Aki Hautamäki & Tomi Sarni

ANDROID-KÄYTTÖJÄRJESTELMÄN MUSIIKKIOMINAISUUDET

MusicBox-musiikkiohjelma

ANDROID-KÄYTTÖJÄRJESTELMÄN MUSIIKKIOMINAISUUDET

MusicBox-musiikkiohjelma

Aki Hautamäki & Tomi Sarni
Opinnäytetyö
Kevät 2011
Tietojenkäsittelyn koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietojenkäsittely koulutusohjelma

Tekijä(t): Aki Hautamäki & Tomi Sarni
Opinnäytetyön nimi: Android-käyttöjärjestelmän musiikkiominaisuudet, MusicBox-musiikkiohjelma
Työn ohjaaja(t): Jouni Juntunen
Työn valmistumislukukausi ja -vuosi: Kevät 2011 Sivumäärä: 43

Opinnäytetyössä keskitytään tutkimaan Android 2.2 -käyttöjärjestelmän soveltuvuutta musiikkiohjelmien luomiseen. Soveltuvuutta tarkastellaan kehitystyökalujen ja käytettävässä olevien ohjelmistoluokkien kannalta sekä teoreettisesti että käytännössä. Tavoitteena on luoda musiikkiohjelma, jolla käytännön osuutta tutkitaan. Musiikkiohjelman toteutuksessa käyttöjärjestelmän äänenkäsittelyyn erikoistuneita luokkia pyritään kokeilemaan eri näkökulmista. Lisäksi raportissa selvitetään syitä Android-käyttöjärjestelmän valintaan ja sen asemaa markkinoilla verrattuna muihin valmistajiin ja käyttöjärjestelmävaihtoehtoihin.

Työssä on käytetty pääasiassa Android Developers -verkkosivuston tarjoamaa dokumentointia ja ohjeistusta. Tutkimusmenetelminä on käytetty suunnitelmallista käyttöjärjestelmän ominaisuuksien selvittämistä sekä valittujen kohteiden kokeilemista käytännössä. Kehityksessä ilmenneet haasteet on kirjattu osaksi raporttia ja ne toimivat kehitysehdotuksina sekä ohjeina muille musiikkiohjelmiston rakentamista suunnitteleville sovelluskehittäjille.

Toiminnallisen osuuden tuloksena syntyi MusicBox-musiikkiohjelma. Ohjelma jakautuu neljään alasovellukseen: Drum Workshop, Piano, Synthesizer ja Voice Scratch. Jatkossa ohjelmistoa kehitetään Android 2.3 -versiolla, joka sisältää uusia ominaisuuksia kuten ääniefektejä, joita voidaan käyttää ohjelmiston parantamiseen.

Asiasanat: Piano, musiikki, nauhoitus, Java, mobiilisisältöpalvelu, mobiililaitteet

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems

Author(s): Tomi Sarni & Aki Hautamäki
Title of thesis: Music qualities of Android operating system, MusicBox music software
Supervisor(s): Jouni Juntunen
Term and year when the thesis was submitted: Spring 2011 Number of pages: 43

The thesis work focuses on studying suitability of Android 2.2 operating system for music software development. In addition to it will give a brief introduction in to global smartphone markets and compare Android to other manufacturers and operating systems. The aim is to study the suitability through a process of creating a music program. The main focus in the process is in exploring specialized audio classes provided by the operating system through different avenues of approach.

Theoretical base relies in the documentation and tutoring provided by the Android Developers web service. Study methods include planned mapping of audio properties of the operating system and testing these properties in practice. The challenges and obstacles encountered in this process have been documented and will provide solutions and guidelines for other music software developers.

The functional part of the thesis work consists of MusicBox software product. The software divides in to four sub programs: Drum Workshop, Piano, Synthesizer and Voice Scratch. Next step will be to develop the software further by implementing new features provided by the future release of Android 2.3.

Keywords: Piano, music, sound recording, Java, mobile services, mobile equipment

SISÄLLYS

1 JOHDANTO 6

2 ANDROID JA ÄLYPUHELIMET 8

2.1 Mitä älypuhelimet ovat? 8

2.2 Älypuhelimien käyttöjärjestelmät 9

2.3 Google Android-käyttöjärjestelmä 11

2.4 Sovelluksen kehittäminen Android-älypuhelimiin 13

3 ANDROID MUSIIKKISOVELLUS 15

3.1 Kosketusrajapinta 15

3.2 Äänen tuottaminen 17

4 MUSICBOX 26

4.1 Sovelluksien esitteleminen 26

4.2 Android-käyttöjärjestelmän soveltuvuus musiikin tuottamiseen 33

5 POHDINTA 36

LÄHTEET 39

1 JOHDANTO

Uuden sukupolven mobiililaitteet tulevat kovaa vauhtia markkinoille. Eletään jälleen eräänlaista murroskautta, jossa suuret yhtiöt kilpailevat markkinaosuuksista ja siitä, kenen luomasta standardista tulee valtavirtaa. Googlen luoma ja ylläpitämä Android-käyttöjärjestelmä on uusimpia yrittäjiä älypuhelinmarkkinoilla. Android-sovellukset toteutetaan pääasiallisesti Java-kielellä, joka yhdistettynä avoimeen sovelluslisenssiin takaa sovelluskehittäjille helposti lähestyttävän kehitysympäristön. Android-älypuhelimien ollessa myös selvästi kilpailijoitaan edullisempia, on Googlen Android vahva haastaja markkinoiden johtajaksi.

Tämän opinnäytetyön tarkoituksena on tutustua Android-älypuhelimien äänenkäsittelyn mahdollisuuksiin sekä pyrkiä tekemään valmis musiikkisovellus. Koska Android-käyttöjärjestelmä on melko uusi ja muuttuu koko ajan, on mielenkiintoista olla ensimmäisten joukossa testaamassa sen sovellusmahdollisuuksia. Tarkastelun kohteeksi valittiin musiikin luominen, koska haluttiin kokeilla kosketusnäytön ja äänen yhdistämisestä. Android-käyttöjärjestelmä valittiin, koska sovellusten kehittämisen aloittaminen siihen on helpompaa verrattuna esimerkiksi Apple iPhone -älypuhelimeen.

Idea musiikkisovelluksesta syntyi Korgin valmistamasta Kaossilator nimisestä elektronisesta soittimesta, jossa soittaja pystyy luomaan musiikkia liikuttelemalla sormia paneelia vasten (KorgUK 2010, hakupäivä 28.14.2011). Nykyisistä älypuhelimista löytyy vastaavanlainen käyttöliittymä, jossa puhelimen käyttäjä ohjaa ohjelmistoa sormillaan. Lisäksi älypuhelimien suorituskyky on kasvanut huimaa vauhtia ja se ylittää jo teknillisiltä ominaisuuksiltaan Kaossilatorin suorituskyvyn. Selvitettäväksi jää, tarjoaako Android-älypuhelimien ohjelmistokehys riittävät mahdollisuudet mallintaa Kaossilator ohjelmistosovellukseksi.

Google Android on luonut älypuhelimillensa rajatun kehitysympäristön, joka sisältää valikoidun kokoelman Java-kirjastoja. Opinnäytetyössä pyritään tutkimaan tämän kehitysympäristön mahdollisuuksia. Löytyykö kyseisistä kirjastoista tapoja tuottaa ääntä eri sävelasteikoilla ja mahdollisesti luoda sointuja usean sävelen yhdistelminä? Voidaanko luoda yhteys sormen liikuttamiseen näytöllä äänen synnyttämiseksi? Näihin peruskysymyksiin pyritään löytämään vastaus, ja mikäli ne osoittautuvat mahdollisiksi, luomaan myös pienimuotoinen sovellus.

Opinnäytetyö on myös oppimisprosessi, joka auttaa tutustumaan älypuhelinsovellusten kehitykseen. Tämä prosessi syventää tietämystä älypuhelinten fyysisten ominaisuuksien hyödyntämisestä, sekä auttaa ymmärtämään niiden tuomat sovellusrajoitteet. Lisäksi opinnäytetyön tuottama musiikkiohjelma on konkreettinen näyttö taidoista, joista on hyötyä työmarkkinoilla.

2 ANDROID JA ÄLYPUHELIMET

Termistä *älypuhelin* on vauhdilla tulossa yleissana mainosteksteissä. Voisi kuvitella, että usealle termin merkitys on hieman hämärän peitossa. Myös *Android*-nimi on joissain määrin outo. Puhelimet ovat totuttu tuntemaan merkkien ja valmistajien perusteella. Monelle on ehkä epäselvää, että useat valmistajat käyttävät järjestelmää puhelimissaan, eikä yhden valmistajan Android-älypuhelin ole sama asia kuin toisen.

2.1 Mitä älypuhelimet ovat?

"A telephone that provides additional information accessing features. Any mobile telephone that combines voice services with e-mail, fax, and pager or Internet access is called a smart phone" (Ceva Mediaroom 2009, hakupäivä 15.10.2010).

Määritelmän mukaan maailman ensimmäinen kuluttajille suunnattu älypuhelin oli IBM:n Simon puhelin, joka tuli laajamittaiseen myyntiin vuonna 1994. Simon sisälsi miltei kaikki älypuhelimelta vaadittavat ominaisuudet. Se yhdisti puhelimen ja dataliikenteen samaan laitteeseen (O'Malley 1994, hakupäivä 15.10.2010). Voidaan puhua älypuhelimen ensimmäisestä sukupolvesta. Simonin myynti jäi kuitenkin vaatimattomaksi mahdollisesti sen korkean hinnan takia. Älypuhelimien läpimurtoa jouduttiin odottamaan aina vuoteen 2007 asti, jolloin Applen iPhone toi kuluttajille teknisesti samat palvelut kuin IBM:n Simon (Masalin 2009, hakupäivä 1.4.2011). Ehkä juuri tämän ansiosta termi *älypuhelin* usein rinnastetaan iPhoneen.

Usein alan erikoislehtien artikkeleissa älypuhelimista käytetään myös termiä *feature phones* eli *toiminnallisuuspuhelimet*. Tämä kertoo siitä, että teknisistä ominaisuuksista ollaan siirtymässä kohti toiminnallisia ominaisuuksia. Puhelimen sovellussisällöllä on enemmän merkitystä kuin koolla, muodolla, painolla tai lisälaitteilla (Pitkänen 2010, hakupäivä 6.4.2011). Usein tämä ilmenee siten, että puhelinvalmistajien tuodessa markkinoille uusia malleja, pyrkivät yhtiöt mainonnassaan painottamaan paremman kameran sijaan puhelimien mukana tulevia ohjelmistoja kuten Skypeä ja Facebookia.

Älypuhelinmarkkinoilla on tällä hetkellä viisi isoa älypuhelinvalmistajaa: Nokia, RIM, Apple, Samsung ja HTC, joista kaksi jälkimmäistä kuuluvat Open Handset Allianceen (OHA) (Kotilainen 2011, hakupäivä 16.2.2011). OHA on Googlen Android-käyttöjärjestelmää käyttävien puhelinvalmistajien (Motorola, LG, HTC ja Samsung muutaman mainitaksemme) muodostama yritysliittouma (Open Handset Alliance 2010, hakupäivä 24.11.2010).

2.2 Älypuhelimien käyttöjärjestelmät

Älypuhelimien teknisten ominaisuuksien lisäksi käyttöjärjestelmät luovat eroja puhelimien välillä. Käyttöjärjestelmät rajoittavat osittain sovelluskehitystä sekä vaikuttavat epäsuorasti myös puhelimien teknisiin ominaisuuksiin muun muassa siten, että toiset käyttöjärjestelmät kuormittavat puhelimien prosessoria eri tavalla. Joissain tapauksissa myös käyttöjärjestelmät rajoittavat sovellusten mahdollisuutta vaikuttaa puhelimen tekniseen käyttäytymiseen. Myös sovellusten kehitysympäristöt ovat erilaisia käyttöjärjestelmien välillä. Lähes kaikki älypuhelimien käyttöjärjestelmät pohjautuvat C- ja C++-ohjelmointikieliin. Usein sovelluskehittäjille tarjotaan mahdollisuus kehittää sovelluksia myös Java-kielellä. Android-käyttöjärjestelmässä sovelluksien kehittäminen tapahtuu pääasiassa Java-kielellä mutta kehittäjille tarjotaan lisäksi mahdollisuus kehittää sovelluksia myös C- ja C++-ohjelmointikielillä. Kuviossa 1 näkyy yhteenveto merkittävimmistä älypuhelintyypeistä ja niiden ominaisuuksista. Kuviossa 1 *tyypillä* tarkoitetaan markkinoilla esiintyvää yleisesti käytettyä nimitystä käyttöjärjestelmästä.

Tyyppi	Google Android	Windows Mobile	RIM BlackBerry	Symbian	Apple IPhone
Käyttöjärjestelmä	Android OS	Windows Phone 7	BlackBerry OS	Symbian OS	iOS
Lisenssi	Apache/GPL v2	Microsoft EULA	RIM EULA	Nokia/Os	Apple EULA
Ydin	Linux	Darwin/Unix	Java based	Linux	Darwin/Unix
Kehityskieli	C/C++/Java (UI)	C++	C++	C/C++	C/C++
Ohjelmointikielet	Java /C/C++	C#	Java	Python/C/C++/Java	Objective-C
Sovelluskehitys	SDK/Eclipse+NB	.NET framework	RIM/Eclipse + NB	QT	iOS SDK/Xcode 3.1

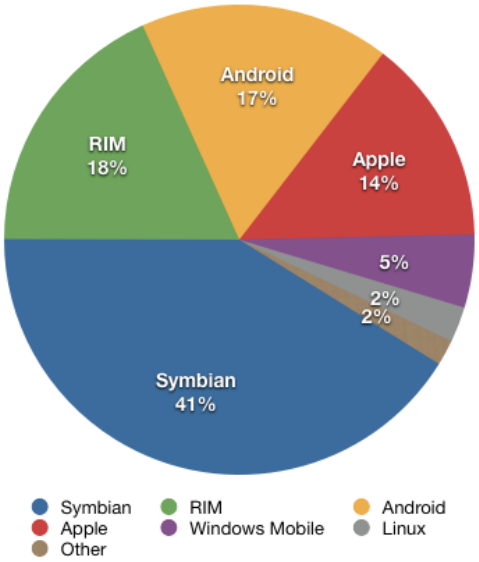
KUVIO 1. Vertailu markkinoiden merkittävimmistä älypuhelintyypeistä ja niiden ominaisuuksista.

Suurin osa vuoden 2010 toisen vuosineljänneksen aikana valmistetuista älypuhelimista oli rakennettu Symbian-käyttöjärjestelmän pohjalle (Kuvio 2). Symbian OS on Nokian omistama mutta sen lisenssi on vapautettu avoimeksi. Symbian-käyttöjärjestelmä on C++-ohjelmointikielellä toteutettu ja sovellusten kehitysympäristönä toimii Qt (Forum.Nokia 2011, hakupäivä 1.4.2011). Nokia ja

Intel ovat ilmoittaneet korvaavansa tulevaisuudessa Maemon (Maemo SDK) yhteisellä MeeGo kehitysympäristöllä, jossa yhdistyy Intelin Linux ytimen päälle rakennettu Moblin-käyttöjärjestelmä sekä Nokian Symbian-käyttöjärjestelmä (Grabham 2010, hakupäivä 18.3.2011). Vuonna 2011 Nokia kuitenkin ilmoitti, että se siirtyy käyttämään Microsoftin Windows Phone 7:ää ensisijaisesti älypuhelimiensa käyttöjärjestelmänä mutta kuitenkin jatkavansa Symbian-käyttöjärjestelmän tukea toistaiseksi (Lehto 2011, hakupäivä 1.3.2011).

Toiseksi suurimman käyttöjärjestelmän paikasta kilpailevat RIM:n BlackBerry, Googlen Android sekä Applen iPhone. Varsinkin RIM on ollut erityisen suosittu Pohjois-Amerikan markkinoilla, jota yleisesti pidetään suunnannäyttäjänä muiden markkinoiden kehitykselle. RIM on kuitenkin menettänyt markkinaosuuttaan iPhone- sekä Android-käyttöjärjestelmille. Näyttäisi siltä, että Pohjois-Amerikan markkinajohtajuus tullaan ratkaisemaan iPhone- ja Android-käyttöjärjestelmien välillä (Dalu 2010, hakupäivä 30.3.2011).

Windows on tullut älypuhelimien käyttöjärjestelmämarkkinoille vasta vuoden 2010 loppupuolella julkaistuaan Windows Phone 7 -käyttöjärjestelmän, jossa on uudistettu versio Windows CE -käyttöjärjestelmästä (Morton 2010, hakupäivä 18.3.2011). Microsoft sinänsä on aina ollut ohjelmistojen ja käyttöjärjestelmien tuottaja eikä varsinaisesti ole valmistanut muita kuin tietokoneiden oheistuotteita. Onkin mielenkiintoista nähdä, miten Microsoft onnistuu sisällyttämään käyttöjärjestelmänsä eri puhelinvalmistajien malleihin vai joutuuko se kenties tuomaan markkinoille omia laitteitaan, kuten esimerkiksi Google on osittain joutunut tekemään. Samsung aikoo sisällyttää Windows Phone 7 -käyttöjärjestelmän 63 prosenttiin älypuhelimistaan vuoden 2011 malleissa. Samsungin Android-käyttöjärjestelmään perustuvat puhelimet ovat kuitenkin kasvattaneet myyntiään reilusti, jolloin jää nähtäväksi muuttuuko Windows Phone 7:n osuus jatkossa (Ojanperä 2010, hakupäivä 24.11.2010). Samsung on lisäksi luonut oman Bada-käyttöjärjestelmän Wave-mallin älypuhelimiin. Bada on vielä uutuus, ja sen osuus myydyistä puhelimista on tämän takia vielä varsin pieni (Sani 2010, hakupäivä 6.4.2011).



KUVIO 2. 2010 vuoden toisen vuosineljänneksen eri käyttöjärjestelmien osuudet Maailman älypuhelinmarkkinoista (Bart 2010, Hakupäivä 24.11 2010).

2.3 Googlen Android-käyttöjärjestelmä

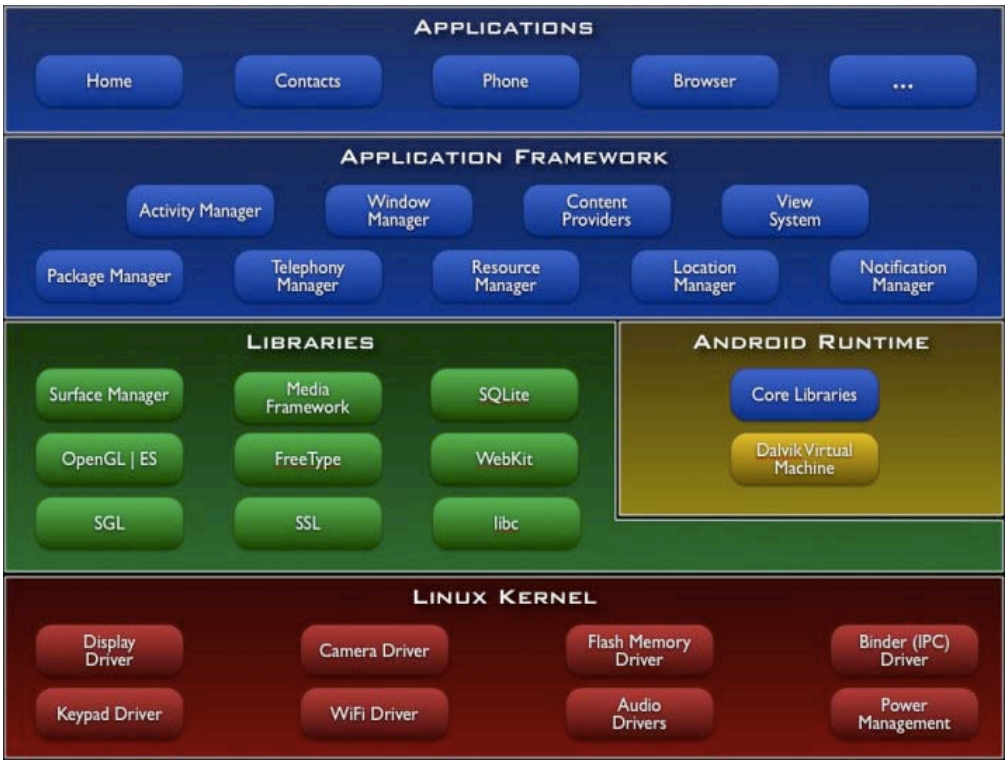
Android-käyttöjärjestelmää kehitti alun perin Android Incorporated -yhtiö, jonka Google osti vuonna 2005. Vuonna 2007 perustettiin OHA, joka otti vastuulleen Android-käyttöjärjestelmän kehittämisen. OHA koostuu useista älypuhelinten laitevalmistajista kuten Motorola, Samsung, LG ja HTC sekä komponenttivalmistajista kuten Intel, Qualcomm ja Texas Instruments. OHA:n vetovastuu on Googlella, jonka valmistamat Nexus-älypuhelimet toimivat Android-käyttöjärjestelmän lippulaivoina (Beavis 2008, hakupäivä 11.3.2011).

Android-käyttöjärjestelmän filosofiana on ollut tuoda markkinoille myös avoin ja lisenssimaksuista vapaa vaihtoehto älypuhelinvalmistajille ja sovelluskehittäjille. OHA on halunnut myös Android Open Source projektissaan varmistaa, ettei osa-alueita voi mikään yritys tai järjestö rajoittaa. Android-käyttöjärjestelmää kehitetään niin sanotulla avoimen lähdekoodin periaatteella (Apache v2 –lisenssi) (Open Handset Alliance 2011, hakupäivä 15.4.2011). OHA kehittää erikseen suljetusti laitteistoja sekä ohjelmistoa seuraavaan julkaistavaan versioon Android-käyttöjärjestelmästä. Kun tämä seuraava versio on valmis, julkaistaan myös järjestelmän kehitysympäristön (Android Software Development Kit eli Android SDK) lähdekoodi avoimena, jotta sovelluskehittäjät voivat halutessaan muokata sitä sopivammaksi omiin projekteihinsa (Android Open Source Project 2010, hakupäivä 24.11.2010). Tulevaisuudessa Android-käyttöjärjestelmä voidaan nähdä myös

muissa laitteissa. Wall Street Journal uutisoi maaliskuussa 2010, että Google on viemässä Android-käyttöjärjestelmää digibokseihin (Vaalisto 2010, hakupäivä 6.4.2011).

Kuviossa 3 on esiteltynä Android-käyttöjärjestelmän rakenne. Alimpana oleva Linux Kernel (Linux ydin) ohjaa puhelimen avaintoimintoja ja fyysisiä komponentteja, kuten ääni- ja kuvapiirejä. Seuraavalla tasolta löytyy luokkakirjastot (Libraries ja Android Runtime), jotka koostuvat Java-luokista. Luokkakirjastot ja niiden sisältämät luokat luovat rajapinnan, mitkä mahdollistavat sen, että sovellukset pääsevät käsiksi osaan puhelimen fyysisistä komponenteista (Android Developers 2011q, hakupäivä 6.4.2011). Esimerkkinä tällaisesta voisi olla luokan metodi, joka palauttaa sovellukselle tiedon siitä, mihin kohtaan kuvaruutua käyttäjän sormi koskettaa metodin kutsuhetkellä.

Application Framework -taso tarjoaa sovelluskehittäjille valmiita ohjelmistokomponentteja ja rajapintoja. Esimerkkinä mainittakoon Notification Manager, joka tarjoaa valmiit raamit kaikille käyttäjälle näkyville ilmoituksille. Sovelluskehittäjä voi näin säästä aikaa lisäämällä omat ilmoituksensa käyttämällä Notification Manageria. Tämä myös tekee ilmoituksista yhtenäisiä muiden puhelimen ilmoitusten kanssa. Ylimpänä tasona ovat valmiit sovellukset (Applications), kuten verkkoselain tai mediasoitin. Tähän tasoon asettuvat myös sovelluskehittäjien luomat omat sovellukset (Android Developers 2011q, hakupäivä 6.4.2011).



KUVIO 3. Android-käyttöjärjestelmän rakenne ja kerrokset (Android Developers 2011q, hakupäivä 6.4.2011).

2.4 Sovelluksen kehittäminen Android-älypuhelimiin

Sovelluskehittäminen Android-älypuhelimiin tapahtuu pääasiassa Java-pohjaisissa ohjelmiston kehittämisympäristöissä. Eclipse- ja IntelliJ IDEA -ohjelmistokehittimiin on asennettavissa Android SDK, joka mahdollistaa sovellusten kehittämisen Android-käyttöjärjestelmälle. Nämä kaksi sovelluskehitintä ovat Android Developers -verkkosivun virallisesti suosittelemia sovelluskehittimiä (Android Developers 2011f, hakupäivä 1.4.2011). Näistä Eclipse on vapaasti käytettävän lisenssin alla ja IntelliJ IDEA on maksullinen ohjelmisto. IntelliJ IDEA:sta on kuitenkin ladattavissa karsittu Community Edition, jolla on mahdollista kehittää Android-sovelluksia (JetBrains 2011, hakupäivä 1.4.2011).

Eclipsen ja IntelliJ IDEA:n lisäksi NetBeans-ohjelmistoon on saatavilla tuki Android-sovelluskehitykseen. NetBeans on Eclipsen tavoin vapaasti käytettävän lisenssin alla (Project Kenai 2011, hakupäivä 1.4.2011). Project Kenai on Sun Microsystemsin perustama kaikille Java-sovelluskehittäjille suunnattu yhteisö, missä muun muassa NetBeansin Android-tuen kehitys tapahtuu vapaaehtoisvoimin. Project Kenain jatkuminen vaikuttaa epävarmalta Oraclen ilmoitettua

suunnitelmistaan sulkea verkkosivu sen ostettua Sun Microsystems (Krill 2010, hakupäivä 1.4.2011).

Android Open Source Project on kehittänyt Eclipseen ADT-liitännän, joka mahdollistaa Android SDK:n asentamisen sekä päivittämisen kätevästi Eclipse-ohjelman kautta. Android Developers -verkkosivulta löytyy myös kattavat ohjeet Android-sovelluskehityksen aloittamiseen juuri Eclipseä käyttämällä (Android Developers 2011a, hakupäivä 1.4.2011). Tässä työssä on päädytty käyttämään Eclipseä sovelluksen kehittämiseen, koska se on ilmainen, Android Developers -verkkosivuston suosittelema ja sen käytöstä on aikaisempaa kokemusta.

Android SDK sisältää kaikki sen julkaisuhetkeä aiemmin ilmestyneet ohjelmointirajapinnan versiot (API), älypuhelimen emulaattorin, dokumentaatiot, ohjeita, esimerkkisovelluksia sekä debuggerin (virheiden jäljittäjä) (Android Developers 2011g, hakupäivä 29.3.2011). Tässä työssä perehdytään Android SDK 2.2 -versioon nimeltään *Froyo*. Vuodelle 2011 on suunniteltu julkaistavan seuraava versio nimeltään Gingerbread (Android SDK 2.3) Google Nexus -sarjan myötä (Gordano 2011, hakupäivä 1.4.2011).

Android NDK (native development kit) tarjoaa mahdollisuuden sisällyttää omia kirjastoja natiivikielellä (C tai C++). NDK:lla tehdyille kirjastoille on varattu käyttöjärjestelmästä erillinen kansio, josta niitä voidaan ladata sovelluksen käyttöön. Sovellusta asennettaessa on sekä sovellustiedostot että kirjastot talletettava puhelimeen. Kirjastot ovat asennuksen jälkeen myös muiden sovellusten käytössä. Natiivikielellä kehotetaan toteuttamaan sellaisia ratkaisuja, joita ei löydy Android SDK:n kirjastovalikosta tai jos halutaan ohittaa Java-ohjelmointirajapinta paremman tehokkuuden saavuttamiseksi. Kaikki SDK:lla kirjoitetut koodit käännetään automaattisesti Java-kielestä C-kieleen ajonaikaisesti, mikä ajoittain voi tuoda lisäkuormitusta sovelluksen ajoon. NDK on tarkoitettu pääasiassa puhelinvalmistajille, jotta ne voivat räätälöidä Android-käyttöjärjestelmää puhelinmalleihin sopivaksi mutta on myös sovelluskehittäjien käytettävissä ilman lisenssimaksuja (Android Developers 2011r, hakupäivä 11.3.2011).

3 ANDROID MUSIIKKISOVELLUS

Opinnäytetyönä kehitettävän musiikkisovelluksen tulisi hyödyntää älypuhelimien fyysisiä ominaisuuksia sekä perinteisiin matkapuhelimiin verrattuna kehittynyttä suorituskykyä. Fyysisillä ominaisuuksilla tarkoitetaan mahdollisuutta käyttää sormia sovellusten ohjaamiseen sekä hyödyntää kasvaneen kuvaruudun koko pinta-alaa. Erityisesti Android 2.2 -version mahdollistama multitouch-ominaisuus on merkittävässä roolissa. Multitouch mahdollistaa sen, että puhelin tunnistaa usean sormen yhtäaikaisen käytön. Tämä ominaisuus on ehdoton vaatimus sointujen muodostamiseen sormia käyttämällä. Älypuhelimet omaavat jo kaikki tarvittavat laitteistot sointujen ja äänen tuottamiseen. Suurin haaste on selvittää, onko käyttöjärjestelmän ohjelmointirajapinnassa tarvittavat luokat siirtämään laitteistolta saatua tietoa sovelluksen käyttöön.

Musiikkisovelluksessa puhelimen näyttöalue toimii kosketuspintana soittimelle, jossa eri alueet näytöstä tuottavat eri ääniä. Näytön vaaka- ja pystyreunat toimivat koordinaatiston *x*- ja *y*-akseleina, joista esimerkiksi yhdelle akselille asetetaan sävelasteikko ja toiselle akselille oktaaveja. Kun kullekin sävelelle ja oktaaville varataan oma pinta-ala pikseleissä, täyttyy näyttöalue neliöistä. Sormen osuessa tällaisen neliön alueelle, toistaa sovellus säveltä ja oktaavia vastaavan äänen. Käyttäjä voi muodostaa soinnun painamalle kahta tai useampaa aluetta (neliötä) näytöllä samanaikaisesti. Ääni kuuluu niin kauan kuin sormi on kiinni näytössä. Käyttäjä voi myös liu'uttaa sormea neliöstä toiseen siten, että ääni vaihtuu katkeamatta toiseen. Käyttäjä voi tavallaan soittaa piirtämällä näytölle. Sovelluksen on tarkoitus olla yksinkertainen ja hauska, eikä käyttäjän tarvitse tietää musiikista tai säveltämisestä. Pelaamisen ja soittamisen tulisi yhdistyä sovelluksessa.

3.1 Kosketusrajapinta

Sovelluksessa puhelimen näyttö toimii kosketuspintana soittimelle, eli sovelluksen täytyy tunnistaa kosketuksen kohta näyttöalueella ja se, millainen kosketus on. Kosketuksella on useita parametreja, jotka määräävät kosketuksen tyylin. Tällaisia parametreja ovat kosketuksen voimakkuus, kosketuksen kesto sekä kosketuksen aloitus- ja lopetuspiste. Lisäksi jos kyseessä on usean sormen yhtäaikainen kosketus, pitää edellä mainitut parametrit huomioida jokaiselle sormelle. Kun kosketuksen parametrit tunnetaan, voidaan muodostaa käsitteitä kosketukselle, kuten onko ky-

seessä pitkä painallus, lyhyt napautus vai sormen liikuttaminen pinnalla. Android-ohjelmointikirjastot sisältävät joitain valmiita luokkia ja metodeja, joilla voidaan tunnistaa kosketustapahtumia ja poimia tapahtumista edellä mainittuja parametreja. Android SDK sisältää muun muassa GestureDetector- ja GestureOverlayView-luokat.

android.view.GestureDetector-luokka tarjoaa tapahtumankäsittelijän *onTouchEvent (MotionEvent event*), joka aktivoituu kun kosketus näytölle tapahtuu ja välittää ohjelmoijan käyttöön kosketuksen parametrit sulkujen sisällä *event*-muuttujassa. Esimerkiksi *event.getY()* kertoo mistä kohdasta kuvaruutua kosketus alkoi. Luokan haittana on se, että se on tarkoitettu ainoastaan kosketuksen tunnistamiseen, ei esimerkiksi kursorilla tai näppäimillä tehtyihin valintoihin (Android Developers 2011i, hakupäivä 11.3.2011). Kuviossa 4 on lueteltuna luokan valmiit metodit, jotka aktivoituvat ainoastaan kun tietyt ehdot ovat täytetty.

onDown(MotionEvent event) Tapahtumankäsittelijä aktivoituu joka kerta kun uusi kosketus alkaa.
onFling(MotionEvent start, MotionEvent end, float velocityX, float velocityY) Tapahtumankäsittelijä aktivoituu kun algoritmi tunnistaa kosketuksen "huitaisuksi".
onLongPress(MotionEvent event) Tapahtumankäsittelijä aktivoituu kun algoritmi tunnistaa pitkän painalluksen.
onScroll(MotionEvent start , MotionEvent end , float distanceX, float distanceY) Tapahtumankäsittelijä aktivoituu kun algoritmi tunnistaa vierityksen.
onShowPress(MotionEvent event) Tapahtumankäsittelijä aktivoituu joka kerta kun uusi kosketus tapahtuu.
onSingleTapUp(MotionEvent event) Kuten *onDown* yllä mutta aktivoituu kun kosketus loppuu.

KUVIO 4. GestureDetector-luokan tapahtumankäsittelijöitä (Android Developers 2011j, hakupäivä 11.3.2011).

android.gesture.GestureOverlayView-luokka tarjoaa yksinkertaisen ja nopean tavan tunnistaa sormilla tehtyjä eleitä. Tämän luokan voi suoraan sijoittaa läpinäkyvänä elementtinä mihin tahansa näytön alueelle. Sormen osuessa elementin päälle, aktivoi se luokalle ominaiset tapahtumankäsittelijät. GestureDetector-luokasta poiketen tässä luokassa on mahdollisuus automaattisesti jäljittää sormen liike graafisesti näytöllä. Luokalla on myös erityisominaisuus kaapata tapahtuma, mikäli samalla näytön alueelle on elementin lisäksi liitetty muita tapahtuman kuuntelijoita (Android

Developers 2011k, hakupäivä 11.3.2011). Kuviossa 5 on lueteltuna luokan valmiit metodit, jotka aktivoituvat ainoastaan kun tietyt ehdot ovat täytetty.

onGesture(GestureOverlayView element, MotionEvent event) Aktivoituu joka kerta kun sormi liikkuu (aktivoituu jokaisella kuljetulla pikselliä).
onGestureCancelled(GestureOverlayView element, MotionEvent event) Aktivoituu jos sormi kulkee elementin ulkopuolelle tai kosketustapahtuma keskeytyy.
onGestureEnded(GestureOverlayView element, MotionEvent event) Aktivoituu kun sormi nousee ylös näytöltä. Kosketustapahtuma päättyy.
onGestureStarted(GestureOverlayView element, MotionEvent event) Aktivoituu kun sormi koskettaa näyttöä ja liikkuu vähintään yhden pikselin verran.

KUVIO 5. GestureOverlayView-luokan tapahtumankäsittelijöitä (Android Developers 2011l, hakupäivä 11.3.2011).

3.2 Äänen tuottaminen

Musiikkiohjelman tarkoitus on antaa käyttäjälle mahdollisuus tuottaa musiikkia ja ääniä sormillaan. Kosketusrajapinnasta kerättyä dataa hyväksi käyttämällä saadaan sovellukseen tuotua tarvittavat muuttujat ohjailemaan ääntä. Äänet joko tuotetaan sovelluksella tai ladataan sovelluksen käyttöön äänitiedostoista. Tällaisia tiedostoja voivat olla yksittäiset instrumentit tai kokonainen musiikkiraita. Huomioitavaa on kuitenkin se, että jokainen instrumentti- tai musiikkiraita vie tilaa älypuhelimen muistista sekä lisää sovelluksen kokoa, sillä ne on sisällytettävä sovelluksen mukana tuotaviin resursseihin. Lisäksi on tutkittava, antavatko äänitiedostojen toistamiseen tarkoitetut luokat riittävästi mahdollisuuksia muokata äänitiedostoa soiton aikana ja vielä siten, että lopputulos on mielekkään kuuloista.

android.media.SoundPool-luokan tarkoitus on hallita ääniresurssien tuomista sovellusten käyttöön. Äänitiedostot voidaan joko sisällyttää sovellukseen, ladata puhelimen muistikortilta tai ladata verkosta. SoundPool-luokka on tarkoitettu kuitenkin vain lyhyiden ääninäytteiden toistamiseen. Mikäli yksittäisen tiedoston koko kasvaa liian isoksi, voi se aiheuttaa sovelluksen ajon aikana virheilmoituksia tai keskeyttää sovelluksen suorittamisen. Näin ollen luokka ei siis sovellu esimerkiksi kokonaisten kappaleiden lataamiseen. SoundPool-luokkaan on mahdollista ladata kerralla

useita tiedostoja ääniraidoiksi. Ääniraidoilla tarkoitetaan tässä työssä ääntä, joka on luokan ilmentymän eli olion käytössä muistivarauksena. Käytettävät metodit ja luokat eivät tallentamisen ja lataamisen lisäksi suorita mitään muokkauksia itse äänitiedostoon. Jokaiselle raidalle luodaan oma kokonaislukutunniste niiden erottamiseksi toisistaan. Tämä pienentää sovelluksen muistin tarvetta, sillä jokaista ääniraitaa kohden ei tarvitse luoda omaa oliota (luokan ilmentymää) (Android Developers 2011o, hakupäivä 11.3.2011).

SoundPool-luokan metodilla *setRate(int streamId, float rate)* voidaan muokata ladatun tiedoston äänenkorkeutta (äänen soittotaajuutta) muuntamalla soittonopeutta (*rate*) suorituksen aikana. Soittonopeutta voi korkeintaan kasvattaa kaksinkertaiseksi tai puolittaa (Android Developers 2011o, hakupäivä 11.3.2011). Äänitiedostona voidaan ladata esimerkiksi kitaralla soitettu vapaa A-kieli, joka soi 440 hertsin taajuudella. Kun kasvatetaan ääniraidan soittonopeutta puolitoistakertaiseksi, muuttuu äänen taajuus 660 hertsiin, mikä vastaa E-säveltä. On otettava huomioon myös se, että äänen *väri* muuttuu ja liian suuret korotukset tai laskut saavat äänen kuulostamaan luonnottomalta. Varsinkin kun kyseessä on sointu jossa ääniaalto koostuu useasta taajuudesta, A-duuri ei välttämättä kuulosta luonnolliselta E-duurilta pelkällä taajuuden muutoksella.

Mahdollisuus muuttaa soittonopeutta ajonaikaisesti mahdollistaa taajuuden liittämisen esimerkiksi sormen koordinaattiin näyttöalueella. Siten sormea liikuttaessa soittonopeus eli ääniraidan taajuus muuttuu suhteessa koordinaatin muutoksen. SoundPool–luokkaa käyttämällä voi ladata useita ääniraitoja kerrallaan muistiin. Näin esimerkiksi *x*- ja *y*-akseleille on mahdollista sitoa useita ääniraitoja. Kunkin ääniraidan play-, stop- ja pause-toimintoja sekä taajuutta voi erikseen muuttaa suorituksen aikaisesti toisistaan riippumatta (Android Developers 2011o, hakupäivä 11.3.2011).

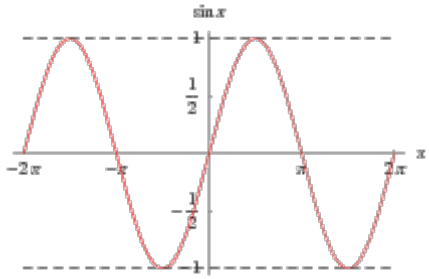
Mikäli SoundPool-luokkaa käyttämällä haluaa rakentaa mahdollisimman autenttisen kuuloisen soittimen, on syytä luoda mahdollisimman monta valmista ääniraitaa vastaamaan useita eri äänitaajuuksia ja käyttää soittonopeuden muuntamista siirtymisiin sävelten välillä. Ääniraitojen otoksista kannattaa tehdä mahdollisimman lyhyet, jotta tiedostojen koot säilyvät pieninä. Esimerkiksi jos haluaa jatkuvaa kitaran ääntä, voi käyttää toistomenetelmää (loop), joka aloittaa ääniraidan soittamisen alusta sen loputtua.

SoundPool-luokan käytöstä tekee epävarman se, että raitojen pituus ja määrä ovat laitteistokohtaisia. On siis mahdollista, että jossain tapauksissa muut sovelluksen osat kärsivät, jos SoundPool-luokka on ylikuormitettu. Luonnollisesti tämä lisää testauksen tarvetta eri alustoilla ja vaatii

poikkeuksien käsittelyä sovellusta rakennettaessa. SoundPool muokkaa äänitiedostot automaattisesti Pulssikoodimodulaatiomuotoon (PCM) käyttämällä MediaPlayer-luokkaa muokkauksen suorittamiseen. Suositeltavaa on, että SoundPool-luokan ilmentymää ei käytetä ääniraitojen soittamiseen ajonaikaisesti, vaan raidat olisi syytä alustaa PCM-muotoisiksi esimerkiksi ohjelman käynnistyessä *load*-metodilla (Android Developers 2011o, hakupäivä 11.3.2011).

android.media.AudioTrack-luokka on tarkoitettu yksittäisen PCM-muotoisen ääniraidan toistamiseen. Merkittävin ero SoundPool-luokkaan on siinä, että se sisältää puskurin, johon ääniraitaa voi kirjoittaa ajonaikaisesti. Toisaalta AudioTrack-luokkaan ei voi ajaa esimerkiksi MP3-muotoista ääniraitaa sillä se ei suorita ääniraidan muuntamista automaattisesti eikä myöskään sisällä tarvittavia purkukodekkeja. Puskurin kokoa ja lukutapaa voi säätää AudioTrack-luokan ilmentymää luodessa. Tämä antaa ohjelmointimahdollisuuksia, joita ei löydy muista Androidin äänentoistoon erikoistuneista luokista. Puskurin voi asettaa joko staattiseen tai striimimuotoon(*stream*). Striimimuotoisessa toistossa puskuria ei tarvitse täyttää kokonaan sillä ääntä toistetaan sitä mukaan kun sitä tuodaan puskuriin. Staattisessa muodossa puskuri on täytettävä kokonaan ennen äänen toistamista (Android Developers 2011d, hakupäivä 11.3.2011).

Syntetisoidun äänen tuottamisessa on oleellisen tärkeää mahdollisuus puskurin dynaamiseen täyttämiseen sekä ääniraidan PCM-muotoinen lukeminen. PCM-ääni muodostetaan näytteistä. Esimerkiksi yleisesti käytettävän CD-tasoisen äänen näytteenottotaajuus on 44100 hertsiä. Toisin sanoen: yksi sekunti ääntä koostuu 44100 näytteestä. Jokaista yksittäistä näytettä kuvaa numeerinen arvo väliltä -32,768 ja 32,768 (16-bittinen). Keinotekoista eli syntetisoitua ääntä tuotettaessa syötetään soittimelle järjestyksessä numeerisia arvoja, jotka soitin muuntaa ääneksi. Satunnaisesti numeerisia arvoja ei kuitenkaan valita, sillä se kuulostaisi pelkästään kohinalta tai rätinältä. Arvojen syöttäminen tapahtuu metodilla *write(short[] audioData, int offSetInShorts, int sizeInShorts)*, jossa arvot syötetään ensin *short [ ]* –muotoiseen (16-bittinen) taulukkoon iteroimalla ja sen jälkeen kyseisellä metodilla soittimeen (Android Developers 2011d, hakupäivä 11.3.2011). Rakentamalla numeerisista arvoista aaltomuotoisia jatkuvia sarjoja, saadaan ääni soimaan. Kuvioissa 6–13 on esiteltynä yleisesti käytettyjä aaltomuotoja syntetisaattoreista.

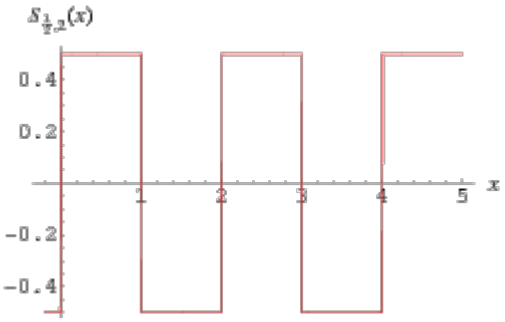


KUVIO 6. Sinimuotoinen aaltokuvio (Weisstein 2011b , hakupäivä 16.2.2011).

$$f(x) = a \sin(\omega x + c).$$

KUVIO 7. Siniaallon funktio (Weisstein 2011c , hakupäivä 16.2.2011).

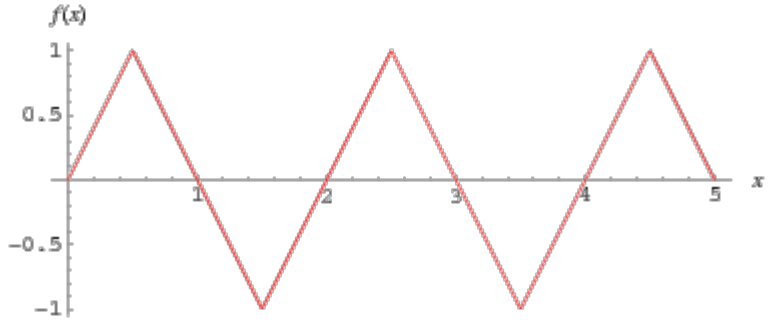
Sinimuotoisessa ääniaallossa kuvion 7 funktiossa muuttuja *x* vastaa aikaa sekunneissa ja kulmataajuus *ω* voidaan korvata *2π·f:llä*, missä *f* on äänen taajuus. Amplitudi *α* kuvaa äänen voimakkuutta ja *c* vaihde-eroa (Weisstein 2011c, hakupäivä 16.2.2011). Sijoittamalla siis haluttu äänen taajuus, voidaan aaltoa piirtää ajan funktiona siten, että funktion arvo kullekin ajan yksikölle syötetään soittimen puskuriin. Arvoista muodostuu kuvion 6 mukainen aaltokuvio siirtämällä ne *xy*-koordinaatistoon. Kuvioissa 8–13 on esiteltynä muita aaltokuvioita, jotka muunnetaan ääneksi vastaavalla tavalla siniaallon kanssa. Eri aaltomuotojen tuottama ääni on sävyltään erilaista. Esimerkiksi siniaalto kuulostaa tasaiselta ja pehmeältä verrattuna kantti- tai sahahammasaaltoon.



KUVIO 8. Kanttiaalto (Weisstein 2011d, hakupäivä 16.2.2011).

$$S(x) = A \operatorname{sgn}\left[\sin\left(\frac{2\pi(x - x_0)}{T}\right)\right]$$

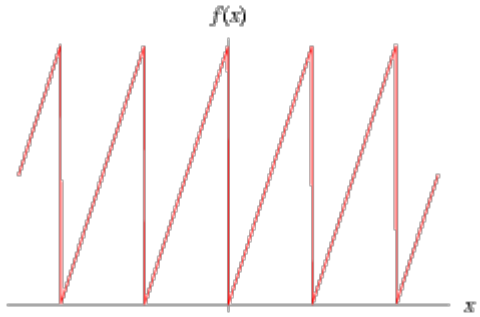
KUVIO 9. Kanttiaallon funktio (Weisstein 2011d, hakupäivä 16.2.2011) .



KUVIO 10. Kolmioaalto (Weisstein 2011e, hakupäivä 16.2.2011).

$$f(x) = \frac{2}{\pi} \sin^{-1}[\sin(\pi x)]$$

KUVIO 11. Kolmioaallon funktio (Weisstein 2011e, hakupäivä 16.2.2011) .



KUVIO 12. Sahahammasaalto (Weisstein 2011a, hakupäivä 16.2.2011).

$$S(x) = A \operatorname{frac}\left(\frac{x}{T} + \phi\right), \operatorname{frac}(x) \equiv x - \lfloor x \rfloor$$

KUVIO 13. Sahahammasaallon funktio (Weisstein 2011a, hakupäivä 16.2.2011).

Syntetisoidun äänen tuottamisen etuja sovelluksen kannalta on sen dynaamisuus. Kun aalto rakennetaan sovelluksessa iteroimalla, ei puhelimeen tarvitse erikseen sovelluksen mukana tuoda ääninäytteitä. Sitomalla ääniaallon taajuus sormen kosketuksen koordinaatteihin, voidaan sulavasti rakentaa ääniaalto vastaamaan mitä tahansa taajuutta. Usein aidon soittimen muodostama ääni koostuu lukuisista siniaalloista, jotka limittäin muodostavat soittimelle ominaisen äänen. Toki tällainen mallintaminen on mahdollista myös syntetisoimalla yksittäiset siniaallot, mutta se on samalla vaativa työ sovelluksen kehittäjälle. Tällöin on käytännöllisempää käyttää esimerkiksi SoundPool-luokkaa näytteen tuomiseksi sovellukseen, ja käyttää soittonopeutta sävelen muuntamiseksi.

android.media.MediaPlayer-luokka on tarkoitettu puhelimen ääni- ja videotiedostojen toistoon. MediaPlayer-luokka sisältää tarvittavat purkukodekit kaikkien yleisimpien tiedostomuotojen toistamiseen. AudioTrack- ja SoundPool-luokista poiketen tämä luokka on tarkoitettu nimenomaan pitkien tiedostojen kuten kokonaisten kappaleiden tai videotiedostojen toistoon (Android Developers 2011m, hakupäivä 11.3.2011). Musiikkisovellukseen MediaPlayer-luokkaa voisi käyttää esimerkiksi taustamusiikin toistamiseen tai yksittäisten pitkien ääninäytteiden toistamiseen.

android.media.ToneGenerator-luokka on tarkoitettu standardisoitujen merkkiäänien toistamiseen. Tällaisia ääniä ovat muun muassa puhelimen eri näppäinten äänet niitä painattaessa ja esimerkiksi *puhelin on varattu* -äänimerkki. Äänten soittamiseen luokka antaa vain muutamia metodeja, joilla voi pääasiassa säätää äänen toistamisen kestoa (Android Developers 2011p, hakupäivä 11.3.2011). Tämä luokka ei sovellu kovin hyvin musiikkisovelluksen käyttöön sen rajoittuneiden äänenhallintamahdollisuuksien takia. Ainoa tapa hyödyntää luokkaa musiikkisovelluksessa on käyttää sitä standardisoitujen merkkiäänien lainaamisessa sovelluksen käyttöön. Tällöin merkkiäänet toimisivat enemmänkin ääniefekteinä.

android.media.MediaRecorder-luokka on tarkoitettu äänen ja videon nauhoittamiseen. Ääntä voi nauhoittaa, joko mikrofonista tai puhelusta. Ääntä ei kuitenkaan pysty nauhoittamaan kaiuttimista tai äänen uloslähtökanavasta. Tämä tarkoittaa sitä, että ääntä ei voi kaapata esimerkiksi muista sovelluksista. Musiikkisovellusten kannalta tämä ongelmallista siksi, että käyttäjä ei voi tallentaa omaa soittamistaan (Android Developers 2011n, hakupäivä 11.3.2011).

MediaRecorder-luokka muuntaa automaattisesti nauhoitetun äänen muutamasta vaihtoehdosta valittavana olevaan ääniformaattiin. Tämä toki helpottaa äänen uudelleen käyttöä sen ollessa valmiiksi formaatissa, mitä voidaan toistaa yleisesti muissa laitteissa. Haittapuolena on se, että muuntaminen nauhoituksen aikana vie paljon resursseja ja voi näin vaikuttaa sovelluksen muiden osien suorituskykyyn. Nauhoitetun äänen laatua ei myöskään voi säätää tällä luokalla ja oletusäänenlaatu on aika heikko (Android Developers 2011n, hakupäivä 11.3.2011). Täten tehokkuutta tai äänen laadun säätämistä tarvitseviin musiikkisovelluksiin tämä luokka ei sovi.

android.media.AudioRecord-luokka antaa enemmän mahdollisuuksia nauhoitusominaisuuksien muokkaamiseen kuin MediaRecorder-luokka. Samalla se vaatii tarkempaa käyttöä sovelluksessa. AudioRecord-luokka tallentaa ääntä PCM-muotoisena, mikä antaa enemmän mahdollisuuksia äänen käsittelyyn. Myös tallentavan äänen laatua voi säätää. Kuten AudioTrack-luokassa, pitää

ääniä lukea ensin puskuriin ennen sen tallentamista. Puskurin koon määrittelemisessä täytyy olla tarkkana, ettei ääninäytteitä katoa nauhoituksen yhteydessä. Puskurin kokoon vaikuttaa nauhoitettavan äänen laatu sekä media, johon tiedosto kirjoitetaan. AudioRecord-luokka voi nauhoittaa ääntä, joko mikrofonista tai puhelusta. Tämä tarkoittaa sitä, että ääntä ei voi kaapata muista sovelluksista (Android Developers 2011e, hakupäivä 11.3.2011).

Teoriassa AudioRecord- ja AudioTrack-luokat toimivat hyvin yhdessä. AudioTrack-luokan puskuria täytettäessä voidaan samaa tietoa syöttää myös AudioRecord-puskuriin. Tässä on kuitenkin huomioitava se, että AudioRecord-luokasta ei löydy dynaamista puskurin täyttömahdollisuutta, joka tarkoittaa sitä, että AudioTrack-luokassa täytyy myös käyttää staattista puskuria. Ongelmaksi saattaa muodostua nauhoitus- ja toisto-operaatioiden synkronisointi. Ohjelmoinnissa on huomioitava se vaihtoehto, että AudioRecord-luokka ei välttämättä pysty kirjoittamaan puskuria yhtä nopeasti kuin AudioTrack-luokka toistaa omaa puskuriansa.

android.media.JetPlayer-luokka on suunniteltu pääasiassa pelien käyttöön. JetPlayer-luokkaa varten tuotavat äänet on pakattava erillisellä JetCreator-ohjelmalla omaksi Jet- tiedostoksi. Kaikki tarvittavat ääninäytteet on ennalta määriteltävä, sillä JetPlayer-luokka ei pysty lataamaan normaaleja äänitiedostoja suoraan. JetPlayer tukee osittain MIDI-rajapintaa (Musical Instrument Digital Interface), jonka avulla voidaan hallita Jet-tiedoston sisältämiä ääninäytteitä. Osittainen tuki tarkoittaa sitä, että MIDI-tiedostot on luotava Jet-tiedoston luonnin yhteydessä. Ei esimerkiksi ole mahdollista dynaamisesti hallita Jet-tiedoston sisältämiä ääninäytteitä JetPlayer-luokan kautta ulkopuolisilla MIDI-komennoilla (Android Developers 2011h, hakupäivä 11.3.2011).

JetPlayer-luokan eräs hyöty on se, että muutamasta ääninäytteestä voidaan yhdistelemällä luoda useita eri kappaleita. Jos esimerkiksi Jet-tiedosto sisältää tarvittavat instrumentit ääninäytteinä, voidaan niistä koostaa kappaleita MIDI-tiedostoilla. Toki MIDI-tiedostot pitää sisällyttää mukaan Jet-tiedostoon, mutta MIDI-tiedostot itsessään eivät paljoa kasvata Jet-tiedoston kokoa. Näin esimerkiksi albumin verran kappaleita voidaan soittaa samoista ääninäytteistä. JetPlayer-luokka antaa mahdollisuuden transponoida tai mykistää soitettavan kappaleen (MIDI-tiedoston) sisäisiä osia tai instrumentteja. Tämä lisää mahdollisuuksia yksittäisen kappaleen käyttämiseen eri tavoin (Android Developers 2011h, hakupäivä 11.3.2011).

Kokonaisuuden hallintaa helpottavat JetPlayer-luokan tuomat MIDI-toiminnot sovellusten käyttöön sekä äänitiedostojen pakkaaminen yhdeksi tiedostoksi. Suurenkin määrän kappaleita voi

luoda samasta JetPlayer-luokan ilmentymästä, mikä varmasti tekee sovelluksesta kevyemmän verrattuna esimerkiksi lukuisten SoundPool-luokan ilmentymien käyttöön (Android Developers 2011h, hakupäivä 11.3.2011).

android.media.AudioManager-luokka on osa puhelimen järjestelmän palveluja. Sen avulla voidaan ohjata ja lukea puhelimen laitteiston ääniominaisuuksia. Tämä luokka alustetaan automaattisesti puhelimen käynnistyttyä ja ainoastaan sen instanssi voidaan hakea sovelluksen käyttöön. Kyseessä on niin sanottu staattinen luokka (Android Developers 2011c, hakupäivä 11.3.2011). Musiikkisovelluksen kannalta se on olennainen, koska sen avulla voidaan hakea tietoja laitteiston ääniominaisuuksista, ja siten muokata sovelluksen käyttäytymistä erilaisissa laitteistoympäristöissä. Lisäksi luokan avulla voi ohjata joitain avaintoimintoja kuten äänen voimakkuuden säätöä, mikrofonien ja kaiuttimien päälle kytkemistä sekä yksittäisten instanssien hallintaa, jos useampi instanssi soittaa ääntä samanaikaisesti. Kuviossa 14 on esitellään kokoelma metodeja, joilla voi säätää tai lukea laitteiston ominaisuuksia.

adjustStreamVolume(int streamType, int direction, int flags) Voi säätää yksittäisten ääniraitojen voimakkuutta. Suunta tarkoittaa sitä, että lasketaanko äänen voimakkuutta vai nostetaanko. Laitteen laajuinen äänen säätö. Viimeisellä parametrillla määritellään, miten voimakkuuden säätö ilmoitetaan käyttäjälle.
getParameters(String keys) Haetaan puhelimen äänilaitteiston ominaisuudet.
isMicrophoneMute() Tarkistetaan onko mikrofoni mykistetty.
isMusicActive() Tarkistetaan onko jokin ääniraita aktiivinen puhelimessa.
isSpeakerphoneOn() Tarkistetaan onko kaiutin on käytössä.
setMicrophoneMute(boolean isMute) Asetetaan mikrofoni mykäksi.
setSpeakerphoneOn(boolean isOn) Asetetaan puhelimen kaiuttimet päälle tai pois päältä.
setStreamMute(int StreamType, boolean state) Mykistetään yksittäinen ääniraita.
setStreamSolo(int streamType, boolean state) Asetetaan yksittäiselle ääniraidalle soolo–tila. Tällöin muut raidat mykistetään.

KUVIO 14. Kokoelma hyödyllisiä AudioManager-luokan metodeja (Android Developers 2011c, hakupäivä 11.3.2011).

Gingerbread eli Android 2.3 -käyttöjärjestelmä tuo mukanaan uudistuksia äänenkäsittelyyn. Gingerbreadin myötä tulee kokonaan uusi luokkakokoelma audiofx, joka sisältää muun muassa EnviromentalReverb-, BassBoost- sekä Equalizer-luokat. Equalizer-luokkalla voi luoda *taajuuskorjaimen*, jolla voi muokata äänen sävyä. EnviromentalReverb-luokalla voi luoda kaikuefektin äänelle. BassBoost-luokkalla voi helposti syventää matalien äänitaajuuksien toistoa (Android Developers 2011b, hakupäivä 11.3.2011). BassBoost-luokkaa voi verrata usein stereolaitteissa tai kannettavissa soittimista löytyvään vastaavan nimiseen toimintoon.

4 MUSICBOX

MusicBox on musiikkiohjelma, joka koostuu neljästä alasovelluksesta: Drum Workshop, Synthesizer, Piano ja Voice Scratch. Kussakin näistä sovelluksista on pyritty kokeilemaan eri tapoja käyttää Android-käyttöjärjestelmän tarjoamia äänen hallintaan erikoistuneita luokkia. Alkuperäinen tarkoitus oli luoda yksi sovellus, jossa äänen hallinta tapahtuu kaksiulotteisesti kosketuksen paikan mukaan. Kehityksen aikana ilmenneiden ongelmien ja haasteiden takia päädyttiin kuitenkin ratkaisuun kokeilla erityyppisiä sovelluksia ja koostaa niistä musiikkiohjelma.

4.1 Sovelluksien esitteleminen

Ohjelma avautuu päänäkymään, jossa eri sovellukset ovat listattuna. Haluttu sovellus käynnistetään painamalle sen nimeä listasta. Sovelluksesta poistuttaessa palataan takaisin päänäkymään Android-älypuhelimien navigointinäppäinten avulla. Tällöin sovelluksella tehty musiikki tai grafiikka ei tallennu lukuun ottamatta Voice Scratch -sovellusta, jossa käyttäjällä on mahdollisuus tallentaa nauhoitettu ääni pysyvästi älypuhelimen muistikortille.

Jokainen sovellus on itsenäisesti toimiva kokonaisuus. Kussakin sovelluksessa on näkyvissä otsikkopalkki, jossa näkyy MusicBox-ohjelman nimi ja logo sekä valittuna olevan sovelluksen nimi ja logo. Jaottelemalla ohjelma erillisiin sovelluksiin, pyrittiin säästämään ohjelman muistinkäyttöä sekä helpottamaan sovelluskehitystyötä. Kuviossa 15 esitellään ohjelman päänäkymä.

KUVIO 15. Music Box -ohjelman aloitusnäkymä.

Drum Workshop -sovellus mallintaa rumpukonetta. Drum Workshopissa kuvaruutu on jaettu neljään samankokoiseen alueeseen, joissa jokaisessa on eri rumpuääni: bassorumpu, hihat, lehmänkello ja virveli. Käyttäjän koskettaessa esimerkiksi bassorumpualuetta, bassorumpu soi kerran ja alueen taustaväri vaihtuu. Kosketuksen loputtua alueen taustaväri vaihtuu takaisin alkuperäiseen. Sovellus tukee useamman sormen yhtäaikaista kosketusta. Kuviossa 16 näkyy sovelluksen päänäkymä. Drum Workshop -sovellus käyttää SoundPool-luokkaa ääniraitojen toistamiseen. Jokainen kosketus aloittaa valitun ääniraidan toistamisen alusta, mikä mahdollistaa nopeiden lyhytaikaisten kosketuksien sarjan.

KUVIO 16. Drum Workshop -sovelluksen päänäkymä.

Synthesizer -sovellus mallintaa klassista analogista syntetisaattoria. Sovelluksen avautuessa käyttäjä valitsee aaltotyypin kolmesta vaihtoehdosta: sini-, neliö tai kanttiaalto. Valinnan jälkeen avautuu soittotila. Soittotilassa käyttäjän koskettaessa kuvaruutua piirtyy kosketuksen rata kuvaruudulle. Sovellus laskee kosketuksen etäisyyden näytön keskipisteeseen joka kerta kun sormi osuu uuteen pikseliin. Tämä etäisyys toimii muuttujana soitettavan äänen taajuudelle. Kuviossa 18 näkyy, kuinka sovelluksen ääntä on ohjattu piirtämällä.

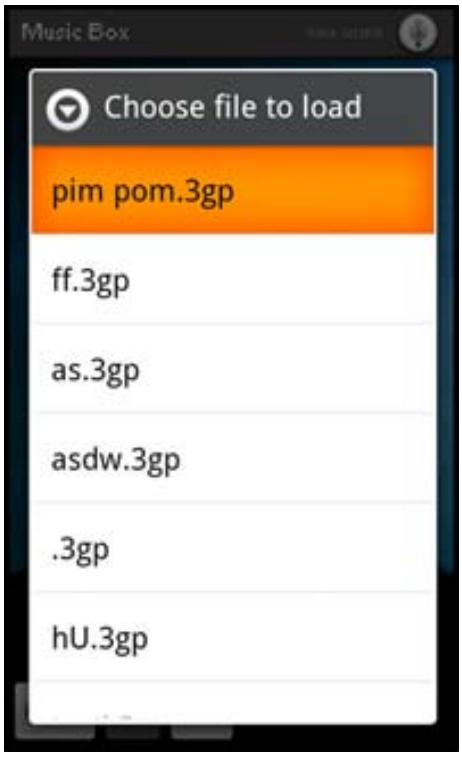
Sovelluksessa kuultavat äänet on rakennettu AudioTrack-luokkaa käyttämällä. Kukin aaltokuvio luodaan siten, että ainoana muuttujana toimii äänen taajuus. Matemaattisesti luotu aaltokuvio syötetään AudioTrack-luokan puskuriin ja se muuntaa saadun tiedon ääneksi. Kuviossa 17 esitellään sovelluksen aloitusvalikko, jossa käyttäjä voi valita käytettävän aaltotyypin. Kukin aaltotyyppi luo erilaisen äänen sävyn.

KUVIO 17. Synthesizer-sovelluksen aloitusnäkymä. Käyttäjä valitsee aaltotyypin soittimelle.

Voice Scratch -sovelluksessa käyttäjä voi nauhoittaa puhelimen mikrofonin kaappaamaa ääntä ja muokata sitä vastaavalla periaatteella kuin Synthetizer-sovelluksessa. Erona Synthesizer-sovellukseen on se, että kosketuksen koordinaatit muuttavat soitettavan äänitiedoston soittonopeutta. Näyttöalueen keskipisteessä käytettävän äänitiedoston soittonopeus on hidastettuna puoleen alkuperäisestä. Näyttöalueen kulmissa soittonopeus on kaksinkertainen alkuperäiseen verrattuna. Sovellus tavallaan matkii vinyylilevyn soittonopeuden muuntamista, *skrätsäystä*. Sovelluksen soittonäkymä on kuvion 18 mukainen.

KUVIO 18. Synthesizer-sovelluksen päänäkymä. Käyttäjä voi soittaa piirtämällä kuvioita.

Sovelluksen avautuessa käyttäjältä kysytään, ladataanko aiemmin nauhoitettu tiedosto tai jokin muu äänitiedosto, vai nauhoitetaanko uusi tiedosto. Mikäli käyttäjä päättää ladata aiemmin luodun tiedoston, sovellus valikoi automaattisesti ladattaviksi vaihtoehdoiksi vain sopivaa tiedostomuotoa olevat äänitiedostot. Kuvio 19 esittää näkymää, jossa käyttäjä valitsee ladattavan äänitiedoston.



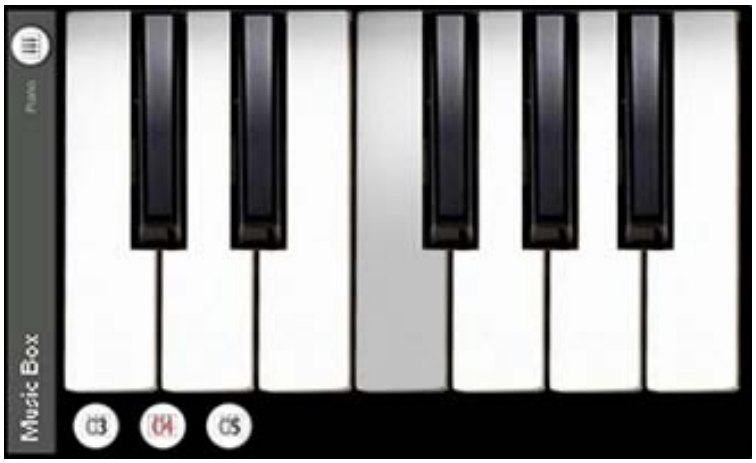
KUVIO 19. Voice Scratch -sovelluksessa käyttäjä voi ladata aiemmin nauhoitettuja ääninäytteitä.

Käyttäjä voi myös halutessaan nauhoittaa uuden äänen valitsemallansa tiedostonimellä. Tällöin avautuu kuvion 20 mukainen näkymä, jossa käyttäjä valitsee nauhoitettavalle tiedostolle nimen. Käyttäjä ei erikseen määritä tiedostopäätettä. Päänäkymästä löytyy myös record-, play- sekä load-napit. Record-nappi käynnistää nauhoituksen, jos käyttäjä on ilmoittanut tiedoston nimen. Muussa tapauksessa sovellus aukaisee kuvion 20 näkymän, jossa käyttäjältä kysytään tiedosto nimeä. Record-napin toiminta muuttuu stop-toiminnoksi nauhoituksen alkaessa. Play-nappi käynnistää äänen toistamisen sekä sovelluksen varsinaisen käyttötilan, jos käyttäjä on joko ladannut äänitiedoston tai suorittanut nauhoituksen loppuun. Play-napin toiminta muuttuu stop-toiminnoksi toistamisen alkaessa. Sovelluksessa käytetään SoundPool- sekä MediaRecorder-luokkia. MediaRecorder-luokalla nauhoitetaan ääntä mikrofonista sekä muunnetaan ääniformaatti sopivaksi. SoundPool-luokalla toistetaan nauhoitettu tai tiedostosta ladattu ääniraita ja muokataan sen soittonopeutta kosketuksen mukaan.

KUVIO 20. Näkymä, jossa käyttäjä kirjoittaa nauhoitettavan tiedoston nimen.

Piano-sovelluksessa avautuu käyttäjälle yksioktaavinen pianokoskettimisto. Poiketen muista ohjelman sovelluksista, Piano-sovellus on suunniteltu käytettävän vaakasuorassa. Sovelluksessa on käytetty SoundPool-luokkaa, johon perussävelet (C4-H4) ovat ladattuna äänitiedostoista. Ylennykset (C#, D#, F#, G# ja A#) on saatu muokkaamalla raidan soittonopeutta. Piano-sovelluksessa on mahdollisuus myös siirtyä oktaavia ylemmäksi tai alemmaksi. Tämä tapahtuu kuvaruudun vasemmassa alalaidassa olevilla ohjainnapeilla. Näiden oktaavien äänet on niin ikään saatu aikaiseksi muuntamalla perussävelien soittonopeutta.

Piano soittaa ääntä riippuen kosketuksen pituudesta ja tukee useamman sormen kosketusta. Kosketinta painettaessa myös painetun koskettimen kuva vaihtuu. Kuviossa 21 kuvataan tapahtuma, jossa käyttäjä on painanut pohjaan *F*-koskettimen *C4*-oktaavista.



KUVIO 21. Piano-sovelluksen päänäkymä. Kuvassa pianolle on valittuna C4-oktaavi ja F-kosketin on painettuna pohjaan.

4.2 Android-käyttöjärjestelmän soveltuvuus musiikin tuottamiseen

Tämän työn sovelluksia rakennettaessa on törmätty useisiin kehitysongelmiin. Ehkä suurimpana ongelmien aiheuttajana on ollut SDK:n tarjoama älypuhelinemulaattori, jolla kehitettävää sovellusta voi testata. Emulaattorista puuttuu useita älypuhelinten fyysisiä ominaisuuksia mallintavia elementtejä. Tämän työn sovellusten kannalta oleellisin puute on usean sormen kosketuksen mallinnuksen puuttuminen. Toinen suuri puute on pysyvien tiedostojen tallentaminen ja niiden hallitseminen. Käytännössä se on suoritettava virtuaaliseen muistilaajennuskorttiin, jota ei välttämättä löydy kaikista markkinoilla olevista puhelimista.

Testaamista ja virheenjäljittämistä (debuggausta) vaikeutti emulaattorin hidas käynnistyminen. Kehittämistä vaikeutti myös emulaattorin käyttämät pc-resurssit. Normaalia raskaamman grafiikan piirtämisessä emulaattorin ruudunpäivitysnopeus laski jopa alle viiteen ruutuun sekunnissa. Internetistä ei löytynyt virallista dokumentointia siitä, kuinka paljon hitaammin emulaattorissa käynnistetyt sovellukset pyörivät oikeisiin laitteisiin verrattuna. Arvio kehittäjien keskuudessa vaikutti kuitenkin olevan, että emulaattori kärsii hitausongelmista oikeaan älypuhelimeen verrattuna. Emulaattorin hitausongelmat ilmenivät myös ääniraitojen toiston ja graafisen piirtämisen tapahtuessa samanaikaisesti. Tällöin ääniraitojen latenssi kasvoi ja usein kaiuttimista kuului *rätinää*.

Latenssi on kaikissa musiikkiohjelmistoissa merkittävä ongelmien aiheuttaja. Latenssilla tarkoitetaan tämän tässä työssä aikaviivettä, joka on käyttäjän kosketuksen ja kaiuttimista kuuluvan ää-

nen välinen aika. Käyttäjän ei ole mielekästä naputella rumpukonetta, jos jokainen ääni kuuluu viiveellä. Latenssista on mahdoton päästä täysin eroon. Sen pituus riippuu paljolti siitä laitteistosta, jolla ääntä toistetaan. Sovelluskehityksen kannalta on vaikeaa ennakoida latenssiin suuruutta, sillä pelkkä puhelimen muistin koko tai suorittimen teho eivät välttämättä ilmaise riittävän tarkasti puhelimen suorituskykyä. Suorituskykyyn vaikuttaa muun muassa se, kuinka paljon taustasovelluksia on yhtäaikaisesti käytössä musiikkisovelluksen kanssa. Huomasimme myös, että jos latenssille herkän toimenpiteen aikana tapahtui paljon graafista piirtämistä, vaikutti se lisäävän merkittävästi latenssin suuruutta.

Eräs tapa pienentää osittain latenssia, oli sijoittaa latenssiherkät toiminnot oman ohjelmasäikeen (*Thread*) suoritettavaksi ja lisäksi priorisoimalla suoritettavia säikeitä. Tällöin erilaiset tapahtumakäsittelijät eivät pystyneet keskeyttämään tai aiheuttamaan katkoksia sovellusrutiinien suoritukseen. Tämä oli tarpeen erityisesti Synthetizer-sovelluksessa, jossa AudioTrack-luokkaan puskuriin syötettään niin sanotulla ikuisesti jatkuvalla silmukalla arvoja.

Yksittäisten luokkien käytössä ilmeni myös ongelmia, joiden takia osissa sovelluksissa päädyttiin hylkäämään jo tehtyjä ratkaisuja. MediaPlayer-luokan käytössä muun muassa ilmeni ongelma, jossa liian lyhyiden äänitiedostojen soittaminen aiheutti rätinää äänen toiston yhteydessä tai ääntä ei kuulunut laisinkaan. Toinen ongelma MediaPlayer-luokassa oli äänen pysäyttäminen (stop-toiminto). Jos ääniraita pysäytettiin ennen sen loppuun soittamista, ei soitin nollannut ääniraitaa. Tämä tarkoittaa sitä, että jos ääniraita käynnistetään pysäytyksen jälkeen uudestaan, jatkaa se toistoa siitä kohdasta, johon se oli pysäytetty. Normaalisti tämä toiminto tapahtuu musiikkisoittimissa pause-komennolla. Varsinainen ongelma on kuitenkin se, että MediaPlayer-luokasta ei löydy takaisinkelausominaisuutta. Ääntä uudelleen soitettaessa on mahdollisuus asettaa toiston alkamiskohta millisekunneissa, mutta tämä ei kuitenkaan toiminut lyhyiden ääninäytteiden kohdalla. Ilmeisesti kyseessä on jokin virhe. Esimerkiksi Drum Workshopissa käytimme tästä syystä SoundPool-luokkaa MediaPlayer-luokan sijasta.

SoundPool-luokassa taas vastaavasti ongelmia alkoi ilmetä yli kahden sekunnin mittaisissa ääniradoissa. Nämä eivät vaikuttaneet olevan riippuvaisia äänitiedoston koosta tai äänen laadusta. Ääniraidat alkoivat tietyn ajan jälkeen pätkiä tai ne rätisivät. Ei ole aivan varmaa, johtuivatko nämä mahdollisesti emulaattorista vai ilmenisivätkö ne myös puhelimissa. SoundPool-luokan olioon pystyi lataamaan useita ääniraitoja. SetRate-metodilla tehty soittonopeuden muutos vaikuttaa kaikkiin luokan olion sisältämiin ääniraitoihin. Tämä voi olla virhe luokassa, sillä setRate-metodiin

on kuitenkin määriteltävä olion sisältämän ääniraidan tunniste. Lisäksi jos kaksi tai useampi luokkaan ladatuista ääniraidoista soi samanaikaisesti setRate-metodin muokatessa yhtä niistä, aiheutti se toisissa ääniradoissa rätinää.

Android-käyttöjärjestelmän äänіominaisuuksissa ei ole mahdollisuutta nauhoittaa ääntä kaiuttimista tai linjauloslähdöstä. Tämä on olennainen puute, jolla on vaikutusta kaiken tyyppisiin musiikkisovelluksiin. Nauhoittamisen voi joissain tapauksista järjestää tallentamalla käyttäjän tekemät toiminnot, kuten kosketuksen alkamis- ja päättymiskohdat sekä näiden aikamerkit. Tallentaminen voidaan suorittaa tekstitiedostoon. Ääntä toistettaessa ajetaan tiedostossa olevat kosketustiedot sovelluksen läpi samaan tapaan kuin ne olisivat käyttäjän tekemiä kosketuksia. Työssä kokeiltiin tällaista nauhoitustyyliä Drum Workshop -sovelluksen yhteydessä siinä kuitenkaan onnistumatta. Ongelmaksi muodostui Java-ohjelmointikielestä periytyvä Timer-luokka, joka ei suorituksen aikana kyennyt pysymään ajassa. Todennäköisesti kyseessä on emulaattorin resurssiongelmat.

JetPlayer-luokka oli ainoa jota ei kokeiltu työn yhteydessä. JetPlayer-luokan käyttö edellyttää käytettävän äänitiedoston kokoamista JetCreator-ohjelmalla, joka taas vaatii toimiakseen Python-ympäristön asennuksen. Pythonin asennuspaketista puuttui Windows 7 -käyttöjärjestelmään kuuluvia dll-tiedostoja, minkä takia ympäristön asennus ei onnistunut työasemaan, joilla musiikkiohjelmaa toteutettiin.

5 POHDINTA

Android-käyttöjärjestelmä on jatkuvien muutostarpeiden edessä. Muutostarpeet eivät aina tarkoita sitä, että olisi tehty välttämättä vääriä ratkaisuja tai virheitä aikaisemmissa kehitysvaiheissa. Enemmänkin tarpeet johtuvat koko älypuhelinteollisuuden osittain hallitsemattomasta kasvusta. Älypuhelimiin lisätään teknisiä ominaisuuksia melkein jokaisessa uudessa mallissa, mikä tuo paineita käyttöjärjestelmälle pysyä näiden muutosten perässä. Vaikka Android-käyttöjärjestelmän tapauksessa yhteistä linjaa yritetään luoda OHA:n avulla, on ymmärrettävä, että sen jäsenet ovat kuitenkin usein toistensa kilpailijoita. Eräs tapa erottua kilpailijoistaan on nimenomaan kehittää puhelimeen jokin uusi tekninen ominaisuus, jota kilpailijoilta ei löydy. Tämä on toisaalta järjestelmää eteenpäin ajava voima. Toisaalta se taas vaikeuttaa kokonaisuuden hallintaa.

Sovelluskehittäjän näkökulmasta tilanne tuntuu ajoittain vaikealta. Eri versioiden välillä on joko poistettu tai ilmoitettu poistettavaksi osia käytössä olevista luokista, ja toisaalta osa viimeksi lisätyistä osista ei ole täysin valmiita. Musiikkisovelluksen kannalta tuki on vaihteleva. Dokumenteista ja luokkien käyttöä kuvaavista esimerkeistä jää tunne, että pääasiallisesti äänio minaisuudet on tehty pelien sekä perustoimintojen kuten video- ja musiikkisoittimien käyttöön. Dokumentointi on yleisesti ottaen hyvin vähäistä, ja iso osa kehitysongelmista jää sovelluskehittäjien kokeilujen varaan. Toki internetistä löytyy lukuisia blogeja ja verkkosivuja, joissa käyttäjät kertovat kokemuksistaan ja antavat apua sitä kyseleville. Näissä verkkosivuissa ilmenneet ratkaisut jäävät kuitenkin useilta osin vajavaisiksi ainakin äänio minaisuuksien osalta.

Tärkeimpänä tiedonlähteenä on käytetty Android Developers -verkkosivustoa. Sivusto on virallinen portaali Android-sovellusten kehittäjille ja sijaitsee android.com-domainissa. Sivustolla julkaistavaa materiaalia voidaan pitää kenties luotettavimpana lähteenä ainakin tässä työssä käytetyn tietoperustan kannalta. Raportissa on kuitenkin varsinaisen sovelluksen kehittämisen lisäksi pyritty perehtymään älypuhelinmarkkinoihin sekä muihin käyttöjärjestelmiin. Luotettavien lähteiden löytäminen muun muassa markkinaosuuksien hahmottamisessa osoittautui erittäin haastavaksi. Kyse on paljolti ennustuksista, joita on yhtä monta kuin niiden tekijää. Lisäksi ennustuksien pohjalla olevat tilastolliset tutkimuksen poikkeavat toisistaan usein huomattavasti. Toki on huomioitava, että tilastot perustuvat laitteistoja myyvien yhtiöiden ilmoittaviin myyntilukuihin, ja on luonnollista olettaa, että jotain virheitä voi ilmoituksissa esiintyä. Raportissa esitetyt luvut markkinaosuuksista on syytä ymmärtää suuntaa-antavina.

Opinnäytetyössä alkuperäisenä suunnitelmana oli tuottaa yksi sovellus, joka käyttää näytön reunoja koordinaatistona. Sovelluksessa ääntä oli tarkoitus muokata riippuen kosketuksen x- ja y-koordinaateista näytöllä. Jos toisena parametrina (*x*-kordinaatti) olisi käytetty äänen korkeutta, olisi toinen parametri (*y*-koordinaatti) muokannut joko äänen sävyä tai jonkin ääniefektin määrää. Android 2.2 -käyttöjärjestelmän äänen käsittelyyn erikoistuneet luokat eivät kuitenkaan tarjonneet mielekästä tapaa muokata ääntä kahdella parametrilla. Päädyimme ratkaisuun, jossa reunojen sijasta käytetään etäisyyttä näytön keskipisteestä kosketuskohtaan. Näin saatiin sidottua molemmat *x*- ja *y*-koordinaatit yhden etäisyys-parametrin alle. Toisaalta työn tarkoituksena on myös tutkia Android-käyttöjärjestelmän musiikkiominaisuuksia. Yksi sovellus ei kuitenkaan vaikuttanut riittävän kaikkien näiden ominaisuuksien tutkimiseen. Tästä syystä päädyimme rakentamaan neljä pienempää sovellusta, joissa pyrimme käyttämään mahdollisimman kattavasti Android-käyttöjärjestelmän tarjoamia luokkia.

Neljästä osasovelluksesta oli tarkoitus muodostaa yksi ohjelmistokokonaisuus, MusicBox-musiikkiohjelma. Ohjelmassa käyttäjällä olisi mahdollisuus nauhoittaa yksittäisten sovellusten tuotokset ja siirtää ne käytettäväksi muihin sovelluksiin. Lopputuloksena syntyisi kokonaisuus, joka koostuisi Drum Workshopilla tehdystä rumpuraidasta, Pianolla ja Synthesizerilla tehdyistä melodioista sekä Voice Scratchilla tehdystä lauluraidasta. Tätä ei kuitenkaan onnistuttu saavuttamaan siitä syystä, että Android-käyttöjärjestelmässä ei ole mahdollista nauhoittaa ääntä uloslähtö-kanavasta tai kaiuttimesta. Muita tallennusvaihtoehtoja myös kokeiltiin niissä onnistumatta. Tässä mielessä toiminnallisessa osuudessa epäonnistuimme saavuttamaan asetetut tavoitteet. Toisaalta onnistuimme selvittämään epäonnistumisen syyt, mikä sinänsä on onnistumista työn päätavoitteessa eli Android-käyttöjärjestelmän musiikkiominaisuuksien tutkimisessa.

Suurin osa työn aikana vastaan tulleista ongelmista ja haasteista ovat johtuneet dokumentoinnin puutteesta, työn kehityksessä käytetystä laitteistosta tai Android-käyttöjärjestelmän ominaisuuksista. Tämä ei kuitenkaan tarkoita sitä, että jatkossa ongelmia ja haasteita ei voisi ratkaista. Esimerkiksi kohta julkaistavassa Android 2.3 -käyttöjärjestelmään on lisätty mahdollisuus vaikuttaa äänen sävyyn Equalizer-luokalla ja mukaan on lisätty muun muassa kaikuefekti (Reverb-luokka). Pelkästään näiden kahden luokan avulla voidaan lisätä kaivattu toinen parametri alkuperäisen suunnitelman mukaiseen Kaossilatoria mallintavaan sovellukseen. On myös mahdollista, että tulevaisuudessa Android-käyttöjärjestelmän ominaisuuksiin on lisätty uloslähtö-kanavasta tai kaiuttimesta nauhoittamisen toiminto. Tämä taas monipuolistaisi MusicBox-ohjelma toiminnallisuutta. Vaikka kaikki mainitut kehityshaasteet saataisiin ratkaistua, on MusicBox-ohjelmaa kui-

tenkin pystyttävä testaamaan oikeilla älypuhelimilla. Tätä opinnäytetyötä tehdessä ei ole ollut mahdollisuutta testata ohjelman toimivuutta muuta kuin sovelluskehittimen tarjoamalla älypuhelin-emulaattorilla. Kyseinen seikka on ensisijainen este tämän hetkisen ohjelman julkaisemiselle.

Työ on ollut haastava mutta antoisa. Kun aloitimme työn, aikaisempaa osaamista Android-käyttöjärjestelmälle kehittämisestä ei ollut. Olemme etsineet ja asentaneet kehitystyökaluja, tutustuneet tietoperustaan sekä löytäneet ja soveltaneet lähteitä työn aikana. Kaikki on aloitettu tyhjältä pöydältä. Opinnäytetyön työstämiseen kului 8 kuukautta, ja vaikka ajoittain kehityksen aikana ilmenneet ongelmat ovat hidastaneet työn valmistumista, on ollut palkitsevaa löytää ratkaisuja näiden ongelmien ohittamiseksi. Android-käyttöjärjestelmä on kaiken kaikkiaan sovelluskehittäjälle helposti lähestyttävä. Tämä yhdistettynä kuluttajien keskuudessa saavuttamaan Android-älypuhelimien suosioon on varmasti resepti pitkäaikaiseen menestykseen. Opinnäytetyömme toimii näytteenä osaamisesta alan töitä haettaessa.

LÄHTEET

Android Developers. 2011a. ADT Plugin for Eclipse. Hakupäivä 1.4.2011.
http://developer.android.com/sdk/eclipse-adt.html.

Android Developers. 2011b. android.media.audiofx. Hakupäivä 11.3.2011.
http://developer.android.com/reference/android/media/audiofx/package-summary.html.

Android Developers. 2011c. AudioManager. Hakupäivä 11.3.2011.
http://developer.android.com/reference/android/media/AudioManager.html.

Android Developers. 2011d. AudioTrack. Hakupäivä 11.3.2011.
http://developer.android.com/reference/android/media/AudioTrack.html.

Android Developers. 2011e. AudioRecord. Hakupäivä 11.3.2011.
http://developer.android.com/reference/android/media/AudioRecord.html.

Android Developers. 2011f. Developing Introduction. Hakupäivä 1.4.2011.
http://developer.android.com/guide/developing/index.html.

Android Developers. 2011g. Installing the SDK. Hakupäivä 29.3.2011.
http://webcache.googleusercontent.com/search?q=cache:zRL4Eyi2YPUJ:developer.android.com/sdk/installing.html+android+sdk+contains&cd=1&hl=fi&ct=clnk&gl=fi&source=www.google.fi.

Android Developers. 2011h. JetPlayer. Hakupäivä 11.3.2011.
http://developer.android.com/reference/android/media/JetPlayer.html.

Android Developers. 2011i. GestureDetector. Hakupäivä 11.3.2011.
http://developer.android.com/reference/android/view/GestureDetector.html#onTouchEvent(android.view.MotionEvent).

Android Developers. 2011j. GestureDetector.OnGestureListener. Hakupäivä 11.3.2011.
http://developer.android.com/reference/android/view/GestureDetector.OnGestureListener.html.

Android Developers. 2011k. GestureOverlayView. Hakupäivä 11.3.2011.
http://developer.android.com/reference/android/gesture/GestureOverlayView.html.

Android Developers. 2011l. GestureOverlayView.OnGestureListener. Hakupäivä 11.3.2011.
http://developer.android.com/reference/android/gesture/GestureOverlayView.OnGestureListener.
html.

Android Developers. 2011m. MediaPlayer. Hakupäivä 11.3.2011.
http://developer.android.com/reference/android/media/MediaPlayer.html.

Android Developers. 2011n. MediaRecorder. Hakupäivä 11.3.2011.
http://developer.android.com/reference/android/media/MediaRecorder.html.

Android Developers. 2011o. SoundPool. Hakupäivä 11.3.2011.
http://developer.android.com/reference/android/media/SoundPool.html.

Android Developers. 2011p. ToneGenerator. Hakupäivä 11.3.2011.
http://developer.android.com/reference/android/media/ToneGenerator.html.

Android Developers. 2011q. What is Android?. Hakupäivä 6.4.2011.
http://developer.android.com/guide/basics/what-is-android.html.

Android Developers. 2011r. What is the NDK?. Hakupäivä 11.3.2011.
http://developer.android.com/sdk/ndk/overview.html.

Android Open Source Project. 2010. Philosophy and goals. Hakupäivä 24.11.2010
http://source.android.com/about/philosophy.html.

Bart, J. 2010. It's Official: Smartphones Dominate. Hakupäivä 24.11.2010
http://hothardware.com/News/Its-Official-Smartphone-Rule-The-Mobile-Web/.

Beavis, G. 2008. A complete history of Android. Hakupäivä 11.3.2011. http://www.techradar.com/news/phone-and-communications/mobile-phones/a-complete-history-of-android-470327.

Ceva Mediaroom. 2010. Glossary of Terms. Hakupäivä 15.10.2010. http://ceva-dsp.mediaroom.com/index.php?s=glossary.

Dalu, M. 2010. RIM vs. Android vs. iPhone: Q3/2010 sales, market share, desirability percentages emerge from recent studies by comScore, Nielsen and Gartner. Hakupäivä 30.3.2011. http://surgeworksmobile.com/iPhone/rim-vs-android-vs-iPhone-q32010-sales-market-share-desirability-percentages-emerge-from-recent-studies-by-comscore-nielsen-and-gartner.

Forum.Nokia. 2011. Nokia Open Source. Hakupäivä 1.4.2011. http://wiki.forum.nokia.com/index.php/Nokia_Open_Source.

Gordano, A. 2011. 52% on Froyo and Gingerbread near invisible!. Hakupäivä 1.4.2011. http://androidheadlines.com/2011/01/52-on-froyo-and-gingerbread-near-invisible.html.

Grabham, D. 2010. Intel and Nokia merge Moblin and Maemo to form Meego. Hakupäivä 18.3.2011. http://www.techradar.com/news/phone-and-communications/mobile-phones/intel-and-nokia-merge-moblin-and-maemo-to-form-meego-670302.

JetBrains. 2011. IntelliJ IDEA Editions Comparison. Hakupäivä 1.4.2011. http://www.jetbrains.com/idea/features/editions_comparison_matrix.html.

Kotilainen, S. 2011. Nokia älypuhelimien ykkönen – mutta kilpailijat hurjina. Hakupäivä 15.2.2011. http://www.tietokone.fi/uutiset/nokia_alypuhelinten_ykkonen_mutta_kilpailijat_hurjina.

KorgUK. 2010. KORG KAOSSILATOR Dynamic Phrase Synthetisizer. Hakupäivä 28.4.2011. http://www.korg.co.uk/products/dance_dj/kaossilator/kaossilator.asp.

Krill, P. 2010. Oracle revises plan to shut down Project Kenai. Hakupäivä 1.4.2011. http://www.infoworld.com/d/developer-world/oracle-revises-plan-shut-down-project-kenai-699.

Lehto, T. 2011. Nokia vaihtaa Windows-älypuhelimiin. Hakupäivä 1.3.2011. http://www.tietokone.fi/uutiset/nokia_vaihtaa_windows_alypuhelimiin.

Masalin, T. 2009. 10 legendaarista kännykkää. Hakupäivä 1.4.2011. http://www.tietokone.fi/lehti/tietokone_07_2009/10_legendaarista_kannykkaa_7814.

Morton, S. 2010. Windows Phone cheat sheet. Hakupäivä 18.3.2011. http://www.techrepublic.com/blog/smartphones/windows-phone-7-cheat-sheet/430.

Ojanperä, V. 2010. Samsung panostaa Windows Phone 7:een. Hakupäivä 24.11.2010 http://www.prosessori.fi/uutiset/uutinen2.asp?id=56824.

O'Malley, C. 1994. BellSouth's communicative Simon is a milestone in the evolution of the PDA. Hakupäivä 15.10.2010. http://web.archive.org/web/19990221174856/byte.com/art/9412/sec11/art3.htm.

Open Handset Alliance. 2010. Handset Manufacturers. Hakupäivä 24.11.2010. http://www.openhandsetalliance.com/oha_members.html.

Open Handset Alliance. 2011. Android Faq. Hakupäivä 15.4.2011. http://www.openhandsetalliance.com/android_faq.html.

Pitkänen, P. 2010. Älypuhelinten merkitys Nokialle kasvaa. http://www.taloussanomat.fi/talous/2010/01/28/alypuhelimien-merkitys-nokialle-kasvaa/20101378/133.

Project Kenai. 2011. Android Plugin For NetBeans. Hakupäivä 1.4.2011. http://kenai.com/projects/nbandroid/.

Sani, I. 2010. Samsung Bada rantautui ensimmäisenä Eurooppaan. http://www.tietoviikko.fi/kehittaja/article423583.ece.

Vaalisto, H. 2010. Google tunkeutuu televisioon. Hakupäivä 6.4.2011. http://www.itviikko.fi/uutiset/2010/03/10/google-tunkeutuu-televisioon/20103618/7.

Weisstein, E W. 2011a. Sawtooth wave. Hakupäivä 16.2.2011.
http://mathworld.wolfram.com/SawtoothWave.html.

Weisstein, E W. 2011b. Sine. Hakupäivä 16.2.2011. http://mathworld.wolfram.com/Sine.html.

Weisstein, E W. 2011c. Sinusoid. Hakupäivä 16.2.2011.
http://mathworld.wolfram.com/Sinusoid.html.

Weisstein, E W. 2011d. Square wave. Hakupäivä 16.2.2011.
http://mathworld.wolfram.com/SquareWave.html.

Weisstein, E W. 2011e. Triangle wave. Hakupäivä 16.2.2011.
http://mathworld.wolfram.com/TriangleWave.html.