Saimaan ammattikorkeakoulu
Liiketoiminta ja kulttuuri Imatra
Kulttuuriyrittäjyyden koulutusohjelma

Jussi M. Hukkanen

HALLITUN KAAOKSEN KOODI

Ylempi ammattikorkeakoulututkinto
Opinnäytetyö 2011

TIIVISTELMÄ

Jussi M. Hukkanen
Hallitun kaaoksen koodi, 49 sivua
Saimaan ammattikorkeakoulu, Imatra
Liiketoiminta ja kulttuuri
Kulttuuriyrittäjyyden koulutusohjelma, ylempi AMK
Opinnäytetyö 2011
Ohjaajat: muotoilija Kimmo Heikkilä, kuvataiteilija Jan Kenneth Weckman

Opinnäytetyön taiteellinen osa on reaaliajassa tapahtuva tietokoneanimaatio *Swarm*, josta tehtiin sekä näyttelytilassa seinälle heijastettava versio että internetistä ladattava näytönsäästäjäversio. Opinnäytetyön kirjallisessa osassa kartoitettiin tietotekniikkaa hyödyntäviä kuvataiteen alueita ja pohdittiin taiteilijan ja *Swarmin* suhdetta niihin. Tekstissä käsiteltiin myös näytönsäästäjiä ja demokulttuuria vapaan kuvataiteilijan näkökulmasta.

Asiasanat: tietokonetaide, generatiivinen taide, ohjelmistotaide, näytönsäästäjä, demoscene

ABSTRACT
Jussi M. Hukkanen
The Code of Organized Chaos, 49 pages
Saimaa University of Applied Sciences, Imatra
Business and Culture
Master's Degree in Cultural Entrepreneurship
Master's Thesis 2011
Instructors: Kimmo Heikkilä, designer, Jan Kenneth Weckman, visual artist

In this thesis, the different forms of computer-based visual art were examined. The purpose was to determine how *Swarm*, a real-time computer animation piece by the author, related to what had previously been done in the field. *Swarm* exists both as a projected version intended for art exhibitions and as a Windows screen saver, which is downloadable via the internet. This paper also offers a computer artist's view at screen savers in general as well as the digital subculture known as the demoscene.

Keywords: Computer Art, Generative Art, Software Art, Screen Saver, Demoscene

SISÄLTÖ

1 JOHDANTO

Kuvataiteen ylemmän ammattikorkeakoulututkinnon opinnäytetyöni koostuu taiteellisesta ja kirjallisesta osasta. Taiteellinen osa on *Swarm*, generatiivinen tietokoneanimaatio. Tämä kirjallinen osa, joka rakentuu taiteellisen osan ympärille, pyrkii hahmottamaan *Swarmin* paikkaa tietokonetaiteen maailmassa.

Aloitan lyhyellä kuvauksella *Swarmista* sekä sen taustoista ja toimintatavasta. Kuvailen myös kolmelle eri käyttöjärjestelmälle tehtyjen versioiden tärkeimpiä eroja. Kolmas luku käsittelee tietokonetaiteen kentän eri alueita. Esitän siinä muutamia esimerkkejä siitä, millä tavoin taiteilijat hyödyntävät tietotekniikkaa nykyään Suomessa ja ulkomailla ja vertaan itseäni ja omaa tuotantoani näihin esimerkkeihin.

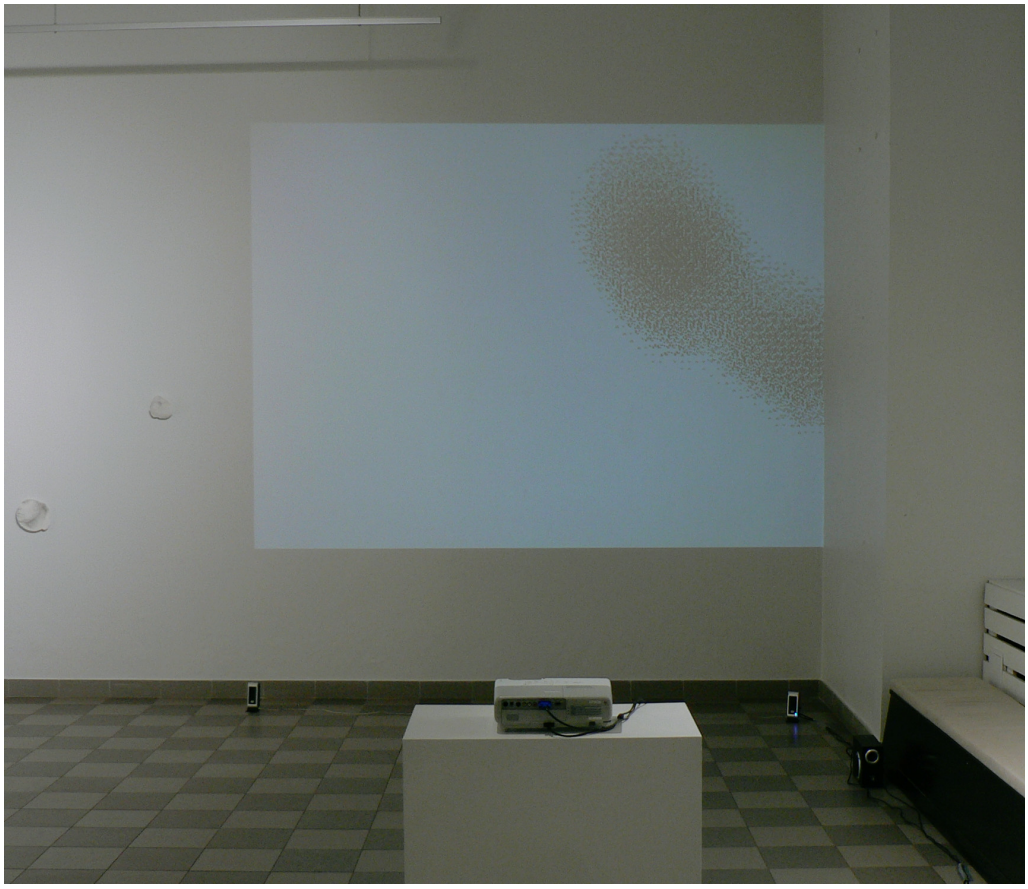
Neljännessä ja viidennessä luvussa käsittelen kahta mielenkiintoista tietokoneohjelmoinnin aluetta: näytönsäästäjiä ja demoja. Näistä ensin mainittua pyrin käsittelemään taidemaailman näkökulmasta ja pohdin näytönsäästäjien roolia tietokonetaiteen muotona. Yritän myös hahmotella määritelmää näytönsäästäjätaiteen käsitteelle. Demojen maailma on otettu käsittelyyn osittain siksi, että *Swarm* on omalla tavallaan demon kaltainen ohjelma, mutta myös siksi, että demokulttuurin ja tietokonetaiteen suhde on mielenkiintoinen.

Kuudes luku keskittyy *Swarmin* teknisiin ominaisuuksiin. Esittelen ohjelman toiminnan keskeiset piirteet mahdollisimman ymmärrettävällä tavalla ja kuvailen jonkin verran sen kehitysprosessia. Viimeisessä luvussa kertaan opinnäytetyöprosessin aikana tekemiäni havaintoja ja pohdin erilaisia näkymiä tulevaa taiteellista toimintaani varten.

Lienee syytä korostaa heti aluksi, että tietokonetaiteessa ja yleisestikin kuvataiteessa mielenkiintoni useimmiten kohdistuu toteutustapoihin ja välineisiin sekä tekniikan ja teknologian merkitykseen: siihen, miten on tehty se, mitä on tehty. Tästä johtuen en tekstissäni juurikaan käsittele kuvataiteen teoreettisia ulottuvuuksia.

2 *SWARM*

Swarm on reaaliajassa tapahtuva generatiivinen kaksiulotteinen tietokoneanimaatio. Se esittää harmaata taivasta vasten piirtyvää miljoonien mustien lintujen tai hyönteisten parvea, joka liikkuu kuvatilassa edestakaisin ja muuttaa jatkuvasti muotoaan. Teoksella on kaksi eri olomuotoa: näyttelytilassa seinälle heijastettu versio ja Windows-ympäristössä toimiva näytönsäästäjäversio, jonka saa ladata ilmaiseksi internetistä. Näyttelyversioon kuuluu ääniraita, joka myös luodaan reaaliaikaisesti ja joka pyrkii kuulostamaan yhtä aikaa tietokoneella toteutetulta ja luonnonmukaisista elementeistä koostuvalta.



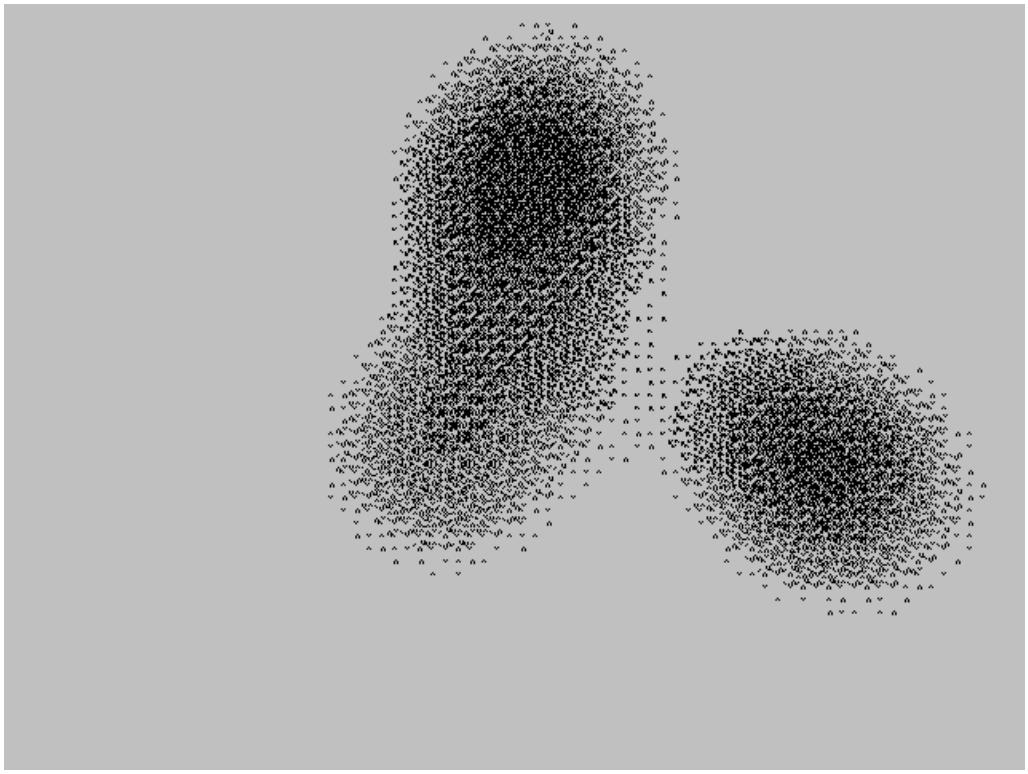
Kuva 2.1 *Swarm*, Galleria Noorus, Tartto, Viro, joulukuu 2010

Teos on suunniteltu siten, että sen jokainen esityskerta on erilainen. *Swarm* on tietokoneohjelma, jonka toimintaa ohjaa yksinkertainen satunnaislukugeneraattori. Sen ansiosta kuvatilassa lentävien olioiden liikkeitä ei ole mahdollista ennakoida. Näin pyritään luomaan vaikutelma siitä, että oliot ikään kuin elävät tieto-

koneessa. Niiden liikehdintä on toisaalta kaoottista ja toisaalta hyvin hallitun oloista.

Swarmin visuaalinen ilme on äärimmilleen pelkistetty: kuva koostuu vain mustista ja harmaista pikseleistä. Yhtenä päämääränä on ollut jäljitellä vanhanaikaisten tietokoneiden graafisia mahdollisuuksia. Teos on suureksi osaksi toteutettu ohjelmointitekniikoilla, jotka ainakin teoriassa voisivat toimia 1990-luvun alun kotitietokoneissa.

Sain idean *Swarmia* varten toukokuussa 2010 nähdessäni televisiodokumentin, jossa esitettiin valtavien kottaraisparvien lentelyä Rooman yllä. Aloin heti miettiä, miten voisin tietokoneohjelman avulla luoda jotain samankaltaiselta näyttävää ryhtymättä kuitenkaan toteuttamaan äärimmäisen pikkutarkkaa parvikäyttäytymisen simulaatiota.



Kuva 2.2 Valmiista *Swarmista* otettu kuvankaappaus

Dokumentin kuvissa minua kiehtoivat pilvien peittämän lähes yksivärisen taivaan ja mustilta pisteiltä näyttävien lintujen välinen kontrasti ja kuva-aiheen

näennäinen minimalistisuus: pysäytyskuva näytti harmaalta kankaalta, johon oli roiskutettu mustia maalipisaroita. Näin minulla oli alusta pitäen mielessäni selvä käsitys siitä, miltä halusin valmiin teoksen näyttävän. Kuva 2.2 esittää *Swarmia* valmiina.

On tärkeää tiedostaa, ettei *Swarm* pyri simuloimaan aidon lintuparven toimintaa vaan ainoastaan jäljittelemään sitä, miltä parvi ja sen liikehdintä näyttää ihmisen silmissä – eikä tämä jäljittelykään ole niin vakavamielistä, että lopputulosta voisi erehtyä pitämään aidonnäköisenä. Tärkeintä on välittää mielleyhtymä ja ajatus siitä, että teos muistuttaa hieman sellaista, mitä luonnossa tapahtuu, kuitenkin niin, että sekä teoksen visuaalinen muoto että äänimaisema ovat selkeästi keinotekoisia.

En ole missään vaiheessa päättänyt, esittävätkö *Swarmin* lentävät oliot lintuja, hyönteisiä vai jotain muuta; tästä syystä kutsun niitä vain olioiksi. Vaikka yksittäisten olioiden muotoa ei valmiissa teoksessa juurikaan kykene hahmottamaan, ne on alun perin piirretty linnun näköisiksi. Jokainen olio ruudulla koostuu alle kymmenestä pikselistä, joten hyvinkin erilaiset tulkinnat siitä, mitä ne esittävät, ovat mahdollisia ja sallittuja.

2.1 Nimi

Teoksen nimeksi tuli *Swarm* (suom. suurparvi) melko luonnollisella tavalla: se on tiedostonimi tai tarkemmin sanoen tiedostokansion eli hakemiston nimi. Uutta ohjelmointiprojektia aloittaessa on mielekkäintä aivan ensimmäiseksi luoda tietokoneen kiintolevylle kansio, jonka sisään ohjelman lähdekoodia ja resursseja aletaan koota. Näin projekti saa yleensä jo alussa työnimen, joka useimmissa tapauksissa myöhemmin korvautuu ohjelman lopullisella nimellä.

En ole keksinyt parempaa nimeä tälle teokselle kuin tuo ensimmäiseksi mieleen tullut työnimi enkä ole nähnyt suurta tarvetta sellaisen keksimiseenkään. Se on lyhyt ja ytimekäs ja kertoo, mitä teos esittää. Lisäksi englanninkielinen nimi on perusteltu silloin, kun teosta on tarkoitus jakaa internetin kautta. *Swarm* on

myös Linux-version ohjelmatiedoston nimi, joten se auttaa täsmentämään, että tässä tapauksessa taideteos ja tietokoneohjelma ovat yksi ja sama asia.

Tiedosto- tai hakemistonimi taideteoksen nimenä ei ole mikään vallankumouksellinen ratkaisu. Esimerkiksi John Klima (2007, 260—261) nimesi *ecosystm*-teoksensa tiedostonimen mukaan, mutta hän halusi pitää tiukasti kiinni vanhojen käyttöjärjestelmien asettamista rajoituksista tiedostojen nimeämiselle. Jotta nimi mahtuisi kahdeksan merkin tilaan, on ”ecosystem”-sanan toinen E pitänyt jättää pois. Niinpä teoksen nimen kirjoittaminen oikein on tuottanut hankaluuksia eri tahoille.

Ohjelmointipiireissä on niin sanotuilla rekursiivisilla lyhenteillä pitkät perinteet. Rekursiivinen lyhenne on esimerkiksi tietokoneohjelman nimeksi annettu lyhenne, jossa jokin kirjaimista, useimmiten ensimmäinen, tarkoittaa nimeä itseään. Monilla ohjelmilla on nimenä humoristinen lyhenne, jonka merkitys on päinvastainen kuin ohjelman todellinen tehtävä, esimerkiksi *LAME* (*LAME Ain’t an MP3 Encoder*). En ole kyennyt välttämään samanhenkisten merkitysten keksimistä myös *Swarmille*: *Swarm Wastes Available Resources Mindlessly* tai *Swarm for Windows Appeals to Roughly Millions*.

2.2 Toimintaperiaate

Jos *Swarm* olisi simulaatio, joka pyrkisi mahdollisimman tarkasti jäljittelemään oikeiden lintujen tai muiden parvessa liikkuvien eliöiden toimintaa, ohjelma vaatisi huomattavasti enemmän järjestelmäresursseja toimiakseen. Ensinnäkin liikkeen pitäisi tapahtua kolmiulotteisessa avaruudessa, kun taas *Swarm* on puhtaasti kaksiulotteisessa tasossa tapahtuva animaatio. Mikä tärkeintä, ohjelma joutuisi pitämään kirjaa jokaisen olioyksilön sijainnista, lentonopeudesta ja suunnasta ja soveltamaan jokaiseen olioon jonkinlaisia käyttäytymissääntöjä, jotka määräisivät niiden liikkeet.

Olioita ohjaavat säännöt voisivat olla joko yksinkertaistettuja tai monimutkaisia, mutta joka tapauksessa parvikäyttäytymisen simulointi edellyttäisi jonkinas-

teisen tekoälyn kehittämistä. Tähän olisi toki olemassa valmiita käyttökelpoisia ratkaisuja, kuten *Boids*-algoritmi (Reynolds 2001).

Swarmiin ei kuitenkaan liity mitään simulaatiopyrkimyksiä. Sen tarkoitus on ainoastaan esittää tietokoneen keinoin jotain luonnossa eläviä suurparvia muistuttavaa. Tämä päämäärä on saavutettu hyvin yksinkertaisella menetelmällä: ruudulle piirretyn parven muoto, kuten myös näyttelyversiossa kuuluva ääni, määräytyy vain kuuden liikkuvan elementin eli ytimen mukaan. Ydinten määrää ei ole rajattu kuuteen, mutten ole kokenut suuremman määrän tuovan teokseen mitään tarpeellista, ja toisaalta alle kuusi olisi mielestäni ollut liian vähän siinä mittakaavassa, jossa halusin parven muotoutuvan ruudulle.

Parvi maalataan ydinten ympärille täyttökuviolla, joka koostuu yksittäisten oliohahmojen yhdistelmistä. Alue, jossa monta ydintä on lähellä toisiaan, näyttää tummemmalta ja tiheämmältä kuin parven reuna-alue, josta voi erottaa yksittäisiä olioita.

Swarmin näyttelyversion äänimaisema tuotetaan niin kutsutuilla generaattoreilla, jotka tarkkailevat eri ydinten liikkeitä ruudulla ja sekoittavat erilaisia ääniä yhteen, vaihdellen niiden korkeutta, voimakkuutta ja panorointia eli vasemman ja oikean äänikanavan tasapainoa. Ytimen pystysuuntainen liike vaikuttaa äänen korkeuteen ja vaakasuuntainen liike panorointiin.

2.3 Toteutus

Muutaman viime vuosikymmenen aikana tapahtunut tietotekniikan kehitys on tarjonnut taiteilijoille rajattomia työskentelymahdollisuuksia ja jatkuvasti uusia ilmaisuvälineitä. Tietokonetta jo pitkään hyödyntäneenä taiteilijana olen kiinnostunut siitä, miten helposti nykyaikaisilla laitteistoilla ja ohjelmistoilla syntyy esimerkiksi näyttävää 3D-animaatiota ja mitä ylimääräisiä haasteita tämä toisaalta tuo mukanaan: tietokoneella työskentelevän taiteilijan on entistä haastavampaa erottua joukosta, kun kaikilla on käytettävissään samat välineet.

Tästä syystä halusin *Swarmin* olevan selkeä irtiotto nykyajan tietotekniikan selviöistä, kuten helposti saatavilla olevasta 3D-grafiikasta. Toinen syy on se, että pidän ohjelmointia erittäin antoisana luovana työnä siinä missä piirtämistä tai maalaamistakin. Olen usein tähdännyt siihen, että tietokoneella toteuttamissani taideteoksissa ja muissa projekteissa mahdollisimman suuri osa olisi oman ohjelmointini tulosta eikä valmiina saatua koodia, joskin tästä periaatteesta oli *Swarmin* kohdalla paikoitellen pakko tinkiä.

2.3.1 DOS

Jo varhaisessa vaiheessa *Swarmin* kehitystyössä päätin, että haluaisin paitsi esittää valmiin teoksen näyttelytilassa, myös jakaa sitä internetin kautta Windows-näytönsäästäjän muodossa. Päätin kuitenkin jättää Windows-version toteutuksen viimeiseksi työvaiheeksi ja tehdä sen varsinaisen näyttelyversion ehdoilla.

Swarmin ulkoasun ovat hyvin pitkälle määritelleet PC-tietokoneissa 1980-luvulta 1990-luvun puoliväliin asti paljon käytetyn DOS-käyttöjärjestelmän asettamat rajoitukset. Alkuperäinen tarkoitukseni oli, että valmis teos olisi DOS-ohjelma ja sen kehitystyö tapahtuikin suurimmaksi osaksi DOS:n sisällä. Vaikka lopulta jouduinkin kääntämään ohjelman Linux-ympäristössä toimivaan muotoon, on lopputulos lähes tarkalleen sen näköinen, miltä sen oli DOS:ssakin tarkoitus näyttää.

DOS oli monellakin tavalla mielekäs valinta. Se on minulle vanhastaan tuttu ympäristö, jossa ohjelmointi on helppoa, ja jossa erityisesti *Swarmin* kaltaisten ohjelmien toteuttaminen on huomattavasti vaivattomampaa kuin kehittyneemmissä käyttöjärjestelmäympäristöissä. Lisäksi halusin valmiin teoksen näyttävän tarkoituksellisesti vanhanaikaiselta ja visuaalisesti vaatimattomalta, joten tasokkaampaa grafiikkaa ja muita mahdollisuuksia tarjoaville järjestelmille ei tässä projektissa alun perin ollut tarvetta.

2.3.2 Linux

Kaksi ratkaisuani johtivat minut siirtämään *Swarmin* kehitystyön Linuxiin. Ensinnäkin halusin animaation tueksi ääntä, mutta äänen tuottaminen on DOS:ssa haastavaa. Toinen syy oli se, että *Swarmin* näyttelyversion oli tarkoitus toimia kannettavassa tietokoneessa, enkä halunnut tinkiä tästä päämäärästä. Kannettavien tietokoneiden äänilaitteistoille on kuitenkin hyvin harvoin tarjolla DOS:n tarvitsemia ajureita, mistä johtuen ääntä tuottavaa DOS-ohjelmaa on vaikeampaa saada toimimaan kannettavassa tietokoneessa kuin pöytätietokoneessa.

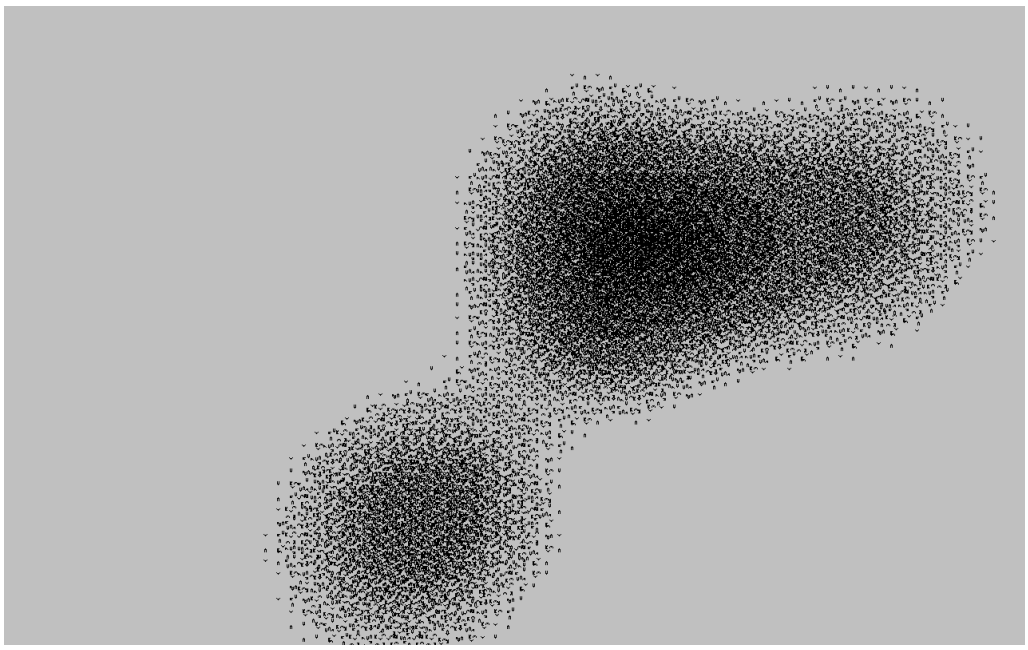
Linuxiin siirtymisen puolesta puhui myös se, että olin jo varhain päättänyt paitsi jakaa *Swarmia* internetin kautta binäärimuodossa eli ajettavana ohjelmatiedostona, myös antaa sen lähdekoodin ilmaiseksi jokaiselle, jota kiinnostaa tutkia ja muokata sitä. Tämä avoimen lähdekoodin toimintamalli on totuttu yhdistämään Linux-maailmaan, vaikkei se olekaan Linuxiin tai mihinkään muuhunkaan yksittäiseen käyttöjärjestelmään sidottu periaate. Toisaalta DOS:lle tehdyn ohjelman lähdekoodista on tätä nykyä iloa vain hyvin pienelle harrastajajoukolle, kun taas Linuxin eri olomuodot kasvattavat yhä suosiotaan ympäri maailmaa.

Swarmin muuntaminen eli porttaaminen Linuxissa toimivaksi ei ollut kovinkaan vaikeaa, vaikka se onkin ensimmäinen Linux-ohjelmani. Olin alusta pitäen pyrkinyt tekemään ohjelmakoodista helposti siirrettävää, koska aioin joka tapauksessa toteuttaa myös Windowsissa toimivan version. Ohjelman tärkeimmät toiminnot ovat laitteistoista ja käyttöjärjestelmistä riippumattomia. Linuxia varten täytyi lähinnä selvittää ajastimen toiminta, jolla animaatio säilyttää halutun nopeuden, sekä grafiikan ja äänen tuottamisen periaatteet. Nämä kolme ovat yleensä asioita, jotka eri käyttöjärjestelmät toteuttavat eri tavalla.

Siinä, missä *Swarmin* DOS-versio ei käytä mitään ulkoisia kirjastoja eli muualta saatua ohjelmakoodia, oli Linuxissa perusteltua delegoida grafiikka ja ääni niitä varten kehitettyjen kirjastojen huolehdittaviksi. Animaation piirtäminen on toteutettu Linux-versiossa *svgalib*-kirjaston avulla ja äänen tuottamisesta vastaa *JACK Audio Connection Kit*.

2.3.3 Windows

Swarmin Windows-versio (kuva 2.3) on näytönsäästäjämuotoinen ohjelma. Sen lähdekoodin rakenne eroaa DOS- ja Linux-versioista usealla tavalla, koska Windows-ohjelmien täytyy noudattaa tiettyä muotoa. Windowsin tarjoamat valmiit rakenteet näytönsäästäjille tekevät niiden toteuttamisesta huomattavasti helpompaa kuin muiden ohjelmien. Tästä huolimatta *Swarmin* Windows-version tekemisessä on ollut omat haasteensa, johtuen osaksi kokemuksen puutteestani Windows-ohjelmoinnin saralla.



Kuva 2.3 *Swarm* Windows XP:ssä

Windows-versioon sisältyy näytönsäästäjien perusvaatimuksiin kuuluva ikkuna, jonka avulla käyttäjä voi asettaa erilaisia animaation ominaisuuksia kuten mittakaavan, nopeuden ja taustan värin. Jotkin näistä ovat saatavilla myös DOS:ssa ja Linuxissa, mutta niiden säätäminen edellyttää ohjelman kääntämistä lähdekoodista uudelleen.

Tämä Windowsin käyttäjälle annettava valinnanvapaus kuului suunnitelmiini alusta alkaen, koska se kuuluu mielestäni hyvän näytönsäästäjän ominaisuuksiin, enkä halunnut luopua siitä, vaikka sen mahdollistaminen teettikin jonkin

verran ylimääräistä päänvaivaa. Esimerkiksi animaation nopeuden vapaa valinta edellyttää sitä, että ohjelma osaa itse tarkkailla piirtämiseen kuluvaa aikaa ja tarvittaessa hidastaa animaatiota.

Oma suhteeni Windowsin *Swarmiin* on hieman ristiriitainen. Ohjelmointi Windows-ympäristössä on melko monimutkaista ja vaatii mielestäni hieman liikaa ylimääräistä opiskelua, vaikka käyttöjärjestelmä tarjoaakin valmiit puitteet hyvin monenlaisten ohjelmien toteuttamiseen – Linuxissa tarvittavia ulkoisia kirjastoja ei *Swarmin* Windows-versio tarvitse lainkaan. Minulle Windows ei näyttäydy kovinkaan mielenkiintoisena ympäristönä, mutta olen hyvin tietoinen sen suuresta käyttäjäkunnasta. Koska *Swarm* oli alun perinkin hyvin näytönsäästäjämäinen teos, oli mielekästä tehdä siitä myös versio, joka voisi paremmin tavoittaa suuren yleisön myös taidenäyttelyjen ulkopuolelta.

3 TIETOKONETAIDE

Swarmin sijoittaminen tiettyyn lokeroon tietokonetaiteen tai yleisemmin taiteen kentällä on haastava tehtävä. Tällaisen lokeroinnin mielekkyys on toki usein kyseenalaistettu, mutta siitä voi olla apua, kun taiteilija esittelee tuotantoaan. Lokeroiden sijaan voitaisiin puhua vaikkapa viitekehyksestä, jonka tietty taidemuoto tarjoaa ja johon yksittäistä teosta peilataan.

Tietotekniikkaa on 1960-luvulta lähtien käytetty kuvataiteen välineenä monin eri tavoin. Ensimmäiset pioneerit, kuten matemaatikon koulutuksen saaneet saksalaiset Georg Nees ja Frieder Nake, käyttivät algoritmeilla ohjattuja piirtureita (Lieser 2009, 19). Vähitellen tätä tietokonetaiteen lajia alettiin kutsua algoritmitaiteeksi.

Algoritmeihin perustuvaa taidetta voidaan kuitenkin sanoa olleen olemassa jo kauan ennen tietokoneita, varsinkin jos kulttuurin kenttää tarkastellaan laajemmin. Kuten Verostko (1999) toteaa, on mikä tahansa nuottikirjoituksen muotoon tallennettu sävellys toistettavissa oleva toimintaohje eli algoritmi, ja sitä paitsi

Stonehenge todistaa matematiikan ja algoritmien ymmärryksen olleen pitkälle kehittynyttä jo esihistoriallisella ajalla.

Monissa M. C. Escherin teoksissa on havaittavissa selkeä geometriaan ja algoritmeihin perustuva rakenne. Henderson (2002) esittelee menetelmän, jolla esimerkiksi Escherin puupiirros *Neliöraja* (1964) voidaan jäljentää jakamalla se perusosiinsa ja kopioimalla näitä osasia yksinkertaisten toimenpiteiden ja sääntöjen avulla. Lopputuloksena syntyy samanlainen algoritmi, jollaista Escher itse on hyvinkin saattanut hyödyntää alkuperäistä piirrosta tehdessään, vaikkei hän tietokonetta käyttänytkään.

Algoritmitaide ei siis ole tietokonetaiteen laji, koska algoritmi ei käsitteenä ole sidottu tietotekniikkaan tai mihinkään muuhunkaan toteutusvälineeseen. Tietokoneiden kehitys on silti kiistatta helpottanut uusien algoritmien suunnittelua, testausta ja käyttöä, käytettiinpä niitä sitten taiteen tekemiseen, tieteelliseen tutkimukseen tai johonkin muuhun.

Algoritmitaiteen tekijät alkoivat 1990-luvun puolivälissä kutsua itseään algoristeiksi. Heille tietotekniikan käyttö ei ole ehdoton edellytys taiteen tekemiselle, mutta he korostavat algoritmin suunnittelun tärkeyttä taiteellisessa prosessissa. Ollakseen vakavasti otettava algoristi taiteilijan on käytettävä itse kehittämiään algoritmeja taiteessaan. (Verostko 2006.)

Swarmin voidaan ajatella olevan algoritmitaideteos, koska se on tietokoneohjelma ja kaikki ohjelmat ovat eräänlaisia algoritmeja tai toisaalta siksi, että parven liikkeet määräytyvät yksinkertaisen algoritmin mukaan. Vuosien saatossa on kuitenkin keksitty uusia kategorioita, joihin *Swarm* on mielekkäämpää sijoittaa.

3.1 Generatiivinen taide

Generatiivisessa taiteessa hyödynnetään tietokoneohjelmaa, konetta tai jotain muuta järjestelmää, joka säädetään toimimaan jossain määrin itsenäisesti ja joka tuottaa lopullisen taideteoksen tai osallistuu sen tuottamiseen. Järjestelmät voivat olla esimerkiksi neuroverkkoja, soluautomaatteja tai erilaisten käyttäytymismallien simulaatioita, ja monet niistä perustuvat kaaoksen ja järjestyksen (satunnaisuuden ja ennalta määräytyvyyden) tasapainoon. (Galanter 2003, 4, 15.)

Generatiivisuus toteutuu *Swarmin* tapauksessa siten, että järjestelmä (tietokoneohjelma) toimii itsenäisesti käynnistämisestään lähtien siihen asti, kunnes sen ajo keskeytetään. Toisin sanoen ohjelman ajon aikana käyttäjältä (teoksen yleisöltä) ei vaadita mitään syötettä eli ohjelman toimintaan vaikuttamista.

Lisäksi kaikki ohjelman synnyttämä kuva- ja äänimateriaali on näennäissatunnaista eli jokaisella ajokerralla erilaista. Kaikki, mitä ohjelmassa tapahtuu, määräytyy sen alkuasetelman mukaan, joka sisältyy kokonaisuudessaan satunnaislukugeneraattorin siemenlukuun. Näin ohjelmasta saadaan haluttaessa monta täysin identtistäkin ajoa syöttämällä sille sama siemenluku uudelleen.

Tässä mielessä *Swarm* voidaan siis laskea generatiivisten taideteosten joukkoon. On kuitenkin tärkeää huomata, ettei edellä mainittu generatiivisen taiteen määritelmä aseta mitään vaatimuksia sille, mistä generatiivisen taideteoksen käyttämä järjestelmä tulee, eli onko se taiteilijan itsensä kehittämä vai jostain valmiina saatu.

3.2 Ohjelmistotaide

Generatiivisen taiteen määritelmästä poiketen ohjelmistotaiteelta (engl. software art) vaaditaan lähtökohtaisesti, että teos on taiteilijan itse tekemä tietokoneohjelma. Ohjelmistotaiteen juuret ovat 1990-luvun nettitaiteessa. Ohjelmistotaide-

teos voi olla audiovisuaalista sisältöä tuottava ohjelma tai se voi vaikkapa manipuloida tietokoneen kiintolevyllä tai internetissä sijaitsevia tiedostoja, mutta pääasia on, että taiteilija on itse laatinut ohjelman. (Lieser 2009, 159.)

Ohjelmistotaideteoksia ovat mm. tietokonepeleissä vallalla olevia ideologioita kyseenalaistavat interaktiiviset peliteokset, hakukoneiden tapaan internetistä dataa keräävät robottiohjelmat sekä taiteelliseen itseilmaisuun kannustavat työkalut. Kyse on siis hyvin laajasta kentästä eikä niinkään yhdestä yhtenäisestä genrestä. (Goriunova 2007.)

Ohjelmistotaiteeseen keskittyvä festivaali Readme on järjestetty Moskovassa 2002, Helsingissä 2003, Århusissa Tanskassa 2004 ja Dortmundissa Saksassa 2005 (Readme 2005). Suomessa ohjelmistotaide, kuten muutkin tietokoneella tehdyn taiteen muodot, näyttää toistaiseksi melko marginaaliselta ilmiöltä. *Swarm* on mielestäni hyvin luontevaa ja mielekästä sijoittaa ohjelmistotaiteen otsikon alle.

Ohjelmistotaide ja generatiivinen taide ovat monelta osin päällekkäin meneviä määritelmiä: ohjelmistotaiteessa usein hyödynnetään olemassa olevia järjestelmiä kuten tietokantoja tai internetiä. Laajemman tulkinnan mukaan kaiken ohjelmistotaiteen voidaan ajatella olevan generatiivista taidetta, koska se vaatii tietotekniikkaa voidakseen olla olemassa, ja tietokone on malliesimerkki omien sääntöjensä mukaisesti toimivasta järjestelmästä.

Olen toisinaan pohtinut tietokoneohjelman lähdekoodin roolia ja erilaisia tapoja ottaa koodi näkyvämmin mukaan osaksi taideteosta. Ohjelmoinnissa liikutaan aina kahden maailman välissä: ohjelman kehitystyö on käytännössä koodin kirjoittamista, mutta valmiin tuotteen käyttäjälle koodi pysyy näkymättömänä. Koodin voidaan jopa ajatella tehneen tehtävänsä siinä vaiheessa, kun ohjelma on valmis.

Ohjelmakoodin näkymättömyys tai läsnäolo kuuluu ohjelmistotaiteen keskeisiin kiinnostuksenkohteisiin. Koodin esittäminen luettavassa muodossa vaikkapa näyttöruudulla on yksi tapa visualisoida tietokoneen sisäistä maailmaa tai ih-

misen ja koneen vuorovaikutusta, mutta yksi siihen liittyvä ongelma on se, että tietokoneisiin perehtymätön yleisö ei näe eroa esimerkiksi C-kielisen lähdekoodin ja HTML-koodin välillä, jotka taas asioihin vihkiytyneille ovat kuin yö ja päivä.

Swarmia tehdessäni harkitsin käyttäväni lähdekoodia teoksen äänimaiseman elementtinä. Tämä ajatus perustui sille havainnolle, että mitä tahansa dataa voidaan syöttää tietokoneen äänilaitteistolle, ikään kuin se olisi aaltoäänitiedoston sisältöä. Tekstitiedostot ja siten myös ohjelmien lähdekooditiedostot kuulostavat tällä tavoin esitettyinä mielenkiintoiselta suhisevalta häiriööäneltä, jollaista olin *Swarmiin* suunnitellut. Luovuin kuitenkin tästä ajatuksesta, koska mielivaltaisesti valitusta datasta tuotettua ääntä kuuntelemalla ei voi päätellä, miten se on tuotettu, joten minun olisi ollut pakko erikseen korostaa, että äänimaisema syntyy nimenomaan lähdekoodista.

3.3 Charles Sandison

Suomalaisista tietokonetaiteen tekijöistä menestyneimpiin lukeutuu skotlantilaissyntyinen Charles Sandison, joka voitti Ars Fennica -palkinnon vuonna 2010. Hänen tilataideteoksissaan, kuten kuvan 3.1 *Chamberissa*, tietokoneen ohjaamat sanat tai symbolit vaeltavat pimeiden huoneiden seinillä, toisinaan myös katossa ja lattiassa.

Teknisesti *Swarm* ja muu oma tietokonetaiteeni on hyvin kaukana siitä, mitä Sandison tekee. Voisi todeta, että meitä yhdistää ainoastaan se, että me kumpikin kirjoitamme tietokoneohjelmamme itse. Sandison on sanonut haluavansa häivyttää tietokoneen läsnäolon installaatioistaan mahdollisimman täydellisesti (Büchler & Sandison 2004, 67) ja pyrkivänsä luomaan vaikutelman sanoista, jotka liikkuvat tilassa itsestään (PEM 2010, 4). Hän ei myöskään halua paljastaa ohjelmakoodinsa toimintaperiaatteita, koska hänen mielestään se rikkoisi tämän vaikutelman ja koska hän toisaalta arvelee, etteivät tekniset yksityiskohdat kiinnosta ihmisiä (PEM 2010, 4).

Kuva 3.1 Charles Sandison: *Chamber*, 2009 (Copyright Charles Sandison)

Olen Sandisonin kanssa samaa mieltä siitä, ettei tietotekniikkaa hyödyntävien taideteosten yleisöltä tarvitse edellyttää tietokoneiden toiminnan ja ohjelmointitekniikoiden tuntemusta, mutta minulle on aivan luonnollista kuvailla myös teosteni teknisiä ratkaisuja kaikille, joita ne kiinnostavat. Tämä liittyy avoimen lähdekoodin periaatteeseen, joka on varsinkin Linuxin suosion kasvun myötä noussut merkittäväksi ohjelmistokentän ilmiöksi, ja jota haluan myös soveltaa *Swarmiin*. Sandisonin lähestymistapa vertautuu ohjelmistokehityksen termeissä suljettuun lähdekoodiin, johon useimmat kaupalliset ohjelmat ja käyttöjärjestelmät perustuvat.

Nämä kaksi näkökulmaa – täydellinen vaikeneminen tekniikasta ja sen kaikkien yksityiskohtien paljastaminen – edustavat saman akselin vastakkaisia ääripäitä. En tietenkään vaadi, että Sandisonin tai kenenkään muunkaan tietokonetaiteen tekijän tulisi seurata omaa esimerkkiäni. Epäilemättä jokainen alalla työskentelevä taiteilija löytää tältä akselilta itselleen sopivimman toimintatavan. Ohjelmistotaiteen piirissä avoin lähdekoodi koetaan yleensä luontevaksi valinnaksi, mutta onkin tärkeää huomata, ettei Sandison itse pidä teoksiaan ohjelmistotai-

teena eikä itseään edes tietokone- tai mediataiteilijana (Büchler & Sandison 2004, 67).

Tietokoneohjelmina Sandisonin teokset ovat huomattavasti *Swarmia* monimutkaisempia, eivätkä vähiten sen johdosta, että ne yleensä käyttävät useampaa kuin yhtä tietokonetta ja videoprojektoria. Tuhannet tai miljoonat sanat, joita Sandison heijastaa seinille ja muille pinnoille, kulkevat omia ratojaan, joten jokaista niistä on luultavasti liikutettava erikseen. Pelkkä olioyksilöistä koostuvan massan vaikutelman luominen ei siis riitä, vaan jokaisen yksittäisen olion on todella oltava olemassa tietokoneiden muisteissa.

Sandison kertoo pitävänsä monia teoksiaan satunnaisuuden tutkimuksina (Jäämeri 2011, 51). Tämä on yksi yhdistävä tekijä Sandisonin töiden ja omieni välillä. Satunnaisuus ja näennäissatunnaisuus ovat keskeisiä generatiivisen taiteen rakennuspalikoita ja *Swarmin* työstämisen yhteydessä olen itsekin jonkin verran tutkinut erilaisia menetelmiä satunnaislukujen tuottamiseen.

On myös mielenkiintoista, että Sandison mainitsee teoksensa *Language as a Mirror of the World* inspiraatioksi lintuparven, jonka hän näki työhuoneensa ikkunasta (YLE Uutiset 2010).

3.4 Scott Draves

Charles Sandisonia, minua ja amerikkalaista Scott ”Spot” Dravesia yhdistävät nuorella iällä aloitettu ohjelmointiharrastus ja vasta myöhemmin saatu oivallus siitä, että tietokoneita ja erityisesti itse tehtyjä ohjelmia voisi käyttää taiteen välineinä (Jäämeri 2011, 46; Triangulation Blog 2011).

Draves kuuluu ohjelmistotaiteen pioneereihin. Hänellä ei ole taiteilijan koulutusta ja hän sanookin päätyneensä taiteilijaksi vahingossa (Triangulation Blog 2011). Hänet tunnetaan *Electric Sheep* -näytönsäästäjän (kuva 3.2; ks. luku 4.4) ja sen pohjana olevan *Flame*-algoritmin keksijänä. *Flame* tuottaa fraktaalikuvia,

Kuva 3.2 *Electric Sheep*, 244. sukupolven lammas 1402 (Artwork by Scott Draves and the Electric Sheep)

jotka perustuvat IFS:iin eli iteratiivisiin funktiojärjestelmiin (engl. iterated function system). Algoritmin toimintaa ohjaa suuri parametrijoukko, jonka ansiosta sillä on mahdollista luoda rajattomasti abstraktia generatiivista tietokonegrafiikkaa (Draves 2000).

Draves kertoo olleensa aina kiinnostunut tavoista saada tietokoneella aikaan jotain odottamatonta ja pyrkineensä tekemään ohjelmia, jotka yllättävät hänet itsensäkin (Triangulation Blog 2011). Hän näkee tässä suoran vertauksen tekoälyyn ja keinotekoiseen elämään, mutta tiedostaa myös tähän liittyvän ongelman: esimerkiksi hänen *Flamella* tuottamansa kuvat ovat hänen ”löytämiään” enemmän kuin hänen tekemiään (Draves 2000).

Yhteisöllisyyden ja avoimen lähdekoodin periaatteet ovat Dravesille tärkeitä. Hän julkaisi *Flamen* ensimmäisen version lähdekoodeineen 1990-luvun alkupuolella, ja sittemmin se on otettu mukaan moniin kuvan- ja videonkäsittelyohjelmistoihin (Draves 2009).

Fraktaalikuvat ovat kiinnostaneet minuakin jo kauan, mutta Draves on vienyt ne todella pitkälle. IFS:t ovat huomattavasti monimutkaisempia kuin mitkään niistä perusfraktaaleista, joita olen itse tutkinut. Vaikka kauneuden kaltaiset käsitteet ovat aina yksilöllisesti määriteltäviä ja tulkinnanvaraisia, on pakko ihailla sitä tapaa, jolla Draves on valjastanut matemaattiset kaavat ja funktiot tuottamaan kauneuden elämyksiä.

3.5 Tietokonetaiteen itsekritiikkiä

Amerikkalainen tietokonetaiteilija ja esseisti Bruce D. Price on voimakkaasti kritisoinut (2005) taiteilijoita, jotka tekevät matematiikkaan vahvasti nojaavia teoksia, kuten algoritmitaidetta tai fraktaalikuvia, ja haluavat esittää niitä vakavasti otettavana digitaalisena kuvataiteena. Price puhuu ”suuresta taiteesta” (engl. great art) ja sanoo edellä mainittujen taidemuotojen sopivan huonosti yhteen tällaisen korkealuokkaisen taiteen ihanteen kanssa. Vaikka hän torjuukin ne näkemykset, joiden mukaan tietokone tekee taiteessakin kaiken työn, kun ihminen vain painaa nappia, hän peräänkuuluttaa inhimillistä kädenjälkeä, joka tietokoneella toteutetusta kliinisen oloisesta kuvataiteesta usein puuttuu.

Ymmärrän Pricen kannan ja olen jossain määrin hänen kanssaan samaa mieltä. On helppoa antautua esimerkiksi fraktaalikuvien vietäväksi ja yllättyä siitä, että puhdas matemaattinen täsmällisyyskin voi tuottaa ihmismieltä puhuttelevia taide-elämyksiä, mutten silti itse laittaisi niitä taidegallerian seinälle. Oma kantani perustuu siihen, että fraktaalit ovat jo melko vanha keksintö enkä usko niillä yksinään olevan paljoa tarjottavaa nyky-yleisölle. Tästä huolimatta on syytä muistaa, että fraktaalikuviakaan ei tietokone tee ihmisen puolesta; inhimillistä luovuutta ja harkintaa tarvitaan esimerkiksi kuva-alan rajaamiseen ja värien valintaan.

Price on kirjoittanut pontevaan sävyyn digitaalisten välineiden tarjoamista mahdollisuuksista ja taiteilijoista, jotka hänen mielestään heittävät nämä mahdollisuudet hukkaan käsitellessään esimerkiksi maalaustaiteen klassikoita tietokonetaiteen keinoin. Price rinnastaa nykyisen tilanteen sadan vuoden takaisiin tai-

desuuntiin kuten futurismiin ja surrealismiin ja ehdottaa, että avantgarden hengen mukaisesti nyt tulisi etsiä uusia ennennäkemättömiä luovuuden muotoja, jotka parhaiten hyödyntäisivät tietotekniikan tarjoamaa potentiaalia. (Price 2004.)

Näiden mielipiteiden valossa on kiinnostavaa katsella Pricen omaa tuotantoa. Hän tekee abstrakteja digitaalisia maalauksia kuvankäsittelyohjelmilla. Hänen mielestään tietokonetaiteen tulee olla kokonaan digitaalista eli sellaista, jota ei olisi mahdollista toteuttaa ilman tietokonetta, joten esimerkiksi valokuva- tai videotaidetta ei pitäisi lainkaan esittää digitaalisen taiteen näyttelyissä (Price 2006). Tästä huolimatta minulle jää monista hänen maalauksistaan sellainen vaikutelma, että asiansa osaava taidemaalari olisi hyvinkin voinut maalata ne siveltimellä kankaalle ilman minkäänlaista tietoteknistä väliintuloa.

Price tuntuu olevan hieman liian jyrkkä määritellessään, millaista taidetta tietokoneilla pitäisi tehdä. En muista kohdanneeni yhtäkään tietokoneella tehtyä taideteosta, joka olisi vaikuttanut auttamattomasti menneisyyteen juuttuneelta, mutten myöskään ymmärrä, miksei taiteilijalle voisi sallia tällaista valinnanvapautta tai miksi tietokonetaide ei saisi ammentaa menneestä siinä missä muutkin taiteenalat. Pyrkiihän *Swarmkin* imitoimaan vanhojen tietokoneiden vaatimatonta grafiikkaa.

En tuomitse Pricen näkemyksiä kokonaan, sillä on tärkeää huomioida, että vaikka hän kritisoi juuri sellaista matematiikkaan pohjautuvaa taidetta, jota esimerkiksi Scott Draves tekee, hän ei lainkaan mainitse Dravesia eikä muitakaan ohjelmistotaiteen tekijöitä. Hän käyttää termiä ”ohjelmoitu taide” (Price 2005), mutta ohjelmistotaide vaikuttaa olevan hänelle tuntematon käsite. Luulen, että se olisi kuitenkin hyvin lähellä sitä uuden digitaalisen ajan taidetta, johon Price toivoo tietokonetaiteilijoiden keskittyvän. Ihmisen kirjoittamaan ohjelmakoodiin sisältyy aina tietty inhimillinen elementti.

4 NÄYTÖNSÄÄSTÄJÄT

Näytönsäästäjät ovat kiehtoneet minua siitä asti, kun yläasteikäisenä ensi kerran näin Windows 3.1:n *Mystify*-näytönsäästäjän. Tein siitä vähän myöhemmin oman jäljitelmäni BASIC-ohjelmointikielellä ja olen sen jälkeen jatkanut omien näytönsäästäjämäisten ohjelmien tekemistä satunnaisena harrastuksena. Vaikka minulla on vain vähän kokemusta varsinaisten Windows-näytönsäästäjien tekemisestä, on *Swarm* selkeää jatkoa tuolle harrastukselle.

Näytönsäästäjän ensisijainen tarkoitus on pidentää tietokoneen näyttöruudun käyttöikää. Perinteisen CRT- eli kuvaputkinäytön ongelma on, että kun se on pitkään päällä ja esittää samaa liikkumatonta kuvaa, tästä kuvasta palaa lähtemätön jälki näytön pinnalle. Tästä syystä tarvitaan näytönsäästäjä, joka tietokoneen ollessa joutilaana pimentää ruudun tai pitää huolen siitä, että ruudulla liikkuu jotain.

Litteisiin näyttöruutuihin, jotka nykyään ovat yleisiä, ei liity tällaista haamukuvan ruutuun palamisen ongelmaa, ja niinpä näytönsäästäjät ovat menettäneet alkuperäisen käyttötarkoituksensa (Baumgärtel 2000). Niiden ensisijainen tehtävä tänä päivänä on mielenkiintoisten visuaalisten elämysten tarjoaminen.

Samalla näytönsäästäjä on myös itseilmaisun väline, paitsi tekijälleen, myös käyttäjälleen. Se, mitä työpaikan tietokoneen ruudulla tapahtuu sillä välin, kun työntekijä on poissa työpisteestään, ei ole suinkaan yhdentekevää, vaan näytönsäästäjän valinta on oman identiteetin määrittelemisen tapa siinä missä muutkin. (Buckhouse, Mirapaulin 2000 mukaan.)

4.1 *Refresh*

Näytönsäästäjät ovat yleensä jääneet vähäiselle huomiolle digitaalisesta taiteesta keskusteltaessa, vaikka niitä voidaan pitää hyvin mielenkiintoisena taidemuotona. Niiden tehtävä on paradoksaalinen: tietokoneen ruudulla esitetään jo-

tain huomionarvoista juuri silloin, kun käyttäjän huomio ei kohdistu siihen. Kaikkein vaikuttavimmat näytönsäästäjät voivat kuitenkin jättää ihmisten mieleen omat pysyvät jälkensä, vaikka ne samalla pyrkivät estämään jälkien jäämisen näyttöruutuihin (Tucker 2009). Vaikkei *After Darkin* lentäviä leivänpaahtimia tai *Avaruusmatkan* tähtitaivasta luultavasti ole tarkoitettu taideteoksiksi, ovat ne omalla kentällään ehdottomia klassikoita.

Taidenäyttelyissä näytönsäästäjiä on tähän asti nähty hyvin vähän. Vuonna 2000 Stanfordin yliopiston museossa Kaliforniassa esillä ollut näyttely *Refresh – The Art of the Screen Saver* oli tässä mielessä uraauurtava. Näyttely oli samaan aikaan nähtävissä internetissä, ja siihen oli koottu 22 näytönsäästäjää, jotka olivat taiteilijoiden tekemiä eivätkä näin ollen suinkaan tavanomaisen kaltaisia. (Baumgärtel 2000; Mirapaul 2000.)

Näyttely osoitti, että näytönsäästäjä voi olla myös yhteiskunnallisiin aiheisiin kantaaottava taideteos: esimerkiksi Tarikh Korulan *Texas Moments* käsitteli Teksasissa teloitettuja vankeja (Mirapaul 2000). Toiset teokset taas ammensivat kekseliäällä tavalla näytönsäästäjien perinteisistä ilmaisutavoista ja jotkin tyytyivät vain esittämään tekijöidensä aiempia maalauksia tai videoita näytönsäästäjän muodossa (Baumgärtel 2000).

Refreshin esittelemät teokset olivat vapaasti ladattavissa internetistä, kuten luontevaa onkin – näytönsäästäjästähän ei ole paljoa iloa, ellei sitä pääse itse kokeilemaan ja näkemään omalla tietokoneellaan. Niiden vapaa levitys pyrittiin kuitenkin estämään tarkasti määritetyillä käyttöehdoilla (Baumgärtel 2000).

4.2 *The Thief*

Yksi *Refresh*-näyttelyssä esitellyistä näytönsäästäjistä oli belgialaissyntyisen Francis Alÿsin *The Thief* (1999). Siitä paljastuu mielenkiintoisella tavalla se muutos, joka näytönsäästäjän roolia ja tehtävää on kohdannut.

The Thief esittää pimeän huoneen seinälle ikkunasta lankeavaa valokeilaa ja siinä liikkuvaa ihmisen varjoa. Kuva on lähes kokonaan musta ja valkoinen valokeila sen keskellä verraten pieni. Ihmishahmon liikkeet on helppo tulkita varjon perusteella: hahmo kävelee ikkunan luo ja kiipeää sen kautta ulos – tai sisään, sillä kaikki paitsi valo ja varjo jää katsojan oman mielikuvituksen ja tulkinnan varaan.

Teos on käytännössä hyvin lyhyt videopätkä, jota toistetaan loputtomana silmukkana. Onkin aiheellista kysyä, onko *The Thief* nimenomaisesti taideteokseksi tarkoitettu näytönsäästäjä vai näytönsäästäjän muodossa esitettävä videoteos. Sen voisi ajatella toimivan vähintään yhtä tehokkaasti videoprojisointina. Toisaalta näytönsäästäjäyhteydessä avonainen ikkuna voidaan myös nähdä vertauksena tietokoneen näyttöruutuun ja käyttöjärjestelmään (Baumgärtel 2000).

Näytönsäästäjän valinta esitystavaksi on tässä tapauksessa ongelmallinen. Jos teosta käyttäisi näytönsäästäjänä kuvaputkinäytöllä varustetussa tietokoneessa, se voisi vaurioittaa näyttöä, koska *The Thief* ei täytä näytönsäästäjälle perinteisesti asetettuja vaatimuksia: kuvasisältö ei liiku ruudulla vaan valkoinen valokeila pysyy kuvassa keskellä. Ajan mittaan puolisuunnikkaan muotoinen kuvio jäisi lopullisesti näkyviin kuvaruudun keskelle, vaikka juuri tällaista näytönsäästäjien pitäisi estää tapahtumasta.

Esitystavan valinta voi olla hyvin pitkälle harkittu ja tietoinen kannanotto, mutta käytännössä se määrittelee, missä laitteistokokoonpanoissa teosta voi katsella ja missä ei. Tämä on hieman yllättävää ottaen huomioon, että näytönsäästäjiä on totuttu pitämään laitteistoista riippumattomina, vaikkakin eri käyttöjärjestelmäympäristöt ovat toki edellyttäneet niiden kääntämistä omiksi versioikseen.

4.3 Ezra Johnson

Myös amerikkalainen taidemaalari Ezra Johnson on tehnyt lukuisia näytönsäästäjätaideteoksia. Hän laittaa töissään maalin elämään maalauspohjalla. Näytön-

säästäjät on tehty stop motion -tekniikalla ottamalla sarja valokuvia maalauksesta samalla kun sitä maalataan ja kokoamalla kuvat animaatioksi.

Johnsonille animaatio ei ole itse tarkoitus vaan tapa tarkastella maalaamisen prosessia (Tucker 2009). Tässäkin tapauksessa näytönsäästäjän valinta välineeksi voidaan kyseenalaistaa ja kysyä, saavatko teokset erityistä lisäarvoa tästä muodosta vai toimisivatko ne yhtä hyvin vaikkapa www-sivulla esitettävinä animaatioina. Päinvastoin kuin Alÿsin *The Thief*, Johnsonin teokset soveltuvat perinteiseen näytönsäästäjäkäyttöön. Teosten tarjoaminen internetin kautta niin, että kuka tahansa voi ne asentaa omalle tietokoneelleen, on yksi tapa tuoda taidetta lähemmäs yleisöään. Tämä sopii mainiosti Johnsonin maalaustyyliin, jossa fyysisyys ja maalipinnan kolmiulotteisuus korostuvat (Tucker 2009).

Sekä Alÿsin että Johnsonin teokset on tehty Flash-animaation muotoon, josta ne on käännetty näytönsäästäjiksi Screentime-ohjelmalla (EAI 2009). Valitettavasti tämä tekee niistä ns. bloatwarea eli ohjelmia, jotka vaativat tietokoneelta suhteettoman paljon kiintolevytilaa tai muita resursseja siihen nähden, miten monimutkaisia tehtäviä ne suorittavat (The Jargon File 2003). Kummassakin tapauksessa on kyse ohjelmista, jotka esittävät ruudulla ennalta tuotettujen bittikarttakuvien jatkumoa. Tähän ei tarvita mitään erityisen monimutkaista ohjelmointia, mutta Flash-muoto vaatii erityisen Flash Player -ohjelmakomponentin sisällyttämistä jokaiseen tällaiseen ohjelmatiedostoon. Tämä paisuttaa tiedoston koon useaan megatavuun, vaikka itse animaatiosisältöä ja ohjelmakoodia olisi huomattavasti vähemmän.

4.4 *Electric Sheep*

Scott Dravesin *Electric Sheep* on poikkeuksellinen ilmiö näytönsäästäjien maailmassa. Se ei tyydy olemaan pelkkä näytönsäästäjä, vaan siitä on kehittynyt kokonainen virtuaalinen maailmanlaajuinen eläintarha tai geenipankki, jolla on tätä nykyä noin puoli miljoonaa käyttäjää (Triangulation Blog 2011).

Electric Sheep koostuu fraktaalianimaatioista, jotka on piirretty Dravesin luomalla *Flame*-algoritmilla. Koska *Flame* on hyvin monimutkainen, jo yksittäisen kuvakehyksen piirtäminen eli renderöinti tavallisella tietokoneella on siinä määrin hidasta, että näytönsäästäjän animaatiosisältöä on mahdotonta tuottaa reaaliajassa. Niinpä *Electric Sheepin* esittämä graafinen sisältö koostuu animaatiotiedostoista, jotka ladataan käyttäjän tietokoneelle palvelimelta (Draves 2009).

Yksi *Electric Sheepin* mielenkiintoisista piirteistä on, että sen esittämät animaatiopätkät renderöidään käyttäjien tietokoneilla. Toiminta perustuu niin sanottuun hajautetun tietojenkäsittelyn malliin. Sillä aikaa, kun tietokone on joutilaana ja *Electric Sheep* on käynnissä, se renderöi uusia kuvakehyksiä animaatioon ja lähettää ne takaisin palvelimelleen. Yksittäisten käyttäjien tietokoneet muodostavat siis suuren vertaisverkon, joka toimii ikään kuin yhtenä suurena tietokoneena. *Electric Sheep*, jonka ensimmäisen version Scott Draves teki vuonna 1999, oli yksi ensimmäisistä vertaisverkkoideaa hyödyntävistä ohjelmista (Draves 2009).

Electric Sheep on kiehtova myös evoluutioelementtiensä takia. Animaatiopätkiä kutsutaan lampaiksi ja jokaisen lampaan ulkoasu määräytyy monimutkaisen geneettisen koodin perusteella. Tämä koodi on pitkä numerosarja, joka ohjaa fraktaalianimaation piirtämistä. Geneettisiä koodeja myös risteytetään keskenään ja niihin syötetään satunnaisia muutoksia, jolloin vanhojen animaatioiden pohjalta syntyy uusia. Lampaat siis lisääntyvät ja niiden ominaisuudet periytyvät uusille lammassukupolville. Käyttäjät voivat myös luoda omia lampaita ja äänestää jo olemassa olevien lampaiden joukosta suosikkejaan. Annetut äänet vaikuttavat uusien lampaiden ominaisuuksiin. (Draves 2005, 4—6.)

4.5 Näytönsäästäjä taidemuotona

Näytönsäästäjätaiteen käsitteelle ei näytä löytyvän mitään olemassa olevaa määritelmää. Tähän lienee yhtenä syynä se, että jo sanaa tarkasteltaessa se

tulee kutakuinkin määritellyksi: mitä muuta näytönsäästäjätaideteos voi tarkoittaa kuin taidekontekstissa esitettävää näytönsäästäjää?

Kuten Alÿsin *The Thief* osoittaa, taideteokseksi tehdyltä näytönsäästäjältä ei voi edes edellyttää niitä ominaisuuksia, joita näytönsäästäjiltä yleisesti vaaditaan, kuten koko näyttöruudun alueella tapahtuvaa animaatiota. Mikä tahansa animaatio- tai videoteos voi olla näytönsäästäjätaidetta, jos se toteutetaan tai käännetään tähän muotoon. Tarkkaan ottaen teoksessa ei tarvitse olla edes liikkuvaa kuvaa – yksinkertaisimmat näytönsäästäjäthän eivät tee muuta kuin pimentävät ruudun – mutta tietenkin sen puuttuessa voidaan kysyä, miten näytönsäästäjä perustellaan välineenä. *The Thief* on esimerkki taidenäytönsäästäjästä, joka modernin taiteen perinteille uskollisesti kyseenalaistaa oman välineensä ja etsii sen rajoja.

Näytönsäästäjätaide ei voi olla ohjelmistotaiteen alalaji, koska näytönsäästäjiä tekeviä taiteilijoita ei voi vaatia ohjelmoimaan teoksensa itse. Edellä mainittu Screentime-ohjelma tekee Flash-muotoisten näytönsäästäjien luomisesta helppoa. Vaikka ActionScript-ohjelmointikieli moninkertaistaakin Flashin potentiaalin, ei monipuolistenkaan animaatioiden toteuttamiseen Flashilla vaadita minkäänlaista ohjelmointia.

Myöskään generatiivinen sisältö ei ole näytönsäästäjätaideteoksen kriteeri. Näytönsäästäjän tulisi toki toimia taukoamattomana ajona käynnistämisestään siihen hetkeen asti, kun käyttäjä keskeyttää sen. Tämä tapahtuu joko siten, että ohjelma tuottaa generatiivisesti uutta sisältöä koko ajan, tai siten, että etukäteen tuotettua sisältöä toistetaan silmukkana uudelleen ja uudelleen, tai toteutustapa voi olla näiden kahden jonkinlainen yhdistelmä. Tämä näennäisesti loputon ajo on kuitenkin vain yksi näytönsäästäjiltä perinteisesti vaadittava ominaisuus. Myös teos, joka esittää jotain liikkuvaa kuvasisältöä jonkin aikaa ja pysähtyy sen jälkeen, voi olla näytönsäästäjätaidetta, vaikkei siitä kovin mielekkääksi näytönsäästäjäksi olisikaan.

Jos näytönsäästäjätaide halutaan laskea omaksi tietokonetaiteen alalajikseen, *Swarmin* voidaan ajatella sopivan varsin luontevasti tähän ryhmään, kuten

myös ohjelmistotaiteen ja generatiivisen taiteen ryhmiin. Luokittelu on sikäli mutkikasta, että ainoastaan *Swarmin* Windows-versio on näytönsäästäjä; DOS- ja Linux-versiot eivät.

Toinen ongelma *Swarmin* luokittelemisessa näytönsäästäjätaideteokseksi on se, että itse haluaisin ihmisten tutustuvan Windowsin *Swarmiin* ensisijaisesti näytönsäästäjänä ja vasta sitten taideteoksena. Toisin sanoen näytönsäästäjämuoto on *Swarmin* tapauksessa enemmän välityskanava tai markkinoinnin väline kuin olennainen osa itse taideteoksen olemusta, vaikkakin minulle on alusta pitäen ollut selvää, että teoksessa on näytönsäästäjän piirteitä ja että se tulee olemaan olemassa myös näytönsäästäjäversiona.

5 DEMOSCENE

Joissain tietokoneella toteuttamissani ohjelmistotaideteoksissa, kuten *Swarmissa*, on havaittavissa yhtymäkohtia demosceneen. Tämä on sikäli yllättävää, etten ole koskaan ollut mukana sillä kentällä ja tietoni siitä ovat varsin rajalliset.

Demoscene, joka suomenkielisissä asiayhteyksissä usein kirjoitetaan tahallisesti muotoon demoskene, on tietokoneohjelmoinnin harrastajien alakulttuuri, jonka keskiössä ovat demot eli reaaliajassa tietokoneella toteutettavat multimediaesitykset. Yksi määritelmä demolle on ”*kuvaa ja ääntä mahdollisimman vaikuttavaksi kokonaisuudeksi*” yhdistävä ohjelma (Saarikoski 2003).

Demoja on verrattu myös musiikkivideoihin (Demoscene Documentary 2010) ja näiden kahden lajin ilmaisukeinoissa onkin paljon yhteistä. Monet demontekijät ovat maininneet musiikkivideot inspiraationsa lähteiksi (Roininen 1998, Reunasen 2010, 61 mukaan). Tärkeimpiä demon tuntomerkkejä on kuitenkin se, että se tuottaa kuvaa ja ääntä reaaliajassa eikä siis ole vain animaatiota tai videota, joka toistetaan nauhalta tai tiedostosta (Tasajärvi, Stamnes & Schustin 2004, 17).

Demoscene asettaa demojen tekijöiden tietotekniikkataidot, etenkin ohjelmointitaidon, korkeaan arvoon. Demontekijät ovat alusta alkaen pyrkineet luomaan niin vaikuttavaa audiovisuaalista sisältöä kuin käytössä olevalla tietokoneella on suinkin mahdollista ja tekemään taidoillaan vaikutuksen muihin alan harrastajiin (Tasajärvi ym. 2004, 15). Omatoiminen tekeminen on myös ollut ensisijaisen tärkeää; valmiin kuva-, ääni- tai muun sisällön kopioimista on pidetty osoituksena omien taitojen puutteesta (Reunanen 2010, 60, 97).

5.1 Demot taiteena

Vaikka ohjelmistotaiteella onkin todettu olevan hyvin läheinen suhde demosceneen (Goriunova 2007), on demopiireissä perinteisesti pyritty välttämään demojen yhdistämistä multimediataiteeseen tai taidemaailmaan yleensäkään (Tasajärvi ym. 2004, 23). Demoscenen jäsenet ovat olleet varsin tyytyväisiä kulttuurinsa underground-asemaan ja näkymättömyyteen. Suomessa demoscene on vuosittain noussut hetkeksi suuren yleisön tietoisuuteen Assembly-tapahtuman kautta.

Näkyvyyttä on kuitenkin alkanut tulla 2000-luvulla lisää. Eräs ensimmäisistä taidenäyttelyistä, joissa esiteltiin demoja ja demosceneä, oli Frankfurtissa 2002—2003 järjestetty *origami digital* (digitalcraft 2002; Reunanen & Silvast 2010). Suomessa demoja tuotiin esiin taiteen kentällä vuonna 2003 nykytaiteen museo Kiasman *demoskene.katastro.fi* -näyttelyssä, joka esitteli 14 demoa vuosilta 1989—2003. Demoja ajettiin näyttelyssä autenttisilla laitteistoilla eli samoilla tietokonemalleilla, jollaisilla ne oli alun perin tehty (Saarikoski 2003).

Muitakin hankkeita, joissa demosceneä on pyritty tekemään tunnetuksi uusmediataiteen muotona muiden joukossa, on nähty. Helsingin Taiteiden yön 2009 *Demowall*-näytöksessä Kiasman ulkoseinään projisoitiin parhaita demoja kolmelta vuosikymmeneltä (Kauppinen 2009). Demoscenellä pitkään vaikuttanut kuvataiteilija Pilvari Pirtola (2010) vastasi näytöksen kuratoinnista.

Demoissa ei alun perin ole ollut muuta ajatuksellista sisältöä kuin ”mepä osaamme tehdä tällä tietokoneella tällaista” tai ”moi vaan, kaikki kaverit” (digitalcraft 2002). Tasajärven (2004, 32) mukaan demot ovat tästä huolimatta kiistattomasti taideteoksia ja niiden tekijät taiteilijoita. Asiaan ei vaikuta se, että vakiintuneessa taidemaailmassa mahdollisesti saavutettava hyväksyntä on ollut heille yhdentekevää verrattuna maineeseen ja kuuluisuuteen demoscenen sisällä. Vaikka hyvin harvoissa demoissa on esitetty poliittisia viestejä, on tietyn tietokoneympäristön valinta luovan itseilmaisun välineeksi Tasajärven mielestä jo viesti sinänsä.

5.2 *Swarm* ja demot

On tärkeää korostaa, että *Swarm* ei ole demo. Mielenkiintoisia yhdenkaltaisuuksia demokulttuuriin nähden siitä toki löytyy, mutta se ei täytä kaikkia vakiintuneita demon tuntomerkkejä. Näitä ovat mm. tiivis rakenne, lyhyehkö mitta ja interaktiivisuuden puute.

Tiivis rakenne viittaa siihen, että demot alun perin koostuivat vain peräkkäin esitetyistä graafisista tehosteista eli demoefekteistä ja niitä säestävästä taustamusiikista. Demoissa alettiin 1990-luvun aikana nähdä vapaampia rakenteita ja harkituille ratkaisuille, kuten yhtenäiselle visuaaliselle ilmeelle, jatkuvuudelle ja narratiivisuudelle, alettiin antaa enemmän arvoa kuin pelkälle efektimyrskylle (Reunanen 2010, 59—60). Lentävien olioiden parvi, jonka *Swarm* esittää, voidaan tulkita demoefektiksi, mutta yksittäistä efektiä ei koskaan ole pidetty demolle riittävänä – ellei kyse ole introsta eli tarkoituksella pieneen kokoon, kuten neljään kilotavuun, tiivistetystä miniatyyridemosta (Reunanen 2010, 52).

Lyhyt mitta on toinen piirre, joka *Swarmista* ilmiselvästi puuttuu; siinä ei ole mittaa lainkaan, vaan se jatkuu niin kauan kunnes sen ajo keskeytetään. Tällainen näytönsäästäjämäinen toimintaperiaate juontaa juurensa generatiivisuudesta ja pyrkimyksestä saada aikaan keinotekoisen elämän halpa illuusio; ennalta määrättyä on vain se, että kuviteltu elämä jatkuu. Demoissa sitä vastoin on aina alku ja loppu. Loputtomasti jatkuvaa ja jokaisella ajokerralla muuttuvaa

rakennetta ei niissä juurikaan ole käytetty, vaan ne tehdään tarkan käsikirjoituksen mukaan, jota niiden on noudatettava esityskerrasta toiseen.

Varsinaista interaktiivisuutta *Swarmissa* ei ole sen enempää kuin demoissakaan, mutta edellä mainittu näytönsäästäjämäinen toiminta edellyttää käyttäjän osallistumisen tarkkailua ainakin sen verran, että ohjelma osaa lopettaa itsensä, kun käyttäjä niin haluaa.

Swarmia tehdessäni ei demomaailma oikeastaan käynyt mielessänikään. Olen silti tietoinen siitä, että ohjelmoinnista kiinnostunut katsoja näkee teoksen todennäköisesti eri tavalla kuin toinen, joka ei piittaa tietokoneista. Demontekijä alkaisi *Swarmin* äärellä epäilemättä miettiä, miten se on toteutettu; niin tekisin tietysti itsekin vastaavassa tilanteessa.

5.3 Kohderyhmä ja kilpailu

Eräs näkökulma on yhtä aikaa erottava ja yhdistävä tekijä *Swarmin* ja demoscenen välillä. Demojen tekemisen päämäärä on niiden esittäminen muille demoscenen jäsenille; se, mitä demoscenen ulkopuolinen maailma niistä ajattelee, on niiden tekijöille yhdentekevää (Reunanen 2010, 29). Demoilla on siis hyvin tarkkaan määrätty kohderyhmä.

Tarkoitukseni ei ole missään vaiheessa ollut rajata *Swarmia* tai muitakaan teoksiani millekään tietylle ryhmälle tai valikoidulle yleisölle, vaan olen lähtenyt siitä periaatteesta, jonka mukaan taide kuuluu kaikille. En tästä huolimatta voi mitään sille, että tietokonetaiteella on taipumus vetää puoleensa tekniikasta kiinnostunutta yleisöä, enkä etenkään sille, että taide sinänsä on oman suljetun sisäpiirinsä huomion kohde. Saadakseen taide-elämyksiä ihmisen on mentävä niiden luo, mieluiten avoimin mielin.

Mainstream-taidekentän vertaaminen demomaailmaan on mielenkiintoinen ajatus. Voidaan jopa pohtia, onko demoscene jossain mielessä eräänlainen taide-

maailman pienoismalli. Taideteoksia luodaan sille yleisölle, joka niitä haluaa nähdä, kuulla ja kokea, aivan kuten demot tehdään toisia demoharrastajia varten. Ihmiset, jotka eivät ole kiinnostuneita taiteesta, eivät vapaaehtoisesti käy taidenäyttelyissä ja kohdatessaan taidetta, varsinkin nykytaidetta, esittävät usein spontaaneja kommentteja kuten ”en voi ymmärtää tätä”. Samalla tavoin demoscenen ulkopuolinen maailma pitää demojen tekemistä nuorten nörttipoikien puuhasteluna, joka on helppo kuitata esimerkiksi sanomalla ”en voi ymmärtää, mikä tuossa viehättää”.

Nuorena minustakin olisi ehkä voinut tulla demontekijä. Luulen, että syy, joka on pitänyt minut erossa demokulttuurista, on siihen keskeisenä osana kuuluva kilpailumentaliteetti. Demotapahtumissa on totuttu järjestämään kilpailuja, joissa demosceneläiset laittavat toistensa aikaansaannoksia paremmuusjärjestykseen. Tämän lisäksi demosceneä on aina leimannut vahva hierarkkisuus, joka on tehnyt maineesta ja kunniasta kilpailemisesta ja eliittiin lukeutumisesta kutakuinkin koko toimintaa ohjaavia päämääriä (Reunanen 2010, 33—34).

Valitettavasti tätä kilpailuhenkisyyttä ei pääse pakoon taidemaailmassakaan, vaikka taideteosten ja taiteilijoiden ansioita arvioivatkin yleensä kriitikot ja kuraattorit eivätkä niinkään muut taiteilijat. Kilpailun voi ajatella olevan jopa raaempaa kuin demomaailmassa, joka on kuitenkin jäsenilleen harrastus eikä ammatti tai elinkeino. Taiteilija joutuu apurahoja saadakseen esittämään todisteita ansioistaan eli käytännössä vakuuttelemaan omaa paremmuuttaan vertaisiinsa nähden. Saadakseen taideteoksiaan kaupaksi on taisteltava ostajista, mikä on ymmärrettävästi vaikeaa ja vierasta monille taiteilijoille.

6 *SWARMIN* RAKENNE

Tässä luvussa pyrin valottamaan *Swarmin* toimintatapaa, rakennetta ja teknisiä ominaisuuksia. Vaikka tietokoneohjelmien toiminnan kuvausten ymmärtäminen on epäilemättä helpointa niille, jotka osaavat ohjelmoida, pyrin esittämään asiat mahdollisimman helppotajuisesti. Tarkoitukseni ei myöskään ole tässä yhtey-

dessä dokumentoida ohjelmakoodin jokaista yksityiskohtaa. Koska *Swarm* on avoimen lähdekoodin ohjelma, voi jokainen halukas tutustua koodiin itse.

6.1 Koodin jäsentely

Swarm on kirjoitettu kokonaan C-kielellä ja sen lähdekoodi koostuu useasta tiedostosta. Koska ohjelma on tehty toimimaan kolmessa eri käyttöjärjestelmässä – DOS:ssa, Windowsissa ja Linuxissa – ja näistä jokainen tarvitsee oman koodinsa, eri käyttöjärjestelmien käyttämät lähdekooditiedostot on sijoitettu omiin alihakemistoihinsa.

Kaikki kolme versiota käyttävät lisäksi samaa käyttöjärjestelmistä riippumatonta päämoduulia, joka koostuu kolmesta tiedostosta ja joka huolehtii ohjelman ydintoiminnoista. Näitä toimintoja ovat animaation toimintojen alustaminen, muistin hallinta, satunnaislukujen tuottaminen sekä itse animaation käynnissä pitämiseen tarvittavat toimet kuten parven ydinten liikuttelu ja parven piirtämiseen liittyvä matematiikka.

Päämoduulilla on oma versionumero, joka on tätä kirjoitettaessa 0.9.08. Versioiden numerointitapa on samansuuntainen kuin monissa avoimen lähdekoodin projekteissa kuten Linuxin ytimessä, *Gnome*-työpöytäympäristössä ja *GIMP*-kuvankäsittelyohjelmassa: ensimmäinen numero on eniten merkitsevä ja sitä nostetaan vain silloin, kun ohjelmassa tehdään merkittäviä muutoksia. Toinen numero kertoo, onko ohjelmaversio vielä kehitystyön alla (pariton numero) vai vakaa ja toimivaksi todettu (parillinen numero). Kolmatta numeroa nostetaan jokaisen pienenkin päivityksen yhteydessä.

Tämän lisäksi *Swarmin* DOS-, Windows- ja Linux-versioilla on omat versionumeronsa, jotka ovat päivämäärämuotoisia ja kertovat suoraan, milloin kyseinen versio on käännetty lähdekoodista ajettavaan muotoon.

6.2 Satunnaisuus

Tietokoneella toteutettu satunnaisuus on joko aitoa tai näennäistä. Aidon sattumanvaraisuuden aikaansaaminen tietotekniikalla on haastava tehtävä. Niinpä satunnaislukugeneraattorista puhuttaessa tarkoitetaan yleensä PRNG:tä eli näennäissatunnaislukugeneraattoria (engl. pseudo-random number generator). PRNG:t ovat algoritmeja, jotka tuottavat toisistaan riippumattomilta ja ennakoimattomilta näyttävien numeroarvojen sarjoja. Saadut arvot ovat kuitenkin täysin riippuvaisia siemenluvusta tai alkutilasta, johon algoritmi alustetaan, joten sama numerosarja on aina toistettavissa samalla alustuksella.

Näennäissatunnaisuus on monissa tilanteissa riittävä ratkaisu ja se voi jopa tulla kysymykseen aitoa satunnaisuutta paremmin. Erityisesti edellä mainittu PRNG:n tuottamien arvojen toistettavuus on usein hyödyllinen ominaisuus. Parhaat ja monikäyttöisimmät PRNG:t ovat monimutkaisia, mutta hyvin yksinkertaisellakin koodilla voidaan tuottaa tarpeeksi satunnaisia lukuja. PRNG:n valinta kannattaa aina suhteuttaa tilanteeseen ja ohjelman tarpeisiin.

Swarmissa käytetty PRNG on itse kehittämäni hyvin lyhyt ja yksinkertainen funktio, josta tietokoneen prosessori suoriutuu nopeasti. Tein sen ensimmäisen version vuonna 2007 *Game of Life* -näytönsäästäjään ja se soveltuu mainiosti *Swarmin* tarpeisiin, mutta se ei kelpaisi yleispäteväksi generaattoriksi. Alla oleva pseudokoodi kuvaa funktion toimintaa. Koodissa A ja B ovat 32-bittisiä kokonaislukumuuttujia eli ne voivat saada arvoja väliltä $[0, 2^{32}[$.

1. Vertaa A:n ja B:n arvoja bitti kerrallaan ja aseta A:n ne bitit nolliksi, jotka ovat samoja kuin B:ssä, ja muut ykkösiksi. Tätä kutsutaan poissulkevaksi TAI-operaatioksi (engl. exclusive or, XOR).
2. Kerro A 1,0625:llä eli toisin sanoen lisää A:han sen arvon kuudestoistaosa. Pyöristä alaspäin.
3. Kerro B 3:lla.
4. A:n saama arvo on PRNG:n tuottama satunnaisluku.

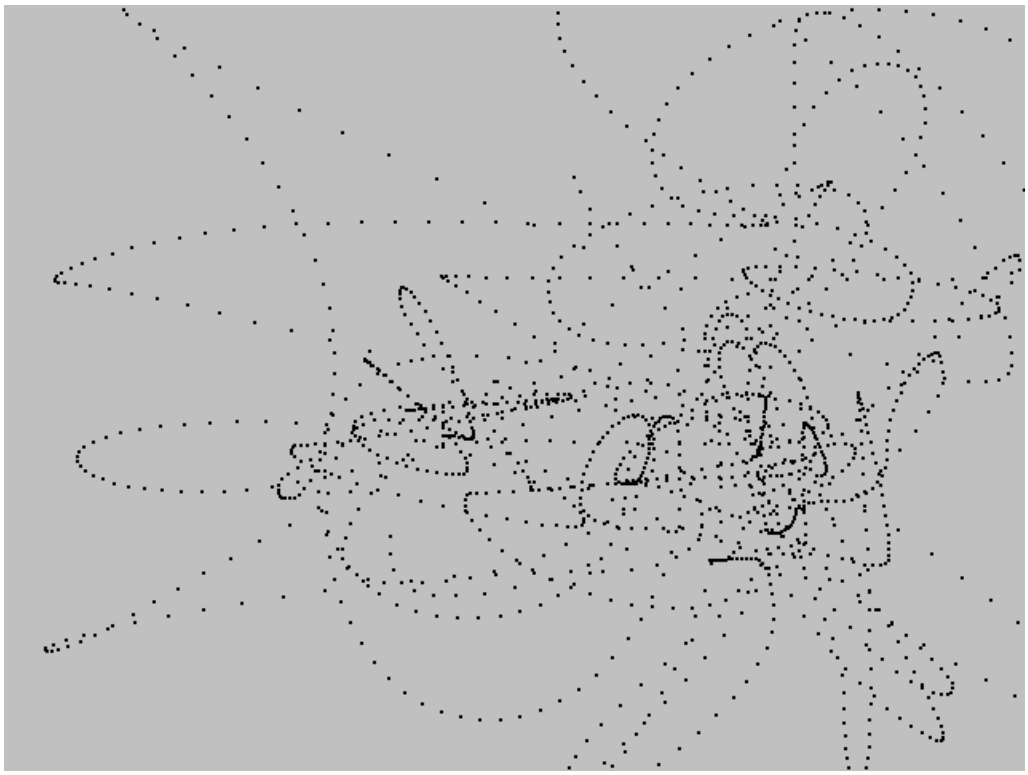
Aina, kun halutaan uusi satunnaisluku, ajetaan tämä sama funktio alkaen kohdasta 1. A:n ja B:n arvot siis säilyvät muistissa eivätkä muutu funktion ulkopuolella. Ennen PRNG:n ensimmäistä käyttöä A ja B on alustettava johonkin alkutilaan. A:n alustusarvoksi asetetaan käyttäjän määrittelemä siemenluku tai sen puuttuessa tietokoneen kellosta luettu numeroarvo, jolloin ohjelman ajo on periaatteessa ainutlaatuinen. B alustetaan mielivaltaisesti valittuun lukuarvoon 3 735 928 559 (heksadesimaaliesityksenä eli 16-kantaisessa lukumuodossa DEADBEEF).

6.3 Liikkeen logiikka

Lentävien olioiden parvi, jota *Swarm* esittää, näyttää koostuvan miljoonista yksittäisistä olioista. Tämä on illuusio, joka on pyritty saamaan aikaan yksinkertaisin keinoin. Ohjelma ei pidä kirjaa kaikkien ruudulla liikkuvien olioiden sijainnista, liikesuunnasta ja nopeudesta, vaan se liikuttelee parven kuutta kuvitteellista ydintä ja piirtää nopeassa tahdissa suuren määrän oliomassaa näiden ydinten ympärille.

Kuudesta ytimestä kutsun kolmea pääytimiksi ja loppuja kolmea häntäytimiksi. Ytimet liikkuvat hyvin yksinkertaisten periaatteiden mukaan: jokainen pääydin vaeltaa omaa kaartelevaa liikerataansa hakeutuen välillä muiden ydinten lähelle ja erkaantuen taas välillä omille teilleen. Häntäytimet liikkuvat suuremmalla nopeudella ja pyrkivät pysyttelemään pääydinten välimaastossa. Kuva 6.1 on piirretty seuraamalla kolmen pääytimen liikehdintää noin minuutin ajan.

Pääydinten liikeradoissa on hieman samanlainen periaate kuin Lissajous'n käyrissä (Weisstein 2004), sillä kunkin ytimen x- ja y-koordinaattia ohjaa erillinen sinifunktio. Tästä syystä ytimet ja niiden mukana koko parvi näyttää keinahtelevan puolelta toiselle pehmeästi.



Kuva 6.1 Kolmen pääytimen liikeradoista muodostuva kuvio

Tämä liikeratojen toteutuksen periaate oli yksi ensimmäisistä keksimistäni ideoista, kun aloitin *Swarmin* suunnittelun, ja se on pysynyt mukana siitä asti. Sitä vastoin se tapa, jolla pääytimet liikkuvat suhteessa toisiinsa, kävi ohjelmointityön ensimmäisten viikkojen aikana läpi muutamia eri vaiheita. Aluksi kokeilin kolmea eri mallia, joissa pääytimet vaeltelivat ruudun puolelta toiselle täysin toisistaan riippumatta. Tällaisilla ratkaisuilla parvi pysyi lähes koko ajan hajaantuneena sinne tänne; oikeastaan ruudulla ei näkynyt yhtä suurta parvea vaan kolme pienempää.

Kirjoitin siis pääytimille uuden satunnaisuuteen perustuvan toimintamallin, jossa ytimet pyrkivät liikkumaan toisiaan kohti 75 %:n todennäköisyydellä ja muulloin kääntymään takaisin tulosuuntaansa. Koska parvi näytti liiankin usein kerääntyvän suureksi möykyksi, vaihdoin todennäköisyyssuhteeksi myöhemmin 5:3 eli 62,5 % vastaan 37,5 %.

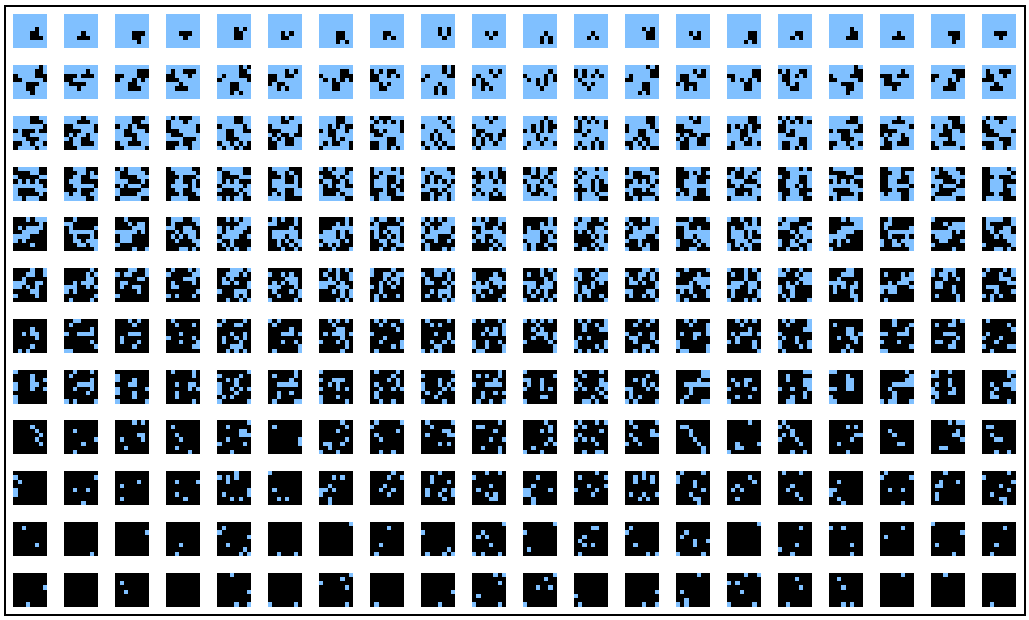
Häntäydinten liikkuminen oli alun perin toteutettu siten, että jokaista pääydintä seuraa yksi häntäydin, joka pyrkii ohjautumaan sitä kohti mutta liikkuu huomat-

tavasti sitä suuremmalla nopeudella. Näin parvelle saatiin elastisesti ja orgaanisesti vellova ulkomuoto. Myöhemmin aloin ajatella, että parven voisi saada näyttämään mielenkiintoisemmalta, jos myös häntäydinten liikkumistapaa kehittäisi hieman pitemmälle. Uudessa mallissa häntäydinten liikenopeus on ennallaan, mutta ne eivät hakeudu kohti pääytimiä vaan näiden väliin jäävää parven sisäosaa.

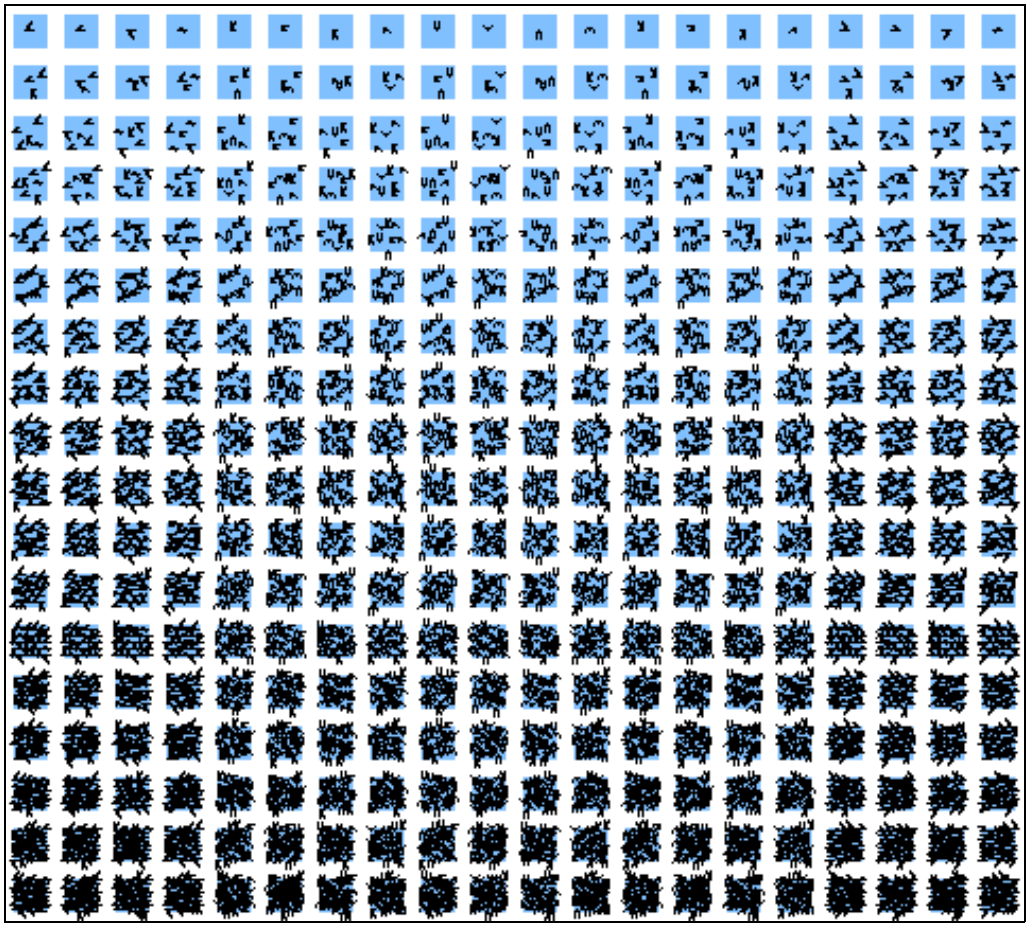
6.4 Kuva ruudulla

Swarmin DOS- ja Linux-versiot piirtävät kymmenen kuvakehystä sekunnissa, kun taas Windows-versiossa käyttäjä voi itse määritellä animaation nopeuden. Jokaisen kuvakehyksen piirtäminen tapahtuu kahdessa vaiheessa: ensin jokainen pää- ja häntäydin siirretään uuteen kohtaan liikeradallaan ja tämän jälkeen renderöidään koko kuvaruudun sisältö.

Swarm käsittelee käytettävissä olevaa kuva-alaa 8 × 8 pikselin kokoisiin neliöihin jaettuna. Tietyn mittaisen säteen rajaama alue ydinten ympärillä täytetään rasterikuviolla. Windows- ja DOS-versioissa on myös mahdollista käyttää 16 × 16 pikselin kokoisiin neliöihin perustuvaa vaihtoehtoista täyttökuviota, jossa lentävät oliot ovat korkeamman resoluution vuoksi lintumaisempia. Tämän lisäksi Windows-versio sallii koko kuvan suurentamisen tai pienentämisen tiettyjen rajojen puitteissa.



Kuva 6.2 Matalaresoluutioinen rasterikuvio



Kuva 6.3 Korkearesoluutioinen rasterikuvio

Kuvan 6.2 esittämä matalaresoluutioinen rasterikuvio koostuu 240 palasta, jotka on jaettu 12 valööriä eli parven tiheysastetta vastaaville riveille. Jokainen pala on 8 × 8 pikseliä. Kuvan 6.3 korkearesoluutioisen täyttökuvion palat sen sijaan ovat 24 × 24 pikselin kokoisia. Niitä piirretään ruudulle 16 × 16 pikselin kokoisiin neliöihin siten, että uloimmat pikselit – kuvassa sinisten neliöiden ulkopuolelle jäävät – menevät päällekkäin vierekkäisten neliöiden kanssa. Tämä auttaa häivyttämään ruudukkomaisuutta ja pehmentämään eroja parven tiheysasteiden välillä. Korkearesoluutioisessa täyttökuviossa on 18 tiheysastetta eli 360 palaa.

Jokaiseen neliöön piirretään sopiva pala rasterikuviosta sen mukaan, kuinka tiheä parvi on kyseisen neliön kohdalla, toisin sanoen kuinka monta ydintä osuu neliön lähipiiriin ja millä etäisyyksillä ne ovat. Palan valintaan vaikuttaa myös se, ovatko neliön lähellä olevat ytimet liikkeessä vasemmalle vai oikealle. Nopeatempoisessa sivusuuntaisessa liikkeessä oleviin parven osiin piirretään sivulta päin kuvattuja olioita.

Varsinaisen kuvan piirtämisestä vastaavat käyttöjärjestelmäkohtaiset rutiinit. Päämoduuli määrittelee, miten sakea parvi on missäkin kohdassa ruutua ja välittää tämän tiedon piirtomoduulille, jonka tehtäväksi jää ratkaista, miten kyseinen parven osa esitetään ruudulla. Piirtämisen toteutustapa on aina viime kädessä riippuvainen käyttöjärjestelmästä ja laitteistosta.

6.5 Äänet

Swarmin eri versioista vain Linux-versiossa on ääntä. Windowsissakin sitä olisi ollut mahdollista ottaa mukaan, mutta mielestäni näytönsäästäjän ei tarvitse äännellä. Toisaalta Windowsin multimediaominaisuuksien hyödyntäminen olisi edellyttänyt jonkin verran ylimääräistä opiskelua. Äänikoodin toteutus ei silti Linuxissakaan tapahtunut täysin vaivattomasti. Olin jo ennen *Swarmia* jonkin verran tutustunut digitaalisen äänen olemukseen ja käsittelytapoihin, mutten paria aiempaa kokeilua lukuun ottamatta ollut tehnyt omia ääntä tuottavia ohjelmia.

Ääntä tuottavat komponentit tietokoneissa ovat siinä määrin erilaisia, että olisi käytännössä mahdotonta kirjoittaa kokonaan itse ohjelma, joka toimisi kaikilla laitteistoilla. Tästä syystä ääniohjelmointiin on Linuxissa, kuten muissakin ympäristöissä, tarjolla valmiita koodikirjastoja, jotka huolehtivat ohjelman, käyttöjärjestelmän ja laitteiston välisestä viestinnästä. Tällaisia kirjastoja kartoittaessani arvelin, että laajalti käytetty *Advanced Linux Sound Architecture* eli *ALSA* olisi hyvä valinta, mutta sitä helppokäyttöisemmäksi kuvailtu *JACK Audio Connection Kit* vei voiton.

Swarmin äänimoduuli luo ohjelman käynnistyessä tietyn määrän abstrakteja rakenteita, joita kutsun generaattoreiksi. Niiden määrä ja ominaisuudet ovat käyttäjän määriteltävissä; nykyisessä versiossa niitä voi olla korkeintaan kuusi. Jokainen generaattori tarkkailee yhtä tai useampaa parven ydintä ja yhdistelee erilaisia digitaalisia ääniaaltomuotoja ydinten liikkeiden mukaan. Generaattoreita ohjaa kolme parametria:

1. Ytimen x-koordinaatti eli vaakasuuntainen sijainti ruudulla määrittelee, miten generaattorin tuottamaa ääntä panoroidaan vasemmalle tai oikealle.
2. Ytimen y-koordinaatti vaikuttaa äänen korkeuteen tiettyjen hertsirajojen sisällä.
3. Äänen voimakkuutta säädetään ytimen pysty- tai vaakasuuntaisen liikkeen nopeuden mukaan. Generaattori voidaan asettaa kytkeytymään päälle, kun nopeus ylittää tietyn kynnyksen.

Näiden kolmen parametrin ei tarvitse seurata saman ytimen x- ja y-koordinaattia, vaan äänen korkeus voi perustua yhteen ytimeen, panorointi toiseen ja voimakkuus kolmanteen.

Yksittäisen generaattorin tuottama ääni voi koostua enimmillään kolmesta erillisestä äänipätkästä, joita yhdistetään ja toistetaan loputtomana silmukkana. *Swarmin* nykyinen versio käyttää hyvin lyhyitä ja monimutkaisesti editoituja äänipätkiä, jotka silmukkatoistona kuulostavat erilaisilta keinotekoisilta surinoilta ja suhinoilta.

Suunnitellessani äänijärjestelmää *Swarm* oli muilta osa-alueiltaan käytännöllisesti katsoen valmis. Pyrin siihen, että äänikoodi olisi mahdollisimman helposti liitettävissä ohjelmaan ilman, että olemassa olevaan koodiin tarvitsisi tehdä merkittäviä muutoksia. Lopputulos toimii mielestäni hyvin, vaikka sitä voidaankin pitää eräänlaisena ”quick and dirty” -ratkaisuna eli monet kohdat on ratkaistu ensimmäisellä mieleen tulleella tavalla. Osittain tämä johtui tiukasta aikataulusta: kirjoitin äänikoodin viikossa, koska halusin *Swarmin* mukaan ryhmänäyttelyyn Tarttoon.

6.6 Avoin lähdekoodi

Kun tietokoneohjelmia levitetään avoimen lähdekoodin periaatteen mukaisesti, tarvitaan lisenssi, jossa määritellään ohjelman tekijän ja käyttäjän oikeudet. *Swarmin* kohdalla käyttämäni lisenssi on Free Software Foundationin kehittämä GPL eli GNU General Public License, joka on yksi eniten käytetyistä avoimen lähdekoodin lisensseistä (Black Duck 2011).

GPL määrää, että ohjelman lähdekoodi on saatavissa siellä, missä itse ohjelmakin. Linux-ohjelmia levitetään tavallisesti pelkkänä lähdekoodina, koska ohjelmointityökalut, kuten kääntäjä, kuuluvat olennaisena osana kaikkiin Linux-jakeluihin ja näin ollen jokainen Linuxin käyttäjä voi kääntää koodin ajettavaksi ohjelmaksi. Avoimen lähdekoodin ohjelmia on nykyään yhä enemmän saatavissa myös Windows-ympäristöön, jossa ohjelmointivälineet on hankittava erikseen.

GPL sallii kenen tahansa tehdä muutoksia lähdekoodiin ja kehittää sen pohjalta uusia muunnelmia. GPL:n keskeisin vaatimus on, että kaikki muunnellut versiot ovat myös saman lisenssin alaisia. Näin pyritään varmistamaan koodin vapaa leviäminen. (FSF 2007.)

Monet tunnetut avoimen lähdekoodin projektit ovat suurten yhteisöjen ponnistuksia, joissa ohjelmoijat saattavat työskennellä eri puolilla maailmaa ja tuottaa pieniä lisäyksiä ja parannuksia koodiin. Ohjelmat hioutuvat vähän kerrallaan pa-

remmiksi. Projektiin osallistujia saattaa tulla ja mennä; yleensä kukaan ei saa palkkaa työpanoksestaan, eikä raha olekaan työssä motiivina.

Swarm sen sijaan on luotu toisenlaisella lähestymistavalla avoimen lähdekoodin toimintamalliin. Vaikka minulle on ollut alusta alkaen selvää, että haluan sen lähdekoodin olevan aikanaan vapaasti saatavilla, olen työstänyt sitä koko ajan itse ottamatta vastaan apua keneltäkään toiselta ohjelmoijalta. En edes ole pyrkinyt tiedottamaan projektistani tai sen avoimuudesta missään. Tämä liittyy tietenkin siihen, että kyseessä on opinnäytetyöni, ja tuntuu mielekkäältä kyetä korostamaan, että se on kokonaan oman työni tulosta. Avoin lähdekoodi on siis tässä tapauksessa enemmän periaatekysymys; se on mielestäni ehdottomasti kannatettava toimintatapa, vaikken olekaan harjoittanut suurta avoimuutta tätä ohjelmaa tehdessäni.

Avoimuus toteutuu *Swarmin* kohdalla siten, että kun ohjelma on valmis, päästän sen käsistäni, jotta joku muu voi halutessaan kehittää sitä eteenpäin. Olen ikään kuin antanut oman panokseni. Periaatteessa voitaisiin sanoa, että avoimen lähdekoodin ohjelmointihankkeissa tavallisempi verkostomainen toimintatapa korvautuu tässä ketju- tai puumaisella kehitysmetodilla: kun yksi ohjelmoija on työstänyt lähdekoodista oman versionsa, se siirtyy seuraavalle.

Tämä kuulostaa toki hienolta, mutta käytäntö lienee hieman toisenlainen. Olen hyvin tietoinen siitä, että nykymaailmassa, jossa avointa lähdekoodia on jo kaikkialla, tällainen joka suhteessa pienieleinen ohjelma katoaa helposti niin kutsuttuun bittiavaruuteen eikä herätä minkäänlaista huomiota. Tämä ei suinkaan häiritse minua, vaan olen tyytyväinen jo siihen, että voin antaa muille ohjelmoinnista ja ohjelmistotaiteesta kiinnostuneille mahdollisuuden tutustua työhöni.

7 YHTEENVETO

Olen tässä tekstissä pyrkinyt hahmottamaan jonkinlaista kuvaa tietokonetaiteen laajasta kentästä ja siitä, miten *Swarm* ja muu oma tuotantoni sijoittuu siihen. Olen löytänyt joitakin osastoja tai lokeroita, joihin *Swarm* voidaan lukea: ensisi-

jaisesti se on ohjelmistotaideteos ja lisäksi siinä on generatiivisen taiteen tunnusmerkkejä. Itse keksimäni näytönsäästäjätaide on kategoria, johon *Swarmin* Windows-versio sopii.

Etsiessäni lähdemateriaalia opinnäytetyöni kirjallista osaa varten olen omaksunut paljon uutta ja hyödyllistä tietoa näytönsäästäjistä, ohjelmistotaiteesta ja erityisesti demoscenestä, joka oli minulle ennestään melko tuntematon maailma, mutta jossa huomaan olevan monia yhteneviä piirteitä taiteelliseen toimintaani ja taiteeseen yleensäkin.

Eräs hyvin mielenkiintoinen ilmiö, johon olen tämän prosessin myötä pintapuolisesti tutustunut ja joka vaikuttaa lähemmän tutkimisen arvoiselta, on *Processing*-ohjelmointiympäristö (http://processing.org). Se on tarkoitettu erityisesti kuvataiteilijoiden välineeksi ja sen kerrotaan olevan helppo oppia, vaikkei ohjelmointitaustaa olisikaan. *Processing* mahdollistaa hyvin monipuolisten tietokonetaideteosten luomisen, ja mikä parasta, se on myös avoimen lähdekoodin projekti. (Whitelaw, Lieserin 2009, 176—177 mukaan.)

Linux-ohjelmoinnin opettelu ehti pitkään olla minulle suunnitelma, jonka täytäntöönpanoon ei koskaan tuntunut olevan aikaa. Nyt voin tyytyväisenä sanoa päässeeni *Swarmin* myötä siihen viimein käsiksi. Jatkossa tulen luultavasti työstämään paljonkin uusia ohjelmointiprojekteja, myös ohjelmistotaideteoksia, nimenomaan Linux-ympäristöön. *JACK*- ja *svgalib*-kirjastot, joihin olen *Swarmin* kautta tutustunut, vaikuttavat hyvin käyttökelpoisilta tulevia hankkeita ajatellen.

Swarm on lähdekoodeineen ladattavissa osoitteessa http://www.tsjh-taide.net.

KUVAT

LÄHTEET

Baumgärtel, T. 2000. Jetzt nicht die Mouse anfassen! http://www.heise.de/tp/r4/artikel/4/4211/1.html (Luettu 18.1.2011)

Black Duck. 2011. Black Duck Open Source Resource Center. http://www.blackducksoftware.com/oss/licenses (Luettu 26.3.2011)

Büchler, P. & Sandison, C. 2004. The Operations Necessary to Solve a Problem. Framework 1, 66—69. http://www.frame-fund.fi/images/stories/pdf/Fw2004/fw-issue2-screen.pdf (Luettu 9.1.2011)

Demoscene Documentary. 2010. Demoscene Documentary Series, Episode 1: Early 1990 Era – Moving from Cracking to Demos. http://www.youtube.com/watch?v=J2Jz1bu2CcA (Luettu 19.1.2011)

digitalcraft. 2002. origami digital – Demos without Restrictions. http://www.digitalcraft.org/index.php?artikel_id=411 (Luettu 18.1.2011)

Draves, S. 2000. Metaprogramming Emergent Graphics. http://draves.org/meg (Luettu 13.3.2011)

Draves, S. 2005. The Electric Sheep Screen-Saver: A Case Study in Aesthetic Evolution. http://draves.org/evomusart05/evomusart05draves.pdf (Luettu 13.3.2011)

Draves, S. 2009. History. http://scottdraves.com/history.html (Luettu 12.3.2011)

EAI. 2009. Electronic Arts Intermix. Interview with Sara Tucker, Dia Art Foundation. http://www.eai.org/resourceguide/preservation/computer/interview_tucker.html (Luettu 27.2.2011)

FSF. 2007. Free Software Foundation. The GNU General Public License v3.0. http://www.gnu.org/licenses/gpl.html (Luettu 26.3.2011)

Galanter, P. 2003. What is Generative Art? http://philipgalanter.com/downloads/ga2003_what_is_genart.pdf (Luettu 15.1.2011)

Goriunova, O. 2007. Software Art. http://www.digitalartistshandbook.org/softwareart (Luettu 20.1.2011)

Henderson, P. 2002. Functional Geometry. Higher Order and Symbolic Computation 15, 349—365. http://www.brics.dk/~hosc/local/HOSC-15-4-pp349-365.pdf (Luettu 17.1.2011)

The Jargon File. 2003. Bloatware. http://catb.org/jargon/html/B/bloatware.html (Luettu 26.2.2011)

Jäämeri, H. 2011. Vaikeinta on luoda sattuma. Suomen Kuvalehti 95, 44—51.

Kauppinen, J. O. 2009. Alt järjestää 50 000 ihmisen demobileet Helsingissä. http://dome.fi/pelit/uutiset/alt-jarjestaa-50-000-ihmisen-demobileet-helsingissa (Luettu 16.1.2011)

Klima, J. 2007. Aesthetics of *ecosystm*. Teoksessa Vesna, V. (toim.) Database Aesthetics: Art in the Age of Information Overflow. Minneapolis & Lontoo: University of Minnesota Press, 260—268.

Lieser, W. 2009. Digital Art. Potsdam: H. F. Ullmann Publishing.

Mirapaul, M. 2000. Screen Savers as Artists' Medium. http://www.nytimes.com/2000/11/23/technology/23save.html (Luettu 18.1.2011)

PEM. 2010. Peabody Essex Museum. Charles Sandison. http://www.pem.org/writable/resources/document/sandison-interview_copy1.pdf (Luettu 9.1.2011)

Pirtola, P. 2010. Curriculum Vitae. http://lowfidelity.org/~nosfe/cv/index.html (Luettu 22.1.2011)

Price, B. D. 2004. Scholarly Perspectives on Digital Art. http://kellysheridan.com/blog/2006/08/31/scholarly-perspectives-on-digital-art (Luettu 27.3.2011)

Price, B. D. 2005. What about Algorithms, Fractals and Programmed Art?? http://digitalrising.blogspot.com/2005_12_01_archive.html (Luettu 16.1.2011)

Price, B. D. 2006. Big Shots Ignore Columnist. World Stunned. http://digitalrising.blogspot.com/2006_04_01_archive.html (Luettu 27.3.2011)

Readme. 2005. Readme 100. http://readme.runme.org (Luettu 20.1.2011)

Reunanen, M. 2010. Computer Demos – What Makes Them Tick? Aalto-yliopisto. Teknillinen korkeakoulu. Informaatio- ja luonnontieteiden tiedekunta. Mediatekniikan laitos. Opinnäytetyö. http://www.kameli.net/demoresearch2/reunanen-licthesis.pdf (Luettu 12.1.2011)

Reunanen, M. & Silvast, A. 2010. Demoscene Research: Bibliography. http://www.kameli.net/demoresearch2/?page_id=4 (Luettu 21.1.2011)

Reynolds, C. 2001. Boids (Flocks, Herds, and Schools: a Distributed Behavioral Model). http://www.red3d.com/cwr/boids (Luettu 15.1.2011)

Saarikoski, P. 2003. Demoskene.katastro.fi -näyttely Kiasman Kontissa 28.3.–15.6.2003. http://www.film-o-holic.com/artikkelit/demoskenekatastrofi-nayttely-kiasman-kontissa (Luettu 13.1.2011)

Tasajärvi, L. (toim.), Stamnes, B. & Schustin, M. 2004. Demoscene: The Art of Real-Time. Helsinki: Even Lake Studios & katastro.fi.

Triangulation Blog. 2011. http://www.triangulationblog.com/2011/01/scott-draves.html (Luettu 12.3.2011)

Tucker, S. 2009. Introduction to *Wrestling with the Blob Beast*. http://awp.diaart.org/johnson/intro.html (Luettu 24.1.2011)

Weisstein, E. W. 2004. Lissajous Curve. http://mathworld.wolfram.com/LissajousCurve.html (Luettu 25.3.2011)

Verostko, R. 1999. Algorithmic Art: Composing the Score for Visual Art. http://www.verostko.com/algorithm.html (Luettu 18.1.2011)

Verostko, R. 2006. The Algorists. http://www.verostko.com/algorist.html (Luettu 16.1.2011)

YLE Uutiset. 2010. Suomen merkittävin kuvataidepalkinto Ars Fennica on annettu skottisyntyiselle Charles Sandisonille. http://areena.yle.fi/video/1479367 (Luettu 15.1.2011)