

Musiikkia kuvista

Lauri Gröhn
metasäveltäjä
Gallery Music from Pictures
www.synesthesia.com

1 Johdanto

Kehittämässäni menetelmässä musiikkia (midi-tiedostoja) generoidaan mistä tahansa kuvista. Menetelmän metaforana voidaan pitää kuvanveistoa, jossa lähdetään kivenkappaleesta tai puunpötkelöstä, joka inspiroi kuvanveistäjää tiettyyn ratkaisuun. Esiteltävä menetelmä perustuu kuvan pikseleiden poissuodattamiseen ja toisaalta useiden musiikillisten valintaparametrien käyttöön. Edelleen metaforaa käyttäen parametrien käyttö on ikään kuin veistosta katselisi eri kuvakulmissa, eri valaistuksessa tai eri ympäristöissä. Kunkin sävellyksen pohjana oleva kuva ja parametriluettelo ovat yhdessä sävellyksen metapartituuri. On vain löydettävä kuvia, josta generoituu mielenkiintoista musiikkia...

Sivuillani www.synesthesia.com on satakunta sävellystä (mp3, midi), joiden generoinnissa käytettyjen kuvien kirjo on laaja: maisemia, maalauksia, piirroksia, viivakoodi, Nokian logo, EU:n lippu, euron kolikko, kassakuitti, WTC:n rauniot, putoava avaruussukkula, käsialanäyte, muotokuva, koira, revontulet ja niin edelleen. Kuva vastaa jossakin mielessä normaalin sävellyksen nimeä tai ensiesitykseen laadittavaa teoksen esittelyä.

Synesthesia on säveltämisen kehitysjärjestelmä, jossa ei ole omaa käyttöliittymää. Ohjelmaa ajetaan JBuilder Java-kehitysympäristön tai muun vastaavan avulla. Tyypillinen konfiguraatio sisältää JBuilderin käyttöliittymän, kuvakansion ja midi-tiedostokansion (liite 1). Kun parametrit on valittu, generoi ohjelma käynnistyksen jälkeen midi-tiedoston muutamassa sekunnissa. (liite 2).

Toistaiseksi en ole suorittanut midi-tiedostojen jälkikäsitteilyä. Raakapartituurista näkee, että jos musiikki halutaan saada muusikoiden kannalta esittämiskelpoiseksi, on midi-tiedostosta saatavaa raakapartituuria (liite 3, liite 5) usein muokattava käyttökelpoisempaan muotoon (liite 4).

Generoinnissa syntyy kokonaisia sävellyksiä, jotka ovat parametreista riippuen minimalistisia, new age –tyylisiä tai vahvasti klusteroituja, genrenä lähinnä ”klassinen nykymusiikki”. Muodostamalla sopivia parametrijoukkoja voidaan rakentaa erilaisia tyyliä tai ”säveltäjäprofiileja”. Sävellysten ovat olleet minuutista 10 tuntiin, Ohjelmassa ei ole lainkaan käytetty satunnaisuutta. Synesthesia-ohjelman spin-offeja voidaan lähteä rakentamaan eri suuntiin (liite 6).

2 Kuvan suodatus

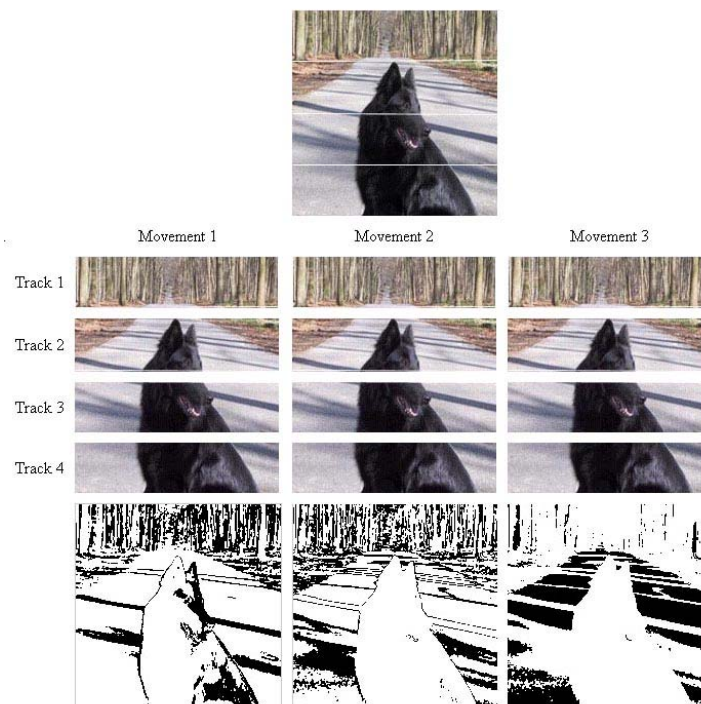
Ohjelmassa valitun kuvan kunkin pikselin r/g/b –arvot lasketaan yhteen (moodi 0) tai kerrotaan keskenään (moodi 1) ja jokaisella suodatuskerralla (parametri) ne pikselit jotka ovat johtaneet yleisimmin esiintyvään arvoon ”veistetään pois”. Kuvassa 1 ylinnä alkuperäinen, seuraavana

moodissa 0 suodatettu 10 ja 400 kertaa ja alinna moodissa 1 suodatettu 100 ja 400 kertaa. Suodatuksessa jäljelle jääneitä (jatkossa käytettävissä olevia) harmaasävy pikseleitä on kuvassa merkitty mustalla. Useimmiten näistäkin käytetään musiikkia generoitaessa alle 10 prosenttia.



3 Kuvan pilkkominen

Raitojen (track, parametri) määrästä riippuen kuva jaetaan pystysuunnassa yhtä suuriin lohkoihin



ja osien (parametri) määrästä riippuen kunkin osan pikseleinä käytetään käytössä olevien pikselien määrää jaettuna osien määrällä. Osassa 1 käytetään tummia pikseleitä ja viimeisessä vaaleimpia. Aika kulkee vasemmalta oikealle. Yleisin väri (musta) on suodattanut pois. (Kuvassa harmaasävy pikselit on merkitty mustalla.)

4 Parametreista

Instrumentit, volyymit jne. voidaan valita parametreilla tai parametrijoukoilla jokaiselle raidalle erikseen. Harmoniat syntyvät raidalle vertikaalinuottien määrästä, minimi-intervallista ja sävelalan laajuudesta.

```

7 // Main parameters =====//default =====
8 int nmov      = 3;          //3: structures (~ movements, piece length)
9 int ntracks   = 8;          //9: number of instruments (tracks) used
10 int mpitch    = 106;        //110: highest pitch of the first track (defines also key
11 int pdi       = 6;          //6: pitch difference to the upper track (harmony)
12 int []pdiff   = {0,pdi,pdi,pdi,pdi,pdi,pdi,pdi,pdi,pdi,pdi,pdi,pdi,pdi,pdi};
13 int thm       = 50;        //36: pitch range of tracks
14 int [] trackhmin = {thm,thm,thm,thm,thm,thm,thm,thm,thm,thm,thm,thm,thm,thm,thm};
15 int mnt       = 1;          //1: maximum number of notes in chords in one track
16 int [] maxnotes = {mnt,mnt,mnt,mnt,mnt,mnt,mnt,mnt,mnt,mnt,mnt,mnt,mnt,mnt,mnt};
17 int itv       = 2;          //1: minimum interval in chords
18 int [] minitv  = {itv,itv,itv,itv,itv,itv,itv,itv,itv,itv,itv,itv,itv,itv,itv};
19 int ins       = 0;          //12: instrument (patch)
20 int [] instrum = {ins,ins,ins,ins,ins,ins,ins,ins,ins,ins,ins,ins,ins,ins,ins};
21 boolean instrumsel = true;  //true:instruments set by SW

```

Jos maksimi sävelkorkeudeksi (parametri) on asetettu midi-arvo 96 ja eri raitojen säveltasojen erotukseksi 6, on ylimmän raidan korkein säveltaso 96, seuraavan 90, seuraavan 84 jne. Ohjelmaan sisältyy taulukko, jossa kukin instrumentin korkein ja mahdollinen nuotti on määritetty.

5 Pikseleiden lukeminen

Pikseleitä (harmaasävyiksi muutettuja) käydään läpi vertikaalisesti ylhäältä alas yksi raita kerrallaan valitun harmonian (parametri i) ja soinnun ”paksuuden” (parametri m) edellyttämällä tavalla. Jos kyseisellä raidalta tai kyseisestä osasta ”puuttuu” pikseli siirrytään matkan i verran alaspäin. Raitaa käydään läpi vasemmalta oikealle niin monta kertaa kuin sävellyksessä on osia (parametri) läpikäyden aina kyseeseen osaan kuuluvia pikseleitä. Aika-akseli kulkee siis vasemmalta oikealle ja osien määrä tempon (parametri) ohella määrää sävellyksen pituuden.

| pikseleitä | i=2* m=4 | i=3* m=1 |
|------------|-------------|-------------|
| * | x | x |
| * | | x |
| * | | x |
| * | | |
| * | x | |
| * | | |
| * | | |
| * | | |
| * | | |
| * | | |
| * | | |
| * | | |
| * | | |
| * | | |
| ... | | |

6 Sävelkorkeudet

Sävelkorkeudet määräytyvät koko sävellyksen korkeimmasta mahdollisesta nuotista (parametri), kunkin raidan suhteellisesta säveltasosta ylempään verrattuna (parametri) ja valitusta asteikosta, joita voidaan muodostaa mielivaltaisesti lisää:

```

47 // Scales =====
48 int[] vscale [] = {
49 {0,1,2,3,4,5,6,7,8,9,10,11}, //#0 chromatic
50 {0,0,2,2,4,5,5,7,7,9, 9,11}, //#1 major
51 {0,0,0,2,2,5,5,5,7,7, 9, 9}, //#2 pentatonic
52 {0,0,2,3,3,5,6,6,8,9, 9,11}, //#3 octatonic
53 {0,0,2,2,4,4,6,6,8,8,10,10}, //#4 whole-tone
54 {0,0,0,3,3,5,5,5,6,6,10,10}, //#5 sakura
55 {0,0,0,0,0,0,0,0,0,0, 0, 0}, //#6 Scelsi
56 {0,1,1,3,4,4,6,7,7,9, 9,11}, //#7 Arabic
57 {0,1,2,2,5,5,6,7,7,8,11,11}, //#8 Messiaen mode 4
58 {0,0,2,2,3,5,5,6,6,9, 9,11}, //#9 harmonic minor
59 {0,0,0,0,0,0,1,1,1,1, 1, 2}, //#10 eyes wide shut
60 {0,0,0,0,0,0,0,0,0,0, 0,11}}; //#11

```

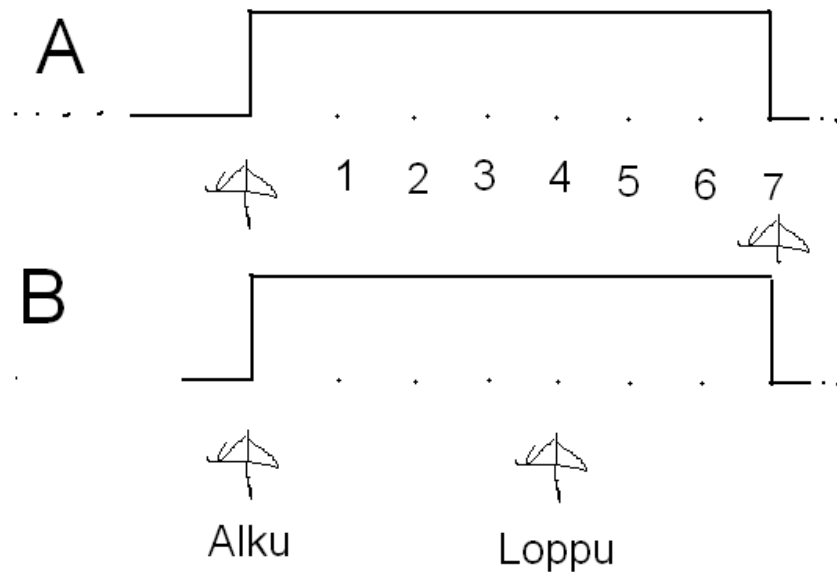
Sävelkorkeudet ja volyyymi saavat arvoikseen midissä käytettäviä arvoja. Jokaisella raidalla voi olla oma sävelasteikkonsa. Ohessa esimerkki pikseleiden kuvautumisesta säveltasoiiksi:

| pikseleitä | kromaattinen | Cduuri | Scelsi |
|------------|--------------|--------|--------|
| ... | | | |
| * | C1 | C1 | C1 |
| * | H | H | C |
| * | A# | A | C |
| * | A | A | C |
| * | G# | G | C |
| * | G | G | C |
| * | F# | F | C |
| * | F | F | C |
| * | E | E | C |
| * | D# | D | C |
| * | D | D | C |
| * | C# | C | C |
| * | C | C | C |
| ... | | | |

Huomattakoon, että valitsemalla alkuperäisestä kuvasta kaksi mielivaltaista pikseliä ei ole mahdollista a priori sanoa kumpi niistä ”soitetaan” ensin ja kumman säveltasoa on korkeampi.

7 Nuottien kestot

Nuotti voi syttyä kun tietyn raidan ja tietyn osan kohdalle osuu ”nouseva reuna” eli vaakatasoa seuraten löytyy pikseli, jota edeltävä pikseli ei ollut käytettävissä. Nuottien kesto määräytyy kunkin raidan vaakasuunnassa käytettävissä olevien perättäisten lukumäärän perusteella. Kunkin raidan nuoteille voidaan asettaa maksimikesto (parametri). Esimerkissä A tuo parametri on yli 7 ja esimerkissä B parametri on 4. Yksikkönä on mininuotti (parametri), joka voi olla 1/8, 1/16 tai 1/32 -nuotti.



8 Volyymit

Kunkin nuotin volyyymi määräytyy seuraavalla tavalla:

```

30 int vom          = 77;          //77: maximum volume
31 int [] volmax    = {vom,vom,vom,vom,vom,vom,vom,vom,vom,vom,vom,vom,vom,vom,vom};
32 int vol          = 1;          //77: minimum volume
33 int [] voltracks = {vol,vol,vol,vol,vol,vol,vol,vol,vol,vol,vol,vol,vol,vol,vol};
34 int vlx         = 1;          //2: crescendo slope (cyclic)
35 int [] volx     = {vlx,vlx,vlx,vlx,vlx,vlx,vlx,vlx,vlx,vlx,vlx,vlx,vlx,vlx,vlx};

193 volumeset = voltracks[tr] + (pp[x][y]+volx[tr]*x)*(127-voltracks[tr]);
194 if (volumeset > volmax[tr])
195     volumeset = volmax[tr];
196 varLen(tr,dt[tr]*minnote);          //time
197 sabyte(tr,0x90+tr);                //track
198 sabyte(tr,note(tr,y));              //pitch
199 sabyte(tr,volumeset);              //volume

```

Keinotekoinen crescendo on kokeiltavana ja jää mahdollisesti jatkossa pois ellei se osoittaudu hyödylliseksi ja ellei sitä voida laskennallisesti kiinnittää johonkin kuvan ominaisuuteen.

9 Muutamia erikoisuuksia

Koska musiikkia voidaan generoida mistä tahansa kuvasta, ovat muutamat yksityiskohdat tuottaneet melkoisesti lisätyötä. Kuvasta riippuen on mahdollista, että musiikkiin tulee liian pitkiä taukoja. Ne on eliminoitu parametrilla, joka määrittelee sallitun maksimitauon. Liian pitkän tauon uhatessa ohjelma supistaa tauon parametrin ilmaisemaan pituuteen. Sävellyksen alun ja lopun pikselit on suodatettava ylimääräisellä tavalla, jottei sävellys ehkä alkaisi voimakkaalla purskeella tai lopu ikään kuin seinään.

10 Lopuksi

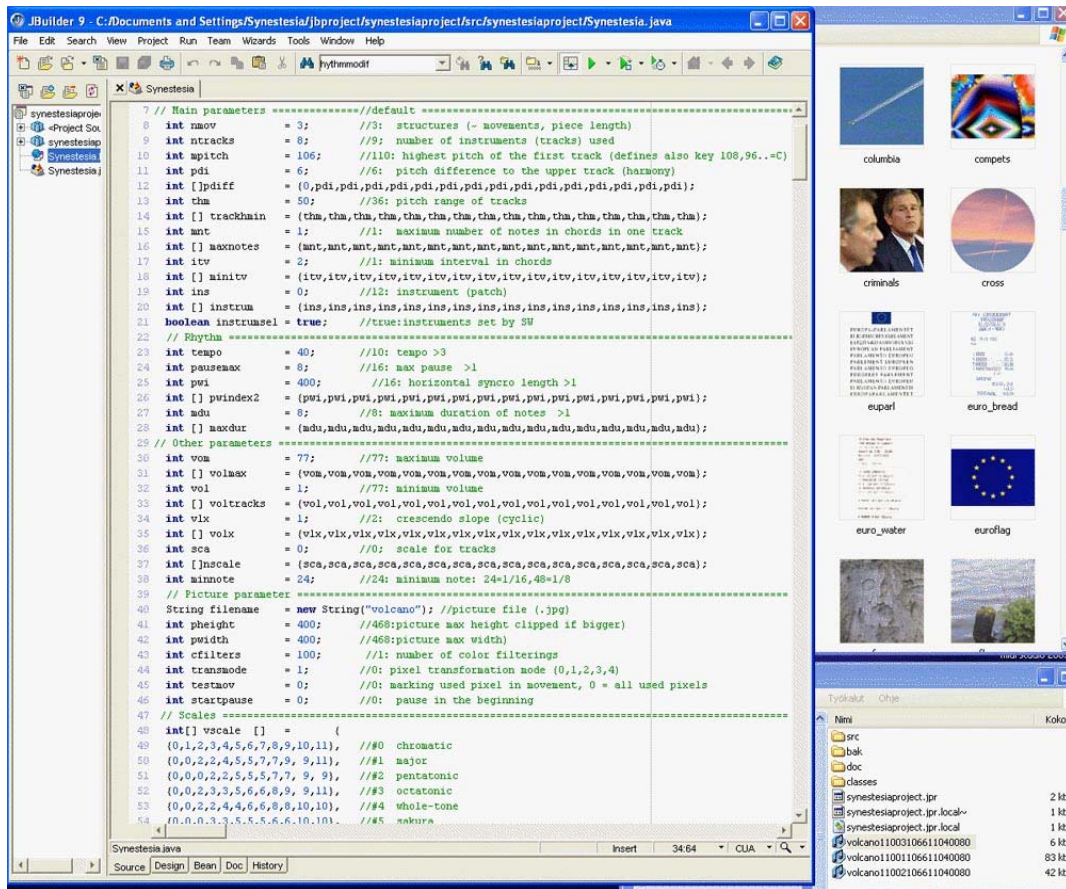
Synesthesia-ohjelmassa on Java-koodia nykyisellään runsaat 500 riviä ja koska käyttöliittymää ei ole, ei olio-ohjelmointia ole toteutettu taiteen sääntöjen mukaisesti enkä edes nykyosaamisellani pystyisi siihen. Kehitysympäristönä ohjelma on toiminut mainiosti, teknisiin rakenteisiin ei ole tarvinnut puuttua. Suoritin ainoa uudelleenohjelmoinnin syksyllä 2001. Viitasaaren sävellyskursseilla saadut muutosehdotukset olen pystynyt toteuttamaan muutaman rivin lisäyksillä tai muutoksilla.

Vaikeimmaksi haasteeksi on osoittautunut midin sävelkorkeudettoman rytmisetin hyödyntäminen. Useiden yritystenkään jälkeen ei sopivia algoritmeja ole löytynyt. Sellaisten löytäminen voisi tuoda mahdollisuuksia siirtyä populaarimpaan suuntaan.

Aivan uutena mahdollisena haasteena voisi olla kehittää käänteinen ohjelma, joka tekisi staattisen kuvan midi-tiedostosta. Näin saatuja kuvia voisi käyttää uusien midi-tiedostojen generointiin, jolloin syntyisi eräänlainen soluautomaatti joka tuottaisi rajattomasti musiikkia.

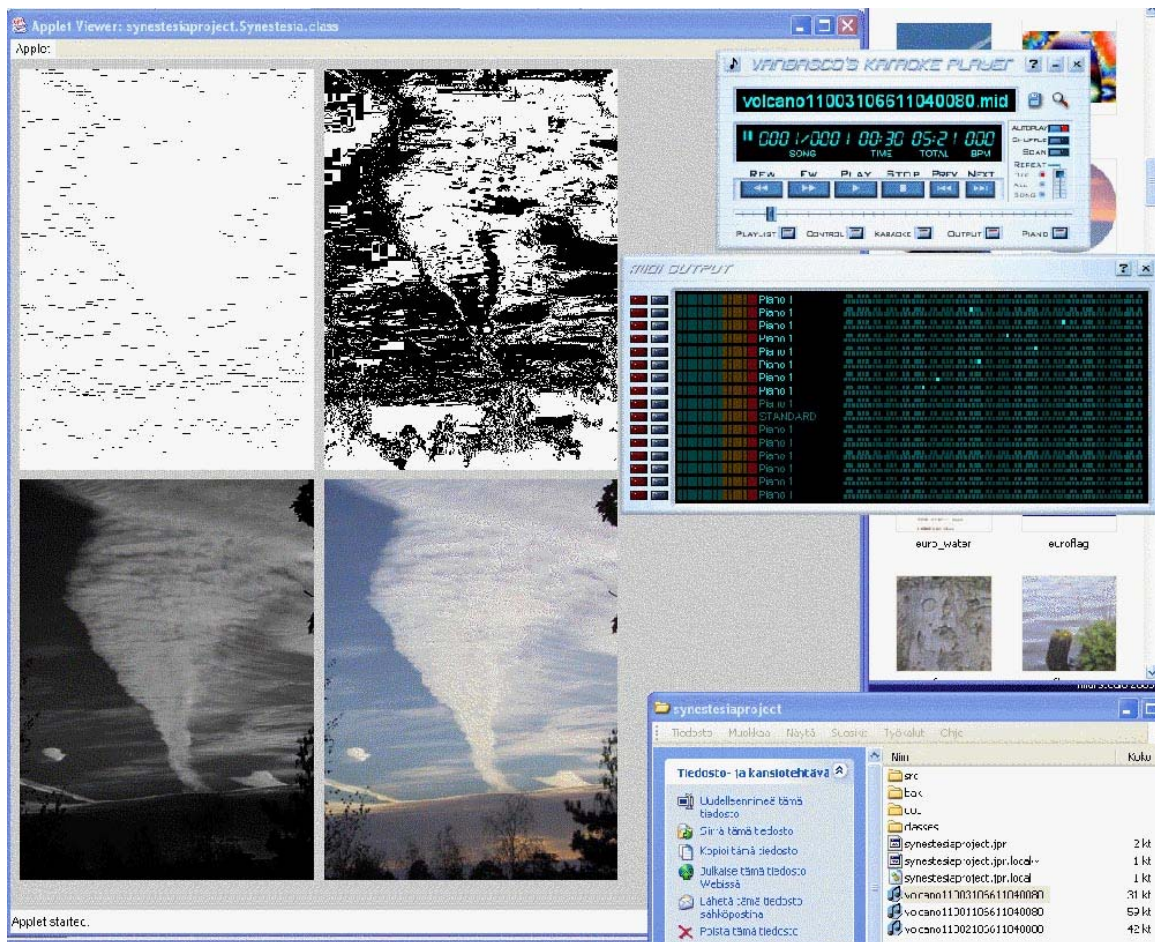
Kehitetyn ohjelman avoin lähdekoodi (kevään 2002 versio) löytyy nettisivulta www.synesthesia.com kuten myös lukuisa joukko midi- ja mp3-tiedostoja kuvineen ja joitakin dokumentteja.

Liite 1: Kehitysympäristö käyttöliittymä parametrien valintavaiheessa



Keskellä Java-lähdekoodia JBuilder 9 ympäristössä. Generointi käynnistyy vasemman olevasta sinisellä maalatusa kohdasta ja generointi kestää muutaman sekunnin. Oikealla pala kuvahakemistoa, josta haluttu kuva (.jpg) voidaan poimia ja alhaalla oikealla kansio, johon midi-tiedosto generoituu.

Liite 2: Näyttö generoinnin tapahduttua



Alhaalla vasemmassa puoliskossa on alkuperäinen kuva, siitä muodostettu harmaasävykuva, josta varsinaisesti suodatus alkaa, ylempänä oikealla suodatettu kuva (kaikki harmaasävyt merkitty mustalla) ja ylävasemmalla kuvan ne pikselit (merkitty mustalla), joita tässä generoinnissa on käytetty musiikin generointiin. Kuvasta ei voi lukea sävelkorkeuksia kuin suuntaa-antavasti eikä kuvasta näy missä sävellyksen osassa kutakin pikseliä on käytetty, eikä siis ajanhetkeä, vaikka aika kulkeekin vasemmalta oikealta. Testiparametrin muutoksella saadaan haluttaessa erikseen näkyviin myös kussakin osassa käytetyt pikselit.

Liite 3: Midi-tiedostosta saatu raakapartituuri, esimerkkinä ”Time is Up”, sivu 1/16

[Title]

[Composer]

Track 1

Track 2

Track 3

Track 4

Track 5

Track 6

Track 7

Track 8

Track 9

Liite 4: Raakapartituurista muokattu lähes muusikoiden soitettavissa oleva partituuri, sivu 1/16

Time is Up
18.3.2003

Synesthesia Software
& Lauri Gröhn

♩ = 60

The score is for the piece "Time is Up" by Synesthesia Software & Lauri Gröhn, dated 18.3.2003. It is in 3/8 time with a tempo of 60 beats per minute. The score consists of five staves: Flute, Bells, Drums, Piano, and a fifth empty staff. The Flute part starts with a forte (*f*) dynamic and plays a melodic line. The Bells part starts with a mezzo-forte (*mf*) dynamic and plays a rhythmic pattern. The Drums part starts with a piano (*p*) dynamic and plays a steady drum pattern. The Piano part starts with a piano (*p*) dynamic and plays a harmonic accompaniment. A small red padlock icon is visible in the top right corner of the score area.

Liite 5: Midi-tiedostosta saatu raakapartituuri, osa

This is a MIDI score for a piece, likely a piano solo. It consists of three staves: a treble clef staff, a grand staff (treble and bass clefs), and a bass clef staff. The music is in 3/8 time and features a complex, flowing melodic line in the treble clef staff, accompanied by a rhythmic accompaniment in the grand staff and bass clef staff. The score is presented as a raw MIDI export, showing note stems, beams, and articulation marks.

Ideas for using Synesthesia generative software

| | |
|---|--|
| <p>Generating music from mobile phone MMS messages</p> | <p>Generating Ring Tones for mobile phones from pictures</p> |
| <p>Tool for composers: idea generator /tester, music writer. Everybody can be a composer, even children</p> | <p>Listening company logos Listening the package covers</p> |
| <p>Tool for music education at schools, educational music, music for ability tests.</p> | <p>Generating music using mobiles phones with digital camera and listening everything around</p> |
| <p>Research tool, new way of analysing music, writing articles of this new approach</p> | <p>Competitions in TV: viewer send picture, the music is presented and the viewers select the best</p> |
| <p>Tool for blind people for "looking" pictures or their drawings, listening structures</p> | <p>Musitives Radio Channel. Unlimited amount of background music, always different</p> |
| <p>Generating musical illusions</p> | <p>Microtonality is easy to add if needed...</p> |