teada, summeeritakse need nii, et järgi jääb 3 sagedusvahemikku, mis salvestatakse. 3 sagedusvahemikku on selleks, et tuvastada madala sagedusega löögid, nagu näiteks basstrumm, keskmise sagedusega löögid, näiteks soolotrumm, ning kõrgsagedusega löögid, nagu näiteks trummitaldrikud. Summeerimine on tehtud seetõttu, et kui jätta kõik sagedusvahemikud alles, siis on löögi tuvastus tundlikum välisele mürale.

Kui 3 sagedusvahemikku on ühe puhvri kohta olemas, siis salvestatakse see ajaloopuhvrisse. Joonisel 6 on selgitatud, kuidas saadakse algsest signaalist ajaloopuhver. Ajaloopuhver on FIFO tüüpi massiiv, kus säilitatakse ühe sekundi jagu sagedusvahemike puhvreid. Selle põhjal arvutatakse iga sagedusvahemiku keskmine ühe sekundi jooksul ning võrreldakse seda iga uue puhvriga. Kui uus puhver on suurem, kui keskmine, mis on korrutatud konstandiga, siis on leitud löök. Iga kord, kui leitakse löök, fikseeritakse selle puhvri salvestamise ajahetk tempo tuvastajale.



Joonis 6. Ajaloopuhvri saamine

3.3 Tempo tuvastamine

Tempo näitab, mitu lööki minutis mängitakse. Seejuures muusikat kirja pannes on ülimalt oluline jälgida ka taktimõõtu. Takt on muusika kirjutamise ning lugemise lihtsustamiseks loodud aja segment. Taktimõõt koosneb kahest naturaalarvust ning märgitakse X/Y. Ülemine arv (antud juhul X) näitab, mitu lööki on ühes taktis ning alumine näitab, kui pikk on üks löök.

Selleks, et tuvastada, kas inimene mängib tempost mööda või ei, on meil vaja mingit algset tempot. Seejärel saame leida, kui pikk oli löök ning kui pikk ta tegelikult oleks pidanud olema. Et saada algset tempot, tuleb antud rakenduses käivitada metronoom algaja režiimis ning seejärel vajutada start nuppu, mille peale hakkab seade salvestama. Seejärel tuleb anda sama intervalliga 4 lööki. Loomulikult ei suuda mängija neid nelja lööki täpselt mängida ning seetõttu arvestatakse ainult neid lööke, mis ei ületa lubatud viga ning nende löökide pealt võetakse keskmine. Kui saadud tempo on väljas lubatud vahemikust 40 kuni 130 lööki minutis, siis jagatakse seda kahega, seni kuni ta jääb lubatud vahemikku. Pärast nelja lööki pannakse ka metronoomi heli käima.

Kui algne tempo on olemas, leitakse iga uue löögi korral kuueteistkümnendiklöögi pikkus ning otsitakse, mitu kuueteistkümnendiklööki oli vahe eelmisest löögist selle löögini. Tempo järgi saadakse teada, kunas oleks pidanud löök toimuma ning lahutades sellest maha löögi tegeliku toimumise aja, saab kätte mängimisel tehtud vea. Selle vea põhjal arvestatakse uus tempo ning muudetakse mängimise tempot, kui vaja. Kui kasutaja poolt määratud aja möödudes ei ole lööki tuvastatakse, pannakse metronoom seisma ning jäädakse ootama uut loo algust.

3.4 Ajalised piirangud

Kuna puhvri suurus on kindlaks määratud, siis on kindel ka standardviga, millega peab arvestama. Et puhvri suurus on ligikaudu 35 millisekundit ning et kahest kõrvuti asuvast puhvrist ei saa kahte lööki leida, siis maksimaalne tempo, mis me saame leida on löökide intervalliga 70 millisekundit ehk 857 lööki minutis. See on maksimaalne tempo tuvastamaks lööke. Samas aga tuleb arvestada, et tempo tuvastamisel tuleb 35 millisekundile juurde liita inimese eksimus tempo mängimisel. Inimese loomulikuks eksimuseks võib arvestada 20 millisekundit. Seega on kogu eksimus (20+35)*2 = 110 millisekundit, mis annab tempoks 545

lööki minutis. Kuna tempo tuvastamisel kasutatakse 1/16 lööke, siis saame et maksimaalne tempo on 60000/(110*4) = 136 lööki minutis. Seega on seatud maksimaalne tempo, mida tempotuvastusega kasutada saab ligikaudu 130 lööki minutis.

4 Heli mängimine

4.1 Heli formaat

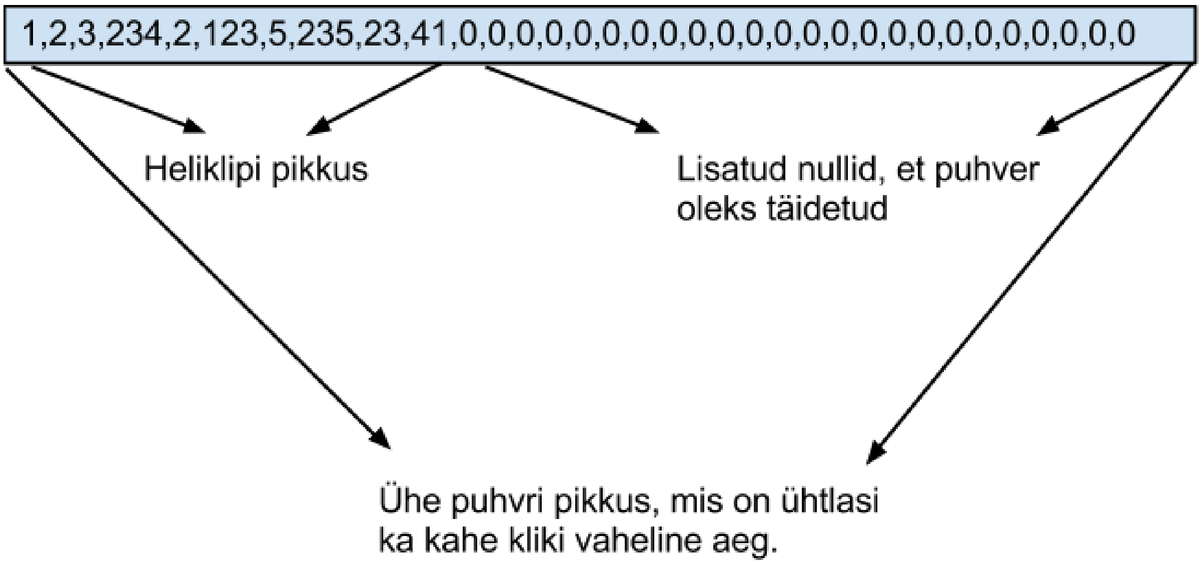
Selleks et mängida ette mingi helifail, peab selle kõigepealt dekodeerima. Dekodeerimine on vajalik, kuna tavaliselt on salvestatakse heli faili kodeerituna, selleks et vähendada faili suurust. Näiteks kasutades 16 bitilisi sämpleid sämplimissagedusega 44100 Hz ning kahte kanalit, tuleb 3 minuti pikkuse helifaili suuruseks 3*60*44100*16*2/8/1024/1024= ~30,3(MB). Seetõttu pakitakse helifaile kokku näiteks MP3 formaati. Android API-s on olemas selline klass nagu, *SoundPool*. Selle klassi abil dekodeeritakse helifail kohe tema sisse lugemisel. Kui fail on dekodeeritud, siis ta salvestatakse PCM (*pulse-code modulation*) andmetena. PCM on pakkimata formaat, mis sisaldabki endas reaalarvulisi väärtusi, mis saadetakse väljundseadmesse. See tagab, et kui antakse käsk klipi mängimiseks, siis on viivitus võimalikult väike. Androidil on ka muid klasse, mille abil heli mängida, nagu näiteks *AudioTrack* ja *MediaPlayer*. Algselt oli rakenduses kasutatud *SoundPool* klassi, kuid järgmises peatükis selgitatud piirangute tõttu tuli see vahetada klassi *AudioTrack* vastu.

4.2 Aja arvestamine

Kuna antud metronoomi tempot peab saama mängimise ajal muuta, siis ei saa kasutada valmis salvestatud metronoomi tiksumist kindlal tempol. Selle asemel tuleb kasutada lühikest heli ning iga löögi ajal see ette mängida. Kuna aga klass *SoundPool* dekodeerib ning salvestab heli PCM andmetena, siis ei võta selle heliklipi ette mängimiseks valmistumine palju aega. Muidugi eeldab see, et hetk millal heli ette tuleb mängida on täpselt välja arvestatud. Selleks oli antud rakenduses algselt kasutatud süsteemi aega. Nimelt pandi iga kord, kui klikk on mängitud, lõim magama, kuid magamise aeg ei saa olla nii pikk nagu peaks olema kahe kliki vaheline paus. See on tingitud asjaolust, et kliki mängimine võtab aega ning vahel võib segada ka Java prügikoristus, mis paneb lõime seisma. Seega tuleb iga kliki ette mängimisel parandada ka võimalikku tekkivat viga ning selleks mõõdetakse süsteemi aega.

Selline lähenemine sobis küll neljandiklöökide heli mängimiseks tempodel kuni 160 lööki minutis, kuid kiirematel tempodel tekitas ta märgatavat hilinemist (kuni 50 millisekundit). See oli tingitud asjaolust, et Androidi ajaarvestus ei ole piisavalt täpne.

Seetõttu ei saa seda lahendust kastutada. Selle asemel on antud rakenduses kasutatud klassi *AudioTrack*, millele saab ise ette anda puhvreid PCM andmetega, mis siis ette mängitakse. Et löökide vahe millisekundites on lihtsalt välja arvutatav ning me teame ka sämplimissagedust, siis saame ka arvutada, kui palju nulle (ehk vaikus) tuleb algsele heliklipile puhvrisse lisada, et klikkide vahe oleks soovitud pikkusega (joonis 7). Et kindla sämplimissageduse hoidmise eest hoolitseb helikaart, võib kindel olla, et heliklippe mängitakse tagasi õige kiirusega ning seega ka, et vaikus klikkide vahe ajal oleks õige pikkusega.



Joonis 7. Kahe kliki vahe pikkune puhver

4.3 Tehnilised piirangud

4.3.1 Lõimed

Antud rakendus koosneb kahest põhilisest osast. Üks on heli mängimine ning teine on heli salvestamine, et jälgida tempot. Kuna mõlema tegumi puhul on vaja kasutada lõpmatut tsüklit, mis alustatakse ja lõpetatakse kasutaja sisendi peale, siis on mõlema tegumi jaoks vaja eraldi lõime. Androidis tehakse vaikimisi kõik toimingud kasutajaliidese lõime peal, kui just ei ole eraldi lõimesid meetoditele eraldatud. Seetõttu ei saa kumbagi toimingut, ei salvestamist ega ka tagasi mängimist, teha vaikimisi lõime peal, kuna see paneks seisma kasutajaliidese. Seega on eraldi heli mängimiseks lõim, mis küll enamus ajast magab, ning heli salvestamise lõim, mis töötab kogu aeg lõpmatus tsüklis.

4.3.2 Java prügikoristus

Nagu eespool ka mainitud, siis on Java üheks oluliseks osaks tema prügikoristus. Java automaatne prügikoristus vastutab mälu halduse eest, et programmeerija selle pärast muretsema ei peaks, nagu see on näiteks keeltes C ning C++. Seega teeb automaatne prügikoristus programmeerija elu lihtsamaks, kuid Androidi platvormil on see tihti ka nuhtluseks. See on tingitud asjaolust, et mobiilsetel seadmetel on mälu palju piiratum, kui PC tüüpi arvutitel ning seetõttu peab automaatne prügikoristus palju tihedamalt programmi tööd segama. Eriti määravaks võib see saada reaalajaliste rakenduste puhul, nagu seda antud rakendus ka on. Seetõttu tuleb jälgida, et asjatult ei loodaks pidevates tsüklites, milleks antud juhul on heli salvestamine ning mängimine, mitte vaja minevaid objekte ning et need, mis luuakse ei sisaldaks kasutut informatsiooni [12].

4.3.3 Sünkroniseerimine

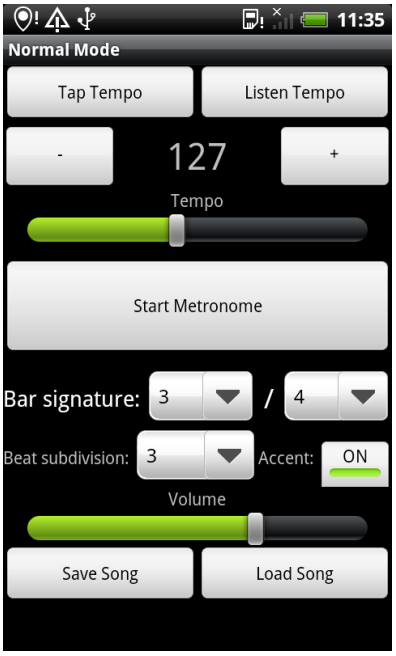
Rakenduses on mitmes kohas ka oluline, et oleks teada, kui suur on ajaline erinevus löögi mängimisest kuni löögi tuvastamiseni, kuna heli töötlus võtab aega. Näiteks algaja režiimis tuleb alguses panna metronoom õigel hetkel käima. See peaks juhtuma pärast nelja esimest lööki, ehk viienda löögi ajal. Antud juhul aga on see küllaltki lihtne, nimelt kuna tempo on teada, ning on teada ka viimase (neljanda) löögi aeg, siis saab viienda löögi täpse aja välja arvutada ning täpselt sel hetkel metronoomi heli mängimist alustada. Teiseks sünkroniseerimist vajavaks kohaks on treeningurežiimis mängimine. Selles režiimis on küll kindel tempo teada, kuid ei ole teada, kui palju võtab aega heli töötlus. Kuna erinevate seadmete peal ei pruugi see aeg kattuda, sest osad protsessorid töötavad kiiremini kui teised, siis ei saa kasutada katse-eksitus meetodil saadud konstantset aega, vaid tuleb see määrata iga seadme peal eraldi. Selleks võiks kasutada metronoomi enda klikke, mis on mängitud ning uuesti salvestatud ning sealt löögi aeg leitud. Kahjuks aga tekitab väljundseadme ümber lülitamine märgatava hilinemise väljundi mängimises, mistõttu ei saa seda kasutada. Seetõttu ongi valitud lahenduseks see, et kasutaja peab mängima metronoomiga kaasa neli lööki ning seejärel võib hakata lugu mängima. Nende nelja löögi seast valitakse välja need, mis kõige täpsemalt vastasid metronoomi tempole ning määratakse neist üks võrdluspunktiks, et edaspidi viga leida.

5 Kasutajaliides

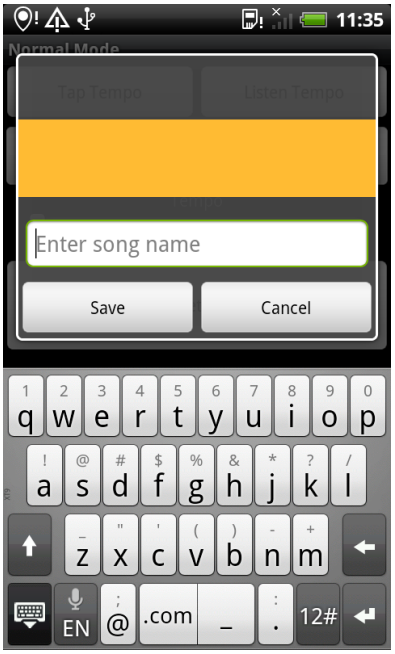
Selles peatükis on kirjeldatud kasutajaliidese disaini. See on jagatud vastavalt rakenduse erinevatele režiimidele.

5.1 Tavarežiim

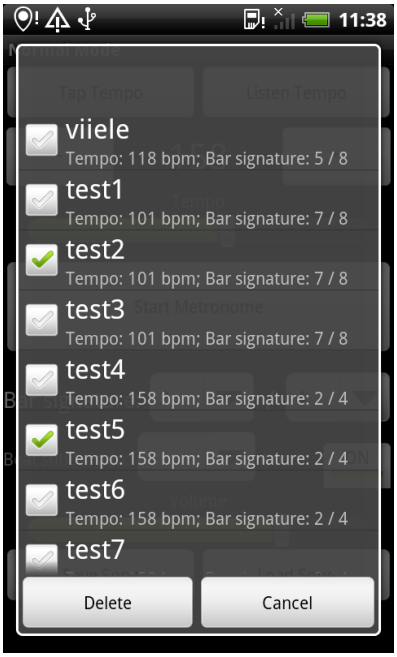
Selle režiimi kasutajaliidest saab näha joonisel 8. Selles režiimis on kasutusel kõik tavapärased metronoomi võimalused. Nimelt on võimalus muuta metronoomi tempot, heli valjust ning taktimõõtu. Lisafunktsioonidena on võimalik vahetada metronoomi "kliki" heli ning salvestada lugude seadeid. Tempo muutmiseks on kasutusel liugur (inglise k. *slider*), kuna nii saab kiiresti leida enam-vähem soovitud tempo. Kuna aga väikeste ekraaniga seadmetel on nii raske saada kätte mingit konkreetset väärtust, siis on lisaks sellele ka võimalik "+" ja "-" nuppudega muuta tempot vastavalt ühe võrra suuremaks ning väiksemaks. Kõige väiksem võimalik tempo on 40 lööki minutis ning kõige suurem on 240 lööki minutis. Heli valjuse jaoks on samuti kasutusel liugur. Kuna aga heli valjuse puhul ei otsi kasutaja mingit kindlat diskreetset väärtust vaid pigem sätitakse see kuulmisele parajalt, siis ei ole heli valjuse luguri juurde lisaks "+" ja "-" nuppe pandud. Taktimõõdu (inglise k. *Bar signature*) määramiseks on kaks menüüd. Esimesest menüüst saab määrata taktimõõdu ülemise komponendi, ehk löökide arvu taktis ning teisest menüüst saab valida ühe löögi pikkuse. Seega alumise numbri abil saame teada, kas löök on neljandiklöök, kaheksandiklöök või hoopis midagi muud. Ülemist numbrit on meil vaja selleks, et juhul kui kasutaja soovib, et esimest lööki rõhutataks, siis me teame, milline on esimene löök. Lisaks on veel takti alguse rõhutamise sisse lülitamiseks loodud nupp (*Accent*) Loo seadete salvestamiseks on eraldi nupp, kuhu vajutades tuleb lahti uus ekraan, kuhu saab sisestada laulu nime (joonis 9). Samuti on loo laadimiseks eraldi nupp, mis avab kõik lood, mille vahel saab siis valida (joonis 10).



Joonis 8 Tavarežiim



Joonis 9. Loo salvestamine



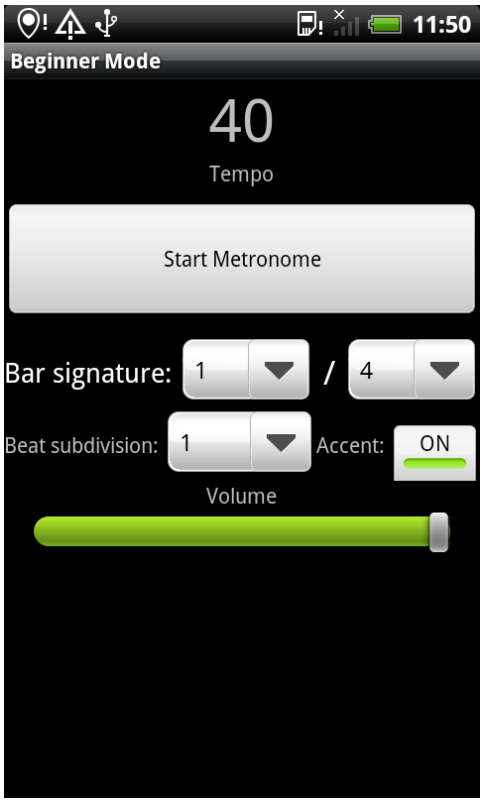
Joonis 10. Loo laadimine

5.2 Algaja režiim

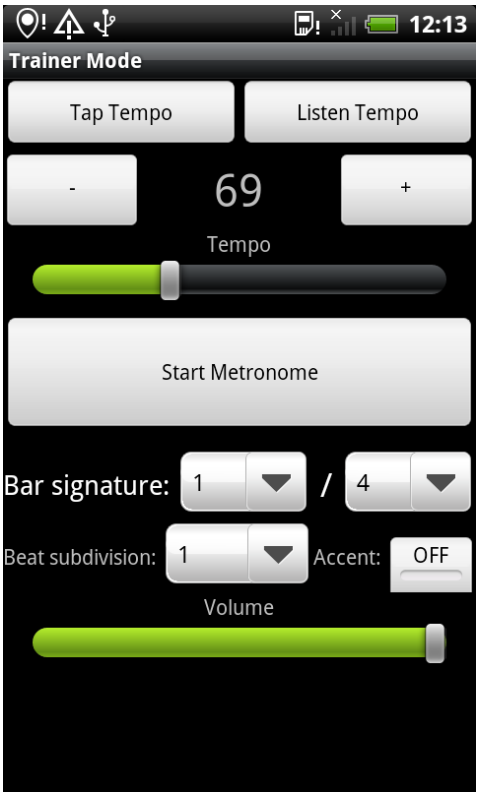
Selle režiimi kasutajaliidest saab nähe joonisel 11. Selles režiimis ei ole midagi võrreldes tavarežiimiga juurde lisatud. Küll aga on mitu elementi kadunud. Nii näiteks on kadunud tempo määramiseks mõeldud liugur ning “+” ja “-” nupud, kuna tempo määratakse mängimise järgi. See režiim on tehtud võimalikult lihtsaks. See tähendab, et kasutaja paneb lihtsalt metronoomi käima ning hakkab mängima. Seejärel tuvastatakse tempo ning rakendus paneb metronoomi mängima. Lisavõimalusena on jäetud alles taktimõõdu ülemise komponendi valimise võimalus, kuid see võimalus muutub aktiivseks üksnes siis, kui on valitud esimese löögi rõhutamine. See eeldab muidugi ka et kasutaja teab, millist taktimõõtu ta kasutab. Lisaks sellele ei ole võimalik selles režiimis ka lugude seadeid salvestada ning seetõttu on ka vastav nupp kadunud.

5.3 Treeningurežiim

Selle režiimi kasutajaliidest saab näha jooniselt 12. See režiim on kahe eelmise vahepealne. See tähendab, et tema jaoks on olemas tavarežiimis kasutusel olevad kasutajaliidese osad, välja arvatud lugude salvestamise ning laadimise nupud. Erinevus on veel see, et ka selles režiimis pannakse koos metronoomi klikiga käima ka salvestamine ning jälgitakse tempot.



Joonis 11. Algaja režiim

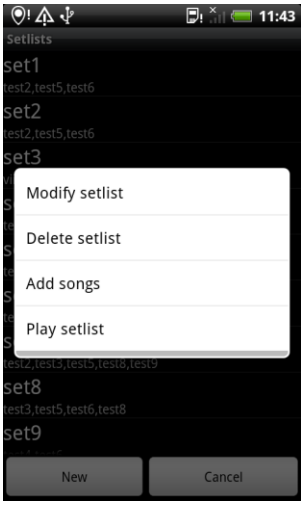


Joonis 12. Treeningurežiim

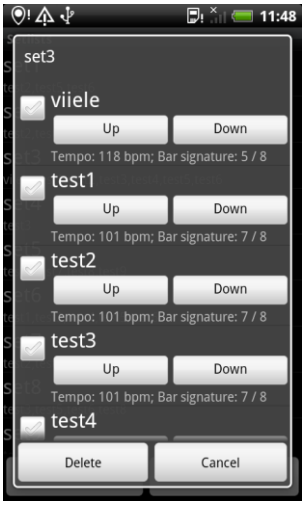
5.4 Seadete listid ning lugude listid

Selleks, et kasutaja saaks mugavalt sirvida erinevate lugude ning seadete liste (joonis 13) on koostatud eraldi vaade. Android platvormil on klass *ListView*, mis muudab listide kuvamise lihtsaks. Küll aga tuli antud projekti jaoks luua sellest kohandatud versioon, kuna lisati nupp, mille abil saab lood ära märkida, et siis neid kas kustutada või seadete listi lisada. Nii ongi iga laulu kõrvale pandud märkekast (inglise k. *checkbox*) ning loo enda peal vajutades saab selle loo seaded laadida rakendusse. Kui teha seadete listi peal pikk vajutus (inglise k. *long click*), siis avaneb väike menüü, kust saab valida listi muutmise, sinna lugude lisamise, selle kustutamise ning mängimise vahel (joonis 14). Kui valida listi muutmine, siis saab laule ka üles alla liigutada kahe nupu abil, et nende järjestust muuta (joonis 15). Kui list mängima panna, avaneb samasugune ekraan nagu tavarežiimis, kuid loo salvestamise ning laadimise nuppude asemel on edasi ning tagasi nupud, et kiiresti eelmine või järgmine lugu valida (joonis 16)

Joonis 13. Seadete list



Joonis 14. Listi menüü



Joonis 15. Lugude järjestamine

Joonis 16. Listi mängimine

6 Uuendused järgmistes versioonides

Kuna tegu on esimese versiooniga rakendusest, on seda võimalik ilmselt palju paremaks muuta. Nii on plaanis näiteks proovida optimeerimist Android NDK abil, lisada MIDI tugi ning lisada muusikalise notatsiooni loomine salvestatud heli järgi.

6.1 Optimeerimine kasutades Android NDK-d

Kuna antud rakenduse programmeerimise alguses ei teadnud autor, kui hästi selline lahendus töötab ning kas see üldse võimalik on, siis on rakenduse esimene versioon programmeeritud kasutades ainult Android SDK-d. Järgmistes versioonides on aga soov optimeerida salvestusprotsessi ning salvestatud andmete töötlemist. Selleks on plaanis kasutada NDK-d, kuna seal programmeeritud koodi peal ei jooksutata Java automaatset prügikoristust ning teiseks jäetakse ära ebavajalikud JNI *(Java Native Interface)* väljakutsed. Kuna *Fourier'* teisendus ning heli salvestamine on küllalt kulukad tegevused, siis võib see anda märgatava võidu programmi töö kiirendamisel.

6.2 MIDI tugi

MIDI protokolli abil antakse edasi muusikalisi signaale. Nende signaalide edasi andmiseks on olemas eraldi MIDI pistikud, kui kasutades MIDI protokolli on võimalik neid signaale anda edasi ka tavalist andmeside ühendust, nagu näiteks USB ühendust kasutades. MIDI väljundit on võimalik saada paljudest elektroonilistest muusikariistadest nagu näiteks süntesaatorid ning mõned elektritrummid. Kuna MIDI signaalid sisaldavad ka täpset aega kunas mingi sündmus toimus, näiteks vajutati alla mingi kindel süntesaatori klahv või löödi konkreetset elektritrummi, siis saab seda infot kasutada ka antud rakenduses tempo tuvastamiseks. Kuna aga Androidil puudub hetkel MIDI tugi, siis tuleks vastav protokoll kas ise programmeerida või otsida mõni teek, kus see juba tehtud on. Kui protokoll on programmeeritud, siis saab seda kasutada USB andmevoost MIDI signaalide tuvastamiseks.

6.3 Muusikalise notatsiooni loomine

Üks põhilisemaid täiendusi, mida rakenduse autor plaanib teha on salvestatud rütmi muusikalisse notatsiooni kirja panemine. See funktsioon on mõeldud trummaritele, kuna

muude instrumentide puhul tuleks leida ka noodi kõrgus. Funktsiooni idee seisneb selles, et kui trummar mängib mingi loo ette, siis pannakse kõik esinenud löögid mingisuguse muusikalise notatsiooni abil kirja. Selleks notatsiooniks võib algul olla ka näiteks MIDI. Kui muusik on loo ära mänginud, genereeritakse talle loost notatsioon, mida ta saab siis oma suva järgi parandada ja muuta. Löögi pikkuste määramine on juba tegelikult antud versioonis loodud, kuna löögi pikkust on kasutatud tempo määramisel. Funktsiooni raskeks osaks on erinevate trummide eristamine, sest iga trummikomplekti osa (nt. basstrumm, soolotrumm) tuleb noodikirjas erinevalt märkida. Ilmselt on keeruline ka noodikirja genereerimine nii et seda interaktiivselt muuta saaks.

Kokkuvõte

Antud bakalaureusetöö eesmärgiks oli Android platvormile mõeldud Adaptiivse Metronoomi rakenduse loomine. Antud rakendus on suunatud muusikutele, eelkõige trummaritele. See rakendus on uudne selle poolest et ta aitab algajal muusikul metronoomiga harjuda, mängides heli kindla tempoga, parandades seda samaaegselt muusiku mängimise järgi.

Töös on põhjalikumalt pööratud tähelepanu Android platvormi arendusvõimaluste kasutamisele sellise projekti puhul, kus on vaja teostada reaalajas signaalitöötlust. Lisaks sellele on töös kirjeldatud ka valminud rakenduse kõiki funktsioone ning kasutajaliidest. Lühiülevaade on toodud ka kasutatud algoritmidest ning edasiste versioonide arenduse plaanidest.

Tööst võib järeldada, et heli mängimine ning samaaegselt reaalajas toimuv rütmituvastus on võimalik ka mobiilsetel platvormidel, kuigi nende ressursid on palju piiratumad kui näiteks personaalarvutitel. Kuigi töö käigus tuli algselt kasutatud tehnoloogiaid välja vahetada, siis lõpuks jõuti ikkagi eesmärgini ning rakenduse põhifunktsioonid said soovitud kujul programmeeritud. Küll aga ei ole rakendus veel valmis sellisel kujul, et teda Google Play poodi üles panna, kuna vajab veel mõningaid muudatusi kasutajaliideses ning funktsioonide optimeerimises. Hetkel valmis olevat installeeritavat rakendust (.apk lõpuga fail) on võimalik alla laadida aadressilt http://sdrv.ms/164XbmU.

Adaptive Metronome For Android Platform

Bachelor's thesis

Andres Nirk

The purpose of this thesis was to develop an Adaptive Metronome application for Android platform. The application is intended for musicians, primarily for drummers. The application is innovative because it helps the musician to get used to playing with a metronome, as it plays the metronome sound with the certain tempo, correcting it if necessary.

It can be concluded from the work that playing the sound and at the same time detecting tempo from audio source in real-time is possible on the mobile platforms, even though they have less resources to use then personal computers for example. Although some of the originally planned techniques had to be replaced, in the end the objective of the project was reached and all the described functions of the application were created.

In the document, attention has been thoroughly paid on developing options on Android platform, when real-time signal processing is needed. In addition the functions and user interface of this application is also described. A short description of used algorithms and future development plans is given.

Viited

[1] Harry McCracken, „Who's Winning, iOS or Android? All the Numbers, All in One Place" http://techland.time.com/2013/04/16/ios-vs-android/ (April 16, 2013)

[2] USA kongressi arhiiv, http://www.digitalpreservation.gov/formats/fdd/fdd000016.shtml

[3] Androidi rakenduste arendajatele mõeldud lehekülg, „SeekBars", http://developer.android.com/design/building-blocks/seek-bars.html

[4] Smartmo, Wikipedia, vaba entsüklopeedia. http://en.wikipedia.org/wiki/File:World_Wide_Smartphone_Sales.png

[5] Sandberg Sound, „Free Metronome ja Drum machine, Google Play, Android rakenduste pood. https://play.google.com/store/apps/details?id=com.sandbergsound.meganomefree&feature=search_result#?t=W251bGwsMSwxLDEsImNvbS5zYW5kYmVyZ3NvdW5kLm1lZ2Fub21lZnJlZSJd (14.06.2012)

[6] Gabriel Simões, „Mobile Metronome", Google Play, Android rakenduste pood. https://play.google.com/store/apps/details?id=gabriel.metronome&feature=search_result#?t=W251bGwsMSwxLDEsImdhYnJpZWwubWV0cm9ub21lIl0.

[7] Stonekick, „Metronome Beats", Google Play, Android rakenduste pood. https://play.google.com/store/apps/details?id=com.andymstone.metronome&feature=search_result#?t=W251bGwsMSwxLDEsImNvbS5hbmR5bXN0b25lLm1ldHJvbm9tZSJd

[8] Dan Lavry, „Sampling Theory For Digital Audio", http://lavryengineering.com/pdfs/lavry-sampling-theory.pdf (2004)

[9] *Ruye Wang* , „The Sampling Theorem", http://fourier.eng.hmc.edu/e161/lectures/sampling/node3.html, (1999-09-28)

[10] Frédéric Patin, „Beat Detection Algorithms, http://www.flipcode.com/misc/BeatDetectionAlgorithms.pdf (21.02.2003)

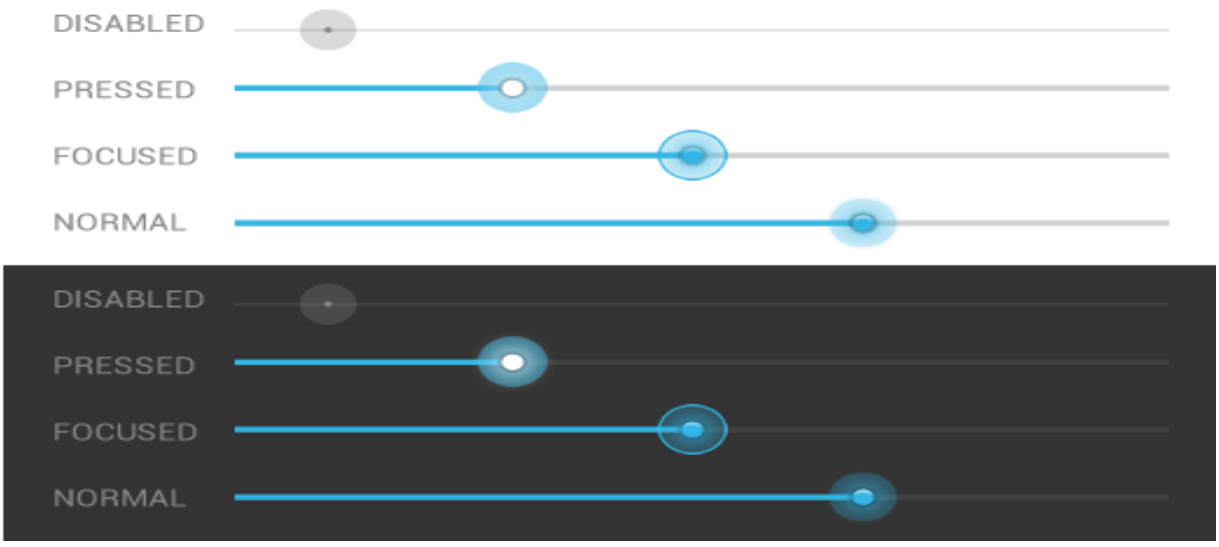
[11] „FFT Analysis", http://www.ni.com/white-paper/3342/en (13.12.2006)

[12] Androidi rakenduste arendajatele mõeldud lehekülg, „Performance Tips", http://developer.android.com/design/building-blocks/seek-bars.html

Lisad

Mõisted ja lühendid

- Klikk – lühike ja terav heli.
- Takt – Muusika notatsioonis kasutatav ühik, mis tähistab uue rõhulise osa algust.
- Liugur – inglise k. *slider* on Androidi kasutajaliidese osa (joonis 17).
- SDK – inglise k. *Software Development Kit* ehk tarkvaraarenduskomplekt on Androidi rakenduste arendamiseks kokku pandud tarkvaratööriistade komplekt.
- NDK – inglise k. *Native Development Kit* on põhimõtteliselt sama mis SDK, kuid sisaldab endas C ja C++ keeles rakenduste arendamise võimalust.
- PCM – inglise k. *pulse-code modulation* on meetod digitaalse signaali väljendamiseks [2]. Selle abil saab väljendada ka helilainet.
- MIDI – inglise k. *Musical Instrument Digital Interface* on standard, milles on kirjeldatud protokoll, digitaalne liides ning ühendustüübid. Selle abil saavad elektroonilised muusikainstrumendid suhelda arvutitega.
- FIFO – inglise k. *first in first out* on järjendi tüüp, kus esimene järjendisse lisatud element võetakse sealt ka esimesena välja.
- Tempo koputamise funktsioon – inglise k: *tap tempo* on tavapärane metronoomi nupp või lüliti, mille abil saab sellele korduvalt vajutades tempot määrata.



Joonis 17. Liuguri näited [3]

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Andres Nirk

(*autori nimi*)

(sünnikuupäev: 13.10.1991)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose Adaptiivne Metronoom Androidile,

   (*lõputöö pealkiri*)

   mille juhendaja on Margus Niitsoo,

   (*juhendaja nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **13.05.2013**