Juho Nieminen

Vertaileva analyysi live coding -sävellysympäristöihin

Tietotekniikan
kandidaatintutkielma
1. helmikuuta 2010

Jyväskylän yliopisto

Tietotekniikan laitos

Jyväskylä

Tekijä: Juho Nieminen
Yhteystiedot: juho.nieminen@jyu.fi
Työn nimi: Vertaileva analyysi live coding -sävellysympäristöihin
Title in English: Comparative analysis of live coding environments
Työ: Tietotekniikan kandidaatintutkielma
Sivumäärä: 26
Tiivistelmä: Tässä tutkielmassa selvitetään mitä live coding on ja tutustutaan neljään live coding -sävellysympäristöön aloittelijan näkökulmasta. Ohjelmia vertaillaan neljän piirteen avulla ensisijaisesti käytettävyyden ja helpon omaksuttavuuden kannalta.
English abstract: In this study it is clarified what live coding is and four live coding composition environments are compared to each other from the beginner's point of view. The comparison is based on four features with the main focus being in usability and ease of assimilation.
Avainsanat: Live coding, algoritminen säveltäminen, helppokäyttöisyys.
Keywords: Live coding, algorithmic composition, usability.

Sisältö

1 Johdanto

Tietokonemusiikilla tarkoitetaan musiikkia, jonka sävellys- tai esitysvaiheessa tietokoneella on keskeinen rooli [1]. Se eroaa siis tältä osin elektronisesta musiikista, joka vain hyödyntää sähköisesti tuotettuja ääniä [2]. Tietokonemusiikki jaetaan karkeasti kahteen pääalueeseen: 1) algoritmiseen säveltämiseen ja 2) äänen käsittelyyn ja äänisynteesiin. Algoritmisessa säveltämisessä tuotetaan ja muokataan tietokonepohjaisesti musiikin rakenteita [1], kun taas äänen käsittelyssä ja äänisynteesissä keskitytään käsittelemään itse äänielementtejä kuten taajuus ja äänen voimakkuus ja vaihe [3].

Tietokonemusiikki on lähtöisin 1950-luvulta, jolloin ensimmäiset tietokoneella tehdyt kappaleet esitettiin [4]. Ensimmäisen varsinaisen musiikkiohjelman teki kuitenkin Max Mathews 1950-luvun lopulla [5]. Ensimmäiset musiikkiohjelmat eivät olleet reaaliaikaisia, vaan muutaman minuutin mittaisen kappaleen laskemiseen saattoi ohjelmalta kulua tunteja tai päiviä. Vasta 1970-luvun mikroprosessorit avasivat tien reaaliaikaisen musiikin tekemiselle, ja 1990-luvulle tultaessa prosessorien tehot olivat kasvaneet niin, että reaaliaikaista tietokonemusiikkia pystyttiin tekemään tavallisemmilla ohjelmilla ja algoritmeilla laboratorio-olosuhteiden sijaan. Tietokoneiden koon pieneneminen mahdollisti myös niiden kipuamisen konserttilavoille, ja näin niitä pystyttiin alkaa käyttämään esiintymisvälineinä. [6] Elävä tietokonemusiikki on siis melko nuori taiteenlaji, ja lisäksi siitä tehty tutkimus painottuu musiikkitieteen puolelle. Tämä tutkimus on tehty näiden syiden pohjalta.

Tutkimuksen kohteena on live coding eli musiikin soittaminen tietokoneen avulla perinteisten instrumenttien tapaan yleisön nähdessä, miten esiintyjä muodostaa, eli ohjelmoi, soivan musiikin. Tämä on mielenkiintoinen yhdistelmä tietoteknistä ja musiikillista osaamista. Tässä tutkimuksessa tutustutaan tämän yhdistelmän mahdollistaviin ohjelmistoihin. Tarkoituksena on lähestyä live coding -käsitettä ja -sävellysympäristöjä tietotekniikan kannalta puuttumatta taiteellisiin seikkoihin. Tutkielman pääajatus voidaan kirjoittaa seuraavanlaisesti: miten eri live coding -sävellysympäristöt eroavat toisistaan ja millaisia piirteitä niistä löytyy. Ohjelmia tarkastellaan aloittelijan näkökulmasta, jolloin helppokäyttöisyys ja helppo omaksuttavuus korostuvat. Live coding -ohjelmistoista valittiin mukaan neljä avoimen lähdekoodin sovellusta, jotka ovat keskittyneet enimmäkseen musiikin tuottamiseen.

Tutkielman luvussa 2 käydään läpi live coding -käsite ja sen lyhyt historia. Luvussa 3 esitellään tutkimusmenetelmä. Luku 4 esittelee käytettävän aineiston, luku 5 kuvaa vertailavat ominaisuudet, luvussa 6 käydään läpi itse vertaileva analyysi, ja viimein yhteenvedossa, luvussa 7, tiivistetään analyysin tulokset ja pohditaan

tutkimuksen herättämiä kysymyksiä.

2 Live coding

Live coding on äänen ja videon luomista ja muokkaamista reaaliajassa yleisön edessä. Esiintyjät kirjoittavat koodia tietokoneelle, jonka monitorin kuva heijastetaan yleisön nähtäville, joten katsojat pääsevät seuraamaan luomisprosessia hyvin tarkasti. Ohjelmaa siis kirjoitetaan sen koko ajan pyöriessä, mikä erottaa live codingin tavallisesta tietokonemusiikista [7]. Ääntä ei välttämättä synnytetä tapahtuman aikana, vaan ääniraita voi olla ennalta tehty, ja sitä vain muokataan erilaisilla alrogitmeilla esiintymisen aikana. Tapahtumat voivat sisältää ääntä, videota tai molempia riippuen käytetyistä ohjelmista ja laitteista. Pääpainon ollessa kuitenkin musiikissa. Live coding tunnetaan myös nimillä interaktiivinen ohjelmointi ja ”on-the-fly” -ohjelmointi. [6]

2.1 Historia

Zmölnigin ja Eckelin tutkielmassa [6] mainitaan, että yksi ensimmäisistä tunnetuista live coding -tapahtumista järjestettiin vuonna 1985 Amsterdamissa Ron Kuivilan esittämänä. Tuohon aikaan esiintymisissä käytetyt ohjelmointikielet olivat Lisp ja Forth. Pian seurannut 1990-luku ei kuitenkaan ollut live codingin suhteen vilkasta aikaa, kun iso osa huomiosta kiinnittyi Internetin nousuun ja avoimeen ohjelmistokehitykseen. On kuitenkin huomattava, että näiden molempien kehityksellä on ollut positiivinen vaikutus myös live codingin kehitykseen ja levinneisyyteen. [6]

Zmölnig ja Eckel kuvaavat tilanteen muuttumisen tultaessa 1990-luvun loppuun: Syntyi uusi taiteen muoto, joka tutki algoritmeja ja käsitteli lähdekoodia ilmaisunmuotona. Tietokoneiden tehot olivat kasvaneet niin, että ohjelmointikielien tulkit olivat tulleet tarpeeksi nopeiksi luomaan ääntä ja videota. Live coding -tapahtumia alettiin taas järjestää useiden ihmisten toimesta uusilla ohjelmointiympäristöillä ja kielillä kuten Perl ja REAL basic. Lopulta vuonna 2003 live codingille luotiin oma järjestö: TOPLAP, *The Transnational Organisation for the Proliferation of Live Audio Programming*, eli kansainvälinen elävän ääniohjelmoinnin edistämisen järjestö. Nykyään TOPLAP esittelee live coding -sovelluksia, organisoi live coding -tapahtumia, julkaisee live coding -musiikkia ja -videoita ja toimii kommunikointikanavana live coding -yhteisössä. [6] [8]

2.2 Määritelmä

Wang ja Cook [7] määrittelevät live coding -käsitteen seuraavilla elementeillä:

- Olkoon P ajettava (mahdollisesti tyhjä) ohjelma.
- Kirjoitetaan uusi koodinpätkä, Q, ja lisätään se P:hen, jolloin saadaan ohjelma P'.
- Muokataan ohjelman P koodia, kun se on ajossa.

Koodinpätkän Q lisääminen P:hen tarkoittaa, että Q suoritetaan nyt P:n osoiteavaruudessa hyödyntäen mahdollisesti samaa dataa ja että Q:n ja P:n välillä on vahva aikariippuvuus. Se taas tarkoittaa, että Q pääsee käsiksi P:n ajoituksiin ja voi synkronoitua P:n kanssa.

2.3 Paradigma

TOPLAPin jäsenten kirjoittama manifesti [9] toteaa muutamia seikkoja live codingista, jotka auttavat selventämään käsiteltävää ajatusmaailmaa. Live codingissa katsotaan olevan enemmänkin kyse algoritmeista kuin työkaluista. Eli periaatteena on, että arvostetaan enemmän ajatuksia kuin esineitä. Algoritmien tutkiminen, kehittäminen ja taitava käsittely ja soveltaminen ohjelmiin onkin keskeinen teema live coding -käsitteessä. Lisäksi rakennettuja musiikkiohjelmia ajatellaan instrumentteina, jotka voivat muuttaa itse itseään. [9]

TOPLAP-manifesti [9] mainitsee sanonnan: ”tietojen pimittäminen on haitallista — näyttäkää kuvaruutunne”. Se viittaa live coding -tapahtumien avoimeen luonteeseen ja käytäntöön, jossa esiintyjien tietokoneiden kuvaruudut heijastetaan valkokankaalle. Koodin halutaan olevan näkyvissä ja kuultavissa niin, että algoritmit ovat selvästi esillä. Live coding -esiintyjät myös tiedostavat, että yleisön ei välttämättä tarvitse ymmärtää koodia voidakseen nauttia siitä, aivan kuten kitaran soitosta voi nauttia ilman, että tietää miten sitä soitetaan. [9]

2.4 Keskeiset käsitteet

Wongin ja Cookin mukaan [7] live codingista voidaan erotella neljä keskeistä käsitettä, joiden avulla live coding -ympäristöjä voidaan arvioida.

- *Modulaarisuus* — jotta ohjelmoija voisi käsitellä koodinpalasia erikseen, on koodin oltava modulaarista, ja moduulien on toimittava samassa osoite- ja nimiavaruudessa.

- *Ajoitukset* — uusien ja jo olemassa olevien ohjelman osien välillä on oltava ajallinen yhtenäisyys ja vahva aikakäsitys; peräkkäisten koodin osien on pysähdyttävä ja käynnistyttävä täsmällisesti.

- *Selkeys ja helppokäyttöisyys* — ohjelman on oltava tarpeeksi selkeä ja helppokäyttöinen, jotta ideoiden toteuttaminen täsmällisesti ja ajan ja datan käsittely on mielekästä.

- *Joustavuus* — kykeneekö ohjelma käyttämään hyväksi ohjelmointikielten tehokkaita rakenteita reaaliaikaisessa ympäristössä?

2.5 Musiikillisia näkökulmia

Uutena käsitteenä live coding herättää myös paljon kysymyksiä. Onko kyseessä vain tietokoneella generoidun musiikin soittamista DJ:n tapaan? Onko esiintyjä aina varma, miltä hänen tuottamansa koodi tulee kuulostamaan tai näyttämään? Eikö virheiden mahdollisuus ole suuri ja niiden vaikutus liian dramaattinen? Eikö koodin heijastaminen valkokankaalle häiritse yleisön keskittymistä itse musiikkiin, vai onko musiikki vain toissijainen asia? Live coding -ohjelmien tekeminen voi viedä paljon aikaa, kuuluuko sinänsä tylsä alkuasettelu esitykseen?

Näitä kysymyksiä TOPLAPin jäsen Nick Collins on pohtinut nettikirjoituksessaan [10]. Hän huomauttaa, että live coding on algoritmeilla leikkimistä, improvisointia, eikä suinkaan DJ-toiminnan tapaista ennalta luodun musiikin soittamista. Improvisoinnista seuraa myös live codingin kokeellinen luonne; koodin seurauksia ei aina lähdetä edes arvailemaan, eikä se ole joskus mahdollistakaan, jos muuttujia on liian monta. Kokeellisuuteen taas liittyvät ohjelmoinnin virheet. Niitä ei kuitenkaan pitäisi katsoa kovin kriittisesti, sillä niiden voidaan katsoa kuuluvan olennaisena osana itse esitykseen. [10]

Valkokankaan käyttäminen live coding -esiintymisissä on seurausta alan avoimesta luonteesta. Tuotettu koodi ei kuitenkaan koskaan saisi olla esityksen pääasia. Halutessaan katsoja voi vaikka sulkea silmänsä nauttiakseen vain musiikista, mutta yleisölle halutaan antaa mahdollisuus seurata ohjelman rakentumista. Tähän rakentamiseen kulunut aika vaihtelee hyvin paljon riippuen siitä, käyttääkö esiintyjä ennalta tehtyjä rakenteita vai luoko hän kokonaan uusia. Aloitukseen kuluvaa aikaa voidaan lyhentää tekemällä työtä etukäteen, mutta ei perinteinen muusikkokaan yleensä ole parhaimmillaan juuri esityksen alussa. [10]

3 Tutkimusmenetelmä

Tässä luvussa kuvataan tutkimuksessa käytetty kvalitatiivinen vertaileva tutkimusmenetelmä (aliluku 3.2) ja sen perustana toimiva perinteinen vertaileva analyysi (aliluku 3.1).

3.1 Perinteinen vertaileva analyysi

Perinteisessä vertailevassa analyysissä jokaista kohdetta pidetään samanarvoisena. Vastakohtana olisi linssianalyysi, jossa jotain kohdetta painotetaan enemmän, ja toisia kohteita käytettäisiin kuin linssejä, joiden läpi tärkeintä kohdetta arvioitaisiin. Perinteisessä vertailevassa analyysissä voidaan edetä kuvaamalla yhden kohteen kaikki piirteet kerralla tai niin, että käydään ensin läpi yksi vertailtava piirre kaikkien kohteiden osalta ja siirrytään sitten seuraavaan piirteeseen. Analyysi voidaan jakaa kolmeen osaan: käytettyyn viitekehykseen, vertailun perusteisiin ja löydettyihin väitteisiin. [11]

- *Viitekehys* — Viitekehys on asiayhteys, jonka valossa vertailu suoritetaan. Se voi koostua ideasta, teemasta, kysymyksestä, ongelmasta, teoriasta, vertailuryhmästä tai biograafisesta tai historiallisesta näkökulmasta. Viitekehys siis antaa tutkimukselle näkökulman, ja ilman sitä mielekkäiden johtopäästösten tekeminen aineistosta olisi mahdodonta.

- *Vertailun perusteet* — Syyt aineiston valinnalle luovat vertailun perusteet. Miksi juuri kyseiset kohteet valittiin mukaan vertailuun? Tutkimuksen tulisi vastata tähän kysymykseen, ja lukijoille pitäisi käydä selväksi, että valinnalle on järkevät perustelut sattuman sijaan.

- *Teesit* — Tutkimuksen väitteet eli teesit pohjautuvat käytettyyn viitekehykseen ja saattavat tutkimuksen ydinajatusta mukanaan. Väitteiden luonne riippuu siitä, miten vertailtavat kohteet oikeastaan liittyvät ja suhtautuvat toinen toisiinsa.

3.2 Vertailu tässä tutkimuksessa

Vertailun tarkoituksena on löytää tutkittujen live coding -sävellysympäristöjen erot ja samanlaisuudet. Toisin sanoen aineistoon tutustumalla ja ohjelmistoja kokeilemalla tarkoitus on etsiä ne ominaisuudet tai piirteet, jotka erottelevat ympäristöt toisistaan tai jakavat ne ryhmiin. Tutkimuksen aineiston viitekehys on kysymys, miltä

live coding -sävellysympäristöt näyttävät aloittelijan silmissä. Erityisesti tarkoitus on selvittää Wangin ja Cookin esittämää yhtä arviointiperustetta (luku 2.4): selkeyttä ja helppokäyttöisyyttä. Tämä kysymys luo pohjan aineiston kohteiden väliselle vertailulle, joka suoritetaan käymällä aineisto läpi piirre kerrallaan ja tutkimalla miten tuo piirre esiintyy kaikissa kohteissa. Aineisto ja vertailtavat piirteet kuvataan ja perustellaan luvuissa 4 ja 5 ja tutkimuksen teesit eli johtopäätökset löytyvät vertailuosion aliluvusta 6.5.

Tutkimuksessa käyttötestaus on rajattu hyvin vähäiseksi, ja suurin osa huomioista on ensivaikutelmia ja aloittelijan näkemyksiä. Tarkemman käyttötestauksen puutteessa turvaudutaan paljon ohjelmien dokumentointiin ja muuhun valmiiseen aineistoon, mikä ei välttämättä anna oikeudenmukaista kuvaa sovelluksesta. Myös jokaisen vertailtavan piirteen syvällinen analysointi ei ole tässä mahdollista, joten vertailussa pyritään ottamaan huomioon vain ydinseikat ja etenkin uutta käyttäjää koskevat ongelmat. Pitää muistaa, että on mahdotonta huomioida vertailtavan aineiston kaikkia puolia. Kvalitatiivisen tutkimuksen luonteen mukaisesti tutkimustulos onkin aina valikoiva [13].

Tutkin ohjelmia live coding -aloittelijan näkökulmasta. Minulla ei ole aikaisempaa kokemusta live coding -sovelluksista, joten uskon pystyväni tarjoamaan melko luotettavan aloittelijan näkökulman. On kuitenkin huomioitava, että olen erittäin kokenut tietokoneenkäyttäjä, joten aloittelijan näkemykseni on värittynyt useiden muiden sovellusten käyttökokemuksista. Ohjelmoinnin olen aloittanut Javalla ja muilla perinteisillä ohjelmointikielillä, enkä ole aikaisemmin kokeillut graafista ohjelmointia. On syytä pitää myös mielessä, että en ole aloittelija musiikinkaan saralla sovellusten tai musiikin teorian näkökulmista.

4 Aineiston kuvaus

Live codingin tietoteknisestä luonteesta johtuen sitä varten on kirjoitettu useita erilaisia ohjelmia [12]. Tekijöinä on ollut joukko harrastajia, koska kaupallinen kiinnostus näin erityistä ja uutta alaa kohtaan on pientä. Ohjelmat ovat pääasiassa avointa lähdekoodia, ja niitä kehittää eteenpäin omat yhteisönsä. Ajan saatossa ohjelmia on syntynyt ja kuollut, mutta tämän vuosituhannen puolella uusia ohjelmistoja on tehty yhä enemmän live codingin kasvattaessa suosiotaan Internetin avustuksella. Keskeistä kaikille live coding -ohjelmistoille on, että käyttäjä pääsee kirjoittamaan omia koodinpätkiään ohjelmaan ja ajamaan niitä reaaliajassa. [6]

Live coding -ohjelmistot voi jakaa karkeasti kahteen ryhmään käyttöliittymänsä mukaan: tekstipohjaisiin ja graafisiin ohjelmistoihin [6]. Tekstipohjaiset ohjelmistot

on selkeästi yleisempi ryhmä niin musiikin kuin videonkin käsittelyssä [12]. Niissä käyttäjä syöttää ohjelmaan vaikuttavien moduulien koodit tekstinä aivan kuten tavallisessa ohjelmoinnissa. Graafisissa ohjelmistoissa sen sijaan rakennetaan ohjelmaa graafisista elementeistä, joita ohjataan ja liitetään toisiinsa tietokoneen hiiren avulla. On kuitenkin tavallista, että myös graafiset live coding -ohjelmistot antavat mahdollisuuden kirjoittaa koodia ja näin luoda uusia elementtejä ohjelmaan.

Tutkimuksessa tullaan vertailemaan neljää eri live coding -sävellysympäristöä: ChucK, SuperCollider, Pure Data ja Impromptu. Nämä neljä valittiin, koska ne ovat avoimia ohjelmistoja, ne toimivat nykyisillä käyttöjärjestelmillä, ne ovat vielä laajalti käytössä ja niitä kehitetään edelleen. Nämä syyt sopivat hyvin tutkimuksen viitekehykseen, eli aloittelijan näkökulmaan. Päätarkoitus on kaikilla ohjelmilla sama, mutta niistä löytyy silti mielekkäitä eroavaisuuksia vertailun suorittamista varten.

4.1 ChucK ja miniAudicle

ChucKin kotisivujen [14] mukaan ChucK on vahvasti ajoitettu, rinnakkaistettu ja lennossa kirjoitettava ääniohjelmointikieli. Se on hyvin monipuolinen ja tarkoitettu erityisesti reaaliaikaiseen äänisynteesiin, säveltämiseen, esiintymiseen ja äänianalyysiin. ChucK-ohjelma, joka siis käyttää kielenään ChucKia, toimii Windowsissa, Linuxissa ja Mac OS X:ssä, ja se tukee monikanavaääntä ja monia rajapintoja kuten MIDI:ä (*Musical Instrument Digital Interface*), OSC:tä (*Open Sound Control*) ja HID:tä (*Human Interface Device*). ChucKin ensimmäinen versio ilmestyi vuonna 2003 Ge Wangin ja Perry Cookin julkaisemana [15]. Käyttöä helpottamaan on tehty sovellus nimeltä miniAudicle, joka on kevyt graafinen käyttöliittymä varsinaiselle ChucK-ohjelmalle [16]. Sen on tehnyt Spencer Salazar ja Ge Wang vuonna 2005. Vertailussa käytän ChucKia miniAudiclen kanssa ja puhuessani ChucKista, viittaan tähän ohjelmayhdistelmään. Käytetyt ohjelmaversiot ovat ChucK 1.2.1.3 (dracula) ja miniAudicle 0.2.0 (gidora).

4.2 SuperCollider

Kehittäjänsä sivuilla [17] SuperCollideria kuvaillaan kehitysympäristöksi ja ohjelmointikieleksi, joka on tarkoitettu reaaliaikaiseen äänisynteesiin. SuperColliderin ohjelmointikieli muistuttaa paljon Smalltalkia. Se on tulkattu, dynaaminen ja oliopohjainen, ja sillä ohjataan erillistä äänisynteesipalvelinta. Tuettuja käyttöjärjestelmiä ovat: Windows, Linux, Mac OS X ja FreeBSD, ja jokaiselle alustalle on useita asiakasohjelmia, joita voidaan käyttää SuperColliderin kanssa äänisynteesipalvelimen ohjaamiseen. Myös SuperCollider tukee MIDI-, OSC- ja HID-rajapintoja. Su-

perColliderin ensimmäisen version julkaisi sen kehittäjä James McCartney maaliskuussa vuonna 1996. [18] Käytetty ohjelmaversio on 3.3.1.

4.3 Pure Data

Pure Data on reaaliaikainen graafinen ohjelmisto musiikin ja videon esittämiseen [20]. Se on kolmas huomionarvoinen jäsen ohjelmointikieliperheessä nimeltä Max, jonka alunperin kehitti Miller Puckette, ja hänen toimesta alkoi myös Pure Datan kehitys 1990-luvun puolivälissä [19]. Sittemmin Pure Data on levittäytynyt monille alustoille, ja siitä on julkaistu versio seuraaville käyttöjärjestelmille: Windows, GNU/Linux, MacOS X, IRIX ja BSD. Pure Data tukee laajalti eri rajapintoja, kuten MIDI:ä, OSC:tä, HID:tä ja VST:tä (*Virtual Studio Technology*). [20] Käytetty ohjelmaversio on Pd 0.41.4-extended.

4.4 Impromptu

Impromptun kehittäjä Andrew Sorensen kuvailee ohjelmistoaan interaktiiviseksi reaaliaikaisen, algoritmipohjaisen musiikin kehitys- ja esiintymisympäristöksi [21]. Vuonna 2005 julkaistu ohjelmisto on kehitetty tukemaan useita käyttäjiä, ja se sisältää oman kehitysympäristönsä, joka on suunniteltu esiintyvää ohjelmointia ja studiotyöskentelyä varten. Impromptun kotisivujen mukaan [22] käytettävä ohjelmointikieli on Scheme, joka kuuluu Lisp-kielten perheeseen, mutta myös C:tä voidaan paikallisesti käyttää. Impromptu on tällä hetkellä saatavilla vain Mac OS X -käyttöjärjestelmälle, ja rajapinnoista se tukee MIDI:ä, OSC:tä ja AU:ta (*Audio Units*), joka on Applen vastine VST:lle. Käytetty ohjelmaversio on 2.1.

5 Vertailtavat piirteet

Tässä luvussa kuvaillaan piirteet, joiden mukaan sävellysympäristöjä vertaillaan ja luokitellaan. Vertailtavat piirteet jakautuvat teemoihin: ohjelman käyttöönotto, käyttöliittymä, avustus ja tuki ja sävellysympäristön liitännäisyys, eli millaisia rajapintoja se voi hyödyntää. Vertailussa käytettävä aloittelijan näkökulma luo paljon painoa ohjelman käytettävyydelle. Jakob Nielsen on määritellyt käytettävyyden näiden viiden ominaisuuden kautta [26]:

- *Opittavuus* — Ohjelma tulisi olla helposti opittava, jotta käyttäjä voi alkaa saamaan tuloksia nopeasti aikaan.

- *Tehokkuus* — Ohjelman käyttö tulisi olla tehokasta niin, että kun käyttäjä on oppinut ohjelman käytön, suuri tuottavuus olisi mahdollista.
- *Muistettavuus* — Ohjelman tulisi olla helposti muistettava, jotta käyttäjä voi tauon jälkeen palata ohjelman pariin ilman uudelleenopettelua.
- *Virheettömyys* — Ohjelmalla tulisi tuottaa mahdollisimman vähän virheitä käytössä, ja jos virheitä tapahtuu, niistä pitäisi pystyä selviämään helposti. Katastrofaalisia virheitä ei saisi tapahtua lainkaan.
- *Tyytyväisyys* — Ohjelman tulisi olla miellyttävä käyttää, jotta käyttäjät olisivat tyytyväisiä ja pitäisivät ohjelmasta.

Miten näitä käytettävyyden ominaisuuksia voitaisiin sitten arvioida ohjelmistoissa? Tutkimuksessa piirteiden vertailussa hyödynnetään seuraavia Nielsenin kehittämiä käytettävyysheuristiikoita [25] käytettävyyden arviointiin:

- *Esteettinen ja minimaalinen suunnittelu* — Käyttötapaukset eivät saa sisältää ylimääräistä informaatiota, sillä kaikki ylimääräinen tieto kilpailee relevantin tiedon kanssa ja vähentää sen suhteellista näkyvyyttä käyttäjälle.
- *Yhtenevyys ja standardit* — Käyttäjän ei pitäisi joutua arvailemaan, tarkoittavatko jotkin sanat, tilanteet tai toiminnot samaa kuin jokin muu. Sovelluksen tulisi noudattaa alustan standardeja.
- *Virheiden estäminen* — Käytettävyyden kannalta hyviä virheilmoituksia parempi ratkaisu on suunnitella ohjelma siten, että virheitä ei pääse tapahtumaan. Virhealttiit tilat tulisi eliminoida kokonaan tai virheitä pitäisi tarkkailla, ja käyttäjältä pitäisi kysyä varmistuksia ennen toimintojen suorittamista.
- *Järjestelmän tilan näkyvyys* — Järjestelmän tulisi aina tiedottaa käyttäjälle, mitä on tapahtumassa sopivalla määrällä palautetta oikeaan aikaan.
- *Käytön vapaus* — Käyttäjät valitsevat usein toimintoja vahingossa, joten he tarvitsevat selkeästi merkityn ”hätäuloskäynnin”, jolla poistua epätoivotusta tilasta ilman hankalaa dialogia. Ohjelmien tulisi tukea undo- ja redo- toimintoja.
- *Avustus ja dokumentaatio* — Vaikka olisi parempi, että ohjelmaa osattaisiin käyttää ilman dokumentointia, voi avustuksen ja dokumentoinnin tarjoaminen olla tarpeellista. Avustukseen ja dokumentointiin pitäisi olla helppo suorittaa hakuja, sen pitäisi paneutua käyttäjän tehtäviin, listata tarkkoja vaiheita tehtävien suorittamiseen ja sen ei tulisi olla liian laaja.

5.1 Käyttöönotto

Käyttöönoton vertailussa tarkastellaan, kuinka ohjelmat ovat toteuttaneet seuraavat Roger Leen esittämät ohjelman käyttöönottoon liittyvät vaiheet [23]:

- *Asennus* — Miten ohjelma asennetaan, vai vaatiiko ohjelma asennusta lainkaan? Leen mukaan asennus voi olla käyttäjävetoinen niin kuin InstallShield, se voi tapahtua paketinhallinan kautta kuten Linuxissa, siihen voi olla tehty Internet-pohjaisia työkaluja kuten Java Web Start tai Windows Update, asennus voi tapahtua järjestelmänhallinnan työkaluilla kuten Microsoft SMS:ssä, se voidaan suorittaa etänä esimerkiksi Citrix:n avulla tai asennus voi olla julkaisetilaa-periaatteella toimiva kuten Sun Java Message Service.

- *Konfigurointi* — Vaaditaanko käyttäjältä asetusten säätämistä heti alussa, vai ovatko esimerkiksi ääniasetukset valmiiksi oikein? Leen vaiheissa konfigurointi tarkoittaa toimintaa, jossa jo asennetun ohjelman asetuksia muutetaan ennen ohjelman aktivointia.

- *Aktivointi* — Mitä vaiheita äänen aikaansaaminen ohjelmassa vaatii? Kertooko ohjelma käyttäjälle ohjeita, miten aloittaa käyttö, jos se ei ole itsestään selvää? Lee kuvaa aktivoinnin ohjelman käynnistämiseksi, ja tässä sitä on tulkittu niin, että käynnistämiseksi katsotaan musiikin ohjelmoinnin aloittaminen.

Käyttöönotto on hyvin tärkeä osa ohjelman toimivuutta, sillä ensikokeilu voi määrätä ottaako käyttäjä ohjelman käyttöönsä vai ei. Ensimmäisten käyttöminuuttien aikana käyttäjä päättää onko ohjelmasta hänelle hyötyä vai ei. [24]

5.2 Käyttöliittymä

Käyttöliittymän vertailussa selvitetään, onko sävellysympäristön käyttöliittymä graafinen vai tekstipohjainen, ja kuinka se toteuttaa viittä ensimmäistä edellä mainittua Nielsenin käytettävyysheuristiikkaa. Käyttöliittymällä on suhteellisen iso rooli live coding -ympäristössä, sillä se tulee hyvin esille, kun esiintyjän tietokoneen monitori heijastetaan valkokankaalle yleisön nähtäville. Käyttöliittymän rooli korostuu myös aloittelijoissa, sillä helppokäyttöinen käyttöliittymä on edellä kuvatun Nielsenin määritelmän mukaan nopeasti opittavissa, mikä rohkaisee käyttäjää ja helpottaa kokeilua. Kokeilu on live codingin oppimisen kannalta tärkeää, sillä perinteisten instrumenttien tapaista harjoitteluperinnettä ei ole [27].

5.3 Avustus ja tuki

Vertaillessa ohjelmia avustuksen ja tuen suhteen tarkastellaan, löytyykö ohjelmasta help-osio, manuaali tai esimerkkejä, ja millaisia ne ovat edellä esitetyn Nilsenin dokumentaatiota koskevan käytettävyysheuristiikan perusteella. Lisäksi selvitetään, onko ohjelmalla postituslistaa, josta voi kysyä apua. Mainitun käytettävyysheuristiikan mukaan ohjelmaa tulisi pystyä käyttämään ilman helpin tukea, mutta live coding -ohjelmissa törmää heti uusiin asioihin, joista pitäisi olla helppo ottaa selvää. Ohjelman perustoimintojen ja käytetyn ohjelmointikielen esitteleminen, sekä esimerkkien ja tutoriaalien tarjoaminen on aloittelijan kannalta erityisen tärkeää.

5.4 Liitännäisyys

Mitä erilaisia rajapintoja sävellysympäristö tukee? Voidaanko sitä ohjata ulkopuolisilla ohjaimilla, tai saadaanko ohjelmistosta ulos erilaisia tiedostoformaatteja? Entä minkälaisia tiedostoformaatteja ohjelma osaa tulkita? Pystytäänkö ohjelmaan lisätä toiminnallisuutta lisäosilla, entä voidaanko ohjelmaa käyttää osana jotain toista sovellusta? Näihin kysymyksiin tutustutaan sävellysympäristöjen vertailussa liitännäisyyden osalta. Liitännäisyys voi tuntua merkityksettömältä aloittelijoille, mutta mahdollisuus käyttää jo tuttuja työvälineitä ja -tapoja live coding -ohjelmien ohessa vähentää uuden opettelemisen taakkaa. Tämä parantaa käytettävyyttä, sillä helppo muistettavuus on osa edellä mainittua käytettävyyden määritelmää.

6 Vertaileva analyysi

6.1 Käyttöönotto

Vertailtavat ohjelmat voi jakaa heti kahteen osaan: niihin, jotka vaativat asennuksen ja niihin, jotka eivät. Pure Data ja SuperCollider ovat niitä, jotka pitää asentaa ennen käyttöä, kun taas ChucKia ja Impromptua ei tarvitse asentaa, vaan riittää, kun purkaa ohjelmapaketin kovalevylle. Voidaan olla monta mieltä siitä, kumpi tapa on helpompi, mutta itse kallistun ei-asennettavien puolelle, sillä silloin käyttöönotto on yleensä vähävaiheisempi kuin käyttäjävetoisissa asennusmalleissa, mikä parantaa käytettävyyttä Nilsenin esteettisyyttä ja minimaalisuutta koskevan käytettävyysheuristiikan mukaan. Lee taas tuo tässä esille käyttäjävetoisten asennusten virhealttiuden [23]. Lisäksi asennusta tarvitsemattomat ohjelmat saa aina sinne, minne haluaa kovalevyllään. Toisin kuin esimerkiksi SuperCollider, joka ei antanut vaihtoehtoa asennuskansioksi, vaan asensi itsensä Windows-käyttöjärjestelmässä suosit-

tuun program files -kansioon. Muutoin SuperColliderin kuin Pure Datankin asennus oli helppo käyttäjävetoinen, wizard-tyyppinen nopea toimitus.

Ohjelmien välille ei syntynyt eroja, kun tarkasteltiin tarvittava konfiguroinnin määrää. Kaikki ohjelmat osasivat alustaa itsensä niin, että ääntä kuului eikä viivettä ollut huomattavasti. Ei ole vallan tavatonta musiikkisovelluksissa, että ensimmäisellä kerralla joutuu kahlaamaan läpi asetuksia, jotta saisi edes jotain ääntä kuulumaan. Äänen aikaansaaminen ei kuitenkaan ollut yhtä helppoa kaikissa sovelluksissa: SuperColliderissa vaaditaan ensin äänipalvelimen käynnistäminen, ChucKissa pitää laittaa virtuaalikone päälle ja Pure Datassa taas täytyy dokumenttiin osata asettaa ääniulostulo. Nämä kaikki tarkoittavat yhtä ylimääräistä vaihetta ennen päämäärää, ohjelman aktivointia, mikä tekee niistä vaikeammin omaksuttavia esteettisyyden ja minimaalisuuden periaatteen mukaan. Sen sijaan Impromptussa ääni kuuluu heti, kun vain kirjoittaa ja ajaa ensimmäisen sopivan komennon. Muista aktivointikäytännöistä SuperColliderin asiakas-palvelin-malli ja ChucKin virtuaalikone tuntuivat tutuimmilta ja osasin käynnistää ne intuitiivisesti, mutta Pure Datassa ääniulostulon merkitys valkeni vasta esimerkin kautta.

Koska live coding -sovellukset ovat pienen joukon harrastus, ja tavallisen käyttäjän oletetaan olevan perillä, mistä ohjelmassa on kyse, ei yksikään neljästä ohjelmasta anna käyttäjälle mitään aloitusvinkkejä. Jää siis käyttäjän tehtäväksi ottaa selvää, mistä aloittaa ja aktivoida ohjelma. Onneksi jokaisessa sovelluksessa on helppi tai manuaali, josta on hyvä lähteä liikkeelle — näistä lisää avustuksen ja tuen vertailussa. Mutta jos vertaillaan, mitä ohjelmat tarjoavat käynnistyttyään, niin ChucK ja Impromptu avaavat käyttäjälle tyhjän koodi-ikkunan valmiiksi, kun taas SuperCollider ja Pure Data antavat käyttäjän avata uuden dokumentin. Käyttöönoton kannalta koin helpommaksi, että ohjelma ohjasi käyttäjää hieman tarjoamalla editorin, jolla aloittaa kokeilu. Tämä toimintatapa tukee Nielsenin käytettävyysperiaatetta yhtenevyydestä ja standardien noudattamisesta, jos on tottunut Windows-ohjelmien tapaan avata uusi dokumentti ohjelman käynnistyessä. Lisäksi tämä käytäntö vähentää jälleen yhden vaiheen ohjelman aktivoinnista edelleen parantaen käytettävyyttä.

Kaikki vertaillut live coding -sovellukset olivat yleisellä tasolla helppoja ottaa käyttöön. Hyvän käytettävyyden periaatteiden mukaisesti sovellukset noudattavat standardeja graafisten ohjelmien käytäntöjä valikoissa ja muissa toiminnoissa, joten navigointi ohjelmissa oli tavanomaista. Aloituskynnykseksi käyttäjälle jää vaiva tutustua ohjelman käyttöohjeisiin. Muista sovelluksista graafisella ohjelmoinnillaan poikkeava Pure Data oli kuitenkin nelikon vaikein lähestyttävä, juurikin erilaisen operointitapansa takia, olettaen että käyttäjällä ei ole kokemusta graafisesta ohjelmoinnista tai vastaavanlaisesta käytöstä esimerkiksi musiikkisovelluksissa. Muis-

sa ohjelmissa ohjelmointi hoituu vain tekstillä, joten koodin suoritusjärjestys ja esitysmuoto näyttää tutulta ja käyttäjä oppii ohjelman nopeammin oppimisen siirron avulla [28]. Pure Datan graafinen järjestely sen sijaan ei kerro aloittelijalle paljoakaan. Tähän ongelmaan sai apua, kun etsi Internetistä Pure Datan käyttövideoita, joissa ohjelman rakennusjärjestys näkyy hyvin.

6.2 Käyttöliittymä

Kuten käyttöönoton vertailussa kävi ilmi, erottuu yksi vertailtavista live coding -sovelluksista selvästi muista käyttöliittymältään. Pure Datan ohjelmointi on pääosin graafista, kun taas muiden sovellusten ohjelmointi on täysin tekstipohjaista. Pure Datassa ohjelmia luodaan sijoittamalla ohjelman editoriin erilaisia elementtejä, joita yhdistellään vetämällä hiirellä ”johtoja” elementtien välille. Elementtien ohjaamista varten luodaan erillisiä kontrollielementtejä, joten kaikki muu hoituu graafisesti paitsi elementtien määritteleminen tehtäviinsä. Graafisen ohjelmoinnin idean tajuaminen oli aluksi melko hankalaa, mutta onneksi Internetistä löytyi videoita, joissa rakennettiin yksinkertaisia esimerkkejä alusta lähtien, ja niiden avulla pääsin selville käytön perusteista. Helpin kautta opetteleminen oli hitaampaa.

ChucK, Impromptu ja SuperCollider käyttävät tekstipohjaista käyttöliittymää ohjelmoimiseen. Ohjelmointi näyttää siis samanlaiselta kuin perinteinen sovellusohjelmointi. Ero tulee siinä, että live coding -ohjelmoinnissa ohjelma on koko ajan ajossa, kun uutta koodia lisätään. Aloittelijan kannalta tekstipohjainen käyttöliittymä on helpompi, sillä oletettavasti live codingia aloitteleva henkilö on ohjelmoinut jotain entuudestaan. Haasteeksi jää vain mahdollisesti uuden ohjelmointikielen opettelu. Käytettävyyden parantamiseksi näiden kolmen tekstipohjaisen sovelluksen koodieditorit käyttävät korostuksia koodissa, jotta koodin lukeminen ja virheiden havaitseminen helpottuisi, kuten Nielsenin virheiden estämisen periaate kannustaa. Automaattista täydennystä editorit eivät kuitenkaan tarjoa.

Järjestelmän tilan näkyvyyden periaate on ohjelmilla hyvin hallussa. Jokainen vertailtu sovellus sisältää jonkinlaisen monitori-ikkunan, josta ohjelman toimintoja ja virheraportteja voi lukea. Testauksen aikana monitoreiden informaatio ei osoittautunut kovin tarpeelliseksi, mutta esimerkiksi ChucKin ja SuperColliderin tapauksissa ne tarjoavat hyvän paikan tarkistaa ohjelman alkuasetukset käynnistyksen yhteydessä. Ohjelman käytön vapauden periaate on myös toteutettu hyvin kaikilla neljällä sovelluksella, mitä sopii odottaakin live-tilanteessa käytettäviltä ohjelmilta. Sovellukset tukevat pikanappeja ja undo- ja redo-toimintoja järjestelmällisesti. Esteettisyyden ja minimaalisen suunnittelun periaate sen sijaan tuo esille eroja ohjel-

mien välillä. Impromptu ja ChucK loistavat tämän periaatteen valossa, sillä niiden hallitsemiseen tarvittavat työkalut ovat ohjelmassa minimaaliset. Esimerkiksi ChucKia voi ohjata koodin lisäksi kolmella napilla: lisää editoitava koodi ajoon, päivitä ajossa oleva koodin tai poista koodi ajosta. SuperCollider taas on hieman runsaampi käyttöliittymän tarjoamien hallintaohjaimien suhteen, sillä siinä ei ole erillisiä helposti nähtäviä nappeja koodin ajoon, vaan käyttäjä joutuu selaamaan ohjelman menuja löytääkseen halutut komennot. Pure Data sen sijaan joutuu graafisen tyylinsä takia tarjoamaan paljon enemmän ohjaimia ohjelman peruskäytön hallintaan, joten minimaalisuudessa tekstipohjaiset sovellukset Impromptu, ChucK ja SuperCollider ovat ymmärrettävästi etevämpiä.

Ohjelman käyttöliittymä vaikuttaa suoraan siihen päätökseen, ottaako käyttäjä ohjelman käyttöön, sillä käyttöliittymän kautta käyttäjä joutuu operoimaan päästäkseen haluttuun lopputulokseen. Aloittelijan näkökulmasta helposti opittava käyttöliittymä on paras, koska se antaa käyttäjälle nopeiten hyötyä, kuten Nilsenin käytettävyyden määritelmässä esitetään. Vertailluista käyttöliittymistä helpoiten lähestyttäviä ovat tekstipohjaiset Impromptu, ChucK ja SuperCollider tutun editointitapansa ansiosta. Näistä Impromptu nousee vielä hitusen helpommaksi sen takia, että siinä ei ole erillistä audiopalvelinta tai virtuaalikonetta, joka pitäisi käynnistää ennen kuin ääntä saa aikaiseksi. Impromptussa soitto voi alkaa heti, kun editorin ensimmäinen ääntätuottava lause on käännetty. Ohjelman arkkitehtuuri siis vaikuttaa käytettävyyteen, mikä selviää myös Bassin, Johnin ja Katesin laatimasta raportista [29]. Tekstipohjaisista ChucK ja Impromptu erottuvat vielä edukseen hyvin yksinkertaisella ja pelkistetyllä käyttöliittymällään.

6.3 Avustus ja tuki

Ohjelman tarjoama avustus jakaa sovellukset kahteen ryhmään. SuperCollider ja Pure Data sisältävät helpin, toisin kuin ChucK ja Impromptu, jotka eivät tarjoa perinteistä helppiä lainkaan ohjelman sisältä. Helpin puute hidastaa ohjelman opettelua, sillä pattitilanteessa apua joutuu kaivamaan muualta, kun taas helpin kanssa voi pyytää ohjelmalta apua juuri tarvitsemastaan asiasta tai aiheesta, mikä on käytettävyyden kannalta iso etu, kuten Nielsenin dokumentaatiota koskevassa heuristiikassa todetaan. SuperColliderin ja Pure Datan helpit toimivat juuri näin, eli melkein mistä tahansa kursorin osoittamasta kohdasta saa aukeamaan help-sivun. Näiden ohjelmien helpit ovat lisäksi interaktiivisia, eli jokainen help-sivu on oma ajettava dokumenttinsa, joten esimerkkien kokeilu onnistuu suoraan helpistä, mikä edelleen nopeuttaa oppimista.

ChucK ja Impromptu tarjoavat apuja ohjelman ulkopuolelta. ChucKin lataamisen yhteydessä saa kattavan manuaalin ja läjän kommentoituja esimerkkejä, ja lisäksi ChucKin kotisivuilta löytyy aloitusoppaita eri aihepiireistä. ChucKin manuaali [30] on hyvin perusteellinen. Se sisältää kaiken tarvittavan ohjelman asennuksesta ChucK-ohjelmointikielen syntaksiin ja live codingin perusteisiin. ChucKin kotisivuilta löytyvät samat ohjeet luonnollisesti html-muodossa. Yhdessä koodiesimerkkien kanssa apu ongelmatilanteisiin löytyy melko varmasti — joskin vaivalloisesti. Impromptu puolestaan tarjoaa ohjelmassaan vain funktiolistauksen ja näppäinkomennot, mutta nettisivuillaan [22] sillä on hyvin kirjoitettuja tutoriaaleja aloittelijoille ja näppäriä videotutoriaaleja jo ohjelman perusteet hallitseville. Sivustolta löytyvät esimerkit ovat hyvin kommentoituja, ja niissä annetaan myös vinkkejä, miten koodia kannattaa kokeilla muuttaa. Lisäksi Impromptun Scheme-ohjelmointikielen ongelmiin annetaan ratkaisuksi linkkejä kielen perusteisiin. Kuitenkin samoin kuin ChucKin tapauksessa myös Impromptun helpin selaaminen on vaivalloista, jos etsii ratkaisua tiettyyn ongelmaan, mikä Nielsenin dokumentaatioperiaatteen mukaisesti on käytettävyyden kannalta huono asia.

SuperColliderin helppi tarjoaa kattavan dokumentoinnin lisäksi oppaan ohjelman käyttöön, jossa lähdetään ihan ohjelmoinnin perusteista, joten aloituskynnys on tässä ohjelmassa matala samoin kuin ChucKissa ja Impromptussa. Helpin selain on kuitenkin avattava tekstikomennolla; ylävalikosta helppiä ei löydy, mikä on vastoin Nielsenin periaatetta avustuksen helposta saatavuudesta. Muut ohjelmat tarjoavat perinteisen help-menun tehtäväpalkista. SuperColliderin jokaiselta help-sivulta löytyy ajettavia esimerkkejä, ja koko helppi on html-muodossa, tosin linkit eivät minulla jostain syystä toimineet. Harmillisesti joskus pitkiltäkin helpsivuilta ei voi etsiä sanan mukaan, vaan sanahaku kohdistuu koko help-kirjastoon. SuperColliderin tapaan myös Pure Datasta löytyy selain helppiä varten, jossa on samaan tapaan opas ja ajettavia esimerkkejä, ja lisäksi html-muotoinen dokumentaatio. Pure Datan helppi on helpompi selata kuin SuperColliderin helppi, mutta SuperColliderin helppi-sivut ovat selkeämmin kirjoitettuja, mikä voi johtua osin tekstipohjaisen ohjelmoinnin selkeämmästä ulkoasusta verrattuna graafisen ohjelmoinnin esittämiseen.

Nielsenin dokumentaatioperiaatteessa varoitetaan liian laajasta dokumentaatiosta, joka voi ajaa käyttäjiä tiehensä, mutta siitä ei ole huolta yhdenkään vertaillun sovelluksen tapauksessa. Käytännössä ohjelmien dokumentaatiot pyrkivät kuvaamaan, mitä ohjelmalla voi tehdä ja miten, ja kaikki ylimääräinen on jätetty ohjelman yhteisön sivuille, keskustelupalstoille ja postilistoille. Kaikilla vertailluilla sovelluksilla on omat yhteisönsä, joten myös siltä suunnalta voi saada tukea. Jokaisella so-

velluksella on oma postituslistansa, ChucKilla ja Pure Datalla niitä on useampia eri tarkoituksiin. Lisäksi postilistoja karttaville löytyy kaikilta muilta paitsi Impromptulta oma nettifooruminsa. ChucK ja Pure Data tarjoavat myös julkista wikiä ongelmien selvittämiseen ja koodin kehittämiseen.

Vertailtaessa ohjelmien tarjoamaa avustusta ja tukea aloittelijan näkökulmasta, erottuvat SuperCollider ja Pure Data edukseen tarjoamalla perinteistä help-selainta ohjelman sisällä. Uuden ohjelmointikielen tai, Pure Datan tapauksessa, kokonaan uuden ohjelmointiparadigman (graafinen ohjelmointi) opetteleminen, jos käyttäjä on tottunut perinteiseen ohjelmointiin, on huomattavasti helpompaa, kun syntaksi ja funktiokutsut ja niiden selitykset löytyvät läheltä. Täytyy kuitenkin mainita, että vaikka Impromptu ei varsinaista helppiä tarjoakaan, niin sen tutoriaalit ja erityisesti videotutoriaalit ovat varsin hyvin tehtyjä ja opettavaisia, ja niillä Impromptu korvaakin osin varsinaisen dokumentoinnin suppeutta verrattuna muihin. ChucK sen sijaan tarjoaa kyllä hyvän dokumentaation sovelluksen käyttöön, mutta sen tutoriaalit ovat suppeammat.

6.4 Liitännäisyys

Erilaisten rajapintojen tukemisessa vertailluissa ohjelmissa ei ole paljon eroa. Kaikki neljä sovellusta tukevat MIDIä ja OSC:tä, kaikki muut paitsi ChucK tukevat VST:tä tai vastaavaa AU:ta ja vuorostaan HID:tä tukevat muut paitsi Impromptu. Liitännäisyys tavanomaisimpiin rajapintoihin on siis erinomainen Pure Datalla ja SuperColliderilla ja hyvä ChucKilla ja Impromptulla. Lyhyesti sanottuna MIDI (*Musical Instrument Digital Interface* [31]) ja OSC (*Open Sound Control* [32]) ovat rajapintoja, joiden avulla elektroniset instrumentit voivat keskustella keskenään ja ohjata toisiaan. VST (*Virtual Studio Technology* [33]) ja AU (*Audio Units* [34]) taas mahdollistavat ulkoisten sovellusten kuten instrumentti- ja efektiohjelmien käytön musiikkisovelluksissa, ja HID (*Human Interface Device* [35]) on rajapinta tietokoneen ja ihmisen ohjaaman laitteen välissä.

Vertaillut sovellukset pystyvät siis kaikki kommunikoimaan sellaisten laitteiden kanssa, jotka tukevat MIDI:ä tai OSC:tä. MIDI:n tapauksessa nämä laitteet ovat yleensä koskettimistoja ja OSC:n tapauksessa sensorilaitteita, jotka lukevat esimerkiksi liikesignaaleja ja muuntavat ne OSC-signaaleiksi. Vastaavasti HID-laitteiden, kuten hiiren tai kaukosäätimen, kanssa pystyvät kommunikoimaan muut paitsi Impromptu. VST- tai AU-tuki löytyy muista paitsi ChucKista, joten SuperCollider, Pure Data ja Impromptu voivat hyödyntää ulkopuolisia musiikkiohjelmia. Impromptun tapauksessa sovellus kykenee toimimaan vain AU-isäntänä, eli Impromptua ei

voi käyttää SuperColliderin tapaan AU-liitännäisenä jossain toisessa sovelluksessa tai kuten Pure Dataa voi käyttää VST-liitännäisenä. ChucKin kanssa on tyydyttävä ohjaamaan muita sovelluksia esimerkiksi MIDI:llä.

Sisääntulevien audiotiedostotyyppien tukemisessa SuperCollider erottuu muista hitusen tukemalla pakattua tiedostomuotoa flac:ia. Sen käytön hyöty on kuitenkin kyseenalainen, sillä käsitellessään flac:ia SuperCollider joutuu purkamaan sitä lennossa, jolloin kuluu ylimääräisiä resursseja. Flac:n lisäksi SuperCollider tukee wav-tiedostoja. Pure Data sen sijaan osaa lukea wav-, aif- ja au-tiedostoja, ChucK ymmärtää vain wav-tiedostoja ja Impromptu vain aif-tiedostoja. Kaikki vertaillut ohjelmat osaavat kirjoittaa audiota suoraan tiedostoon. Pure Data hallitsee tässäkin eniten tiedostomuotoja tukemalla wav-, aif- ja au-formaatteja. SuperCollider ja Chuck kirjoittavat wav-formaattiin ja Impromptu aif-formaattiin. MIDI-tiedostoja ohjelmat eivät tuota, vaikka tukevatkin MIDI-rajapintaa, sillä live codig -ohjelmien luonne on ymmärrettävästi täysin erilainen kuin perinteisen nuoteilla kirjoitetun kappaleen. Jokaisella sovelluksella on käytössään oma ohjelmointikielensä, joka tekstipohjaisissa sovelluksissa (SuperCollider, ChucK ja Impromptu) voidaan lukea mistä tahansa tekstitiedostosta, vaikka ohjelmat käyttävätkin omia tiedostopäätteitään. Pure Data käyttää ohjelmien tallettamiseen patch-tiedostoja, jotka ovat samoja kuin esikuvana toimineella sovelluksella MAXilla.

Aloittelijan näkökulmasta tarkasteltuna vertailtavat ohjelmat eivät eroa toisistaan liitännäisyyden osalta. Kaikki sovellukset tukevat hyvin perinteisimpiä rajapintoja, paitsi ChucK, jossa ei ole VST-tukea. Myöskään tuetuissa tiedostotyypeissä ei ole kovinkaan suurta eroa ohjelmien välillä. Pure Data tukee suurinta joukkoa eri audiotiedostoja, mutta muut ohjelmat tyytyvät toimimaan vain perinteisillä wav- ja aif-tiedostoilla. Dokumenttitiedostojen tallentaminen on myös samantapaista joka sovelluksella. Ainoana erona on tekstipohjaisten sovellusten, SuperColliderin, ChucKin ja Impromptun, luomien dokumenttien muokattavuus millä tahansa tekstieditorilla, mikä antaa niille pienen edun tässä suhteessa. Pienen siksi, että esiintymistilanteessa ulkoisten tekstieditorien käyttö ei tiettävästi onnistu samaan tapaan kuin sisäisten editorien, sillä sisäisestä editorista koodin voi ajaa suoraan tallentamatta, mutta ulkoisella editorilla tiedoston joutuu ensin tallentamaan, ja sitten vasta ajamaan, mikä hidastaa käsittelyä. Ulkoista editoria voi kuitenkin käyttää vaivatta kaikkeen muuhun editointiin kuten esityksen alustustiedostojen rakentamiseen.

	käyttöönotto	käyttöliittymä	avustus ja tuki	liitännäisyys
Impromptu	3	3	2	2
SuperCollider	2	2	3	2
ChucK	3	2	2	1
Pure Data	1	1	3	2

Taulukko 1: Live coding -ohjelmien helppokäyttöisyys eri osa-alueilla asteikolla 1-3.

6.5 Johtopäätökset

Vertailtujen neljän live coding -sovelluksen suoriutuminen valitulla neljällä osa-alueella on esitetty taulukossa 1. Numeroiden merkitykset ovat: 1 välttävä, 2 hyvä ja 3 kiitettävä. Suurimmat erot tulivat ilmi käyttöönotossa ja käyttöliittymässä, joissa aloittelijan kannalta helppokäyttöisimmäksi ja nopeiten omaksuttavimmaksi osoittautui Impromptu. Impromptun käyttöönotto sujuu ilman asennusta, ja käynnistyksen jälkeen ohjelma on valmis tuottamaan ääntä. Lisäksi ohjelman pelkistetty käyttöliittymä keskittää käyttäjän huomion tärkeimpään, eli tekstieditoriin, jolla koodi musiikin tuottamiseksi kirjoitetaan, mutta silti käyttöliittymä tarjoaa käytettävyyden kannalta olennaiset osat mieluisaan käyttökokemukseen. Ainoa rajoittava tekijä Impromptun käyttöönotossa on sen saatavuus vain Mac OS X -käyttöliittymälle.

Avustuksen ja tuen osalta tilanne on eri sovellusten välillä tasaisempi. Aloittelijalle parasta avustusta ja tukea vertailluista ohjelmista tarjoavat SuperCollider ja Pure Data, jotka erottuvat muista tarjoamalla perinteisen sovelluksensisäisen helpin. Molempien ohjelmien helpin interaktiivisuus, eli jokainen help-sivu on oma ajettava dokumenttinsa, toimii erinomaisesti uuden opettelussa ja esimerkkien kokeilussa. ChucK ja Impromptu eivät kuitenkaan häviä paljon kahdelle edelliselle, sillä niiden tarjoama dokumentointi ja opastus Internet-sivuillaan on vähintään yhtä pätevää. Verkko-oppaiden käytettävyys vaan ei ole yhtä hyvä kuin ohjelmaan istutetun help-selaimen.

Liitännäisyydessä eroja ohjelmien välille ei syntynyt paljoakaan. Sovellukset tukevat yleisimpiä rajapintoja (MIDI, OSC, VST, HID) ulkoisten laitteiden ohjaukseen ja lukemiseen lukuun ottamatta ChucKia, joka ei tue VST-rajapintaa. Tiedostotyyppien tukemisessa ohjelmat ovat myös samalla viivalla: ohjelmat osaavat lukea ja kirjoittaa pakkaamatonta audiota ja käyttävät dokumenttitiedostoinaan omia formaattejaan. Tekstipohjaisten SuperColliderin, ChucKin ja Impromptun dokumenttitiedostot ovat kuitenkin muokattavissa millä tahansa tekstieditorilla. Aloittelijan kannalta jokainen vertailluista sovelluksista on liitännäisyyden kannalta yhtä helposti lähestyttävä, sillä esimerkiksi suositun mp3-formaatin tukea ei oletuksena löytynyt

yhdestäkään ohjelmasta.

Kokonaisuudessaan kaikki vertaillut osa-alueet huomioiden Impromptu on ohjelmista sopivin aloittelijoille tämän tutkimuksen perusteella. Impromptu pärjää erityisesti käyttöönotossa ja käytettävyydessä. Jos käytettävissä ei kuitenkaan ole Mac OS X -käyttöjörjestelmää, jota Impromptu vaatii, on seuraavaksi paras vaihtoehto live coding -ohjelmaksi SuperCollider. Se tarjoaa nopeasti omaksuttavan perinteisen asiakas-palvelin-mallin ja hyvän avustuksen ja tuen käyttäjälle. Kovin paljon SuperColliderista ei jää ChucK, jonka parhaita puolia on yksinkertainen käyttöliittymä ja kattava manuaali. Vaikeimmin lähestyttäväksi ohjelmaksi aloittelijalle paljastui Pure Data, jonka graafinen käyttöperiaate on hyvin vieras perinteiseen ohjelmointiin tottuneille. Pure Datan vahvuuksia ovat kuitenkin hyvä avustus ja tuki sekä sen suosiosta johtuva kattava liitännäisyys eri ohjelmistoihin.

7 Yhteenveto

Tämän tutkimuksen tarkoituksena oli löytää aloittelijan näkökulmasta paras live coding -sävellysympäristö vertaillusta ohjelmien joukosta. Ensin kartoitettiin live coding -käsitettä ja live codingin taustoja ja esiteltiin käytettävä vertaileva tutkimusmenetelmä. Tämän jälkeen kuvattiin neljän ohjelman muodostama aineisto ja neljä eri vertailtavaa piirrettä. Sitten suoritettiin varsinainen vertaileva analyysi käyttäen hyväksi Jakob Nielsenin käytettävyysheuristiikoita. Lopuksi esiteltiin tutkimuksen johtopäätökset.

Tutkimuksessa selvisi, että Impromptu on vertailtavista ohjelmista aloittelijaystävällisin. Suurimmat erot ohjelmien välille syntyivät käyttöönotossa ja käyttöliittymässä. Käyttöönotossa ohjelmat erosivat eniten asennuksen tavassa ja käyttövalmiuteen johtavien vaiheiden määrässä. Käyttöliittymässä ohjelmia jakoi erilleen eniten graafisten ja tekstipohjaisten ohjelmien käyttötapojen ero. Avustuksen ja tuen ja liitännäisyyden osalta erot ohjelmien välillä olivat pienempiä. Tutkimus paljasti myös, että tekstipohjaiset live coding -sovellukset ovat sopivampia aloittelijalle kuin graafisen ohjelmoinnin sovellukset. Kvalitatiivisen tutkimuksen luonteen mukaisesti tutkimustulos on riippuvainen siitä, mitä aineistoa (luku 4) on otettu mukaan ja mihin asioihin (luku 5) on kiinnitetty huomiota. Lisäksi tulokseen vaikuttaa menetelmä (luku 3), jolla tutkimus on tehty. Subjektiivisten mieltymysten vaikutusta tuloksiin pyrittiin pienentämään kiinnittämällä vertailtavat piirteet tunnettuihin käytettävyysheuristiikkoihin.

Luonnollinen jatko tutkimukselle olisi muiden live coding -ohjelmien vertailu samoilla kriteereillä. Kaupallisia ja avoimia live coding -sovelluksia on tarjolla

useita niin äänen kuin videonkin tekemiseen. Aloittelijanäkökulman parantamiseksi olisi hyvä suorittaa käyttötestaus. Ohjelmien vertailua voisi myös laajentaa muihin piirteisiin kuten ohjelmointikielen tehokkuuteen, ohjelman suorituskykyyn ja äänenlaatuun. Lisäksi olisi mielenkiintoista tutkia, vaikuttaako musiikillinen tietämys live coding -ohjelmien käyttöön. Onko live coding vain koodaamista vai tuoko musiikillinen intuitio siihen jotain lisää?

Lähteet

[1] Ulla Pohjannoro, *Tietokonemusiikki, Sibelius Akatemia*, saatavilla WWW-muodossa <URL: `http://www2.siba.fi/historia/1900/-germaaniartikkelit/tietokonemusiikki_germ.html`>, viitattu 8.3.2009.

[2] Ulla Pohjannoro, *Elektronimusiikki, Sibelius Akatemia*, saatavilla WWW-muodossa <URL: `http://www2.siba.fi/historia/1900/-germaaniartikkelit/elektronimusiikki_germ.html`>, viitattu 8.3.2009.

[3] Center for Audio Recording Arts, *Elements of sound - Series 1*, saatavilla WWW-muodossa <URL: `http://cara.gsu.edu/courses/MI_3110/sound1/-elementsofsound1.htm`>, viitattu 27.3.2009.

[4] Kristine H. Burns, *History of electronic and computer music including automatic instruments and composition machines*, saatavilla WWW-muodossa <URL: `http://eamusic.dartmouth.edu/~wowem/electronmedia/-music/eamhistory.html`>, viitattu 8.3.2009.

[5] Eric Kühn, *A Brief History of Computer Music*, saatavilla WWW-muodossa <URL: `http://music.calarts.edu/~eric/cm.html`>, viitattu 19.11.2009.

[6] Iohannes M. Zmölnig ja Gerhard Eckel, *Live Coding: An Overview*, University of Music and Dramatic Arts, Graz Institute of Electronic Music and Acoustics, Itävalta, 2007, s. 1-3, saatavana PDF-muodossa <URL: `http://iem.at/projekte/publications/paper/live/live.pdf`>.

[7] Ge Wang ja Perry R. Cook, *On-the-fly Programming: Using Code as an Expressive Musical Instrument*, In Proceedings of the

2004 International Conference on New Interfaces for Musical Expression (NIME), USA, 2004, s. 1-2, saatavana PDF-muodossa <URL: http://soundlab.cs.princeton.edu/publications/on-the-fly_nime2004.pdf>.

[8] TOPLAP-yhteisö, *TOPLAP-kotisivu*, 1.9.2009, saatavilla WWW-muodossa <URL: http://toplap.org/index.php/Main_Page>.

[9] TOPLAP-yhteisö, *TOPLAP Manifesto*, 14.7.2008, saatavilla WWW-muodossa <URL: http://toplap.org/index.php/ManifestoDraft>.

[10] Nick Collins, *Some thoughts on live coding (of music)*, 21.8.2008, saatavana WWW-muodossa <URL: http://www.toplap.org/index.php/Some_thoughts>.

[11] Kerry Walk, *How to Write a Comparative Analysis*, Writing Center at Harvard University, 1998, saatavilla WWW-muodossa <URL: http://www.fas.harvard.edu/~wricntr/documents/-CompAnalysis.html>.

[12] TOPLAP-yhteisö, *TOPLAP Systems*, 12.11.2009, saatavilla WWW-muodossa <URL: http://toplap.org/index.php/ToplapSystems>.

[13] Pekka Räsänen, Anu-Hanna Nattila, Harri Melin, *Tutkimusmenetelmien pyörteissä*, PS-kustannus, Jyväskylä, 2005, s. 59.

[14] Ge Wang ja Perry R. Cook, *ChucK : Strongly-timed, Concurrent, and On-the-fly Audio Programming Language*, saatavilla WWW-muodossa <URL: http://chuck.cs.princeton.edu/>, viitattu 8.3.2009.

[15] Ge Wang ja Perry R. Cook, *ChucK: A Concurrent, On-the-fly, Audio Programming Language*, Computer Science Department, Princeton University, In Proceedings of the 2003 International Computer Music Conference, 2003, saatavilla PDF-muodossa: <URL: http://soundlab.cs.princeton.edu/publications/-chuck_icmc2003.pdf>.

[16] Spencer Salazar ja Ge Wang, *miniAudicle (windows)*, saatavilla WWW-muodossa <URL: http://audicle.cs.princeton.edu/mini/windows/>, viitattu 19.11.2009.

[17] James McCartney, *The SuperCollider home page*, saatavilla WWW-muodossa <URL: http://www.audiosynth.com/>, viitattu 8.3.2009.

[18] Dan Palkowski, Fredrik Olofsson, *SuperCollider yhteisösivut*, 19.11.2009, saatavilla WWW-muodossa <URL: http://supercollider.sourceforge.net/>.

[19] Miller S. Puckette, *Pure Data: another integrated computer music environment*, 1996, toimittanut Hans-Christoph Steiner, 6.5.2007, saatavana WWW-muodossa <URL: http://puredata.info/docs/articles/puredata1997>.

[20] Iohannes M. Zmölnig, *Pure Data -kotisivut*, 3.8.2008, saatavilla WWW-muodossa <URL: http://puredata.info/>.

[21] Andrew Sorensen, *Impromptu: An interactive programming environment for composition and performance*, Australasian Computer Music Conference, 2005, Brisbane: ACMA, saatavilla PDF-muodossa <URL: http://impromptu.moso.com.au/extras/sorensen_acmc_05.pdf>.

[22] Andrew Sorensen, *Impromptu -kotisivut*, saatavilla WWW-muodossa <URL: http://impromptu.moso.com.au/>, viitattu 8.3.2009.

[23] Roger Lee,*Software Engineering Research, Management and Applications*, Springer, 2008, s. 31-35.

[24] A. Dillon, *User Interface Design, MacMillan Encyclopedia of Cognitive Science, Vol. 4*, MacMillan, London, 2003, s. 453-458.

[25] J. Nielsen ja R.L. Mack, *Usability Inspection Methods*, John Wiley & Sons, New York, NY, 1994, s. 30.

[26] Jakob Nielsen, *Usability Engineering*, Morgan Kaufmann, San Francisco, 1994, s. 26.

[27] Nick Nilson, *Live Coding Practice*, University of Sussex, Department of Informatics, Brighton, 2007, saatavilla PDF-muodossa: <URL: http://www.informatics.sussex.ac.uk/users/nc81/research/-livecodingpractice.pdf>.

[28] Robert haskell, *Transfer of learning: cognition, instruction, and reasoning*, Academic Press, 2001, s. xiii.

[29] Len Bass, Bonnie E. John ja Jesse Kates, *Achieving Usability Through Software Architecture*, Carnegie Mellon University, Software Engineering Instirute, 2001, s. 82, saatavilla PDF-muodossa <URL: http://www.sei.cmu.edu/reports/01tr005.pdf>.

[30] Ge Wang ja Perry Cook, *The ChucK Manual*, 2007, saatavilla PDF-muodossa <URL: http://chuck.cs.princeton.edu/release/-files/chuck_manual.pdf>.

[31] Joseph Rothstein, *MIDI: a comprehensive introduction, 2nd ed.*, A-R Editions, Wisconsin, 1995, s. 1.

[32] Matthew Wright, *Open sound control: an enabling technology for musical networking*, Organised Sound -lehti vsk/osa:10 iss:3, 2005, s. 193.

[33] Christos Karavasileiadis ja Stephan O'Bryan, *Modeling a VST Synthesizer*, 2008, s. 30, saatavilla PDF-muodossa <URL: http://rudar.ruc.dk/bitstream/1800/3246/1/-Group12%20project.pdf>.

[34] Apple Inc., *Core Audio Overview*, 2007, s. 10, saatavilla PDF-muodossa: <URL: http://developer.apple.com/mac/library/-documentation/MusicAudio/Conceptual/CoreAudioOverview/-CoreAudioOverview.pdf>.

[35] USB Implementers Forum, *Device Class Definition for Human Interface Devices (HID)*, 2001, s. 1-2, saatavilla PDF-muodossa: <URL: http://www.usb.org/developers/devclass_docs/HID1_11.pdf/>.