SISÄLLYS

1 JOHDANTO

Opinnäytetyöni käsittelee musiikkisovellutusten ohjelmointia Commodore 64 –kotitietokoneella, mikä on varsin marginaalista ja pitkälle erikoistunutta tietoa jopa musiikkiteknologian kentässä. Tästä seikasta johtuen olen tukeutunut monen ihmisen apuun tutkielman eri työvaiheissa ja hajanaisen taustatiedon keräämisessä. Nimeltä haluan mainita seuraavat tutkielmani kannalta erityisen merkittävät henkilöt ja tahot:

Alistair Bowness – Arkanoid-musiikkirutiinin lähdeaineiston selvennykset ja tarkennukset

Arto Koivisto – Ääninäytteissä käytetyn MIDIbox SID –syntetisaattorin rakentaminen

Anthony McSweeney – Monty On The Run –musiikkirutiinin lähdeaineisto

Harri Kentala – Liitteenä olevan cd-levyn koostaminen

High Voltage SID Collection (HVSC) – Liitteinä olevat pelimusiikit

Per Håkan Sundell – SoundMonitor-ohjelman sekä Commodore 64 –tietokoneen ääniominaisuuksien testauksessa käytetyn CCS64-emulaattorin ohjelmoija

Stefano Tognon – Driller- ja Arkanoid-musiikkirutiinien lähdeaineisto

1.1 Commodore 64

Johdannon Commodore Business Machines –yhtiötä ja sen tuotteita koskeva asiasisältö on koottu kokonaisuudessaan WWW-dokumentista Commodore Computer History *ja* IEEE Spectrum (3/1985) *-lehden artikkelista* Design Case History: The Commodore 64. *Kaikki muut johdannon lähteet on merkitty viitteillä asiayhteyteensä.*

Vuoden 1982 tammikuussa Yhdysvaltain Las Vegasissa järjestettiin kulutuselektroniikan messutapahtuma Winter Consumer Electronics Show (Winter CES), jossa Yhdysvaltalainen Commodore Business Machines –yhtiö (CBM) esitteli uusimman luomuksensa, Commodore 64 –kotitietokoneen (C64). Tietokone oli alunperin suunniteltu valtaamaan kasvavat konsoli- ja videopelimarkkinat, mutta viime hetken linjamuutoksella yhtiön silloinen johtaja Jack Tramiel vaihtoi tavoitteeksi luoda lippulaiva ohjaamaan kotitietokoneiden kehityksen uusille urille. Ja juuri sen Commodore 64 teki.

Menestyksekkään ensiesittelynsä jälkeen C64 kiirehdittiin tuotantoon suuren kiinnostuksen takia ja elokuussa 1982 tuotanto oli jo niin suurta, että koneita voitiin toimittaa myös jälleenmyyjille. Vielä vuoteen 2001 mennessä ei yksikään kotitietokone ollut yltänyt C64:n hulppeaan rajapyykkiin, 30 miljoonaan myytyyn yksikköön[1] (Newport, S. 2001). C64:n sisältämä tekniikka sekä hinta/laatusuhde mullistivat kotitietokonemarkkinat ja nostivat kotitietokoneiden suorituskyvyn uudelle asteelle.

Aikanaan helppo C64:n käyttöliittymä ja poikkeuksellisen matala hinta ominaisuuksiinsa nähden tekivät tietokoneharrastuksesta massojen hupia; muiden tietokonevalmistajien oli pakko laskea hintojaan ja lisätä ominaisuuksia omiin koneisiinsa pysyäkseen mukana kilpailussa. Vuoteen 1983 mennessä kotitietokonemarkkinoille oli myös ilmaantunut äkillisesti huomattava määrä uusia yrityksiä, mikä entisestään lietsoi hinnanalennuksia ja kilpailua (Huhtamo, E. & Kangas, S. (toim.) 2002, 55).

Suurille tietokonemarkkinoille tuotettiin myös paljon ohjelmia. Brittiläinen tietokonevalmistaja Sinclair keksi ensimmäisenä mainostaa kotitietokonettaan

[1] Vertailun vuoksi mainittakoon, että Apple Computer –yhtiön kaikkia Macintosh–tietokonemalleja ei ole tähän päiväänkään mennessä valmistettu yhtä paljon kuin pelkästään Commodore 64 –sarjan tietokoneita (Commodore 64 2005).

vuonna 1981 välineenä oppia ”tietotekninen lukutaito”, ja tämä tietokoneharrastuksen suuntaus säilyikin pitkään nimenomaan Britanniassa vaikuttaen myös ohjelmisto- ja peliteollisuuteen, joiden innovatiivisimpina johtomaita 1980–luvulla Britannia ja Japani olivat (Huhtamo, E. & Kangas, S. (toim.) 2002, 55. sekä MacKenzie, J. 1996).

Myös C64:n mukaan pakatut ohjekirjat johdattelivat uuden omistajansa ohjelmoinnin pariin kehittämään ”tietoteknistä lukutaitoaan”, puhumattakaan säännöllisistä tietokonealan julkaisuista (esim. Compute! (USA) 1983-1991, Commodore User (Englanti) 1982-1998, Your Commodore (Englanti) 1984-1990, MikroBitti (Suomi) 1984->), jotka sisälsivät joka numerossaan monia sivuja ohjelmakoodia, jota innokkaat lukijat pystyivät omalla kotikoneillaan kokeilla.

Nuorten ohjelmoijien määrä alkoikin kasvaa eksponentiaalisesti. Sen sijaan, että tietokoneen omistaja olisi ostanut tarvitsemansa ohjelmiston valmiina, hän saattoi itse kehittää ohjelman tekemään tarvittavan työn. Nuorempien ohjelmointiharrastajien keskuudessa epäilemättä pelien ja muiden hupiohjelmien tekeminen oli suositumpaa puuhastelua koneen parissa kuin esimerkiksi tekstinkäsittelyohjelman suunnittelu. Tosin hyötyohjelmien koodaaminen tuotti myös ensimmäiset teinimiljönäärinsä 1980-luvulla[2].

C64:n tekniset ominaisuudet innoittivat ohjelmoijia kokeilemaan uusia mahdollisuuksia ja ennen kaikkea C64:sta varten suunnitellut uudet ääni- ja videomikrosirut houkuttivat tietokoneiden käyttäjiä siirtymään kilpailijoista Commodoren kannattajiksi. C64:n aikanaan erinomaiset ominaisuudet audiovisuaalisella alueella olivat perua alkuperäisestä ajatuksesta luoda nimenomaan uusi pelikone haastamaan konsolimarkkinat. Vaikka C64 viime hetkillä päätettiin muuttaa pelkästä pelikoneesta yleishyödylliseksi ja tehokkaaksi kotitietokoneeksi, sen asema markkinoilla kääntyi hyvin pian toimiston apulaisesta ja kodin dokumenttien järjestäjästä viihteen keskukseksi[3]. Sen lisäksi että koneessa oli laadukkaat ääni- ja grafiikkaominaisuudet, se sisälsi myös vaativaan käyttöön tarkoitetun

[2] Ranskalainen Cyrille de Vigdemont vaihtoi 14-vuotiaana jokapäiväisen koulunkäyntinsä kirjekurssiksi, sillä hänen ohjelmoimiensa hyötyohjelmien (Caissor & Remember) markkinointi ja tuottojen hallinnointi kärsivät tavanomaisista koulupäivistä (Rapo, P. 1986, 53-54).

[3] Tämä suuntaus koski myös muiden laitevalmistajien kotimikroja (Huhtamo E. & Kangas S. (toim.) 2002, 56).

mikroprosessorin ja muistimäärän. Nämä ominaisuudet yhdistämällä C64:lla toteutetuista peleistä ja ohjelmista tuli väistämättä aikansa eliittiä[4].

Commodoren tytäryhtiön, MOS Technologyn, C64:een suunnittelema äänisiru, SID (Sound Interface Device), oli jotain aivan uutta tietokoneiden äänentuotossa – itse asiassa C64 oli ensimmäinen kotitietokone, jossa oli sisäänrakennettu syntetisaattori. Ennen C64:n ilmestymistä kotitietokoneissa olleet ääniominaisuudet olivat lähinnä ääniefektejä varten, eivätkä soveltuneet laadukkaan musiikin tuottamiseen. Äänisirun suunnitellut Robert Yannes pyrkikin tekemään äänisirusta itsenäisen tuotteen, jota voitaisiin käyttää muissakin sovelluksissa, mutta pian C64:n tuotannon alettua kävi ilmi, että koko mikrosirutuotanto menee tietokoneiden kalustamiseen rajusti kasvaneen kysynnän takia, eikä muita äänisirua käyttäviä sovelluksia ehditty saada tuotantoon Commodore–yhtiön elinaikana (Varga, A., 1996).

1.2 Ohjelmointisovellusten kehittyminen

Commodore–yhtiön tuotteiden huima menekki ja tekniset ominaisuudet olivat suurena vaikutteena myös ohjelmoinnin alakulttuurien muodostumiseen ja laajentumiseen. Nuorten, valtaosin miespuolisten, tietokoneharrastajien keskuudessa ohjelmoinnista ja tietokoneen hallinnasta tuli tapa ilmentää omaa luovuuttaan ja nokkeluuttaan usein hyvin kyseenalaisissakin yhteyksissä. Kaupallisesti tuotettujen pelien ”kräkkääminen” (eng. cracking, murtaminen), eli kopiointisuojausten poistaminen, oli erityisen suosittu tapa tehdä taitojaan tunnetuksi[5]. Lähes poikkeuksetta koodimurron suorittanut ohjelmoija(t) lisäsi pelin alkuun ylimääräisen intron, missä hän pystyi

[4] Jo ensimmäisestä tietokonepelistä (Steve Russelin Spacewar vuodelta 1961) lähtien pelejä alettiin käyttämään myös tietokoneen ominaisuuksien ja toimintatehon esittelijöinä myynninedistämistarkoituksessa (Huhtamo E. & Kangas S. (toim.) 2002, 50-51). Näyttävät pelit vaikuttivatkin suuresti C64:n nousuun kotitietokoneiden markkinajohtajaksi 1980-luvulla.

[5] Vuonna 1986 Englannissa luettiin ensimmäiset tuomiot tietomurroista ohjelmoinnin harrastajille Steve Gold ja Robert Schifree. Näin ”hakkeroinnista” tuli virallisesti rikollista toimintaa Englannissa (Uutisia 1986, 4). Mielenkiintoista on huomata, että mainostaessaan uusinta modeemiaan ”hakkereille ja muille ammattilaisille” (Nokian modeemimainos 1987) Nokia haluaa selkeästi erottaa termit hacker ja cracker toisistaan: hakkeri on mainoksen mukaan ”kehittynyt harrastaja, ’sähköpostin ritari’ ”, ja cracker taas ”rikollinen tietoverkkoihin murtautuja”. Näihin termeihin liittyikin voimakas asenteellinen vivahde, ja pitkään niiden merkitys (varsinkin maallikkojen keskuudessa) olikin epäselvä – tässä yhteydessä Nokia ei selvästikään halunnut leimautua ”crackereiden” tukijaksi. Harrastajapiireissä cracking tarkoittaa myös jonkin ohjelman toimintaperiaatteen selvittämistä, eli koodin purkamista – ei välttämättä aina kopiointisuojauksen poistamista. Termi hacking liittyy taas ennen kaikkea omien innovaatioiden tekemiseen tekniikan alalla.

väläyttämään muitakin ohjelmointitaitojaan ja esittelemään itsensä nimimerkin suojissa. Samaan aikaan yleistynyt modeemien käyttö tietokoneiden välisessä kommunikoinnissa aiheutti piraattipelien leviämisen kulovalkean tavoin ympäri maailman.

Toinen tapa aiheuttaa kateutta muissa harrastajissa oli tehdä demoja, eli itsenäisiä esittelyohjelmia, jotka koostuivat erilaisista kuvista, animaatioista, musiikeista ja muista ohjelmoijan taitoja esittelevistä efekteistä ja ohjelmointikikoista. Demot eivät olleet yleensä interaktiivisia tai käytännöllisiä ohjelmia, vaan ohjelmoinnin harrastajien, hakkereiden (eng. hack, selviytyä jostakin), tuottamaa tietokonetaidetta. Tuotoksia arvioitiin esteettisyyden, ohjelmointitekniikan ja omaperäisyyden perusteella. Oivaltaminen ja innovaatioiden tekeminen olivat itsetarkoitus, kuten jo 1950-luvulla Yhdysvalloissa syntyneen alkuperäisen hacking–periaatteen mukaan pitääkin olla (Huhtamo, E. & Kangas, S. (toim.) 2002, 49-50).

Nämä tietokoneharrastajat pitivät yhteyttä toisiinsa ympäri maapallon ns. purkkien (eng. box) kautta, joita voi verrata nykyään internetistä löytyviin keskustelupalstoihin. Purkki toimi palvelimena eli yhteystietokoneena sekä tietokantana tietokoneiden käyttäjille, jotka lähettivät viestejä toisilleen modeemien avulla puhelinverkkoja pitkin. Tiedostojen vaihdon lisäksi käytiin keskustelua kaikista mahdollisista ajankohtaisista ohjelmointiin ja tietokonekulttuuriin liittyvistä aiheista. Tälle tietokonemaailman alakulttuurille vakiintui englanninkielinen nimitys ”the scene” (eng., tapahtumapaikka, näyttämö), ja tietenkin eri tietokonemerkkejä käyttävät harrastajat pyörivät oman konemerkkinsä ”tapahtumapaikoilla” – esimerkiksi C64-scene, Atari-scene.

Harva ohjelmoija pystyi hallitsemaan kaikkia koneen osa-alueita, joten oli tyypillistä, että samanhenkiset harrastajat päätyivät muodostamaan ryhmittymiä, joissa yksi jäsen erikoistui grafiikan tekoon, yksi musiikkiin, yksi animaatioiden toteuttamiseen jne. Näin oli laita myös pelien ohjelmoinnin suhteen. Varsinkin musiikin ohjelmointi eriytyi nopeasti omaksi alakseen peliteollisuuden tuotannossa. Säveltäjät työskentelivät usein enemmän tai vähemmän urakkapalkalla eikä heidän taiteelliseen työhönsä pahemmin tilaajan puolelta puututtu. Tähän oli ainakin kaksi syytä: sovittaminen ja sävellysten ammattimainen työstäminen vaatii paljon perehtyneisyyttä

musiikkiin ohjelmointitaidon lisäksi (maallikon on vaikea ohjata säveltäjän työtä tarkasti ilman yhteistä kieltä tai säveltapailun taitoja) ja toisekseen pelien musiikki oli helppo kirjoittaa itse pelin ohjelmakoodista erillään ja jälkeenpäin liittää osaksi kokonaisuutta, toisin kuin pelin sisältämä grafiikka tai animaatio.

Pelimusiikkien ohjelmoijista kehittyikin oma ammattikuntansa, jonka arvostus ilmeni tietokonelehtien peliarvosteluissakin, joissa pelien äänet ja musiikki arvosteltiin erillisenä ominaisuutena[6]. Kuuluisan säveltäjän nimen saaminen pelipaketin kanteen oli hyvä myyntivaltti pelien tuottajille (MacKenzie, J. 1996 ja Interview with Rob Hubbard 1986), ja menestyksekkäimmät säveltäjät saattoivatkin toimia vapaina työntekijöinä ohjelmointimarkkinoilla ja saada paremman tuoton sävellyksilleen kuin ollessaan vakituisessa työsuhteessa.

1.3 Commodore 64 ja musiikki

Todistuksena C64:n tehdyn musiikin elinvoimaisuudesta kertoo haku internetistä Google–hakukoneella sanoilla *sid*, *music* ja *c64*, jotka tuottivat 93 900 osumaa. URL-osoitteesta www.hvsc.c64.org löytyy SID–musiikkiin erikoistunut tietokanta, The High Voltage SID Collection (HVSC), johon pyritään keräämään kaikkien C64:lle julkaistujen pelien musiikit ja äänitehosteet. Tämän lisäksi tietokanta sisältää demoihin sävellettyä musiikkia ja muita SID-äänisirulla sävellettyjä teoksia. Kirjoittamishetkellä arkistossa oli 29 055 musiikkitiedostoa.

Internetissä toimii myös radio nimeltä Kohina (www.kohina.com), joka on erikoistunut soittamaan ainoastaan 8- ja 16-bittisten tietokoneiden ja konsolien demo- ja pelimusiikkia. Www.c64audio.com -sivuston kautta voi tilata cd–julkaisuja vanhoista C64:n pelimusiikeista: alkuperäismusiikkeja soitettuna C64:n äänisirulla, C64:lla tehtyjä remix–versioita suosituimmista pelimusiikeista, akustisilla ja sähköisillä soittimilla tehtyjä tarkkoja jäljennöksiä pelien tunnussävelmistä jne. Mainittakoon vielä, että numerossaan 9/1995 BYTE–lehti listasi SID-mikrosirun maailman 20 merkittävimmän mikrosirun joukkoon (Most Important Chips 1995).

Robert Yannesin alkuperäinen ajatus SID–sirun käyttämisestä muissakin kaupallisissa sovellutuksissa kuin kotitietokoneissa on myös toteutunut. Asialla ei tosin ole Commodore eikä mikrosirun suunnitellut MOS Technology, vaan aivan toisenlaiset yhtiöt, kuten elektronisiin soittimiin erikoistunut Elektron, joka julkaisi vuonna 1997 ainoastaan SID-sirua äänenlähteenään käyttävän MIDI-syntetisaattorin nimeltä SidStation. Myös SID–siruun pohjautuvia PC-äänikortteja löytyy markkinoilta, esimerkkinä mainittakoon Hard Softwaren HardSid.

Toisin sanoen ”C64-skene” on tänä päivänä kaikkea muuta kuin kuollut. Commodoren tietokoneiden parissa ohjelmointiharrastuksensa aloittaneet teinit ovat tänä päivänä ohjelmoinnin ensimmäisiä itseoppineita ammattilaisia, jotka ovat suurelta osin vastuussa tietoyhteiskunnan käsitteen muovaamisesta ja tietotekniikan muuttumisesta ihmisten jokapäiväiseksi apuvälineeksi. Internet-julkaisuissa voidaan nyt vapaasti puhua aroistakin aiheista Commodore 64:n ohjelmiin ja laitteisiin liittyen[7], sillä C64:n tuotanto lopetettiin vuonna 1993 (mikä lopetti samalla myös ohjelmistojen uustuotannon) eikä koneella ja sille tehdyillä tuotteilla enää ole merkittävää kaupallista arvoa[8].

Samoin tekijänoikeuksien osalta tilanne on ollut 80-luvun puolella sekava[9], eivätkä asianosaiset ymmärrettävästi mitenkään osanneet varautua 2000–luvun ”C64–retrobuumiin” ja Internetin räjähtävään kasvamiseen

---

[6] Esim. MikroBitti jaotteli arvostelun ääniin, grafiikkaan ja pelattavuuteen/kiinnostavuuteen, sekä yleisarvosanaan. Pelkästään peleihin erikoistunut englantilainen Zzap!64–lehti taas jaotteli arvion ulkoasuun, grafiikkaan, ääniin, pelattavuuteen, pitkäikäisyyteen, hinta/laatu-suhteeseen ja yleisarvioon.

[7] Esim. SID –mikrosirun patenttihakemus löytyy Internet-osoitteesta http://www.delphion.com/details?pn=US04677890_ ja monien hyötyohjelmien sekä pelien lähdekoodit ovat julkisessa jakelussa lukemattomilla Commodorelle omistetuilla nettisivustoilla.

[8] Vielä viime vuonna Commodore–tuotemerkin kaikki oikeudet omistanut Hollantilainen Tulip Computers tosin yritti sulkea Internetistä kaikki Commodore–tuotemerkkiä käyttävät sivustot, jos ne eivät suorittaneet lisenssimaksua merkin käytöstä. Hanke kuitenkin kaatui C64–sivustojen valtavaan määrään, retroharrastajien vastustukseen ja loppujen lopuksi riittämättömään taloudelliseen hyötyyn oikeusteitse perittävistä lisenssimaksuista. Vuoden 2004 lopulla tuotemerkin oikeudet siirtyivät 24 miljoonan euron yrityskaupassa Yhdysvaltalaiselle Yeahronimo Media Ventures Inc. (YMV) –yhtiölle. (Tulip Computers Sells Commodore To Yeahronimo Media Ventures Inc. 2004.) Commodore–yhtiön virallinen kotisivu löytyy osoitteesta http://www.commodoreworld.com/.

[9] Kotitietokonemarkkinat kävivät ylikierroksilla varsinkin 1980-luvun alku- ja keskivaiheilla, jolloin oli tärkeintä vain työntää mahdollisimman paljon ja nopeasti potentiaalisia hittituotteita markkinoille voittojen kotiuttamiseksi. Varsinkin pelimusiikkien tekijänoikeuksien osalta tilanne ajautui monimutkaiseksi, sillä yleensä säveltäjät saivat kertakorvauksen tekemästään sävellyksestä, mutta eivät välttämättä luopuneet oikeuksista teoksiinsa. Myöskään tietous tekijänoikeuksista, niiden valvomisesta tai soveltamisesta ei varsin nuorella kotitietokoneteollisuuden alalla ollut ehtinyt selkiytyä ja järjestäytyä. Chris Abbott on toiminut esimerkillisesti tekijänoikeuksien puolesta SID–musiikin parissa, ja hänen työstään sekä SID–musiikin tekijänoikeuksista löytyy tietoa Internet-osoitteesta http://c64audio.valuehost.co.uk/edge/licencing.htm.

ihmisten arkipäiväiseksi tiedonlähteeksi. Hakkerit ja krakkeritkin esiintyvät nykyään ylpeinä omilla nettisivuillaan omalla nimellään - eivät enää nimimerkin suojissa.

Nostalgia ja lämpimät muistot ensimmäistä kotikonettaan kohtaan ovat saaneet ohjelmoijat tekemään kunniaa C64:n mullistavalle vaikutukselle tietokoneiden kehitykseen ja omaan elämäänsä tekemällä sivustoja, jotka pyrkivät säilyttämään niin tarkasti kuin mahdollista kaiken tiedon, joka Commodore–yhtiön tuotteita ympäröi. Muille tietokonemalleille on ohjelmoitu emulaattoreita, eli ohjelmia, jotka jäljittelevät äärimmäisen tarkasti C64:n käyttöjärjestelmää ja ominaisuuksia – jopa niin tarkasti, että alkuperäisellä C64:lla tehtyjä ohjelmia voi ladata ja käyttää sellaisenaan yhtä hyvin myös esim. PC-tietokoneilla. Commodore 64 on näin irroitettu fyysisestä olemuksestaan, muutettu digitaaliseen muotoon ja on siten kopioitavissa identtisenä tiedostona aina uuteen fyysiseen sijaintiin. Commodore 64:sta on tehty kuolematon.

Tältä pohjalta tarkasteltuna työni aihe ei ole merkityksetön tai vanhentunut. C64:lla sävelletty musiikki ja SID–mikrosirun tuottama tunnusomainen ääni toimivat edelleen innoittajina ja osatekijöinä lukuisissa uusissa sävellyksissä, projekteissa ja yhtyeissä, jotka eivät välttämättä ole missään tekemisissä tietokoneiden retrokulttuurin kanssa. Jopa ainoastaan C64:sta käyttämällä tehtyä musiikkia on levytetty aivan viime vuosinakin. Tästä esimerkkinä suomalaisartisti Teron Rikos Records –levymerkillä julkaisemat levyt First Blood (2000), Bombs Away (2001) ja Cracker’s Revenge (2003).

1.4 Työn tavoite

Työni ensisijaisena lähtökohtana on henkilökohtainen kiinnostukseni tarkastella lähemmin tapoja, joilla 80-luvun puolessa välissä pinnalla olleet C64:n suosituimpien pelimusiikkien säveltäjät ohjelmoivat musiikkinsa. Erityisen kiinnostavaa on selvittää miten heidän käyttämänsä konekieliset ohjelmointirutiinit mahdollisesti vaikuttivat itse sävellyksiin, miten paljon heidän musiikillinen tyylinsä oli riippuvainen heidän ohjelmointitaidoistaan ja millä tavoin ohjelmointitekniikat erosivat toisistaan eri säveltäjien välillä.

Toinen tarkastelun aihe on C64:n ja SID–sirun ominaisuuksien tutkiminen yleisemmällä tasolla. Nykypäivän mittapuulla teknisesti varsin vaatimattomasta syntetisaattorista on 20 vuoden aikana puristettu ulos mitä ihmeellisimpiä ääniä käyttämällä ohjelmointitekniikoita, joita laitteen suunnittelijat eivät olleet tarkoittaneet tai osanneet ennustaa laitteessa käytettävän.

Kysymyksessä on siis vertaileva tutkielma 1980-luvun Commodore 64:n pelimusiikkien säveltäjien käyttämistä instrumenttitekniikoista. Soitin on tässä tapauksessa tietokoneen sisälle rakennettu syntetisaattori ja äänen tuottaminen tapahtuu konekielisen ohjelmoinnin kautta. Tässä suhteessa soittotekniikka poikkeaa yleisistä akustisten ja sähköisten musiikki-instrumenttien käyttötavoista suurestikin, mutta periaatteiltaan vastaavan tutkimuksen voisi tehdä minkä tahansa soittimen parissa.

Tutkittavien säveltäjien valintaperusteena on heidän aikanaan saamansa arvostus tietokonemusiikin säveltäjinä ja ohjelmoijina alan lehdistössä, krakkeri-, ohjelmointi- ja harrastuspiireissä, pelejä ostavassa yleisössä, sekä pelejä tuottaneissa yrityksissä (työtilausten määrä). Sävellykset, joiden ohjelmakoodia analysoin, on valittu vaihtelevin perustein. Tärkein ominaisuus kunkin ohjelmointirutiinin kohdalla on sen edustavuus säveltäjän taidonnäytteitä yleisemmällä tasolla esiteltäessä, ts. säveltäjä on käyttänyt ko. teoksen ohjelmointirutiinia useassa sävellyksessään ja se edustaa säveltäjän kehittyneempää ohjelmointivaihetta, ei ensimmäisiä haparoivia askelia tai C64–uran viimeisimpiä, kokeilevia tai monimutkaisia ohjelmoinnin teknisiä taidonnäytteitä. Tutkielman analyysiosiossa käyn tarkemmin läpi kunkin säveltäjän ja sävellyksen valintaperusteita sekä taustoja.

Analyyseissa ei huomioida pelien käyttämiä ääniefektejä vaikka ne olisivatkin ohjelmoitu käyttäen samaa musiikkirutiinia jota pelimusiikitkin käyttävät. Oleellisinta on hahmottaa ohjelmointirutiinien yleiset toimintaperiaatteet sekä ominaisuudet nimenomaan musiikkikappaleita sävellettäessä.

1.5 Työmenetelmät

Jokainen tutkielmaan valittu ohjelmointirutiini (lukuunottamatta Chris Hülsbeckin SoundMonitor –ohjelmaa) on ensin käännetty ”ihmisten kielelle” Commodore 64:n mikroprosessorin (MOS Technologyn valmistama malli 6510) käyttämästä konekielisestä käsky- ja datajonosta. Jokaisen rutiinin kohdalla tämä purkaminen on tehty hieman eri tavalla.

Matt Grayn soittorutiini löytyy purettuna ja kommentoituna lähdekoodeineen SIDin–verkkolehdestä (Tognon, S. 2002).

Martin Galwayn Arkanoid-musiikkirutiinin analysoinnissa on käytetty pääasiallisena lähteenä Stefano Tognonin kyseistä musiikkirutiinia käsittelevää verkkoartikkelia (Tognon, S. 2003), joka sisältää myös rutiinin lähdekoodin. Suurena apuna rutiinin selittämisessä on ollut Alistair “Buz” Bowness, johon olen ollut henkilökohtaisesti yhteydessä sähköpostin välityksellä. Kootessaan Martin Galwayn säveltämistä pelimusiikeista koostuvaa, mahdollisimman autenttisen kuuloista tuplalevyä ”Project: Galway” (Bowness, A. 2003), Alistair joutui suorittamaan myös itse ohjelmointirutiinin purkamisen yhteistyössä säveltäjän kanssa - hänellä on siis korvaamatonta ensikäden tietoa musiikkirutiinin toimintaperiaatteista.

Rob Hubbardin soittorutiinin on purkanut ja selittänyt Anthony McSweeney Internet-artikkelissaan ”Rob Hubbard’s Music: Disassembled, Commented and Explained by Anthony McSweeney” (MacSweeney, A. 1993). Rutiinin lähdekoodi löytyy artikkelin yhteydestä.

Chris Hülsbeckin käyttämä editori SoundMonitor löytyy osoitteesta http://utenti.lycos.it/ice00/HVMEC/CONTROL/, johon hän on itse antanut käyttöohjeet vuonna 1986 ilmestyneessä 64’er–lehdessä (Hülsbeck, C. 1986). Ohjeiden lisäksi olen itse analysoinut musiikkirutiinia käyttämällä SoundMonitor-ohjelmaa Commodore 64:n toimintaa tarkasti jäljittelevän ccs64–emulaattorin kautta PC–koneella. Soundmonitor–ohjelman analyysissa ei siis ole käytetty lähteenä konekielistä koodia.

Analysoitujen ohjelmointirutiinien lähdekoodeja ei ole sisällytetty tutkielmaan niiden vaatiman suuren sivumäärän takia.

Musiikkirutiinien esittelyn yhteydessä tarkastelen ohjelmien toimintaperiaatteita tarkemmin: missä järjestyksessä ohjelma suorittaa

komennot, paljonko viivettä käskyt aiheuttavat, millä tavoin ohjelma suorittaa musiikillisia toimintoja (esim. vibrato, portamento, arpeggio), miten sovitukset on rakennettu koodissa, millä tavoin kappale on jaettu osiin jne. Näitä soittorutiinin ominaisuuksia tutkimalla voi selvittää mitä vaikutuksia ohjelmointitekniikalla, eli soittimen hallinnalla, on säveltäjien musiikilliseen ilmaisuun. Olennainen osa on vertailla eri säveltäjien työskentelytapoja toisiinsa, jolloin voidaan tarkastella eri ohjelmointiratkaisujen mahdollisia etuja tai esteitä monimutkaisen musiikin tuottamisessa ja saada selkeämpi ja laajempi yleiskuva SID–mikrosirun moninaisista käyttömahdollisuuksista.

Varsinaisen analyysin yhteydessä kuvailen tarkemmin käyttämiäni työmenetelmiä ja selvitän yksityiskohtaisemmin käyttämieni ohjelmien toimintaa sekä lähdeaineistojen luonnetta.

1.6 Termeistä ja työn luonteesta

Tutkielma on painottunut asiasisällöltään hyvin vahvasti musiikkiteknologian ja tietokoneohjelmoinnin piiriin kuuluviin aiheisiin, ja sisältää ajoittain erittäin yksityiskohtaista tietoa sekä SID-mikrosirun että käsiteltyjen ohjelmointirutiinien teknisistä ratkaisuista. Tämä on kuitenkin välttämätöntä riittävän ymmärryksen saavuttamiseksi Commodore 64 –tietokoneen syntetisointimahdollisuuksista sekä analysoitujen ohjelmointirutiinien toimivuudesta musiikin säveltämisessä ja sovittamisessa.

Näistä syistä johtuen teksti sisältää runsaasti erikoissanastoa ja voi välillä olla haastavaa luettavaa musiikin ammattilaisillekin. Olen kuitenkin yrittänyt sisällyttää (opinnäytetyön laajuuden rajoissa) tutkielman taustaosioon yleistajuisen katsauksen syntetisoinnin ja ohjelmoinnin periaatteista Commodore 64 –tietokoneelle sovitettuna, jotta aihepiiri avautuisi myös laajemmalle lukijakunnalle.

Analyysien ja vertailun tuottamat tulokset ja näistä tuloksista johdetut päätelmät on käsitelty luvuissa 4 – 4.5 yleisemmällä tasolla, jotta saatua uutta tietoa voitaisiin käyttää kimmokkeena uusissa sovelluksissa musiikin säveltämisessä, sovittamisessa ja soittamisessa. Analyysilukujen yhteydessä tehdyt päätelmät taas koskevat enemmän tutkittavien tietokoneohjelmien teknisiä ominaisuuksia tai yksittäisiä toimintoja.

Koin mahdottomaksi selittää erikseen kaikkia tutkielmassa esiintyviä termejä, joten jonkinasteinen pohjatietämys musiikin teoriasta sekä kokemus tietokoneiden ja syntetisaattoreiden käytöstä on oletettavaa. Joitakin keskeisimpiä tutkielmassa esiintyviä termejä on käsitelty alla olevassa listassa. *Osa listassa olevien termien merkityksistä on sovitettu tutkielmakohtaisiksi, eikä niillä siten välttämättä ole yleispätevää merkitystä*. Tällaiset poikkeukset on merkitty tähdellä (*). Kaikki tutkielmaa varten tehdyt käännökset muista kielistä ovat omaa käsialaani.

aliohjelma (eng. sub routine) = itsenäinen ohjelma laajemman ohjelman sisällä

editori (eng. editor)* = graafisen käyttöliittymän omaava musiikin luomiseen ja muokkaamiseen käytettävä tietokoneohjelma

efekti (eng. effect)* = erikoistoiminto, tehokeino

emulaattori (eng. emulator) = toisen laitteen toimintoja jäljittelevä tietokoneohjelma

ohjelmointirutiini (eng. programming routine)* = tiettyjä toimintoja rutiininomaisesti suorittava tietokoneohjelma, jossa muuttujien arvot ohjelmoidaan erillään varsinaiset toiminnot sisältävästä ohjelman käskyosasta. Tutkielmassa tällä termillä viitataan lähtökohtaisesti musiikillisia toimintoja suorittavaan ohjelmaan, eli musiikkirutiiniin

oskillaattori (eng. oscillator) = aaltomuodon ja taajuuden tuottava elektronisen instrumentin komponentti

parametri* = muuttuja, ohjelman tietyssä toiminnossa tai ominaisuudessa käytettävä muunneltava arvo

raituri (eng. tracker) = musiikkiohjelma, jossa musiikkitieto syötetään aikayksikköjä vastaaviin askeliin yksiäänisille raidoille

rekisteri = muistipaikka, joka säilyttää tietokoneen sisäisiä toimintoja ohjaavia arvoja. Tietokoneen suorittamia toimintoja voidaan ohjata syöttämällä suoraan erisuuruisia arvoja rekistereihin ja vastaavasti koneen toimintaa voidaan tutkia tarkastelemalla eri komentojen rekistereihin tallentamia tietoja

suodin (eng. filter) = rajoitin, joka estää haluttujen taajuuksien soimisen äänessä

synkronisointi (eng. synchonization) = tahdistaminen, samanaikaistaminen

syntetisaattori (eng. synthesizer) = elektroninen soitin, joka tuottaa ääniä keinotekoisesti muokkaamalla sähkövirran ominaisuuksia akustisen värähtelyn sijaan

verhokäyrä (eng. envelope) = Instrumentilla tuotettujen äänien äänenvoimakkuusmuutoksia kuvaava muuttuja. Verhokäyrä koostuu neljästä eri vaiheesta: alukeaika (attack), heikentymisaika (decay), sointivoimakkuus (sustain) ja vaimenemisaika (release)

1.7 Ääniesimerkit

Liitteenä olevan cd-levyn sisältämät Commodore 64:n pelimusiikit tai niiden katkelmat on ensin tallennettu .sid–tiedostomuodossa[10] PC–tietokoneelle High Voltage SID Collection –arkistosta (HVSC 2005). Tämän jälkeen tiedostot on muutettu .wav–muotoon[11] (käyttäen SidPlay2–emulointiohjelmaa

[10] .sid–tiedostot sisältävät C64:lla ohjelmoitua ääntä alkuperäisessä konekielisessä muodossaan. Kysymyksessä ei siis ole audiotiedosto, vaan datatiedosto, joka sisältää konekielisen musiikkirutiinin sekä kaiken sävellyksen käyttämän nuottitiedon. Tässä mielessä .sid-tiedostot muistuttavat MIDI–tietoa sillä erotuksella, että toimiakseen oikein SID-nuottidata vaatii toimiakseen nimenomaan sen musiikkirutiinin, jolla se on alunperinkin ohjelmoitu. MIDI–standardi on taas luotu nimenomaan yhtenäistämään eri laitteiden toimintatapoja, jolloin samaa nuottitietoa voidaan käyttää erilaisissa laitteissa.

[11] .wav–tiedostomuoto on Microsoftin ja IBM:n yhdessä kehittelemä standardi äänitiedostoille PC–koneissa.

sekä AudaCity–äänenkäsittelyohjelmaa) ja tallennettu cd–levylle. Kaikki muut ääniesimerkit on tuotettu käyttäen MidiBox SID –syntetisaattoria, tallennettu audiotietona PC–koneelle .wav–muotoon ja poltettu cd–levylle. Liitelevyä voi kuunnella tavallisella cd-soittimella ilman lisälaitteita.

HVSC:n arkistosta ladatut kappaleet on eristetty alkuperäisten C64:n pelien konekielisestä ohjelmakoodista käyttäen alkuperäistä Commodore 64–tietokonetta. Tämän jälkeen konekielinen sävellys on muutettu PSIDv2NG–formaattiin käyttäen SIDEdit–ohjelmaa ja siirretty digitaalisesti HVSC:n tietokantaan. HVSC:n arkistossa olevat kappaleet (.sid–tiedostot) ovat siis käytännössä identtisiä kopioita alkuperäisistä musiikkikappaleista lukuunottamatta tiedostojen alkuun liitettyä osaa, joka sisältää sävellyksen tiedot (säveltäjä, julkaisuvuosi, peliyhtiö jne.) mutta ei vaikuta millään tavalla tiedoston musiikkidataan tai audiotoistoon. Todistuksena tästä on mahdollisuus soittaa samoja .sid–tiedostoja joko alkuperäisellä C64–koneella tai esim. PC–koneella.

En paneudu tässä työssä tarkemmin kappaleiden emulointiin ja aihetta koskeviin teknisiin kysymyksiin, sillä erot alkuperäisellä C64–koneella soitetun ja SidPlay2–emulaattorilla toistetun äänen välillä ovat häviävän pienet. Sitä paitsi tutkielmani käsittelee ensisijaisesti musiikin konekielistä ohjelmointia Commodore 64 –kotitietokoneella, ei tarkasteltujen sävellysten kuulokuvaa. Kaikki ääniesimerkit ovat tarkoitettu lukijalle referenssiksi ja hahmottamista auttaviksi välineiksi.

SidPlay2–emulaattoriohjelma toistaa .sid–tiedostot audiotietona, joka on taas tallennettu AudaCity–ohjelman välityksellä .wav–formaattiin PC–koneen kovalevylle. Tällä menetelmällä alkuperäinen C64:n pelimusiikki on saatettu muotoon, jota voidaan toistaa tavallisella cd-soittimella tavalliselta cd-levyltä. Toinen vaihtoehto olisi ollut laittaa liitteeksi .sid–tiedostot sellaisenaan, mutta niiden saattaminen kuunneltavaan muotoon olisi vaatinut lukijalta huomattavasti enemmän vaivannäköä ja häirinnyt lukukokemusta. Tästä syystä päädyin mainitsemaani muutosprosessiin, vaikka se saattaakin ”värittää” ääniesimerkkejä aavistuksen verran.

Muut ääniesimerkit kuin pelimusiikit on tallennettu AudaCity–ohjelman välityksellä PC–koneelle MIDIbox SID –syntetisaattorilla soitettuna ja poltettu .wav–muodossa cd-levylle (poikkeuksena ESIMCD 10 & ESIMCD 11, jotka

on tallennettu .mp3 –formaatissa tietokoneen kovalevylle ja poltettu edelleen cd-levylle). MIDIbox SID –syntetisaattori käyttää neljää Commodore 64 –tietokoneista irroitettua SID–mikrosirua (vanhempi malli SID 6581, kts. 2.2.3, sivut 32-34) äänenmuodostuksessaan, joten syntetisaattorin tuottama ääni on täysin autenttista. Muuten laitteen käyttöliittymä on MIDI–teknologiaan perustuva.

Äänikorttina PC–koneessa käytettiin ST Audion korttimallia dsp24. Audiotieto tallennettiin tietokoneelle 24 bitin resoluutiolla käyttäen 44.1 kHz näytteenottotaajuutta.

Ääniesimerkkeihin viittaan tekstillä ESIMCD 1, ESIMCD 2, ... ja kappaleisiin tai niiden osiin tekstillä SIDCD 1, SIDCD 2, jne.

HVSC–arkisto ja tietoa sen historiasta sekä toimintaperiaatteista löytyy osoitteesta www.hvsc.c64.org - suosittelen lämpimästi tarkempaa tutustumista. AudaCity–äänenkäsittelyohjelma on ilmaisessa levityksessä oleva vapaan lähdekoodin ohjelma, joka on ladattavissa Internet–osoitteesta http://audacity.sourceforge.net/. ST Audion tuotteiden edustus löytyy Internet–osoitteesta www.staudio.de. Michael Schwendt ohjelmoi alkuperäisen SidPlay–emulaattorin, jonka pohjalta Simon White on tehnyt SidPlay2–ohjelman. SidPlay2 on vapaasti ladattavissa osoitteesta http://sidplay2.sourceforge.net/. Thorsten Klosen kehittelemän MIDIbox SID –syntetisaattorin kaikki tekniset tiedot löytyvät osoitteesta www.ucapps.de.

2 TAUSTATIETOA

Elektronisten pelien historiasta ja kehityksestä on vasta aivan viimeisten vuosien aikana tehty laajempia kirjallisia tutkimuksia (esim. Huhtamo E. & Kangas S. 2002. *Mariosofia. Elektronisten pelien kulttuuri*. Helsinki: Gaudeamus; King L. (toim.) 2002. *Game On. The History and Culture of Videogames*. London: Laurence King Publications; Kent S. 2001. *The Ultimate History of Video Games*. Roseville, CA: Prima Publishing) mutta peleissä käytetty musiikki on jäänyt mielestäni tyystin vaille tutkimuksellista huomiota. Monet tutkimukset tarkastelevat pelejä ja tietokonekulttuuria sosiologisena ilmiönä, mutta tekniset ja erityisesti taiteelliset aspektit ja näkemykset tietokonepeleistä on julkaistu lähinnä populääriteoksina tai kuvitettuina historian katsauksina – ei tieteellisinä tutkimuksina.

Pelkästään tietokonepelien musiikin historiaan keskittyvää teosta ei tietääkseni ole julkaistu, ja Suomen korkeakoulujen opinnäytetöiden tietokannoista ei löydy yhtäkään työtä, joka käsittelisi ohjelmoinnin ja musiikin välistä suhdetta tietokonepeleissä.

Kattavia yleisteoksia ja ohjekirjoja musiikin tekemiseen tietokoneella on yleisesti saatavilla (myös Commodoreen soveltuvia teoksia) mutta niissäkään ei analysoida jo sävellettyä tietokonemusiikkia tai käydä läpi peleissä käytettyjä ohjelmointirutiineja, vaan keskitytään antamaan ohjeita ja esimerkkejä oman musiikkiohjelman tai tietokonemusiikin ohjelmoimiseen. Tietysti on ymmärrettävää, että pelien ja musiikkien ohjelmoijien koodaustekniikat olivat (ja ovat tietysti edelleen) tarkoin varjeltuja salaisuuksia[12], joiden selvittämisestä krakkerit ja muut ohjelmoinnin harrastajat olivat kiinnostuneita.

Tämän takia C64:n ohjelmointikikat ja –rutiinit jäivät hyvin pienen ja teknisesti osaavan harrastajapiirin tietoon, eikä tällaista tietoa voinut julkaista missään alan foorumissa ilman tekijän lupaa - mitä kukaan tuskin olisi myöntänytkään. Nykyään pelitalojen ohjelmointikoodeja myydäänkin

[12] Ohjelmoijat harhauttivat koodinpurkajia asettamalla ohjelmointirutiineihin käskyjä, jotka viittasivat olemattomiin koodinpätkiin tai sijoittivat komentoja muiden käskyjen väliin niin, ettei ohjelma koskaan edes käyttänyt niitä (Tognon, S. 2003).

lisensseillä kilpaileville firmoille ja niiden suunnitteluun ja toteutukseen osallistuu lukuisia eri alueisiin erikoistuneita koodaajia.

Internet on kuitenkin osoittautunut tämän kaltaisen, tekijänoikeuksien kannalta harmaalla alueella liikkuvan tiedon ehtymättömäksi lähteeksi; tietoverkon ihanteisiin kuului jo luomisvaiheessa ajatus vapaasta ja nopeasta tiedonvälityksestä ja käytännössä rajoittamattomista tietovarannoista, jotka olisivat kaikkien käytössä, ja joita kukin käyttäjä voisi itse laajentaa ja muokata. Niinpä myös kaikki tarvitsemani analyysiaineisto tätä tutkielmaa varten löytyi Commodore 64 –tietokonetta käsitteleviltä Internet-sivustoilta[13].

Toivon ja odotan kovasti elektronisten pelien musiikkia käsittelevän yleisteoksen ilmestymistä, joka keräisi yhteen esimerkkejä monista eri tietokonemerkeille ja -malleille ohjelmoiduista pelimusiikeista vuosien varrelta, sekä vertailisi eri tietokoneiden ominaisuuksia äänen tuottamisessa ja muokkaamisessa. Sitä odotellessa tämä tutkielma pyrkii osaltaan kokoamaan yhteen arvokasta mutta hyvin hajautunutta tietoa musiikin ohjelmoimisesta Commodore 64 –kotitietokoneella.

2.1 SID:n historiaa

MOS Technologyn (Commodore Business Machines –yhtiön tytäryhtiö) suunnittelemia mikrosiruja oli käytetty Commodoren vuonna 1980 julkistamassa VIC-20–tietokoneessa ja tammikuussa 1981 Albert J. Charpentier, MOS-yhtiön piirisuunnitteluosaston silloinen johtaja, ehdottikin emoyhtiölle uusien korkealaatuisten ääni- ja videosirujen tuottamista (Wallich, P. 1985). Commodoren teknisen osaston johtaja Charles Winterble myönsi luvan sirujen suunnitteluun, jotka tultaisiin sijoittamaan Commodoren uuteen pelikonsoliin. Insinöörit saivatkin työskennellä varsin itsenäisesti uuden projektin parissa, kunnes mikrosirut valmistuivat marraskuun puolivälissä 1981.

Aloittaessaan SID 6581 –äänisirun suunnittelun vuoden 1981 alussa MOS Technology –yhtiössä, Robert Yannes ei tiennyt mitään VLSI-sirujen[14]

[13] Kts. osio 1.5, sivut 14-15.
[14] VLSI = very-large-scale integration. Tässä tapauksessa on kyse mikrosirusta, joka sisältää vähintään tuhat piiriä, jotka toimivat ”portteina” virran kulkiessa sirussa. Pienelle alueelle on siis yhdistetty suuri määrä komponentteja – siitä nimitys VLSI.

suunnittelusta (Varga, A. 1996). Hänen aikaisemmat työnsä Commodorelle[15] ja hänen kokemuksensa äänisynteesistä kuitenkin puolsivat hänen valitsemistaan uusien ääniominaisuuksien suunnittelijaksi Commodore ja MOS–yhtiöiden tuotteisiin.

Yannes oli tunnettu Commodorella kustannustehokkuudestaan (ja Commodore oli tunnettu pihinä yhtiönä) ja noudatti audiosirun suunnittelussa samaa periaatetta - hän ei käyttänyt yhtään enempää materiaalia siruun kuin oli välttämätöntä saavuttaakseen vastaavat ominaisuudet kuin 80-luvun musiikkiteollisuuden valmistamissa syntetisaattoreissa oli käytössä (Wallich, P. 1985).

Apunaan hänellä oli kaksi luonnosten tekijää, yksi CAD[16]–suunnitteluohjelman käyttäjä sekä yhtiön oma, sillä hetkellä puoliteholla käyvä mikrosirujen tuotantolinja. Tämä mahdollisti mikrosirun virtapiirien tehokkaan ja luotettavan testauksen eristämällä ne itse sirusta. Silti aika oli suurempi rajoite suunnittelijoille kuin taloudelliset resurssit. Esimerkiksi SID–sirun oskillaattorit olisi voitu monistaa ohjelmallisesti yhdestä moduulista, jolloin rakennustilaa olisi jäänyt enemmän muillekin audio-ominaisuuksille, mutta ajanpuutteen vuoksi Yannes päätti rakentaa kolme erillistä oskillaattoria (Wallich, P. 1985). Samoin mikrosirun tekniikassa oli vajavaisuuksia (varsinkin suotimen toiminnassa), joita Yannesilla ei ollut mahdollisuutta paikata ennen C64:n siirtymistä tuotantoon.

Vuonna 1985 Commodore julkaisi tehokkaamman version C64–tietokoneestaan, joka sisälsi 128 kilotavua muistia ja ristittiin sen mukaan Commodore 128:ksi. Uusi malli sisälsi myös teknisesti uudelleen suunnitellun äänisirun (SID 8580), joka sisälsi samat ääniominaisuudet kuin edeltäjänsä, mutta toimi teknisten parannusten takia tasaisemmin. Robert Yannesilla ei tosin enää ollut mitään tekemistä uuden sirun kanssa, sillä hän oli jo alkuvuodesta 1983 lähtenyt Commodoren palveluksesta ja perustanut Peripheral Visions –yhtiön yhdessä Al Charpentierin, Charles Winterblen, David Ziembeickin ja Bruce Crocketten kanssa. Yhtiö vaihtui pian Ensoniq–nimiseksi ja 1990–luvun lopulla siitä tuli osa Creative Labs –yhtiötä (Commodore Computer History 2005).

---

[15] Yannes suunnitteli yksin Commodore 64:sta edeltäneen VIC-20 –tietokoneen kotonaan ylimääräiselle piirilevyn mallipalaselle (Wallich, P. 1985).

[16] CAD = Computer Aided Design (eng.), tietokoneavusteinen suunnittelu.

SID–sirun suurimmat eroavaisuudet sen aikaisiin kilpailijoihin tietokoneteollisuudessa olivat mahdollisuus ohjelmoida jokaiselle äänelle oma verhokäyränsä, sekä varsin tarkka taajuusohjaus (Wallich, P. 1985). Yannes oli sisällyttänyt siruun myös valmiin taulukon, joka sisälsi tasavireisen järjestelmän nuottien taajuudet valmiiksi ohjelmoituna – Commodoren tekninen johtaja Charles Winterble tosin poisti taulukon, sillä se kulutti liikaa silikonia.

Marraskuun lopulla 1981 audio- ja videomikrosirujen ollessa valmiit Commodore–yhtiön johto piti kokouksen, jossa pääjohtaja Jack Tramiel päättikin sijoittaa uudet sirut tammikuun toisella viikolla julkistettavaan uuteen 8-bittiseen kotitietokoneeseen pelikonsolin sijaan. Tietokone tosin täytyi vielä suunnitella ja rakentaa.

Kahdessa päivässä insinöörit loivat paperille uuden C64 –koneen perusrakenteen, sijoittivat siihen uudet mikrosirut ja lainasivat surutta edellisen konemallinsa, VIC-20:n, osia sekä ohjelmakoodia saadakseen tietokoneen esittelykuntoon. Ja niin, juuri ennen uutta vuotta 1982 viisi toimivaa koekappaletta olivat valmiina esittelykäyttöön tammikuun CES–messuille (Wallich, P. 1985).

C64:n alhainen hinta (595 $/kpl tuotantokustannusten ollessa 135 $/kpl) ja huippuluokan ominaisuudet saivat kilpailijat hämmennyksiin ja kuluttajat hymyilemään. Valtava kysyntä pakotti Commodoren toimimaan nopeasti tuotannon käynnistämiseksi eikä suunnittelijoille siten jäänyt enää aikaa muuttaa prototyypin kaikkia vikoja. Myös Commodore 64 –kotitietokoneen ohjekirjaan painettiin virheellistä tietoa laitteen ominaisuuksista, mikä aiheutti laitteen käyttäjille ylimääräistä päänvaivaa (Wallich, P. 1985).

2.2 SID -mikrosirun ääniominaisuudet

Tämän luvun lähdeaineistona on käytetty SID–mikrosirun alkuperäistä patenttihakemusta (Yannes, R. 1983) *sekä Robert Yannesin haastattelua* (Varga, A. 1996). *Aaltomuotojen matemaattiset kaavat ja äänisignaalien fysikaalisia ominaisuuksia koskeva tieto on lähtöisin Stephen L. Juddin verkkoartikkelista (Judd, S. 1996). Muut lähteet on merkitty vitteillä asiayhteyksiinsä.*

FEATURES

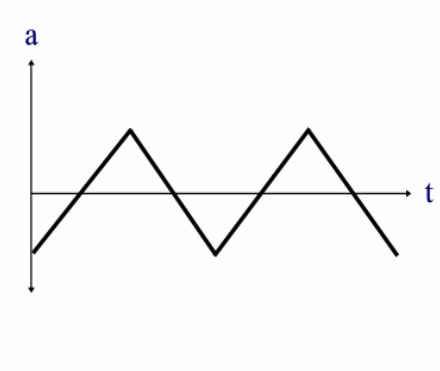
o **3 TONE OSCILLATORS**
Range: 0-4 kHz
o **4 WAVEFORMS PER OSCILLATOR**
Triangle, Sawtooth,
Variable Pulse, Noise
o **3 AMPLITUDE MODULATORS**
Range: 48 dB
o **3 ENVELOPE GENERATORS**
Exponential response
Attack Rate: 2 ms-8 s
Decay Rate: 6 ms-24 s
Sustain Level: 0-peak volume
Release Rate: 6 ms-24 s
o **OSCILLATOR SYNCHRONIZATION**
o **RING MODULATION**
o **PROGRAMMABLE FILTER**
Cutoff range: 30 Hz-12 kHz
12 dB/octave Rolloff
Low pass, Bandpass,
High pass, Notch outputs
Variable Resonance
o **MASTER VOLUME CONTROL**
o **2 A/D POT INTERFACES**
o **RANDOM NUMBER/MODULATION GENERATOR**
o **EXTERNAL AUDIO INPUT**

KUVIO 1. *SID–äänisirun ominaisuudet (Commodore 64 Programmer's Reference Guide 1983)*

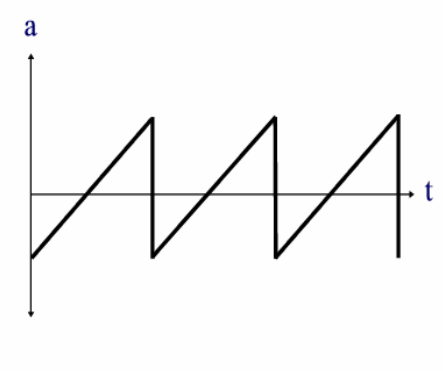
2.2.1 Aaltomuodot

Sound Interface Device (SID) sisältää kolme digitaalista oskillaattoria, jotka määrittelevät äänen taajuuden ja aaltomuodon, eli harmonisten yläsävelten voimakkuussuhteet äänessä. Koska äänilähteitä on käytössä vain kolme kappaletta, voi SID tuottaa samanaikaisesti vain kolmea eri taajuutta/aaltomuotoa. Käytössä olevia aaltomuotoja on neljä erilaista: kolmio, nouseva sahalaita, muunneltava pulssi ja kohina. Seuraavat kuviot esittävät SID:n tuottaman äänen ideaalisia[17] perusaaltomuotoja.

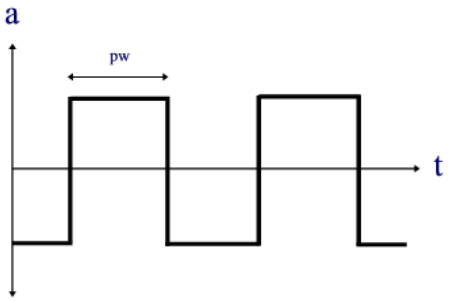
[17] Kuviot eivät vastaa oskillaattoreiden tuottamia todellisia signaaleja. Fysikaalisesti ei ole mahdollista tuottaa aaltoa, jonka amplitudi nousisi tai laskisi tasan 90 asteen kulmassa (kuten pulssiaaallon ideaalissa), vaan asteluku on aina pienempi. Samoin signaalin asettuessa amplitudin muutoksen jälkeen tasaiselle jänniteasteelle syntyy taitekohtaan aina voimakkuudeltaan vaihteleva amplitudipiikki.



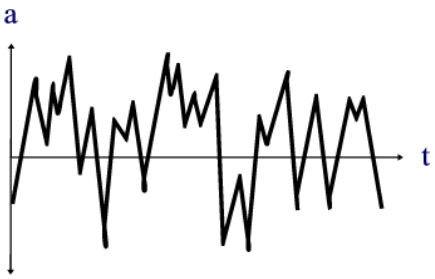
KUVIO 2. Kolmioaalto (ESIMCD 1)



KUVIO 3. Sahalaita-aalto (ESIMCD 2)



KUVIO 4. Pulssiaalto (ESIMCD 3)



KUVIO 5. Kohina-aalto (ESIMCD 4)

Kolmioaalto on lähimpänä yksinkertaisimman fysikaalisen aaltomuodon, eli siniaallon, hahmoa - kolmioaalto sisältääkin vähiten ylä-ääniä SID:n tuottamista aaltomuodoista. Kaikki aaltomuodot koostuvat yksittäisistä eri taajuuksilla olevista siniaalloista, joten matemaattisesti kolmioaalto voidaan kuvata seuraavasti:

sin(x) - sin(3x)/9 + sin(5x)/25 - sin(7x)/49 + ...

Kolmioaalto sisältää siis vain parittomia perusäänen kerrannaisaaltoja, joiden voimakkuus heikkenee nopeasti ylä-äänen taajuuden kasvaessa. Johtuen Yannesin ohjelmointiratkaisusta SID–mikrosirussa kolmioaallon resoluutio on yksitoista bittiä, eli puolet sahalaita-aallon 12-bittisestä resoluutiosta (amplitudin ja taajuuden osalta ominaisuudet ovat kuitenkin samat).

Sahalaita-aalto sisältää sekä parittomia että parillisia perusäänen kerrannaisaaltoja. Ylä-äänten voimakkuudet myös heikkenevät hitaammin kuin kolmioaallossa. Seuraava kaava kuvaa sahalaita-aaltoa matemaattisesti:

sin(x) + sin(2x)/2 + sin(3x)/3 + sin(4x)/4 + ...

Pulssiaalto koostuu vain kahdesta vaiheesta; aalto on joko tasaisesti positiivinen tai vastaavalla arvolla tasaisesti negatiivinen. Pulssiaallon leveyttä (muuttuja pw kuviossa 4) säätelemällä pystytään vaikuttamaan äänen sisältämien ylä-äänten määrään ja siten muuttamaan äänen väriä (ESIMCD 5). Leveyttä voi muuttaa lineaarisesti 12-bittisellä luvulla alla olevan kaavan[18] mukaisesti:

PULSSIN LEVEYS = (REKISTERINARVO / 40,95)%

Rekisteri voi siis saada arvokseen luvun väliltä 0 – 4095 ja pulssin leveys annetaan prosenttina koko aallon leveydestä. Antamalla rekisterille arvon 0 oskillaattori tuottaa pienimmän mahdollisen tasavirtajännitteen ulostulostaan – samoin tapahtuu rekisterin arvolla 4095 mutta suurimmalla oskillaattorin jännitteellä. Kanttiaallon voi muodostaa käyttämällä rekisterin arvoa 2048, jolloin pulssin leveys on 50,0 % yhden desimaalin tarkkuudella. Kanttiaallon matemaattinen vastine on:

sin(x) + sin(3x)/3 + sin(5x)/5 + ...

Kohina-aaltoa ei voi vastaavasti matemaattisesti kuvailla, sillä sen ylä-äänten taajuudet vaihtuvat sattumanvaraisesti. SID:n tuottama ääni on niin kutsuttua ”sinistä kohinaa” (Judd, S. 1996), missä valitun perusäänen yläpuolelle syntyy sattumanvaraisesti mitä tahansa taajuuksia, joista kullakin on yhtä suuri todennäköisyys esiintyä[19]. Ylä-äänet eivät siis välttämättä ole kokonaislukukerrannaisia perusäänelle. Ylemmillä taajuuksilla sävelten välisen intervallin sisään mahtuu huomattavasti enemmän kokolukutaajuuksia

[18] Patenttidokumentissa annetun kaavan mukaan jakajana olisi 49,95 - tällöin rekisterin arvolla 4095 pulssin leveydeksi tulisi kuitenkin 81,981981… %. Todennäköisesti kyseessä on kirjoitusvirhe, joka on syntynyt siirrettäessä patenttidokumenttia sähköiseen muotoon. Jotta saataisiin 100 %:n pulssin leveys, tulee jakajan siis olla 40,95 – ei 49,95.

[19] Todellisuudessa SID:n kohina ei koostu sattumanvaraisesti arvotuista taajuuksista, vaan kohina-aaltomuodon valinta tuottaa hyvin pitkän numerosarjan, joka toistaa itseään säännöllisin väliajoin (Mäkelä, M. & Alstrup, A. 2005). Sarjan uusiutumista on kuitenkin käytännössä mahdotonta erottaa kuulokuvassa, joten ratkaisu on musiikkiohjelmoinnissa täysin toimiva.

kuin alemmilla taajuuksilla. Tästä johtuen sinisen kohinan äänienergia on tasaisemmin jakaantunut intervallien sisällä kuin valkoisessa kohinassa, missä kaikki taajuudet ovat läsnä.

2.2.2 Signaalin eteneminen

SID–mikrosirun sisältämät oskillaattorit saavat toimintaohjeensa mikroprosessorin (MOS Technology 6510) lähettämän digitaalisen tiedon perusteella. Saatuaan tiedon valitusta äänimuodosta ja taajuudesta oskillaattori lähettää digitaalisen tiedon aaltomuodon valitsijaan ja edelleen muuntimeen, joka tuottaa valintojen mukaisen analogisen sähkövirran[20]. Sähkövirta ohjautuu taas digitaaliseen verhokäyrägeneraattoriin, joka muuttaa signaalin amplitudia annettujen arvojen mukaisesti. Tämän jälkeen analoginen signaali voidaan ajaa joko suoraan tai analogisen suotimen kautta yleistä äänenvoimakkuutta säätelevään vahvistimeen ja tietokoneen ääniulostuloon.

Oskillaattorit toimivat ns. ”phase-accumulation”–periaatteella[21], missä digitaalinen oskillaattori laskee yhden kokonaisen aallonpituuden muodon kerrallaan ja syöttää sen ulostuloonsa; näin aaltomuoto toistuu samanlaisena ja samassa vaiheessa. Tietokoneen laskentateho määrittelee kuinka korkeita ääniä ja monimutkaisia aaltomuotoja digitaalinen oskillaattori pystyy tuottamaan. Commodore 64:n äänenkorkeuksia rajoittaa kuitenkin SID:n oskillaattorien 16–bittinen taajuuksille varattu rekisteri, jonka arvo voi vaihdella välillä 0 – 65536. SID:n tuottaman äänen korkeus määräytyy seuraavan kaavan mukaisesti:

TAAJUUS = (REKISTERIN ARVO x KELLOTAAJUUS) / 16 777 216

[20] Aaltomuodot ovat siis pelkkiä nollia ja ykkösiä lähtiessään oskillaattorin ulostulosta. Wavetable–synteesissä (käytössä useissa PC–äänikorteissa) aaltomuoto taas valitaan valmiiksi ohjelmoiduista digitaalisista signaaleista, jotka sen jälkeen syötetään edelleen A/D–muuntimeen. Yannesilla ei kuitenkaan ollut mahdollisuutta sijoittaa tällaisia valmiita aaltomuotoja sirulle tilanpuutteen vuoksi. Niinpä oskillaattorit joutuvat laskemaan aaltomuodot reaaliajassa mikroprosessorilta tulevan tiedon mukaisesti. Tästä syystä SID:n tuottamat aaltomuodot ovat varsin yksinkertaisia verrattuna esim. akustisten soittimien aaltomuotoihin, joiden reaaliaikainen mallintaminen on käytännössä mahdotonta Commodore 64:n laskentateholla.

[21] Phase (eng.), vaihe; accumulation (eng.) kasautuminen.

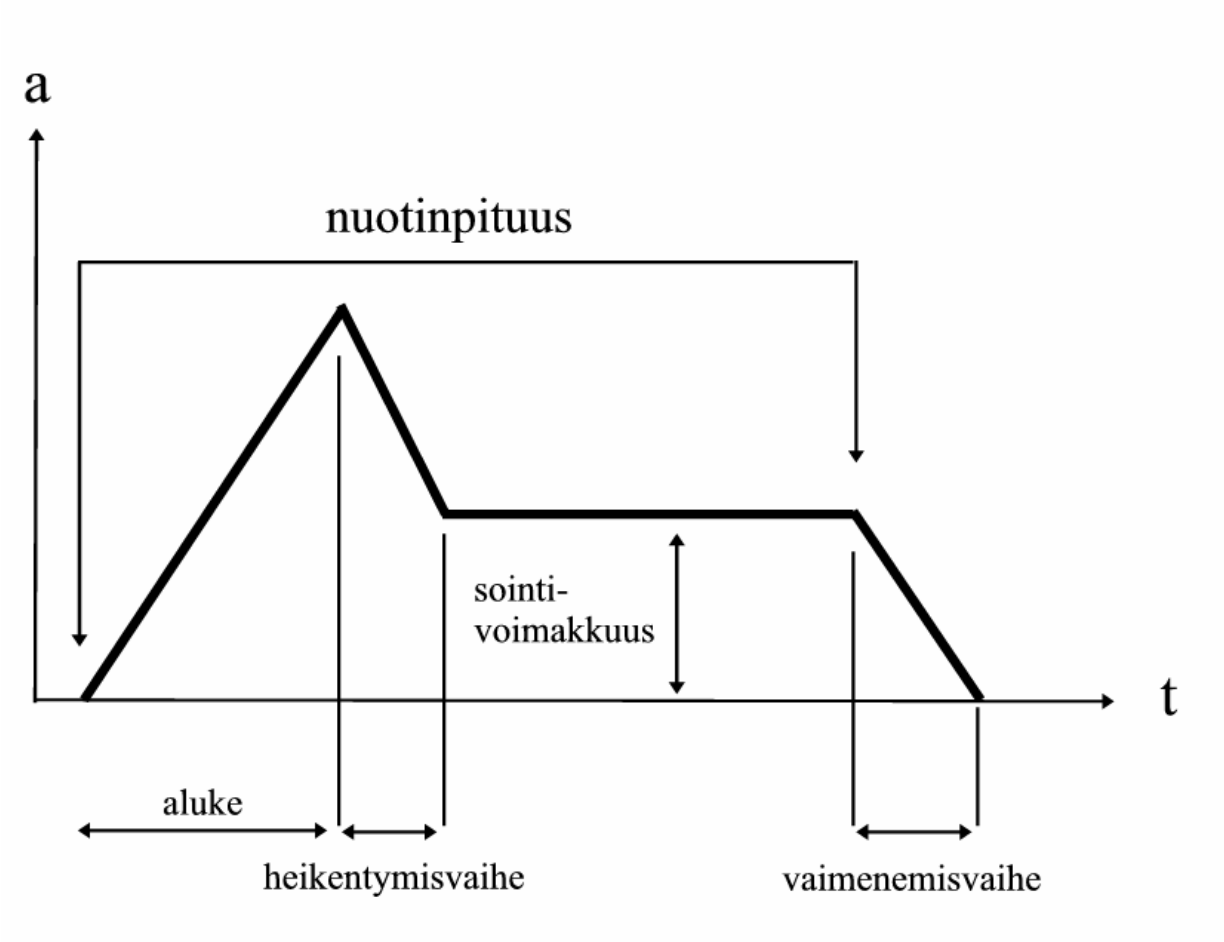
Rekisterin arvo on siis muistipaikkaan ohjelmoitu arvo väliltä 0 – 65 536 ja kellotaajuus on mikroprosessorin laskentanopeus. Kellotaajuuden 6510 -mikroprosessorissa määrittää ~1 MHz taajuudella sykkivä kvartsikide (1 022 730 Hz NTSC–järjestelmässä ja 985 250 Hz PAL–järjestelmässä[22], (Judd, S. 1996). Suorittamalla edellä mainitun laskutoimituksen rekisterin pienimmällä ja suurimmalla arvolla saadaan SID:n äänialueeksi 0 – 3995 Hz (NTSC) ja 3848,6 Hz (PAL) yhden desimaalin tarkkuudella. Tarkempia tietoja aaltomuotojen digitaalisesta muodostumisesta C64–koneella löytyy Robert Yannesin haastattelusta (Varga, A. 1996).

Oskillaattorin jälkeen tuleva aaltomuodon valitsin ei pysty käsittelemään kuin yhtä taajuutta yhdistettynä yhteen aaltomuotoon kerrallaan, mistä johtuen aaltomuotoja ei voi yhdistää toisiinsa monimutkaisemmiksi aaltomuodoiksi[23] - toinen aaltomuotojen moninaisuutta rajoittava tekijä on 6510 -mikroprosessorin vaatimaton laskentateho (kts. alaviite 20, sivu 27).

Aaltomuodon valitsijan jälkeen signaali muuttuu analogiseksi D/A–muuntimessa ja ohjautuu verhokäyrän muokkaimeen. Verhokäyrä (kuvio 6) muokataan digitaalisesti haluttuun muotoon, mutta itse suodin ja sen käsittelemä signaali ovat analogisia. Tässä mielessä SID on itse asiassa substraktiiviseen synteesiin perustuva analogisyntetisaattori. Yhdistelemällä analogisia ja digitaalisia osia Yannes pystyi saavuttamaan nykymittapuulla hyvin vaatimattoman prosessoritehon ja muistin rajoissa erinomaisesti toimivan digitaalisesti ohjailtavan syntetisaattorin.

[22] Erot johtuvat vaihtosähkövirtojen erisuuruisesta taajuudesta ja jännitteestä NTSC ja PAL-järjestelmissä. PAL–standardi on käytössä Euroopassa ja NTSC mm. Yhdysvalloissa. Tämän tutkielman kaikki ääniesimerkit pohjautuvat PAL –järjestelmän taajuuksiin.

[23] Yannes väittää haastattelussaan aaltomuotojen yhdistämisen tuottavan loogisen AND–operaation aaltomuotojen arvoille (kuten myös Jürgen Oppermann [Oppermann, J. 2004]), mutta mitä todellisuudessa koneen sisällä tapahtuu aaltomuotoja yhdistettäessä on edelleen varmistamatta. Yhdistelmämuodoilla ei kuitenkaan ole varsinaista musikaalista käyttöä, sillä aaltomuotoja yhdistelemällä tuotetut äänet ovat epämääräisiä ja vaimeita (ominaisuutta on paranneltu uudemmassa 8580-sirumallissa; kts. 2.2.3, s. 33), ja tällainen komento saattaa lukita tietokoneen satunnaislukugeneraattorin täyttämällä sen pelkillä nolla-arvoilla.



KUVIO 6. Verhokäyrän vaiheet

Seuraava taulukko sisältää kaikki aika-arvot mitä C64:n verhokäyrägeneraattori voi käyttää.

TAULUKKO 1. Verhokäyrän muutosnopeudet (Windisch, S. 1983)

ARVO		**ALUKEAIKA**	**HEIKENTYMISAIKA**	**VAIMENEMISAIKA**
Desim.	Heksadesim.			
0	0	2 ms	6 ms	6 ms
1	1	8 ms	24 ms	24 ms
2	2	16 ms	48 ms	48 ms
3	3	24 ms	72 ms	72 ms
4	4	38 ms	114 ms	114 ms
5	5	56 ms	168 ms	168 ms
6	6	68 ms	204 ms	204 ms
7	7	80 ms	240 ms	240 ms
8	8	100 ms	300 ms	300 ms
9	9	250 ms	750 ms	750 ms

TAULUKKO 1. (jatkuu)

10	A	500 ms	1,5 s	1,5 s
11	B	800 ms	2,4 s	2,4 s
12	C	1 s	3 s	3 s
13	D	3 s	9 s	9 s
14	E	5 s	15 s	15 s
15	F	8 s	24 s	24 s

Äänenvoimakkuuden vaihteluväli on 48 dB, joka on jaettavissa lineaarisesti neljällä bitillä kuuteentoista eri voimakkuuteen (sointivoimakkuus), jolla ääni soi kunnes se vapautetaan vaimenemisvaiheeseensa nuotinpituuden loppuessa.

Verhokäyrägeneraattori itsessään on 8-bittinen laskuri, joka käynnistettäessä nousee 0:sta 255:teen (0 – 48 dB) alukeajaksi annetulla nopeudella, laskee valittuun soimisvoimakkuuteen annetulla heikkenemisajalla ja lopuksi laskee takaisin nollaan annetulla vaimenemisajalla. Jos äänenvoimakkuutta laskee kesken nuotin soimisajan, verhokäyrän laskuri aloittaa vaimentamisen vasta uudesta voimakkuustasosta – äänenvoimakkuutta nostettaessa laskuri kuitenkin aloittaa vaimentamisen ensin annetusta äänenvoimakkuudesta.

Yllä olevassa taulukossa annetut muutosajat ovat Yannesin itsensä subjektiivisesti kuulokuvansa pohjalta määrittelemiä ja kiinteästi sirulle asentamia - ne eivät ole ohjelmallisesti muutettavissa. Verhokäyrägeneraattorin lähettämä data ohjautuu D/A –muuntimeen, missä se moduloituu oskillaattorista lähteneeseen analogiseksi muutettuun signaaliin. Lisätietoja SID –mikrosirun verhokäyrän toiminnasta löytyy Robert Yannesin haastattelusta (Varga, A. 1996).

Seuraavaksi signaali saapuu analogiseen suotimeen, joka tuottaa yhtäaikaisesti kolmea erilaista suodinvaihtoehtoa: alipäästö-, ylipäästö- ja kaistapäästösuotimen. Näistä kolmesta voi kytkeä päälle minkä tahansa yhdistelmän. Yli- ja alipäästösuotimissa leikkaustaajuuden kohdalta alkaen äänenvoimakkuus laskee 12 dB oktaavia kohden ja kaistapäästösuotimessa 6 dB/oktaavi (ESIMCD 6, ESIMCD 7, ESIMCD 8). Suodin ei sisällä ns. ”key follow” –toimintoa, eli leikkaustaajuus ei ole suhteellinen oskillaattorin

tuottamaan taajuuteen nähden, vaan itsenäinen, absoluuttinen leikkausarvo. SID:n patenttihakemuksen tietojen mukaan tarkka leikkaustaajuus määräytyy seuraavan kaavan mukaan:

TAAJUUS = (REKISTERIN ARVO * 5.8) + 30 Hz

Rekisterin arvoksi pystyy syöttämään 11-bittisen luvun, joten ylläolevan kaavan mukaan alin suotimen taajuus on 30 Hz ja ylin vastaavasti 11 902,6 Hz. Leikkaustaajuutta voi halutessaan korostaa (resonanssi) 4-bittisellä arvolla (kuusitoista voimakkuusvaihtoehtoa). SID sisältää vain yhden suotimen, joten yhdellä kertaa ei voi käyttää kuin yhdenlaisia suotimen asetuksia - suotimen voi kuitenkin kytkeä vaikuttamaan samanaikaisesti vaikka kaikkiin kolmeen signaaliin tai mihin tahansa näiden yhdistelmistä.

Näiden ominaisuuksien lisäksi mitkä tahansa kaksi oskillaattoria pystyvät toimimaan synkronoidusti - moduloitu oskillaattori aloittaa aaltomuotonsa alusta samaan aikaan moduloivan oskillaattorin kanssa riippumatta moduloitavan äänen vaiheesta. Tällä tavoin toinen oskillaatori voidaan pakottaa soimaan samassa vaiheessa toisen oskillaattorin kanssa vaikka taajuudet olisivat erisuuruiset.

”Ring modulation” –efekti (eng., kehämodulointi) tuottaa erotuksen ja summan kahden oskillaattorin taajuuksista. Moduloitavan oskillaattorin aaltomuodon on oltava kolmio ja moduloivan oskillaattorin taajuuden suurempi kuin nolla, jotta efekti voidaan kytkeä päälle. Millään muilla moduloivan oskillaattorin asetuksilla ei ole vaikutusta efektiin.

Jokainen SID:n aaltomuodon sisältää jo itsessään runsaasti ylä-ääniä, joten kehämodulointiefekti tuottaa erittäin laajan yläsävelspektrin ääniaaltojen yläsäveltenkin luodessa uusia taajuuksia summautumalla ja vähentymällä. Tällä tavoin voidaan tuottaa aaltomuotoja, jotka sisältävät enharmonisia yläsäveliä (eli muita taajuuksia kuin perusäänen kokolukukerrannaisia) ja ääniä, joiden perusäänten taajuudet ylittävät rekisterin kautta annettavien taajuuksien ylärajan ~4 KHz.

Viimeisessä vaiheessa kaikki kolme ääntä ohjautuvat 4-bittiseen vahvistimeen (16 voimakkuustasoa), joka säätelee yleistä äänenvoimakkuutta. Myös tätä vahvistinta ohjataan siis digitaalisella käskyllä,

joka muuntuu vahvistimessa analogiseksi sähkövirraksi ja moduloi sisääntulevia kolmea äänisignaalia.

Commodore 64 –tietokoneeseen on mahdollista liittää myös ulkoinen äänilähde (esimerkiksi toinen C64), jonka lähettämä audiosignaali voidaan ohjata joko suoraan yleisvahvistimeen tai prosessoida se suotimen kautta. SID voi myös vastaanottaa analogista signaalia ulkoisista potentiometriohjaimista. Nämä kaksi A/D POT INTERFACE –liitäntää pystyvät muuttamaan analogisen signaalin digitaaliseksi, joten tällaista ohjaussignaalia voidaan käyttää pelisovelluksissa tai ohjaamaan SID:n äänenmuokkausominaisuuksia kuten syntetisaattoreissa.

Mikroprosessorin on mahdollista lukea suoraan kolmannen oskillaattorin tuottaman aaltomuodon kahdeksaa ylintä bittiä. Tällä tavoin oskillaattorin tuottaman äänen aaltomuotoa voidaan käyttää toisten äänien moduloinnissa. Esimerkiksi hyvin matalataajuuksinen ja hiljainen kolmioaalto voidaan ohjata moduloimaan toisen oskillaattorin äänenkorkeutta, jolloin saadaan aikaiseksi vibrato. Myös muissa ohjelmasovelluksissa kolmannen oskillaattorin aaltomuotoa voi käyttää hyväkseen; esimerkiksi satunnaislukugeneraattori (eng. random number generator) saadaan aikaiseksi niin, että mikroprosessori käyttää lukuarvoina kohina-aaltomuodon tuottamia ylä-ääniä. Tällaisessa tapauksessa oskillaattorin tuottamaa dataa ei siis käytetäkään musiikin ohjelmointiin, vaan käsitellään sellaisenaan satunnaisena numerotietona toisentyyppisissä sovelluksissa, esimerkiksi tietokonepeleissä.

Yleensä kolmannen oskillaattorin tuottama modulointiaalto ei ole musiikillisesti kaunista kuultavaa, joten musiikkisovelluksissa tätä modulointitapaa käytettäessä kolmas kanava säädetään yleensä äänettömälle. Tämä äänenmuokkaustapa kuitenkin poistaa yhden äänen käytöstä, minkä takia käytännössä kaikki efektit pelien musiikeissa tuotettiin erillisillä ohjelmointikomennoilla, kuten kolmannen luvun analyysit osoittavat.

2.2.3 6581 ja 8580

Tämän luvun lähteenä on käytetty C=Hacking–verkkolehden artikkelia ”The C64 Digi” (Harbron, R., Harsfalvi, L. & Judd, S. 2001) *ja Wikipedia–*

verkkotietosanakirjan SID–mikrosiruja käsittelevää osiota (MOS Technology SID 2005).

SID 6581 –mikrosiru sisälsi paljon vajavaisuuksia Commodore 64:n siirtyessä vuonna 1982 esittelyversiosta tuotantoon. Robert Yannes tiesi sirussa olevan vielä heikkouksia ja osia, jotka eivät toimineet niin kuin hän oli suunnitellut, mutta hänellä ei ollut mahdollisuutta paikata puutteita erittäin kiireisen tuotantoaikataulun takia.

Suurimpana ongelmana säveltäjät sekä muut Commodoren käyttäjät mainitsevat analogisen suotimen epätarkkuuden; suotimen vaimennusteho vaihtelee äänenvoimakkuuden mukaan, ja johtuen laadultaan epätasaisista komponenteista leikkaustaajuuden määrittävä kaava (kts. s. 31) ei pidä ollenkaan paikkaansa, vaan vaihtelee sirukohtaisesti.

Toisena laadullisena ongelmana mikrosirua vaivaa suuri taustakohinan määrä (ESIMCD 9), joka johtuu verhokäyrän D/A–muuntimien tuottamasta audioulostuloon kulkevasta tasasähkövirrasta. Myös yleisvoimakkuuden säätimessä on pieni ylimääräinen tasajännite. Lisäksi verhokäyrän digitaalisissa ohjaimissa on toimintavirheitä[24]. Näistä syistä kahta täysin samankuuloista SID 6581 –mikrosirua on tuskin löydettävissä.

Vuonna 1985 ilmestyneessä SID 8580 –sirussa suurin osa yllä mainituista vioista on korjattu. Uudempien sirujen ominaisuudet olivat keskenään hyvin samanlaiset johtuen ennen kaikkea suotimeen tehdyistä parannuksista. Samoin audioulostulossa ei ollut enää havaittavaa tasavirtavuotoa ja koko siru käytti pienempää toimintajännitettä. Aaltomuotojen yhdisteleminen tuottaa uudella sirumallilla ainakin osittain käyttökelpoisia ja ylipäätään kuuluvia ääniä.

6581-sirumallissa ollutta jännitevuotoa osattiin tosin käyttää myös hyödyksi; audioulostulossa olevaa jännitettä pystyy muuttamaan ohjelmallisesti nostamalla tai laskemalla 4-bittisen vahvistimen rekisteriarvoa nopealla taajuudella, ja siten luomaan uuden analogisen signaalin, joka

[24] Vanhemmassa sirumallissa (6581) verhokäyrän todelliset aika-arvot eivät täsmää Yannesin määrittelemien arvojen kanssa. Ongelma esiintyy Martin Galwayn mukaan varsinkin pitkillä aluke- ja vaimenemisajoilla, jolloin ääni syttyy tai vaimenee aivan liian aikaisin. Todennäköisesti ongelma liittyy mikrosirun aaltomuotorekisteriin, jossa määritellään nuotin syttymis- ja sammumishetki. (Carr, N. & Abbott, C. (toim.) 2001.) Ongelman pystyy kiertämään suorittamalla ns. “kylmäkäynnistyksen” ennen jokaista uutta nuottia.

sekoittuu oskillaattoreista tuleviin kolmeen signaaliin[25]. Näin pystytään käyttämään neljää yhtäaikaista ääntä. Uudemmassa 8580–versiossa jännitevuoto on vähennetty niin pieneksi, että vahvistimen voimakkuudenmuutoksilla aikaansaadun signaalin voimakkuus on liian pieni kuultavaksi.

Yleisesti ottaen 6581–sirun ääntä voi luonnehtia lämpimämmäksi ja ”analogisemmaksi” kuin uudemman 8580–version tuottamaa audiosignaalia. Varsin hyvin tämä ero on kuultavissa suodinta käytettäessä. Liitteenä olevan cd-levyn kappaleet ESIMCD 10 ja ESIMCD 11 (Klose, T. 2005) sisältävät suodattimen läpi kulkevan ääninäytteen soitettuna vuorotellen vanhemmalla ja uudemmalla sirumallilla käyttäen MIDIbox SID –syntetisaattoria.

Luvuissa 2 – 2.2.3 on käsitelty kaikki itse SID–mikrosirun äänentuottoon ja –muokkaukseen liittyvät ominaisuudet. Seuraava luku valottaa lyhyesti C64–tietokoneen sisäistä elämää sekä konekielisen ohjelmoinnin toimintaperiaatteita. Kolmannessa luvussa analysoin pelimusiikkien ohjelmoinnissa käytettyjä musiikkirutiineja, eli eri säveltäjien ”SID-soittotekniikoita”. SID:n ääniominaisuudet itsessään voivat vaikuttaa varsin vaatimattomilta, mutta Commodore 64 –tietokoneen ominaisuuksien hyödyntäminen ohjelmoimalla luo suuren määrän erilaisia mahdollisuuksia äänenvärien ja efektien tuottamiseen edellä läpikäytyjen SID-mikrosirun ”kiinteiden” ominaisuuksien lisäksi.

2.3 6510 –prosessorin käyttämän konekielen erittäin lyhyt oppimäärä

*Tämän luvun lähteenä on käytetty James Butterfieldin ohjelmointiopasta ”Machine Language For The Commodore 64 and other Commodore computers” (*Butterfield, J. 1984*), sekä ohjelmointia käsittelevää nettiartikkelia Programming – Introduction (*Programming – Introduction 2005*).*

[25] SID:n kellotaajuus eli laskentanopeus on ~1MHz, joten Nyqvistin teoreeman (Nyquist-Shannon sampling theorem 2005) mukaan yleisvoimakkuutta ohjelmallisesti säätelemällä olisi *periaatteessa* mahdollista tuottaa taajuudeltaan n. 500 kHz ääni 4-bittisellä resoluutiolla. Yleensä tätä metodia käytetään ”sample”–äänien (eng., näyte), eli digitaalisesti tallennettujen audiosignaalien toistoon. Varsin vaatimaton resoluutio (16 voimakkuusvaihtoehtoa) tuottaa ääninäytteisiin kuitenkin paljon kohinaa, jota ei voi SID:n suotimella poistaa suotimen sijaitessa signaaliketjussa ennen vahvistinta. Lisätietoja digitoitujen näytteiden soittamisesta Commodore 64 –tietokoneella löytyy artikkelista ”The C64 Digi” (Harbron, R., Harsfalvi, L. & Judd, S. 2001).

Kaikki tässä tutkielmassa käsiteltävät musiikkirutiinit on ohjelmoitu Commodore 64 –kotitietokoneen sisältämän 6510–mikroprosessorin käyttämällä konekielellä. Chris Hülsbeckin ohjelmoima Soundmonitor–ohjelma poikkeaa käyttötavaltaan muista tutkielmassa analysoiduista ohjelmista siten, että itse ohjelman käyttäminen ei vaadi konekielen tuntemista, vaan sisältää graafisen käyttöliittymän - monet krakkeriryhmät ja hakkerit olivat epäilemättä onnessaan löytäessään tämän helposti käytettävän musiikkieditorin[26]. Vaikka konekielen hallitseekin, on helppo ja suhteellisen tehokas käyttöliittymä epäilemättä miellyttävämpi käyttää, jolloin pääpaino siirtyy ohjelmointikikkojen pähkäilemisestä musiikin säveltämiseen. Varjopuolena on yleensä tällaisten editorien viemä suuri muistitila, viiveet toimintojen suorittamisessa sekä rajalliset äänenmuokkausmahdollisuudet verrattuna konekielisesti ohjailtaviin ohjelmiin.

Jokaisen tietokoneen toiminta perustuu sähkövirran kulkemiseen elektronisissa piireissä. Nämä piirit käsittelevät sähkövirtaa tasan kahdella tavalla – joko ne välittävät sen eteenpäin tai estävät sen kulkemisen. Tästä johtuen tietokoneissa on käytössä ns. binäärinen järjestelmä, joka koostuu ainoastaan kahden numeron, 1 & 0, yhdistelmistä. Elektronisen piirin ollessa ”ykköstilassa” virta kulkee piirin lävitse ja ”nollatilassa” reitti on tukittu. Tällaista pienintä digitaalisen tiedon yksikköä kutsutaan bitiksi (eng. bit = pala, muru) ja kahdeksan bitin yhdistelmää tavuksi (eng. byte).

Jokaisen tietokoneen sisältä löytyy myös mikroprosessori, joka ohjaa kaikkia tietokoneen tekemiä toimintoja. Mikroprosessori on yhteydessä tietokoneen muistipiireihin, joista se voi ottaa käyttöönsä tietoa tai joihin se voi tallentaa väliaikaisesti tai pysyvästi digitaalista tietoa. Commodore 64 sisältää 65 536 muistipaikkaa, jotka ovat 8-bittisen tietoväylän välityksellä yhteydessä mikroprosessoriin. Tästä seuraa, että jokainen muistipaikka voi käsitellä kahdeksan bitin verran tietoa, mikä vastaa 256 erilaista nollan ja ykkösen

[26] Xerox–yhtiön Palo Altossa sijainnut tutkimuskeskus kehitti ensimmäisenä graafisiin kuvakkeisiin perustuvan tietokoneen käyttöjärjestelmän 1970-luvulla, mutta tämän tyyppinen käyttäjäpinta yleistyi vasta Apple Macintosh –yhtiön tuotua vuonna 1983 markkinoille Lisa-kotitietokoneensa (Inventors of the Modern Computer 2005). Microsoft omaksui idean omaan Windows 1.0 -käyttöjärjestelmäänsä ja nykyisin käytännössä kaikki tietokoneet toimivatkin “ikkunaperiaatteella”. Helpommin hahmotettavan graafisen käyttöliittymän aikaisemman yleistymisen esteenä oli sen vaatima suuri prosessoriteho sekä kallis näyttö. Prosessori- ja näyttötekniikan kehittyessä 1980-luvun alkupuolella myös vaadittavat ominaisuudet sisältävien tuotteiden hinnat alenivat kotitalouksien kukkaroille sopiviksi.

yhdistelmää. Samoin mikroprosessori ymmärtää kahdeksan bitin verran erilaisia komentoja – C64 on siten 8-bittinen tietokone.

6510–mikroprosessorin ymmärtämä konekieli koostuu siis periaatteessa 256 erilaisesta sanasta, eli käskystä. Vuonna 1983 ilmestynyt "The Commodore 64 Programmer's Reference Guide" –ohjekirja listaa kuitenkin vain 151 “virallista” käskyä muiden bittiyhdistelmien saadessa merkinnän ”future expansion” (eng., tuleva laajennus). Konekielen hyvin hallitsevat ohjelmoijat kuitenkin tiesivät miten nämä ”ylimääräiset” 105 käskyä toimivat ja pystyivät hyödyntämään niitä myös työssään.

Kaikki Commodore 64:sen ohjelmat koostuvat 256 erilaisesta käskystä, sekä heksadesimaalisista lukuarvoista, joita käskyt ja rekisterit käyttävät toiminnoissaan. Commodore 64 -tietokoneen ruudulla tietokoneohjelman konekielinen käsky- ja datajono voisi näyttää seuraavanlaiselta:

```
- - -

.:1000 11 3A E4 00 21 32 04 AA
.:1008 20 4A 49 4D 20 42 55 54
.:1010 54 45 52 46 49 45 4C 44

- - -
```

Konekielessä kaikki luvut ilmaistaan muistitilan säästämiseksi kymmenlukujärjestelmän sijaan kuusitoistalukujärjestelmässä, eli heksadesimaaleina[27] (erotuksena desimaaliluvuista käytetään heksadesimaalisten numeroiden edellä merkkiä $). Nelinumeroinen luku vasemmassa laidassa kertoo rivin ensimmäisen arvon muistipaikan numeron, eli osoitteen. Sen oikealla puolella olevat kaksinumeroiset luvut kertovat muistipaikkojen sisältämän arvon. Heksadesimaalein sanottuna annetussa esimerkissä muistipaikkaan 1000 on talletettu arvo 11, muistipaikkaan 1001 arvo 3A, muistipaikkaan 1002 arvo E4 jne. Muistipaikassa oleva heksadesimaalinen luku edustaa tietokoneelle joko *käskyä tai lukuarvoa* riippuen niiden viittaussuhteista ohjelman sisällä.

[27] Heksadesimaalisessa järjestelmässä numerot 10-15 on korvattu kirjaimilla A, B, C, D, E ja F. Esim. luku 15 ilmaistaan heksadesimaaleina 8-bittisessä järjestelmässä luvulla $0F. Heksadesimaalinen luku muutetaan desimaaleiksi kertomalla vasemmanpuoleinen merkki kuudellatoista ja lisäämällä oikeanpuoleinen merkki tulokseen. $21 = 2 * 16 + 1 = 33; $AC = 10 * 16 + 12 = 172 jne. Kaksi heksadesimaalista merkkiä vastaa siis 256 eri arvoa ($00 - $FF), kun desimaalijärjestelmässä kahdella merkillä pystytään tuottamaan vain 100 erilaista arvoa (00-99).

Kaikki tietokoneen toiminnot ovat kontrolloitavissa antamalla mikroprosessorille heksadesimaalisia käskyjä. Jotkin käskyt toteutuvat välittömästi käyttäjän painaessa kirjoitetun komennon jälkeen RETURN–näppäintä, toiset käskyt taas täytyy tallentaa ensin tietokoneen muistiin ja erillisellä käskyllä käynnistää niiden suoritus. Kaikki varsinaiset tietokoneohjelmat ovat muistiin tallennettuja käskyjonoja, joiden läpikäynti alkaa käyttäjän erillisellä ruudulle kirjoittamalla suorituskäskyllä.

Käytännössä tietokoneohjelman koodaus tapahtuu kirjoittamalla käskyjä tai lukuarvoja tietokoneen muistipaikkoihin ns. konekielimonitorin avulla[28]. Konekielimonitori on apuohjelma, joka käynnistetään liittämällä tietokoneen korttipaikkaan ohjelman sisältävä lisälaite. Konekielimonitori näyttää tietokoneen ruudulla muistipaikkojen sisällön numeroina (kts. edellinen sivu) ja mahdollistaa muistipaikkojen sisällön muuttamisen. Näppäimistöä käyttäen osoitin siirretään halutun muistipaikan kohdalle, tieto kirjataan tietokoneen muistiin ja lopuksi ohjelma ajetaan kirjoittamalla ruudulle ohjelman suorituskäsky. Assembler–ohjelma helpottaa työskentelyä antamalla numeromuodossa oleville käskyille symboliset vastineet käyttämällä kirjaimia, mutta säilyttää *lukuarvoina käsiteltävän tiedon numeromuodossa.*

Esimerkiksi käsky, joka vähentää luvun 1 valitun tavun arvosta, ilmaistaan konekielimonitorissa arvolla $C6, mutta assembler–ohjelmaa käytettäessä sama käsky annetaan kirjaimilla DEC (eng. DECrease byte value). Vastaavasti tiedot DEC $C6 $C6 assembler–ohjelmassa tarkoittaisivat, että vähennä luku 1 *muistipaikan* $C6 $C6 *sisältämästä arvosta*. Tällä tavoin ilmaistun ohjelmakoodin hahmottaminen on huomattavasti vaivattomampaa kuin pelkkien ”numerokielisten” toimintojen tulkkaaminen konekielimonitorissa.

Jokaisella tietokoneen muistipaikalla on oma tehtävänsä tietokoneen toiminnassa; muuttamalla rekisteriksi kutsutun muistipaikan arvoa voidaan muuttaa kyseisen muistipaikan ohjaamaa tietokoneen toimintaa. Osa muistipaikoista on taas varattu puhtaasti ohjelmatiedon (komentojen ja lukuarvojen) säilyttämistä varten.

---

[28] Jokainen tietokone sisältää myös kiinteän käyttöjärjestelmän eli konekielisen ohjelman, joka on pysyvästi tallennettu koneen muistiin (C64:ssa CBM-BASIC). Tämä ohjelma käynnistyy automaattisesti koneen käynnistyessä. Käyttöjärjestelmä vastaa mm. tietokoneen näppäimistön, näyttöruudun sekä osoittimen toiminnasta.

Kaikki konekieliset ohjelmat koostuvat siis muistipaikkoihin tallennetuista lukuarvoista sekä 256 erilaisen käskyn yhdistelmistä. Kaikki käskyt perustuvat kahdeksan bitin yhdistelmien lukemiseen, tallentamiseen ja muuttamiseen halutulla tavalla. Esimerkiksi peräkkäin annetut käskyt LDA $0380, ASL, STA $0380 tuottavat 6510 –mikroprosessorin sisältävissä tietokoneissa seuraavanlaisen tapahtumaketjun:

1. lataa muistipaikan $0380 sisältämä lukuarvo A-rekisteriin
2. suorita luvulle aritmeettinen siirros vasempaan
3. tallenna saatu lukuarvo takaisin muistipaikkaan $0380

Oletetaan että muistipaikassa $0380 on heksadesimaalinen lukuarvo $64, jonka vastine binäärisessä järjestelmässä on 0110 0100. Tämä arvo ladataan A-rekisteriin, jossa prosessori suorittaa matemaattisia toimintoja. Luvulle suoritetaan käsky ASL (eng. Arithmetic Shift Left), jolloin kaikki luvut binäärisessä järjestelmässä siirtyvät yhden pykälän vasemmalle ja oikeanpuolimmainen bitti korvautuu nollalla. Täten luku saa uuden binäärisen muodon 1100 1000, jonka lukuarvo $C8 on kaksinkertainen alkuperäiseen verrattuna. Kolmas käsky (STA, eng. Store value in A-register) tallettaa tuplaantuneen lukuarvon takaisin muistipaikkaan $0380.

Tällaiset konekieliset käskyt ovat hyvin yksinkertaisia ja suoraviivaisia, joten niiden toteuttaminen vaatii vain vähän prosessoritehoa. Ne kuluttavat myös vähemmän muistitilaa kuin muilla ohjelmointikielillä kirjoitetut ohjelmat, sillä mikroprosessori ei tarvitse erillistä ohjelmaa ”tulkkaamaan” komentoja. Näistä syistä johtuen käytännössä kaikki Commodore 64:lle ohjelmoidut pelit koodattiin konekielellä; samoin peleissä käytetyt musiikit sekä niiden soittamiseen tarvittavat ohjelmat, joita tämä tutkielma käsittelee.

Tietokoneohjelmaa, jonka koodissa käskytieto on erillään lukutiedosta, kutsutaan ohjelmointirutiiniksi. Tällä periaatteella ohjelmoidun ohjelman voi kopioida sellaisenaan toiseen sovellukseen, ja pelkästään ohjelman lukemia numerotietoja vaihtamalla voi samalla ohjelmalla tuottaa täysin erilaisia lopputuloksia. Kaikki tässä tutkielmassani analysoidut musiikkiohjelmat ovat tällaisia ohjelmointirutiineja – pelkästään nuottitietoa muuttamalla voi samalla musiikkiohjelmalla soittaa lukemattoman määrän eri sävellyksiä.

3 OHJELMOINTIRUTIINIEN ANALYYSIT JA PÄÄTELMIÄ

Juuri “oikeiden” säveltäjien valitseminen tähän vertailevaan tutkielmaan on mahdotonta, onhan kuitenkin kyse musiikillisista makuasioista. Joitakin perusteita valinnoille voidaan kuitenkin antaa:

- kuinka monipuolisesti säveltäjä on käyttänyt C64:n mahdollisuuksia äänentuottamisessa
- kuinka paljon säveltäjä on saanut huomiota alan julkaisuissa (esim. peliarvosteluissa)
- montako julkaistua sävellystä säveltäjä on tehnyt
- muut ansiot ja huomionosoitukset säveltäjänä/muusikkona

Kullekin säveltäjälle ja hänen käyttämälleen soittorutiinille on omistettu oma alalukunsa, jossa käydään läpi itse analyysin lisäksi säveltäjän taustaa sekä syitä, miksi juuri hänen tapansa työskennellä on valittu tarkastelun kohteeksi tähän tutkielmaan.

3.1 Matt Gray

Englantilaisen Matt Grayn ensimmäiset levitykseen tulleet C64–sävellykset ovat vuodelta 1986, jolloin C64 oli jo vakiinnuttanut asemansa pelitietokoneena Euroopassa ja Amerikassa - myös uudempi versio SID–sirusta oli ollut vuoden ajan markkinoilla. Ensimmäiset C64:lle tuotetut pelit myytiin cartridge–muodossa (eng., patruuna), eli koneen takaosaan liitettävinä moduuleina. Halvemmassa kasettiformaatissa julkaistujen pelien yleistyessä myös piraattiversiot ohjelmista yleistyivät ja hakkereiden työstämät demot ja ohjelmat levisivät käyttäjiltä toisille kätevästi paitsi modeemin, myös kasettien välityksellä. Myös Grayn ensimmäiset musiikkijulkaisut ovatkin hänen omalla nimellään julkistettuja, eivät peleihin sävellettyä musiikkia.

Vuodesta 1987 lähtien hänen sävellyksiään päätyi myös C64:n peleihin, joista tunnetuin on epäilemättä musiikki peliin Last Ninja 2 (System 3, 1988).

Muita tunnettuja sävellyksiä ovat musiikit peleihin Driller (Incentive, 1988), Vendetta (System 3, 1990), Deliverance[29] (Hewson, 1990), Rob Levyn – elokuvan pohjalta tehty Hunter's Moon (Thalamus, 1987) sekä latausmusiikki peliin Bangkok Nights (System 3, 1987).

Hänen taustoistaan ja muista töistään tietokoneohjelmien parissa ei löydy mitään tietoa Commodorelle omistautuneista Internet–sivustoilta tai lehtijulkaisuista. Hänen nimensä mainitaan toisten säveltäjien haastatteluissa, (Carr, N. 2001a & Haste Töne? 1989) mutta henkilökohtaisesti hän ei ole antanut kommentteja työstään pelimusiikin saralla julkisuuteen missään foorumissa - Matt ei ole tullut parrasvaloihin edes hänen nimeään kantavien cd-levyjen julkistamisen yhteydessä.

Kyseisten kokoelmalevyjen (The Best Of Matt Gray & Last Ninja 2) julkaisija ja tuottaja Jason MacKenzie toivookin Internet-sivuillaan www.binaryzone.co.uk Matt Grayn tulevan esiin kommentoimaan menneitä ja kertomaan urastaan pelimusiikkien parissa. Myös Chris Abbottin perustama levy-yhtiö C64AUDIO (www.c64audio.com) myy Matt Grayn levyä, joten jo tekijänoikeudellisista syistä johtuen Matt on varmasti tietoinen hänen töitään sisältävistä julkaisuista – todennäköisesti hän ei vain halua olla tekemisissä 80-luvun retrohenkeilyn kanssa. Toinen syy voi olla hänen nykyinen uransa, joka jatkuu musiikin tuottamisen puolella Britanniassa[30] – nörttimenneisyys ensimmäisten kotitietokoneiden parissa ei välttämättä ole positiivinen merkintä CV:n ensimmäisellä sivulla.

Hänen sävellyksensä painottuvat tyylilajiltaan raskaamman rockmusiikin puolelle ja kaikki originaalit kappaleet tuntuvat sisältävän rumpuraidan, toisin kuin esimerkiksi Martin Galwayn hyvin melodispainotteiset sävellykset. Hän epäilemättä osasi myös hyödyntää SID 6581 –sirun jännitevuotoa rumpuäänien tuottamiseen – esimerkkinä tästä käy Maze Mania –pelin musiikki (SIDCD 1), jossa kyseistä mikrosirun ominaisuutta on käytetty tasaista 16-osarytmiä soittavien hi-hat –symbaalien äänen imitoimiseen.

---

[29] Deliverance-pelin musiikkien tekijäksi oli ilmeisesti ehdolla useampia säveltäjiä. Deliverance nimellä varustettuja sävellyksiä löytyy ainakin seuraavilta C64–säveltäjiltä; Johannes Bjerregaard, Laxity, Link.

[30] Hän on ollut vaikuttamassa Xenomania Music –nimen alla mm. seuraavien artistien levyillä soittajana, sovittajana ja/tai tuottajana: Cher, Kylie Minogue, Suga Babes, Girls Aloud ja Saint Etienne.

Maze Mania -kappaleessa basso- ja virvelirumpu sekä 8-osa takapotkuilla oleva sointu on tuotettu käyttämällä vain yhtä oskillaattoria - kaikki äänet soivat limittäin (eivät milloinkaan yhtäaikaisesti), ja takapotkuilla oleva sointu on itse asiassa illuusio harmonisesta intervallista, joka on saatu aikaan kahden äänen hyvin nopealla vuorottelulla. Bassokuvio ja melodiaääni taas vaativat kumpikin yhden oskillaattorin. Äänenvoimakkuusrekisterin hyvin nopeilla muutoksilla tuotettu hi-hat soi siis ikäänkuin oskillaattoreiden tuottamien äänien ”päällä”.

Matt käyttää useissa sävellyksissään rumpujen lisäksi myös tasaisista 16-osanuoteista koostuvaa bassolinjaa, kuten Maze Maniassakin on kuultavissa.

Matt Grayn analysoitava musiikkirutiini on poimittu Driller–pelistä, ja rutiinin lähdekoodin on purkanut ja kommentoinut Stefano Tognon itse julkaisemassaan SIDin–verkkolehdessä (Tognon, S. 2002). Toinen vahva ehdokas soittorutiinin lähteeksi oli Last Ninja 2 –pelissä käytetty ohjelmointirutiini. Tosin hän käytti mitä suurimmalla todennäköisyydellä kyseisissä peleissä samaa ohjelmarunkoa, mutta Last Ninja 2 sisältää huomattavasti enemmän itse musiikkidataa sekä varsin paljon erilaisia ”soittimia” verrattuna Driller–pelimusiikkiin. Tärkeintä analyysissä onkin keskittyminen itse ohjelman toimintaan musiikillisessa mielessä, ei ainoastaan yksittäisiin erilaisiin efekteihin tai soittimiin paneutuminen. Last Ninja 2 olisi sisältänytkin todennäköisesti turhan paljon informaatiota tämän opinnäytetyöksi tehdyn tutkielman laajuuteen suhteutettuna.

Stefano Tognon huomauttaa artikkelissaan ettei hänen tekemänsä lähdekoodin purkaminen tuota 100 %:sta kopiota Matt Grayn käyttämästä ohjelmointirutiinista. Tämä johtuu siitä, että HVSC:n arkiston .sid-tiedostot eivät yleensä sisällä peleissä käytettyjä ääniefektejä eivätkä siten myöskään niiden ohjelmakoodia. Tästä syystä .sid–tiedostot sisältävät vääriä muistipaikkaviittauksia ja nuottidataa, johon ohjelma ei missään vaiheessa viittaa. Pelimusiikit on kuitenkin haluttu tallentaa HVSC–arkistoon sellaisenaan ilman ääniefektejä. Peliä pelatessa ääniefektit syntyvät pelaajan toimintojen seurauksena ja katkaisevat taustalla soivan musiikin – arkiston kappaleversiot ovat kuitenkin tarkoitettu nimenomaan kuuntelua varten, jolloin ääniefektit vain ärsyttäisivät kuulijaa pätkimällä musiikkia. Joihinkin HVSC:n

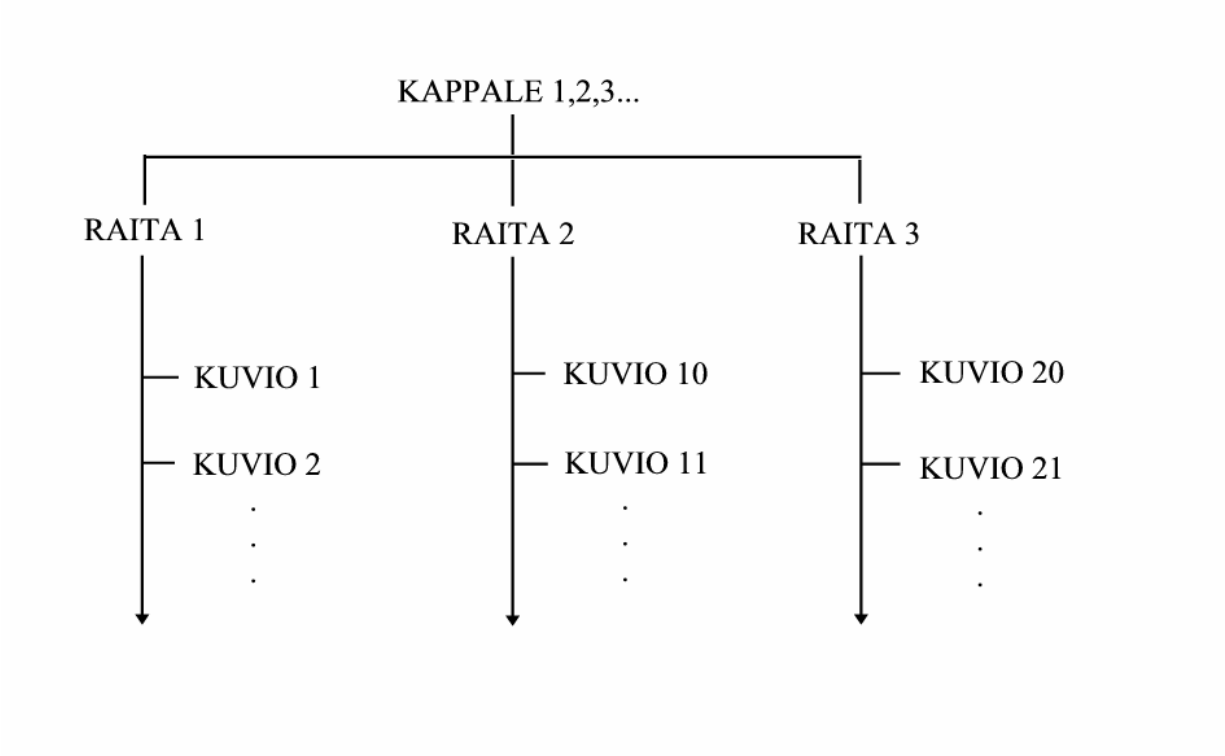
arkiston kappaleisiin on lisätty myös ääniefektit, jolloin ne ovat kuunneltavissa erillään itse pelimusiikeista.

3.1.1 Driller–musiikkirutiini

Tässä luvussa käsitellyn musiikkirutiinin on purkanut ja selittänyt Stefano Tognon Internet-lehdessään SIDin (Tognon, S. 2002). Kuviot, taulukot, johtopäätökset musiikkirutiinin toimivuudesta käytännössä sekä yhteenvedon sisältö ovat omaa käsialaani, ellei toisin ole mainittu. Driller–pelimusiikki löytyy liitteenä olevalta cd-levyltä (SIDCD 2).

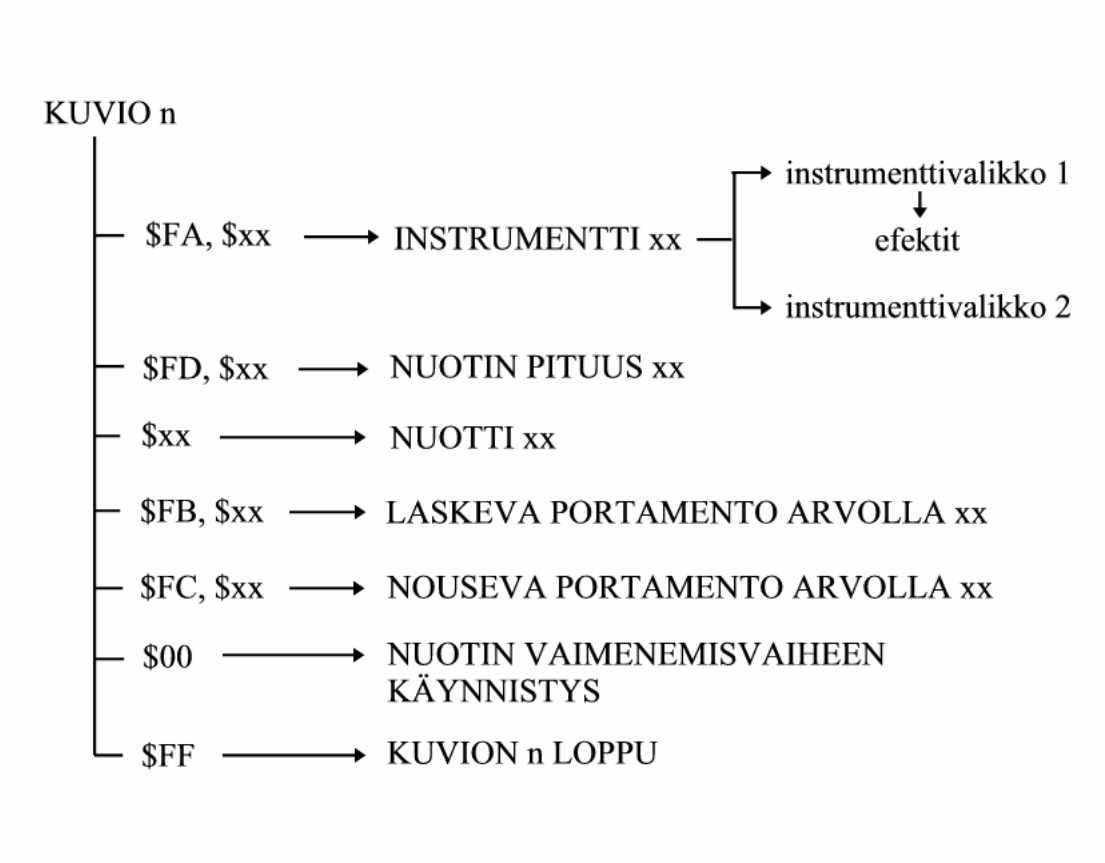
Matt Grayn ohjelmointirutiinissa pelimusiikki on jaoteltu osiin nuottitiedon käsittelyn helpottamiseksi; ensin ohjelma valitsee soitettavan kappaleen (eng. song), sitten raidan (eng. track) ja lopuksi kuvion (eng. pattern). Kuvioita voi ajatella tahteina ja kappaleen osina. Yksi raita taas vastaa yhden oskillaattorin toimintaa. Toisin sanottuna musiikkiohjelma valitsee pelitilanteeseen kuuluvan kappaleen soitettavaksi itse pelin ohjelmakoodin käskemänä ja aloittaa musiikin soiton komentamalla ensimmäisen oskillaattorin soittamaan esim. kuviota 1, toisen oskillaattorin kuvioita 4 sekä 5, kolmannen oskillaattorin kuviota 2 jne. Kaikki nämä komennot on kirjoitettu tietokoneen muistiin konekielisinä käskyinä luvussa 2.3 kuvatulla tavalla; mitään graafista käyttöliittymää ei ole ollut käytössä sävellystä ohjelmoitaessa[31].

[31] Matt on tosin käyttänyt joitakin sävellyksiä tehdessään myös ns. tracker–ohjelmia, kuten Chris Hülsbeckin vuonna 1986 ohjelmoimaa MusicMaster–ohjelmaa (versiosta 1.0 eteenpäin nimellä SoundMonitor), sekä Dutch-USA Team –kollektiivin (Oscar Giesen & Marco Swagerman) vuonna 1987 ohjelmoimaa Rockmonitor–sovellusta (Tognon, S. 2005).



KUVIO 7. DRILLER–musiikkirutiinin toimintaperiaate 1

Jokainen kappale sisältää siis kolme raitaa, joista kukin vastaa yhden oskillaattorin toimintaa - yksi raita taas koostuu useammasta kuviosta. Tällä tavoin kappaleen rakennetta on helppo muutella vaihtamalla kuvioiden soittojärjestystä ohjelmassa tai kertaamalla tiettyä kuviota. Mikä tahansa kuvio voidaan tietysti soittaa millä tahansa raidalla.



KUVIO 8. Driller–musiikkirutiinin toimintaperiaate 2

Kuviot sisältävät tiedon soitettavista nuoteista, instrumenteista (oskillaattoreiden asetuksista), nuottien pituudesta jne. Itse musiikki sijaitsee siis kuvioissa. Vaihtamalla kuvioissa olevan nuottidatan ja vaihtamalla kuvioiden soittojärjestystä voi luoda samalla ohjelmalla uutta musiikkia.

Kuvioissa oleva tieto muodostuu seitsemästä erilaisesta toiminnosta (kts. kuvio 8). $FA kertoo tietokoneelle, että seuraava lukuarvo tarkoittaa instrumentin numeroa, esimerkiksi instrumenttia numero $01. Instrumentit taas koostuvat kahdesta kahdeksan tavun kokoisesta valikosta, jotka määrittelevät instrumentin ominaisuudet.

Käsky $FD tarkoittaa, että seuraava lukuarvo käskyjonossa edustaa nuotin kestoaikaa. Mitä suurempi arvo, sen pidempi on nuotin sointiaika. *Kaikki käskyä $FD seuraavat nuotit soitetaan samalla nuotinpituudella, kunnes uusi nuotinpituus on annettu*. Musiikkirutiinin ajastin laskee nuotin keston vähentämällä annetusta luvusta arvon $01 jokaisella ajastimen

kierroksella kunnes pääsee arvoon $00. Ajastin taas pohjautuu tietokoneen ruudun päivitysnopeuteen, eli virkistystaajuuteen[32].

Käsky $FB aloittaa laskevan liu'un nuotille arvolla $xx ja komento $FC vastaavasti nousevan liu'un arvolla $xx. Jos lukuarvoa ei edellä mikään edellä mainituista käskyistä ($FA, $FB, $FC, $FD), niin kyseessä on sävelkorkeus. SID –siru tarvitsee kaksi tavua määritelläkseen soitettavan taajuuden, mutta Driller –musiikkirutiinissa nuottien taajuudet on sijoitettu valmiiksi laskettuun taulukkoon, josta nuotit voidaan poimia käyttämällä viittaamiseen vain yhtä tavua. Näin nuottitiedon kirjoittaminen nopeutuu ja helpottuu, mutta vastaavasti valmis nuottitaulukko kasvattaa ohjelman kokoa ja aiheuttaa ”sakkokierroksen” ohjelman ajoreitissä.

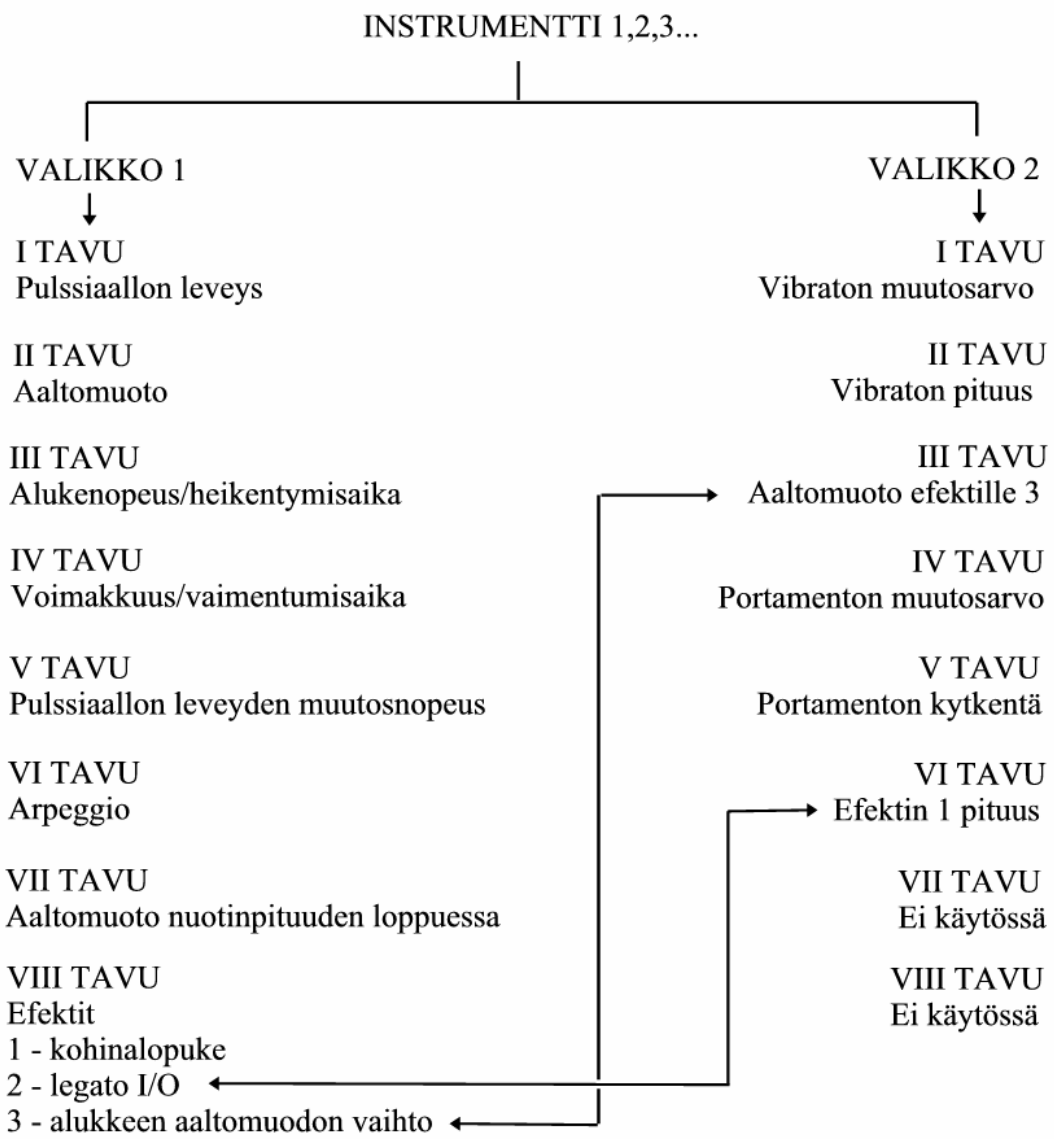
Nuottiarvoille $00 ja $FF on varattu erityinen tarkoitus; $00 ilmoittaa ohjelmalle, että edellisen nuotin verhokäyrän vaimenemisvaihe tulee käynnistää ja $FF taas kertoo ohjelmalle: ”Tämä kuvio päättyy tähän, siirry seuraavaan kuvioon tällä raidalla”. Ilmeisesti jokainen annettu nuotti siis lähtökohtaisesti katkeaa ”seinään” ilman vaimenemisvaihetta, ellei sitä erikseen käynnistetä nuottiarvolla (komennolla) $00.

---

[32] Kuvaruutu koostuu useista riveistä (eng. raster lines), jotka näyttö piirtää järjestyksessä ylhäältä alas, vasemmalta oikealle. Saapuessaan kuvaruudun rivin loppuun, kuvaputki katkaisee piirtosäteensä siirtyäkseen uudestaan kuvaruudun vasempaan laitaan mistä se aloittaa uuden rivin piirtämisen. Toinen pidempi tauko kuvaputken toimintaan tulee piirtimen siirtyessä pidemmän matkan valmiin kuvaruudun oikeasta alakulmasta takaisin vasempaan yläkulmaan uuden ruudun piirtämistä varten. Näitä kuvaputken säännöllisiä toimintajaksoja voi käyttää ajastimina tietokoneohjelmissa.

Commodore 64 päivittää näyttöruutua vakionopeudella, joka on riippumaton muista koneen tapahtumista ja ohjelmoidusta koodista. PAL–järjestelmässä koko ruutu on piirretty uusiksi 1/50 sekunnissa, NTSC–järjestelmässä 1/60 sekunnissa. Tästä johtuen kyseinen Driller–teema soi NTSC-järjestelmässä liian nopeasti, koska sävellys on suunniteltu PAL–järjestelmässä toimivaksi. Kappaleen yleisnopeutta voi vaihtaa muuttamalla ajastimen jaksonpituutta musiikkirutiinissa. Koska nuotin pituus annetaan ohjelmassa aina yhden tavun avulla (256 eri pituutta), myös nuottien absoluuttiset pituudet vaihtuvat suhteessa ajastimen nopeuteen.

3.1.1.1 Istrumentit



KUVIO 9. Driller–musiikkirutiinin toimintaperiaate 3

Instrumentit, eli äänen värit ja ominaisuudet, koostuvat kahdesta kahdeksan tavun valikosta, jotka ohjelma käy kokonaisuudessaan läpi ottaessaan uuden instrumentin käyttöön. Valikon 1 ensimäinen tavu määrittelee pulssiaallon leveyden ja toinen tavu aaltomuodon (jos pulssiaaltoa ei haluta äänessä käyttää, on ensimmäisen tavun arvo yhdentekevä). Kuten taajuudet, myös pulssin leveys määritellään SID–sirulle kahdella tavulla. Matt on rutiinissaan korvannut tavut kahdella tavun puolikkaalla, joilla saadaan aikaiseksi

kuitenkin laaja skaala eri pulssin leveyksiä puolet pienemmällä tiedon määrällä.

Kolmas ja neljäs tavu määrittelevät verhokäyrän arvot SID–mikrosirulle ohjelmoitujen valmiiden taulukoiden mukaan (kts. taulukko 1, sivut 29-30).

Viidennellä tavulla annetaan arvo pulssin leveyden jatkuvalle muutokselle äänessä, eli ns. ”pulse sweep” -toiminnolle (eng., pulssipyyhkäisy), joka on kuultavissa heti Driller–kappaleen alussa (SIDCD 2). Pulssin leveys vaihtelee vakionopeudella pulssin leveyden määrittävän SID -rekisterin enemmän merkitsevän tavun arvojen $08 ja $0E (2048 – 3584) välillä. Saavuttaessaan pienimmän tai suurimman arvonsa liike vaihtaa suuntaa. Mitä suuremman arvon viidennelle tavulle antaa, sitä isommilla askelilla pulssin leveys muuttuu. Muutosarvolla 0 pulssin leveyden vaihtelua ei tapahdu.

Kuudes tavu määrittelee arpeggio–toiminnon ominaisuudet. Arpeggio–rutiinille on omistettu oma alalukunsa 3.1.1.2.

Seitsemännen tavun toiminta ei ole täysin selkeä, mutta käsittääkseni sen avulla voidaan muuttaa äänen aaltomuotoa hyvin pieneksi hetkeksi nuotin lopussa, jolloin jokainen kyseisellä soittimella soitettu ääni saa ikään kuin ”lopukkeen”. Tämä aaltomuodon vaihdos tapahtuu samalla ruudun päivityskerralla kuin uuden nuotin syttyminen, joten parhaimmillaankin tämä lopuke soi vain 1/50 sekunnin ajan. Aika on kuitenkin riittävä, jotta aaltomuodon vaihdoksella saadaan ääneen uusi vivahde.

Toinen vaihtoehto seitsemännen tavun käytölle on yksinkertaisesti ”key-off” –komento, eli nuotin loppumisesta ilmoittava käsky (sama kuin käsky $00 nuottidatassa annettuna). Verhokäyrä nimittäin käynnistetään ja lopetetaan antamalla käsky samaan rekisteriin, joka määrittelee aaltomuodonkin. Jos tämä vaimenemiskäsky on ohjelmoitu suoraan soittimen ominaisuudeksi, ei käskyä tarvitse erikseen kirjoittaa nuottidatan joukkoon, vaan vaimeneminen käynnistyy automaattisesti kyseisellä instrumentilla soitetun nuotin nuotinpituuden loppuessa.

Kahdeksannella tavulla voidaan kytkeä päälle kolme erilaista efektiä: bitillä 0 kohinalopuke, bitillä 1 legato tai aaltomuodon vaihto alukkeen aikana bitillä numero 2. Kohinalopuke saadaan aikaiseksi lisäämällä äänen loppuun hyvin nopeasti laskeva korkeataajuuksinen kohina-aaltomuoto. Kohinan

taajuuden laskettua tarpeeksi alas ääni sammuu kokonaan. Stefano Tognonin artikkelista ei käy ilmi millä tavoin lopukkeen pituutta/taajuuden laskunopeutta voi muuttaa. Efekti on kuultavissa 16-osanuoteissa Driller-kappaleen kohdasta 0:35 eteenpäin (SIDCD 2).

Legato–efekti kytkee uuden nuotin edelliseen ilman erillistä aluketta tai liukua.

Kolmas efekti muuttaa äänen aaltomuodon kahden ruudunpäivityksen ajaksi (1/25 s) valikko 2:n kolmannen tavun mukaiseksi. Tämän efektin avulla voi imitoida esim. rumpuja tai vaskipuhaltimia, joiden aluke on hyvin nopea ja/tai erilainen verrattuna soittimen soivaan äänenväriin.

Valikon 2 kahdella ensimmäisellä tavulla määritellään vibratorutiinin parametrit. Vibraton toimintaa käsitellään omassa alaluvussaan 3.1.1.3. Kolmas tavu valikossa 2 antaa aaltomuodon efektille nro 3 (kts. ed.).

Neljännellä ja viidennellä tavulla määritellään portamenton toiminta. Itse portamento toimii vakionopeudella, mutta askelten arvonmuutosta voidaan vaihtaa neljännen tavun arvolla. Viides tavu taas määrittelee liu'un suunnan ja moduloitavan SID–taajuusrekisterin. Arvo $00 ei kytke portamentoa ollenkaan ääneen, $01 tekee laskevan liu'un muuttamalla vähemmän merkitsevää taajuusrekisteriä ja $03 tekee saman enemmän merkitsevälle taajuusrekisterille. $02 tekee vastaavasti nousevan liu'un vähemmän merkitsevälle taajuusrekisterille ja arvot $04:stä ylöspäin tuottavat nousevan liu'un moduloimalla enemmän merkitsevää taajuusrekisteriä.

Kuudes tavu määrittelee kohinalopukkeen, eli efektin 1, pituuden (kts. ed.). Seitsemäs ja kahdeksas tavu eivät ole käytössä Driller–kappaleen musiikkirutiinissa. Näitä tavuja voi käyttää viitteinä muihin efekteihin ja ääniominaisuuksiin, joita musiikkirutiiniin halutaan liittää.

3.1.1.2 Arpeggio

Arpeggio koostuu valmiista taulukoista, joihin instrumenttivalikon 1 kuudennella tavulla viitataan. Arpeggio–tavulla määritellään mitä nuottisekvenssiä (arpeggiotaulukkoa) käytetään ja kuinka monta peräkkäistä nuottia taulukosta soitetaan. Taulukoihin on mahdollista ohjelmoida sävelkulkuja pienimmillään puolen sävelaskeleen välein ja suurimmillaan

hypyn ylimpään mahdolliseen säveleen asti. Nuottien määrää arpeggiossa rajoittaa vain vapaana oleva muistitila.

Esimerkiksi arvo $ 51 kuudennessa tavussa valitsee arpeggiotaulukon nro 1 ja soittaa siitä viisi ensimmäistä ääntä riippumatta siitä, kuinka monta ääntä taulukko kokonaisuudessaan sisältää. Taulukossa yhtä arpeggion nuottia vastaa yksi tavu, ja tämä tavu ilmoittaa montako puolisävelaskelta lisätään perusääneen, eli taajuuteen joka on määritelty kuviossa (pattern). Esimerkiksi arpeggiotaulukon arvot $07, $0C, $07, $0C ja $00 tuottaisivat ensin kvintin perusääneen suhteutettuna, sitten oktaavin, kvintin, oktaavin ja lopuksi perusäänen.

Myös arpeggio toimii ohjelmassa vakionopeudella. Arpeggiota voi kuitenkin hidastaa antamalla arpeggiotaulukolle aina kaksi kertaa saman arvon peräkkäin, jolloin yksittäisen nuotin sointiaika tuplaantuu.

3.1.1.3 Vibrato

TAULUKKO 2. Vibraton toiminta

ASKEL nro	0 * L	1 * L	2 * L	3 * L	4 * L
TOIMINTA	F – V	F + V	F + V	F – V	F – V

Muuttuja F on nuotin taajuus kuviossa, V vibraton muutosarvo (valikko 2, 1. tavu) ja muuttujalla L (valikko 2, 2. tavu) voidaan kertoa montako kertaa askeleen toiminta suoritetaan peräkkäin. Askelten nopeus on tässäkin toiminnossa vakio, mutta vibraton nopeutta kuulokuvassa pystyy säätelemään erilaisilla muuttujien arvojen yhdistelmillä.

Vibrato pysyy aktivoituna koko nuotin soimisajan ja aloittaa toimintansa askeleesta 0. Laskettuaan taajuuden arvon neljännessä askeleessa ohjelma siirtyy takaisin askeleeseen 1 ja jatkaa vibratoa askeleiden 1 – 4 välillä. Esimerkiksi taajuuden F ollessa 440 Hz, vibraton muutosarvo 10 ja vibraton syvyys 2, saadaan seuraavanlainen taajuudenvaihtelu: 430, 420; 430, 440; 450, 460; 450, 440; 430, 420; 430, 440; 450, 460; jne.

3.1.2 Yhteenveto Driller-musiikkirutiinista

Matt Grayn käyttämä musiikkirutiini kuvastaa hyvää ohjelmointitaitoa ja ilmiselvää ymmärrystä musiikissa yleensä käytettävistä tehokeinoista ja rakenteista. Rutiinissa käytetty kappaleen rakenne muistuttaa paljon ns. tracker–musiikkiohjelmissa käytettyä tapaa jakaa tieto raidoille ja tietyn jakson mittaisiin kuvioihin. Samoin instrumenttien valinta ja portamenton ohjelmointi muistuttavat tällaista raituripohjaista ajattelutapaa.

Ensimmäisen virallisen ja kaupallisen raiturin julkaisi Karsten Obarski vuonna 1987 Commodore Amiga –tietokoneelle. Tässä tutkielmassa käsiteltävä Chris Hülsbeckin Commodore 64:lle ohjelmoima SoundMonitor (alunperin MusicMaster) on kuitenkin tietääkseni maailman ensimmäinen varsinainen graafisella käyttöliittymällä varustettu raituri. Matt on Stefanon mukaan käyttänyt mm. SoundMonitor–ohjelmaa sävellyksissään, joten on hyvin mahdollista että hänen vuonna 1988 käyttämänsä Driller-ohjelmointirutiini on saanut vaikutteita tracker–ohjelmien toimintaperiaatteista. Tiedon luotettava varmistaminen edellyttäisi tietenkin haastattelua itse säveltäjän kanssa.

Yleisellä tasolla arvioituna rutiini on varsin monipuolinen mutta myös virtaviivainen sävellyskäytössä. Instrumenttien sijoittaminen efekteineen erillisiin valikoihin helpottaa ohjelman käyttöä ja nopeuttaa nuottitiedon muuntelua (nuottidata on siistin näköistä). Samoin kappaleen rakenteen muokkaus sujuu kuvioiden ansiosta varsin helposti.

Pulssipyyhkäisyn, vibraton, arpeggion ja portamenton ohjelmoiminen kiinteästi soitinvalikoihin mahdollistaa hyvin monimutkaisten äänenvärien käytön varsin pienellä vaivalla itse sävellysvaiheessa. Samoin instrumentteihin lisättävät efektit ovat varsin hyvin toteutettuja ja käyttökelpoisia. Varsinkin rumpuja jäljittelevät äänet on helppo luoda käyttämällä efektejä 1 & 2.

Suurimpana puutteena rutiinissa pidän suotimen käytön hankaluutta. Suodinta ei ole liitetty osaksi mitään soitinta, vaan sen aktivoiminen vaatisi erillisen käskyosion. Driller–kappaleessa suotimen laatu on valittavissa samassa ohjelman osiossa, missä SID–sirun rekisterit valmistellaan käyttöä varten. Missään vaiheessa kappaletta suodinta ei kuitenkaan ole aktivoitu eikä sille ole annettu leikkaustaajuutta.

Suotimen jättäminen kokonaan pois käytöstä on aika vahva musiikillinen linjanveto – tosin silloin säästytään myös suotimien eroihin liittyvistä konekohtaisista ongelmista (kts. 2.2.3, ss. 32-34). Instrumenttivalikossa 2 on vieläpä vapaana kaksi tavua, joita ei Stefanon artikkelin mukaan käytetä mihinkään toimintoon musiikkirutiinissa – miksi suodinta ei voisi ohjata näillä tyhjillä tavuilla, jos ohjelma kuitenkin käy valikot kokonaisuudessaan läpi aktivoidessaan soittimen?

Myös ”ring modulation” –efekti on sivuutettu kokonaan Matt Grayn rutiinissa. Syy efektin poisjättämiseen voi olla yksinkertaisesti musikaalinen – efektillä tuotetut äänet ovat kuitenkin hyvin omalaatuisia ja niiden käyttömahdollisuudet sävellyksissä rajalliset.

Todennäköisesti ohjelman suoritusnopeus kärsii instrumenttien varsin laajan valikkojärjestelmän takia. Mitään ongelmia tuskin ilmenee soitettaessa kappaletta irrallaan pelistä, mutta pelin yhteydessä musiikkirutiinin pieni koko ja suoritusnopeus nousevat hyvin merkitseviksi tekijöiksi. Monet muutkin pelissä olevat toiminnot nimittäin pohjautuvat kuvaruudun päivittämisessä syntyviin jaksoihin ja muihin tietokoneen jatkuvasti suorittamiin ”taustatehtäviin”. Mitä enemmän pelissä tapahtuu asioita yhtäaikaa eri tasoilla, sitä suuremmalla syyllä ohjelmakoodin tulisi olla mahdollisimman virtaviivaista ja tehokasta, jotta peli ei hidastu, pätki tai kaada koko koneen toimintaa.

Vaikka soitin muodostuisi pelkästä sahalaita-aallosta ja pakollisista verhokäyrän asetuksista, joutuu musiikkirutiini kuitenkin käymään läpi 16 tavua komentoja (joista kaksi on pois käytöstä) vaadittavan kolmen sijasta määrittääkseen tämän varsin yksinkertaisen soittimen ominaisuudet. Jos soittimessa haluaisi käyttää vielä suodinta ja kehämodulointia, täytyy niitä koskevat asetukset tunkea nuottidatan joukkoon tai vastaavasti muuttamaan instrumenttitaulukkoa niin, että käyttämättömillä tavuilla viitataan erilliseen aliohjelmaan, joka määrittelee kahden tavun avulla tarvittavat suotimen ja kehämodulaation asetukset.

Parhaimmillaan rutiini kuluttaa aikaa pelitilanteen vaatimien ääniefektien[33] tuottamiseen vain noin 630 mikroprosessorin kierrosta (n. 0,6 ms), mutta nuottitiedon ja kuvioiden toistamiseen kuluu pahimmillaan jopa 2

[33] Ääniefektejä ei käsitellä tämän tutkielman analyyseissä. Kts. 1.4, s. 13.

142 kierrosta (n. 2,2 ms) (Tognon, S. 2002). Kahden millisekunnin viive musiikissa yhdistettynä muihin pelitapahtumien aiheuttamiin viiveisiin on jo huomattavissa korvakuuloltakin.

3.2 Martin Galway

Tämän luvun teksti on koottu Martin Galwayn hastatteluista (Varga, A. 1996 ja Interview With Martin Galway 2005).

Ensimmäiset tietokoneella sävelletyt kappaleensa brittiläinen Martin Galway teki jo vuonna 1983 koulunsa BBC–tietokoneilla. Keväällä 1984 hän teki musiikit koulukaverinsa ohjelmoimaan peliin, jonka he yhdessä onnistuivat kauppaamaan Ocean–pelitalolle Manchesterissa. Martin tarjosi samantien muitakin tietokoneella säveltämiään kappaleita kyseiselle pelifirmalle, jonka henkilöstö kehotti häntä siirtymään BBC–tietokoneista Commodore 64:n käyttäjäksi. Ocean antoi hänen käyttöönsä yhden C64–koneen, assembler–ohjelman ja lähdekoodin Oceanin käyttämään musiikkirutiiniin, ja jo vuoden 1985 helmikuussa Ocean palkkasi hänet ohjelmoijaksi peliyhtiöön.

Käyttämällä Oceanin musiikkirutiinia hän ohjelmoi musiikit ensimmäiseen C64-pelijulkaisuunsa “Daley Thompson’s Decathlon”, mutta ohjelmoi jatkossa itse musiikkirutiininsa. Ensimmäinen oma rutiini oli käytössä pelissä Kong Strikes Back, joka julkaistiin vuonna 1984 Martinin ollessa vasta 18-vuotias.

Vuoden 1987 aikana Martin siirtyi käyttämään Atari ST –tietokonetta musiikkirutiinien tekoon. Vaikka hän kirjoitti koodin eri tietokoneella, ei se muuttanut mitenkään Commodore 64:sen ääniominaisuuksia - tehokkaamman Atari ST -tietokoneen käyttäminen kuitenkin helpotti ja nopeutti itse ohjelmointityöskentelyä tarjoamalla paremmat muokkausmahdollisuudet ja enemmän muistitilaa.

Hänen säveltämänsä musiikki päätyi yhteensä 34:ään C64–tietokonepeliin ja lopetettuaan työskentelyn C64:sen parissa hän siirtyi tekemään musiikkia uudemmilla tietokonemalleilla. Nykyään Martin tuottaa musiikkia ja äänitehosteita peleihin Digital Anvil –yhtiössä.

Hänen valttinsa C64-pelimusiikin tekijänä oli kokemus tietokoneiden ohjelmoinnista yleisellä tasolla, sillä uransa alussa Ocean–pelitalossa hän

osallistui myös pelien ohjelmointiin. Hän oli ensimmäinen C64–säveltäjä, joka käytti kaupallisesti julkaistun pelin musiikeissa hyväkseen SID–sirussa olevaa jännitevuotoa (Arkanoid [Imagine, 1987]) ja myöhemmin hän hioi taitojaan sample–tekniikan käytössä sävellyksissään (mm. Game Over [Imagine, 1987]). Hänellä on myös musikaalinen koti, sillä hänen isänsä on musiikinopettaja ja hän soitatti aikanaan pojallaan huilua, viulua, klarinettia ja pianoa. Piano oli kuitenkin ainoa soitin, jonka opiskelua Martin jonkin aikaa jatkoi.

Martinin C64–sävellyksiä on koottu mm. Back In Time ja The Sound Interface Device –kokoelmille ja pelkästään Martinin sävellyksistä on koottu Project: Galway –tuplalevy[34], jonka kappaleet on äänitetty käyttäen soittimena Martinin omaa Commodore–tietokonetta, jolla sävellykset on alunperinkin tehty. Tällä tavoin äänitettynä kappaleet kuulostavat juuri sellaisilta kuin Martin halusikin niiden kuulostavan (koskien SID–mikrosirujen kappalekohtaisia eroja kts. luku 2.2.3 ss. 32-34).

Parhaimmillaan Martinin sävellykset pelastivat muuten markkinoilla tuhoon tuomitut pelit kadotukselta – tietokoneharrastajat nimittäin ostivat erittäin kehnosti ohjelmoituja pelejä pelkästään Galwayn ohjelmoiman musiikin takia (mm. Miami Vice (Ocean, 1986) ja Highlander (Ocean, 1986). Erityisesti hänen kykynsä saada SID “laulamaan” (esim. Martinin versio Jean Michel Jarren “Magnetic Fields” –kappaleesta pelissä Yie Ar Kung Fu (Imagine, 1985; SIDCD 3) on aiheuttanut ihastusta tietokoneharrastajien parissa (Carr, N. 2001b).

3.2.1 Arkanoid -musiikkirutiini

Tässä luvussa käsitellyn musiikkirutiinin on purkanut ja selittänyt Stefano Tognon internetlehdessään SIDin (Tognon, S. 2003). Kuviot, taulukot, johtopäätökset musiikkirutiinin toimivuudesta käytännössä sekä yhteenvedon sisältö ovat omaa käsialaani, ellei toisin ole mainittu. Arkanoid –pelimusiikki löytyy liitteenä olevalta cd-levyltä (SIDCD 4).

[34] Martin Galwayn pelimusiikkeja sisältäviä cd-levyjä voi tilata mm. Internet–osoitteista www.binaryzone.co.uk ja www.c64audio.com.

Martin Galwayn ohjelmoima musiikkirutiini jakautuu kappaleisiin, jotka koostuvat kolmesta raidasta, kuten Matt Grayn käyttämässä soittorutiinissa. Martin ei kuitenkaan jaa raitoja erikseen kuvioiksi, vaan nuottidata sijoitetaan suoraan raidoille. Kuvioiden määrittelemisen sijaan Martin käyttää konekielistä hyppykäskyä JSR (eng. Jump to SubRoutine) muistuttavaa käskyä, joka käskee ohjelmaa siirtymään senhetkisestä kohdasta annettuun osoitteeseen (muistipaikkaan) ja suorittamaan siellä annetut komennot (aliohjelman) ensin. Käskyä RTS (eng. ReTurn from Subroutine) muistuttavalla komennolla musiikkirutiini palaa kohtaan, mistä lähtikin aliohjelmaan.

Tällä tavoin nuottidatasta tulee vaikeammin hallittavan näköistä ja kappaleen rakenteen muuntelu ei ole yhtä selkeää kuin käytettäessä jakoa raitoihin ja kuvioihin. Etuna on kuitenkin ohjelman suoritusnopeuden lisääntyminen, kun haluttuun kappaleen kohtaan siirrytään käyttämällä vain yhtä konekielistä hyppykäskyä.

Arkanoid–musiikkirutiinin nuottidatassa voi käyttää hyvin monenlaisia käskyjä, joilla ohjataan käytännössä kaikkia musiikkirutiinin toimintoja lukuunottamatta kappaleiden ja raitojen jakamista sekä kappaleen tempon muuttamista. Käytännössä nämä käskyt voi jakaa kahteen luokkaan; komentoihin, jotka koskevat nuottiarvoja sekä kappaleen rakennetta ja käskyihin, jotka määrittelevät instrumenttien ominaisuudet. Taulukossa 3 on lueteltu kaikki käskyt, joita käytetään nuottidatan antamiseen sekä kappaleen rakenteen muuttamiseen.

TAULUKKO 3. Arkanoid–musiikkirutiinin nuottidataa ja kappalerakennetta koskevat käskyt (Tognon, S. 2003, muunnelma)

KÄSKYKOODI	**TOIMINTA**
$00 – 5F nn	Soita nuotti $00 – 5F taulukosta otettavalla pituudella nn
$60 – BF nn	Soita nuotti $60 – BF pituudella nn
$C0	Palaa kohtaan, josta hyppy suoritettiin (kts. $C2)
$C2 xx, yy	Suorita aliohjelma, joka sijaitsee osoitteessa xx yy
$C4 xx, yy	Hyppää uuteen osoitteeseen

TAULUKKO 3. (jatkuu)

$C6 nn, xx, yy	Suorita osoitteen xx, yy aliohjelma nn puolsävelaskelta ylempää/alempaa
$CC nn	Toista ohjelman osa nn kertaa
$CE	Siirry seuraavaan toistoon (kts. $CC)
$D8 xx, yy	Suorita konekielinen aliohjelma osoitteessa xx yy

Arkanoid–rutiinissa on käytössä 80 erikorkuista nuottia, joiden taajuudet on määritelty erillisessä taulukossa. Suurin mahdollinen nuottien määrä rutiinissa olisi 95 ($00 – 5E), mutta Martin ei koskaan löytänyt käyttöä aivan ylimmille nuoteille, eikä siksi ohjelmoinut niitä nuottitaulukkoon (Bowness, A. 2-4/2005). Tästä johtuen matalimman taulukon äänen saa arvolla $00 tai $60, ja korkeimman arvolla $4F tai $AF. Arvot $5F ja $BF tuottavat tauon.

Nuotin pituus voidaan määritellä rutiinissa kahdella tavalla; jos nuotin sävelkorkeus ilmaistaan arvolla $00 – 5F, nuotin keston ilmaiseva tavu nn viittaa nuotinpituustaulukkoon, ja vastavasti arvojen $60 – BF yhteydessä arvo nn ilmaisee nuotin keston absoluuttisesti (nn ajastimen yksikköä). Nuotinpituustaulukko sisältää 32 eri pituutta, joihin viitataan nuottidatassa arvoilla $01 – 20. Taulukon pituuksien perusarvo määritellään absoluuttisina ajastimen yksikköinä (1/50 sekunteina)[35]. *Kaikki taulukon nuotinpituudet* ovat neljän kerrannaisia tästä absoluuttisesta arvosta.

Esimerkiksi *taulukon absoluuttisen arvon* ollessa $02, on taulukon ensimmäinen nuotinpituus $08 (8/50 s), seuraava arvo on $10 (16/50 s), seuraava on $18 (24/50 s) jne. Tempoa voidaan siis vaihtaa muuttamalla taulukon lyhyimmän nuotinpituuden absoluuttista kestoa - tällöin taulukkoon viittaavat nuotinpituudet muuttuvat suhteessa lyhyimpään mahdolliseen nuotinpituuteen, mutta *absoluuttisesti määritellyt nuotinpituudet pysyvät samoina*. Tämän ansiosta rutiinilla voidaan tuottaa monimutkaisia polyrytmejä

[35] Nuottien pituudet pohjautuvat C64-tietokoneen CIA–komponentin (Complex Interface Adapter) keskeytyksiin, jotka tapahtuvat 1/50 tai 1/60 s. välein riippuen sähkönjakeluverkon käyttämästä taajuudesta. Vaikka näyttöruudun uudelleen piirtämiseen kuluva aika (1/50 s PAL, 1/60 s NTSC) täsmääkin CIA–ajastimen keskeytysnopeuden kanssa, ne eivät ole toiminnallisessa yhteydessä toisiinsa. Näyttöruudun päivitykseen pohjautuvat ajastimet mahdollistavat vielä pienempien aikayksiköiden käytön hyödyntämällä näytön yksittäisen vaakalinjan piirtämiseen kuluvaa ajanjaksoa.

tai suorittaa metrisiä modulaatioita. Esimerkki metrisestä modulaatiosta löytyy Martinin Ocean–pelitalolle ohjelmoimasta latausmusiikista Ocean Loader 2 (SIDCD 5, 0:48).

Komennolla $C2 voidaan nuottidatassa siirtyä haluttuun kohtaan ja tällä tavoin voidaan muutella esim. kappaleen rakennetta. Käsky $C2 vaatii aina parikseen käskyn $C0, joka ilmoittaa mistä kohdasta aliohjelmaa rutiini palaa takaisin hyppypaikkaan. Komentoa $C0 käytetään myös raitojen lopussa ilmoituksena nuottidatan loppumisesta. Myös komento $C4 ohjaa ohjelman uuteen osoitteeseen, mutta ilman velvoitusta palata takaisin hyppykohtaan.

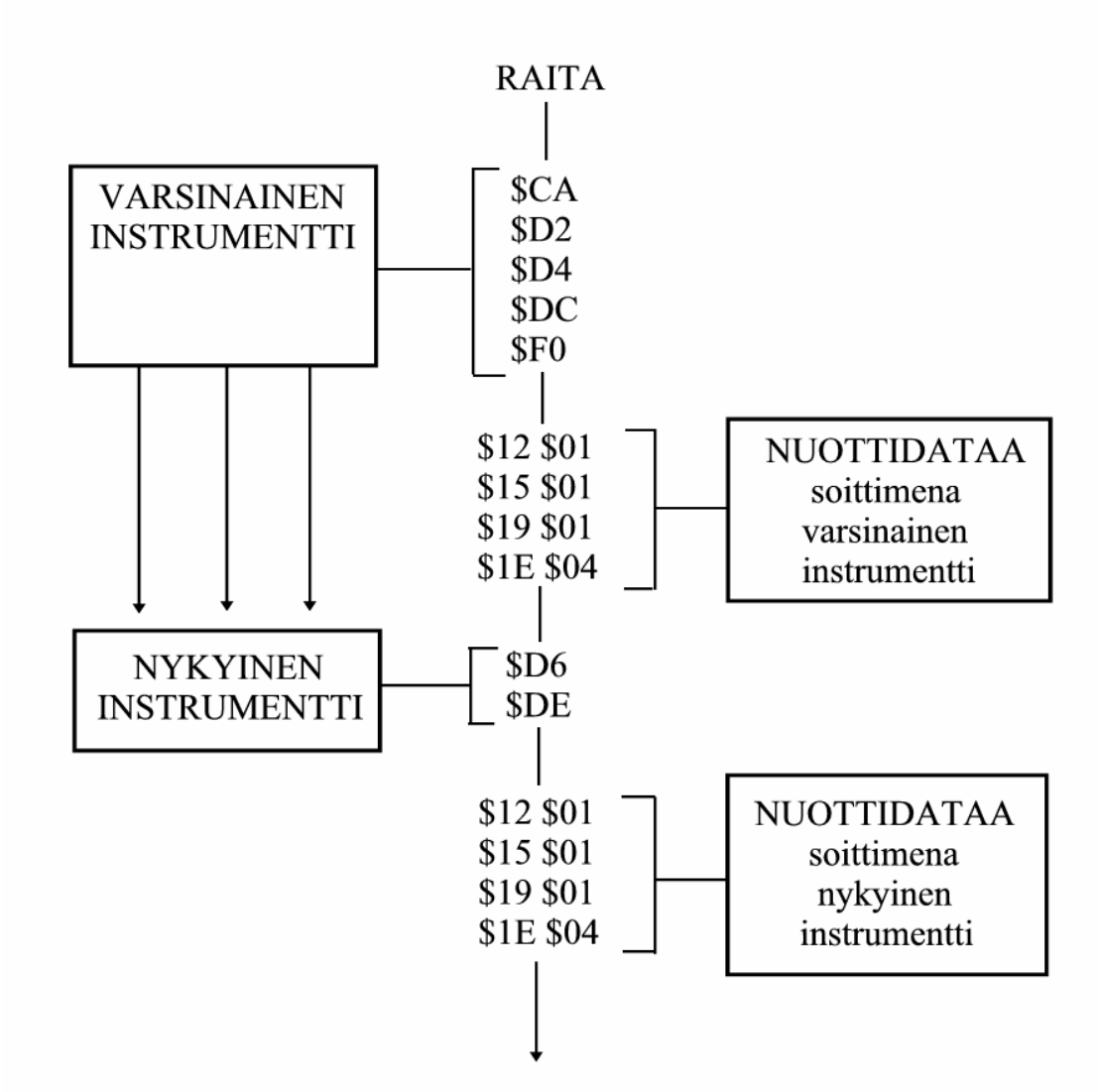
Käskyllä $C6 rutiini siirtyy suorittamaan musiikkidataa käsittelevää aliohjelmaa, mutta transponoi nuottidatan nn puolsävelaskelta ylös- tai alaspäin (negatiivisten lukujen ilmaisemisesta heksadesimaaliluvuilla kts. alaviite 36, s. 64).

Neljäs hyppykäsky $D8 on tarkoitettu siirtymiseen itsenäiseen konekieliseen aliohjelmaan – ei musiikkidataa sisältävään aliohjelmaan.

Komennot $CC ja $CE toimivat aina yhteistyössä; käsky $CC kertoo ohjelmalle montako kertaa (nn) seuraava ohjelman osa, ns. silmukka, toistetaan ja käsky $CE määrittelee silmukan loppupisteen, josta ohjelma palaa takaisin silmukan alkuun (käskyyn $CC).

3.2.1.1 Instrumentit

Hyvin omalaatuinen piirre Martin Galwayn musiikkirutiinissa on instrumenttien ohjelmointikäytäntö. Sen sijaan, että ohjelma sisältäisi x kappaletta erilaisia etukäteen ohjelmoituja soittimia, Martinin käyttämä musiikkirutiini sisältää ”soitinaihioita”, joiden ominaisuuksia voi ”muuttaa lennosta” nuottidatan seassa annettavilla käskyillä muuttamatta itse aihioita, eli varsinaisia instrumentteja. Instrumenttien sointivärin varioiminen tällä tavoin on sulavaa ja tuottaa musiikkiin hienovaraisia nyansseja – juuri niitä ominaisuuksia, joiden johdosta Martinin sävellyksiä on ylistetty (Carr, N. 2001 & Bowness, A. 2-4/2005).



KUVIO 10. Arkanoid–musiikkirutiinin instrumenttien toimintaperiaate

Taulukko 4 sisältää käskyt, joilla voidaan muodostaa ja muunnella kaikkia käytettävien instrumenttien ominaisuuksia.

TAULUKKO 4. Arkanoid–musiikkirutiinin instrumenttien ominaisuuksia koskevat käskyt (Tognon, S. 2003, muunnelma)

KÄSKYKOODI	**TOIMINTA**
$CA id, nn	Anna arvo nn instrumentin parametrille id
$D2 nn, xx, yy	Anna arvot instrumentin ensimmäisille parametreille 1 - nn osoitteesta xx yy
$D4 xx, yy	Anna arvot instrumentin verhokäyrälle ja aaltomuodolle osoitteesta xx yy
$D6 id, nn	Anna arvo nn nykyisen instrumentin parametrille id
$DC id, n1, n2	Anna arvot n1 ja n2 instrumentin parametrille id
$DE id, n1, n2	Anna arvot n1 ja n2 nykyisen instrumentin parametrille id
$E0	Kytke alipäästösuodin kolmannelle oskillaattorille suurimmalla mahdollisella resonanssin arvolla
$E2 xx, yy	Anna arvot suotimen parametreille osoitteesta xx yy
$F0 xx, yy	Anna arvot taajuusefektin parametreille osoitteesta xx yy

Ennen nuottitiedon kirjaamista raidalle musiikkirutiinin täytyy tietää millä soittimella nuotit soitetaan. Varsinaisen instrumentin ominaisuudet määritellään käyttämällä komentoja $CA, $D2, $D4, $DC sekä $F0, ja nykyisen instrumentin ominaisuudet komennoilla $D6 ja $DE. Kaikki varsinaisen instrumentin arvot kopioituvat samoihin kohtiin nykyisen instrumentin taulukkoon lukuunottamatta parametrejä $18, $19 ja $1A (kts. taulukko 5), joita voidaan muuttaa ainoastaan muuttamalla varsinaista instrumenttia. Varsinaisen instrumentin muuttujien arvot ovat ikään kuin soittimen pysyviä ominaisuuksia ja nykyisen instrumentin arvot taas senhetkisiä, oskillaattorin väliaikaisesti käyttämän soittimen arvoja (kts. kuvio 10).

Käskyllä $CA voidaan määritellä varsinaisen instrumentin ominaisuuksia, jotka vaativat yhdellä tavulla määritellyn arvon (0-255). Haluttu arvo (nn) sijoitetaan valittuun muuttujaan käyttämällä muuttujaa vastaavaa tunnusnumeroa (id).

Käskyllä $D2 voidaan antaa halutulle varsinaisen instrumentin ominaisuuksien joukolle (nn) arvot, jotka on sijoitettu muistipaikkaan xx yy.

Muutettavat ominaisuudet valitaan järjestyksessä ensimmäisestä varsinaisen instrumentin taulukossa (taulukko 5) olevasta muuttujasta laskien. Esimerkiksi määrittämällä joukoksi $03 (nn), voi instrumentin parametreille $00 - 03 antaa muistipaikasta xx yy lähtien löytyvät neljä peräkkäistä arvoa.

Käsky $D6 määrittelee arvon nn nykyisen instrumentin muuttujalle, jonka tunnusluku on id. *Nykyisen instrumentin ominaisuuksien muutoksilla ei ole vaikutusta varsinaisen instrumentin ominaisuuksiin.*

Käskyllä $DC määritellään varsinaisen instrumentin muuttujat (id), jotka vaativat kaksitavuisen arvon (n1 n2) ja käskyllä $DE vastaavat muuttujat *nykyiselle instrumentille*.

Käskyllä $E0 kytketään alisuodin päälle kolmannelle oskillaattorille käyttäen suurinta mahdollista resonanssin arvoa. Vastaavan käskyn voisi ohjelmoida myös ensimmäiselle ja toiselle oskillaattorille, mutta muistin säästämiseksi Martin on jättänyt tämän toiminnon pois muilta oskillaattoreilta, jotka eivät kappaleessa suodinta muutenkaan käyttäisi.

Arkanoid–musiikkirutiinissa itse asiassa jokainen käskytaulukon toiminto on ohjelmoitu lähtökohtaisesti kolmeen kertaan, eli erikseen jokaiselle kanavalle. Jos jotakin toimintoa/käskyä ei kappaleessa tarvitsekaan käyttää kaikilla oskillaattoreilla, voidaan tällaiset ylimääräiset toimintaohjeet helposti poistaa vaarantamatta rutiinin rakennetta tai toimintaa millään tavoin. Näin rutiini säilyttää yksinkertaisemman ja helposti muunneltavan muodon, vaikka viekin enemmän muistitilaa kuin vastaava rutiini, jossa olisi käytetty kolmeen kertaan kirjoitettujen taulukoiden sijasta viitenumerointiin perustuvaa ohjelmointitapaa.

Suotimen muuttujat asetetaan käskyllä $E2, jolloin osoitteessa xx yy olevat arvot siirretään suotimen ominaisuudet määrittävään taulukkoon. Samalla periaatteella asetetaan taajuusefektin parametrit käskyllä $F0.

Taulukko 5. Varsinaisen ja nykyisen instrumentin parametrit (Tognon, S. 2003, muunnelma)

VARSINAINEN INSTRUMENTTI	**NYKYINEN INSTRUMENTTI**

PARAMETRIN TUNNUSLUKU (id)	**MUUTTUJAN TOIMINTA**	**PARAMETRIN TUNNUSLUKU (id)**	**MUUTTUJAN TOIMINTA**
$00 - 01	Efektin 1 taajuuden lisäys/kierros 1. vaiheessa	$00 - 01	Efektin 1 taajuuden lisäys/kierros 1. vaiheessa
$02 - 03	Efektin 1 taajuuden lisäys/kierros 2. vaiheessa	$02 - 03	Efektin 1 taajuuden lisäys/kierros 2. vaiheessa
$04 - 05	Efektin 1 taajuuden lisäys/kierros 3. vaiheessa	$04 - 05	Efektin 1 taajuuden lisäys/kierros 3. vaiheessa
$06 - 07	Efektin 1 taajuuden lisäys/kierros 4. vaiheessa	$06 - 07	Efektin 1 taajuuden lisäys/kierros 4. vaiheessa
$08	Efektin 1 kierrosten määrä 1. vaiheessa	$08	Efektin 1 kierrosten määrä 1. vaiheessa
$09	Efektin 1 kierrosten määrä 2. vaiheessa	$09	Efektin 1 kierrosten määrä 2. vaiheessa
$0A	Efektin 1 kierrosten määrä 3. vaiheessa	$0A	Efektin 1 kierrosten määrä 3. vaiheessa
$0B	Efektin 1 kierrosten määrä 4. vaiheessa	$0B	Efektin 1 kierrosten määrä 4. vaiheessa
$0C	Viivekierrosten määrä ennen efektin 1 alkamista	$0C	Viivekierrosten määrä ennen efektin 1 alkamista
$0D	Määrittelee instrumentin toiminnan efekti 1:sen 4. vaiheen päätyttyä	$0D	Määrittelee instrumentin toiminnan efekti 1:sen 4. vaiheen päätyttyä
$0E	Efektin 2 kierrosten määrä 1. vaiheessa	$0E	Efektin 2 kierrosten määrä 1. vaiheessa
$0F	Efektin 2 kierrosten määrä 2. vaiheessa	$0F	Efektin 2 kierrosten määrä 2. vaiheessa
$10	Viivekierrosten määrä ennen efektin 2 alkamista	$10	Viivekierrosten määrä ennen efektin 2 alkamista
$11	Määrittelee instrumentin toiminnan efekti 2:sen 4. vaiheen päätyttyä	$11	Määrittelee instrumentin toiminnan efekti 2:sen 4. vaiheen päätyttyä
$12 - 13	Efektin 2 pulssinleveyden muutos/kierros 1. vaiheessa	$12 - 13	Efektin 2 pulssinleveyden muutos/kierros 1. vaiheessa
$14 - 15	Efektin 2 pulssinleveyden muutos/kierros 2. vaiheessa	$14 - 15	Efektin 2 pulssinleveyden muutos/kierros 2. vaiheessa
$16 - 17	Pulssin leveys	$16 - 17	Pulssin leveys
$18	**Aaltomuoto/kontrollirekisteri**	**$18**	**Nykyinen taajuus, vähemmän merkitsevä tavu**
$19	**Aluke-/heikentymisaika**	**$19**	**Nykyinen taajuus, enemmän merkitsevä tavu**

TAULUKKO 5. (jatkuu)

$1A	**Soimisvoimakkuus/vaimentumis aika**	**$1A**	**Aaltomuoto/kontrollirekisteri**
$1B	Nuotinkesto ennen vaimentumisvaiheen aloittamista	$1B	Nuotinkesto ennen vaimentumisvaiheen aloittamista
$1C	Aika ennen ”kylmäkäynnistyksen” aloittamista	$1C	Aika ennen ”kylmäkäynnistyksen” aloittamista
$1D	-	$1D	Efektin 1 kierrosten määrä 1. vaiheessa
$1E	-	$1E	Efektin 1 kierrosten määrä 2. vaiheessa
$1F	-	$1F	Efektin 1 kierrosten määrä 3. vaiheessa
$20	-	$20	Efektin 1 kierrosten määrä 4. vaiheessa
$21	-	$21	Efektin 2 kierrosten määrä 1. vaiheessa
$22	-	$22	Efektin 2 kierrosten määrä 2. vaiheessa

Merkittävimmät soittimen ominaisuudet ovat varsinaisen instrumentin parametrit $18 – 1A, jotka määrittelevät soittimen aaltomuodon sekä verhokäyrän ominaisuudet - pulssiaalto vaatii toimiakseen lisäksi arvot muuttujille $16 ja $17. Nykyisen soittimen parametrilla $1A voi muuttaa käytettävän instrumentin aaltomuotoa vauhdissa.

Parametri $1B määrittelee kiinteästi, kuinka kauan kyseisellä soittimella soitettu nuotti soi ennen siirtymistä vaimentumisvaiheeseensa. Tämä ominaisuus on hyödyllinen lyhyitä, perkussiivisia ääniä käytettäessä. Esimerkiksi äänen kestoksi voi antaa nuottidatassa yhden neljäsosan, mutta ääni soisi kuitenkin vain yhden 16-osan ajan ja loput kolme 16-osaa olisivat hiljaisia, ilman että näitä kolmea 16-osataukoa tarvitsisi erikseen sijoittaa nuottidataan. Jos muuttuja $1B saa arvon $00, ei vaimentumisvaiheeseen siirrytä laisinkaan, jolloin soittimen kaikki äänet sidotaan kiinni toisiinsa (legato).

Käskyllä $1C määritellään aika, joka kuluu ”kylmäkäynnistyksen” suorittamisessa nuotin vaimenemisvaiheen päätyttyä (arvo $FF ohittaa toiminnon). Kylmäkäynnistys on tarpeellinen haluttaessa varmistaa

mahdollisimman tarkka verhokäyrän toiminta seuraavalle äänelle (verhokäyrän epätarkkuuksista kts. 2.2.3, s. 33, alaviite 24). Käytännössä tämä tapahtuu ”nollaamalla” oskillaattori ennen seuraavan nuotin syttymistä, eli asettamalla arvo $00 SID–sirun rekistereihin, jotka määrittelevät oskillaattorin taajuuden, pulssinleveyden, aaltomuodon ja verhokäyrän (Kristensen, A. 2005). Kylmäkäynnistys katkaisee siis oskillaattorin toiminnan kokonaan, joten kylmäkäynnistysvaihe aktivoituu vasta edellisen toiminnon päätyttyä ($1B).

Kaikki muut instrumenttien parametrit koskevat taajuuden ja pulssinleveyden muuntelua, joita käsitellään luvuissa 3.2.1.2 ja 3.2.1.3.

3.2.1.2 Efekti 1: taajuusmodulaatio

Taajuusmodulaatio toimii neljässä eri vaiheessa. Jokainen vaihe koostuu muunneltavasta määrästä kierroksia, joiden aikana nuotin taajuus muuttuu annetun arvon mukaisesti (taulukko 6).

TAULUKKO 6. Taajuusmodulaation toiminta

TAPAHTUMA	KIERROSTEN LKM VAIHEESSA	LISÄTTÄVÄ TAAJUUS/KIERROS
Viive ennen efektiä 1	Parametri $0C	-
Modulointivaihe 1	Parametri $08	Parametrit $00 - 01
Modulointivaihe 2	Parametri $09	Parametrit $02 – 03
Modulointivaihe 3	Parametri $0A	Parametrit $04 – 05
Modulointivaihe 4	Parametri $0B	Parametrit $06 – 07
Jatkotoiminta, parametri $0D		

Parametrilla $0D määritellään miten taajuusmodulointi jatkuu 4. vaiheen päätyttyä. Jos parametrin $0D eniten merkitsevä bitti (bitti 7) on päällä, jatkuu modulointi nykyisen instrumentin taulukon modulaatioarvoilla siitä taajuudesta, josta modulointi aloitettiin - tämä viimeisin nuottidatassa annettu taajuus tallentuu aina nykyisen instrumentin taulukon kohtiin $18 ja $19.

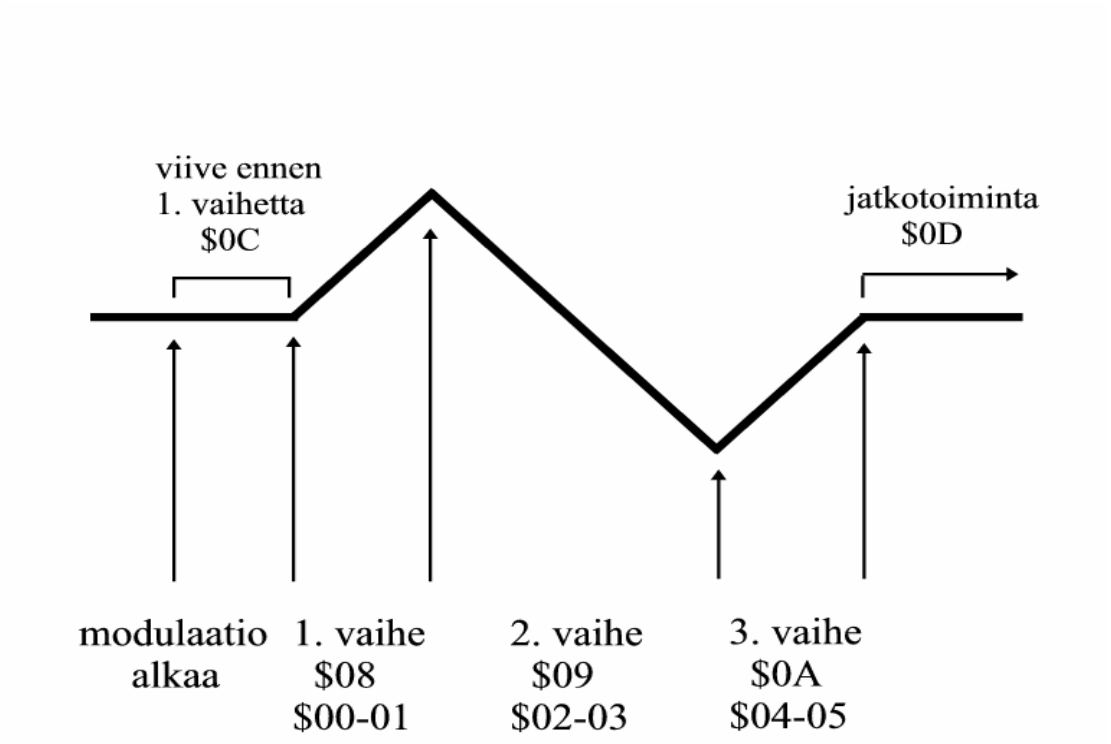
Vähiten merkitsevä bitti (bitti 0) toimii samoin kuin bitti 7, mutta modulointi jatkuu siitä taajuudesta mihin neljäs modulointivaihe sen jätti.

Jos jokin muu kuin eniten tai vähiten merkitsevä bitti parametrissa $0D on kytketty päälle, modulointi lakkaa ja nuotin taajuus palaa nuottidatassa viimeksi annettuun korkeuteen (taajuuteen ennen modulointia).

Esimerkiksi vibrato (kuvio 11) voidaan tuottaa seuraavanlaisilla taajuusmodulaation arvoilla (taulukko 7):

TAULUKKO 7. Esimerkkiarvot taajuusmodulaatiolle

TAPAHTUMA	KIERROSTEN LKM VAIHEESSA	LISÄTTÄVÄ TAAJUUS/KIERROS
Viive ennen efektiä 1	$05	-
Modulointivaihe 1	$02	$10 $00
Modulointivaihe 2	$04	$EF $FF
Modulointivaihe 3	$02	$10 $00
Modulointivaihe 4	$00	$00 $00
Jatkotoiminta $02		



KUVIO 11. Esimerkki taajuusmodulaation toiminnasta

Moduloitava ääni soi ensin viiden kierroksen ajan nuottidatassa annetulla äänenkorkeudella, jonka jälkeen 1. modulointivaihe käynnistyy. Vaiheessa 1 äänenkorkeuteen lisätään arvo $10 kahden kierroksen ajan, vaiheessa 2 taajuus laskee arvolla $10 neljän kierroksen ajan[36] ja kolmannessa vaiheessa ääni nousee kahden kierroksen ajan arvolla $10. Lopuksi parametrin $0D arvo $02 määrittelee, että modulointi loppuu tähän ja ääni jää soimaan alkuperäiseen korkeuteensa.

Taajuusmodulaatiolla voidaan luoda lukematon määrä erilaisia ja hyvin monimutkaisia liikkeitä äänelle varsin vaivattomasti muuttamalla taulukon vaiheiden pituutta, suuntaa ja muutosvoimakkuutta.

Nykyisen instrumentin parametrit $1D – 22 vastaavat varsinaisen instrumentin taulukon arvoja $08 – 0B sekä $0E – 0F. Nykyiselle instrumentille voidaan siis määrittää omat, varsinaisen instrumentin taulukon arvoista poikkeavat efektien kierrosmäärät, jotka otetaan käyttöön kytkemällä bitti 0 tai 7 päälle parametrista $0D.

3.2.1.3 Efekti 2: pulssimodulaatio

Pulssimodulaatio noudattaa täysin samoja periaatteita kuin taajuusmodulaatiokin – nimen mukaisesti äänenkorkeuden sijaan muunnellaankin pulssiaallon pulssinleveyttä. Pulssimodulaatiossa on käytössä kuitenkin vain kaksi vaihetta taajuusmodulaation neljän vaiheen sijaan. Myös pulssimodulaation jatkotoiminnasta vastaava parametri $11 toimii täysin identtisesti verrattuna taajuusmodulaation vastaavaan muuttujaan $0D.

[36] Binäärinen järjestelmä mahdollistaa negatiivisten arvojen ilmaisemisen heksadesimaaliluvuilla käyttämällä ns. kahden komplementin logiikkaa. Positiivinen arvo muutetaan negatiiviseksi muuttamalla kaikki tavun/tavujen bitit käänteisiksi ja lisäämällä “käänteislukuun” arvo $01. Esimerkkitapauksessa heksadesimaalinen lukuarvo $EF $FF vastaa desimaalilukua –16 ja vastaavasti lukuarvo $10 $00 desimaalilukua +16.

3.2.1.4 Suodin

Suotimen asetukset toimivat samalla periaatteella kuin taajuusmodulaatio; jokainen vaihe (4 kpl) koostuu muunneltavasta määrästä kierroksia, joiden aikana suotimen leikkaustaajuus muuttuu annetun arvon mukaisesti. Suotimen jatkotoimintaa käsittelevä parametri toimii myös identtisesti parametrin $0D kanssa.

3.2.1.5 Rumpuäänet

Martin käyttää Arkanoid–pelimusiikeissa myös rumpuääniä (yhteensä 6 erilaista), jotka on ohjelmoitu käyttäen hyväksi SID–sirun jännitevuotoa äänenvoimakkuudensäätimessä. Nämä rumpuäänet koostuvat asteittaisista, hyvin nopeista äänenvoimakkuuden nostoista, jotka aloittavat nousunsa uudestaan matalimmasta arvosta äänenvoimakkuuden saavuttaessa huippunsa. Rumpuäänien soittamista varten tarvitaan erillinen aliohjelma, joten niitä ei voi ohjailla normaalin nuottidatan seasta kuten muita instrumentteja.

Nämä rumpuäänet eivät siis ole digitaalisesti tallennettuja “näytteitä” (eng. sample) oikeista rumpuäänistä, vaan Martinin itsensä ohjelmoimia ääniä[37]. Vaikka rumpuäänien käyttäminen vaatiikin oman aliohjelmansa, ei rumpuäänien käyttö muiden instrumentin kanssa samaan aikaan tuota ongelmia - Arkanoid–pelimusiikissa yksi raita on varattu kokonaan pelkästään rumpuäänien soittamiseen kahden muun kanavan käyttäessä normaaleja instrumentteja.

Martin itse ei pidä Arkanoid–peliin luomiaan rumpuääniä kovin kummoisina tuotoksina ja mainitsee, että myöhemmin samana vuonna hän sisällytti peliin Game Over (Imagine, 1987) hienostuneempia, oikeasti digitoituja rumpunäytteitä (Interview With Martin Galway 2005). Omien sanojensa mukaan hän ei osannut hyödyntää SID–sirua rumpuäänien tuottamiseen ohjelmallisesti kuten Rob Hubbard, vaan suosi näytteiden käyttöä rytmisten elementtien luomiseksi (Sánchez, C. 2003).

---

[37] Haastattelussaan (Interview With Martin Galway 2005) Martin myöntää varastaneensa idean rumpuäänien luomiseen äänenvoimakkuussäädöillä Digidrum–ohjelmasta. Ohjelman sisältämät rumpuäänet olivat digitoituja näytteitä rummuista, joita hän imitoi omalla ohjelmointitekniikallaan.

Arkanoid–pelin käyttämät ääniefektit on myös luotu käyttämällä musiikkirutiinia. Erotuksena musiikin ohjelmoinnista ääniefektit vaativat vain tiedon instrumentin ominaisuuksista ja lisäksi kaksi tavua, jotka määrittävät ääniefektin taajuuden – ääniefektin kesto määräytyy instrumentin modulaatioefektin pituuden mukaan.

3.2.2 Yhteenveto Arkanoid-musiikkirutiinista

Martin Galwayn ohjelmointirutiini sisältää monia suorastaan nerokkaita oivalluksia. Kaikkein kirkkain oivallus on instrumenttien ohjelmointikäytäntö – jatkuvasti muuttuvien pienten nyanssien ohjelmoiminen musiikkiin on vaivatonta kahta instrumenttitaulukkoa käyttämällä. Samoin modulointiefektien (taajuus-, pulssinleveys. ja suodinefekti) muuntelumahdollisuudet ovat todella laajat ja mahdollistavat hyvin erilaisten sointivärien ja vivahteiden tuottamisen (vibrato, portamento, glissando, legato jne.) vaivattomasti *nuottidatan seassa* annettavilla käskyillä. Myös nuotinpituuksien antaminen absoluuttisella tai suhteellisella arvolla on kätevä tapa tuottaa polyrytmejä sekä tempon vaihdoksia.

Martinin tapa kirjoittaa identtiset ominaisuusaihiot jokaiselle kanavalle erikseen vie paljon muistitilaa mutta tuottaa selkeämmin hahmotettavan lähdekoodin, jonka muuntelu on siten myös nopeampaa.

Tracker–periaatteella toimivien ohjelmien etuna on kappaleen rakenteen vaivaton hahmottaminen kuvioiden avulla. Arkanoid–musiikkirutiinissa rakenteen muuntelu tapahtuu hyppykomennoilla nuottidatan eri kohtiin. Hyppykomentojen käyttö on hieman monimutkaisempaa kuin kuvioihin viittaaminen, mutta ohjelma suoriutuu nopeammin konekielisten hyppykäskyjen suorittamisesta kuin erikseen määriteltyihin kuvioihin viittaamisesta (kuvioiden aiheuttamasta viiveestä kts. 3.1.2, sivut 51-52).

Musiikkirutiini ottaa myös huomioon 6581–sirumalliin liittyvän verhokäyrän epätäsmällisyysongelman mahdollistamalla ”kylmäkäynnistyksen” käytön tarvittaessa jokaiselle syttyvälle äänelle.

Oikeastaan ainoat puutteet Martin Galwayn musiikkirutiinissa ovat kehämodulaatio- ja synkronisointiefektien jättäminen pois käskytaulukosta, sekä varsinaisten rumpuinstrumenttien puuttuminen. Jälkimmäisen ongelman

Martin siis ratkaisi käyttämällä rumpuääniin sample-äänien soittamisessa käytettävää ohjelmointitekniikkaa (kts. 3.2.1.5, s. 65). Muilta ominaisuuksiltaan rutiini on varsin monipuolinen ja hyvin käyttökelpoinen erilaisissa musiikin tyylilajeissa.

Kokonaisuudessaan Arkanoid–musiikkirutiini on hyvin virtaviivainen ja ennen kaikkea muuntautumiskykyinen ohjelma - koko musiikkirutiinia (lukuunottamatta rumpuääniä) voi ohjailla pelkästään raidoille sijoitettavalla datalla. Tämä ajatusmalli erottaa Martin Galwayn rutiinin Matt Grayn käyttämästä, enemmän tracker –ohjelmaa muistuttavasta musiikkirutiinista.

3.3 Rob Hubbard

Rob Hubbardin taustoja koskeva tieto on hankittu kokonaisuudessaan hänen epävirallisen kotisivunsa "The Complete Works Of Rob Hubbard" (The Complete Works Of Rob Hubbard 2005) *kautta.*

Ensimmäiset Commodore 64:lle ohjelmoidut pelit, kuten myös pelimusiikit, hakivat vielä muotoaan ja kokeilivat rajojaan uuden tietokoneen suorituskyvyn puitteissa 1980-luvun alkupuolella. Pelimusiikkien taso kuitenkin revähti kertaheitolla uudelle asteelle brittiläisen Rob Hubbardin siirtyessä opetusohjelmien koodaamisesta pelimusiikkien säveltäjäksi vuosien 1984-85 aikana. Hänen ensimmäinen hittituotteensa oli musiikki peliin Thing On A Spring (Gremlin Graphics) vuonna 1985, ja lopullinen todistus hänen lahjoistaan, sekä säveltäjänä että musiikin ohjelmoijana, tulivat ilmi jo samana vuonna julkaistussa pelissä Monty On The Run (Gremlin Graphics).

Seuraavat kolme vuotta Rob hallitsi pelimusiikin kenttää suvereenisti laadullaan[38] ja työtilauksillaan – hänen musiikkinsa päätyivät yhteensä 73:een C64 –peliin (puhumattakaan muista tietokonemerkeistä) ennen kuin hän siirtyi vuonna 1988 freelancer–säveltäjän töistä EA–peliyhtiön vakituiseen palvelukseen Yhdysvaltoihin. Varsinkin hänen rytminen osaamisensa ja SID–sirulla tuotetut rumpuäänet ansaitsevat kunniamaininnan (Sánchez, C. 2003).

[38] Tietokonelehdessä Zapp! 64 julkaistiin vuosina 1986-89 listaa lukijoiden parhaaksi äänestämistä pelimusiikeista, ja Hubbardin tekemä sävellys löytyy jokaisesta listasta sijalta 1. Myös Martin Galwayn nimi löytyy jokaisesta julkaistusta Top Ten –listasta ja Matt Grayn musiikki peliin Driller nousi parhaimmillaan sijalle 3 viiden listoilla viettäneen kuukautensa aikana.

Hänen vahva osaamisensa pelimusiikkien ohjelmoijana perustui musiikinopiskelun kautta saavutettuun tietotaitoon, mutta myös kypsään ja monimuotoiseen ajatteluun ohjelmointitekniikoissa[39]. Hän oli myös suhteellisen vanha (29-vuotias) tullessaan hektisesti kasvaville 80-luvun kotitietokonemarkkinoille mukaan. Commodore 64:n aikanaan helppo käyttöliittymä ja hyvät ominaisuudet pelejä ajatellen imivät mukaansa paljon nimenomaan nuoria tietokoneharrastajia, jotka kehittivät hyvin lyhyessä ajassa täysin omatoimisesti taitonsa ammattimaiselle tasolle ja siirtyivät kiperästi osaavia tekijöitä kaipaavan peliteollisuuden palvelukseen – vanhoja osaajia kun ei varsinaisesti ollut olemassakaan.

Hakusanat *rob*, *hubbard*, *c64* ja *music* tuottavat pelkästään Google–hakukoneella 4 620 osumaa ja hänelle on omistettu jopa kokonainen Internet–sivusto, “The Complete Works Of Rob Hubbard”, osoitteessa www.robhubbard.co.uk. Myös lukuisia hänen sävellyksiään sisältäviä cd-levyjä on julkaistu viimeisten vuosien aikana, joita voi tilata osoitteista www.binaryzone.co.uk, www.c64audio.com, www.pressplayontape.com ja www.livet.se/visa. Game Audio Network Guild palkitsi Rob Hubbardin elämäntyöstään tietokonemusiikin parissa vuoden 2004 maaliskuussa ja epäilemättä seuraava suuri hetki hänen urallaan on hänen säveltämänsä International Karate (System 3, 1986) –pelimusiikin pohjalta tehty sovitus sinfoniaorkesterille, jonka ensiesittää kapellimestari Andy Brickin johdolla The FILMharmonic Orchestra Prague.

3.3.1 Monty On The Run –musiikkirutiini

Rob Hubbardin soittorutiinin on purkanut ja selittänyt Anthony McSweeney internet -artikkelissaan ”Rob Hubbard’s Music: Disassembled, Commented and Explained by Anthony McSweeney” (MacSweeney, A. 1993). *Kuviot, taulukot, johtopäätökset musiikkirutiinin toimivuudesta käytännössä sekä yhteenvedon sisältö ovat omaa käsialaani, ellei toisin ole mainittu. Monty On The Run –pelimusiikki löytyy liitteenä olevalta cd-levyltä (SIDCD 6).*

[39] Rob ohjelmoi myös yhden pelin uransa alkuaikoina; Robin suurimmilta osin ohjelmoima, perheen pienimmille suunnattu Razzmatazz–peli ei kuitenkaan ollut myyntimenestys ja pelin julkaissut firma ajautui konkurssiin pian pelin julkaisun jälkeen.

Rob Hubbardin käyttämän ohjelmointirutiinin suurin musiikkidatan yksikkö on ns. moduuli, joka sisältää kaikki pelin käyttämät kappaleet yhdessä datanipussa - itse ohjelmointirutiini ja soittimet taas ovat erillään nuottidatasta. Tämä mahdollistaa soittorutiinin ja instrumenttien vaivattoman muuntelun, sekä nuottitiedon siirtämisen vaikkapa levykkeen muistiin tarvitsematta liittää itse soittoohjelmaa mukaan.

Kuten Matt Grayn käyttämä musiikkirutiini, myös Hubbardin rutiini jakautuu kappaleiden sisällä kolmeksi raidaksi, jotka taas koostuvat varsinaisen nuottidatan sisältämistä kuvioista. Jokainen raita päättyy joko komentoon $FF tai $FE. Komento $FF käskee ohjelmaa toistamaan raidan ja $FE vastaavasti lopettamaan musiikin soiton raidan loppuessa. Samoin kuvion loppuminen ilmaistaan käyttämällä arvoa $FF nuottidatassa, jolloin ohjelma siirtyy seuraavaan kuvioon.

Jokainen kuviossa oleva nuotti vaatii soidakseen tiedot nuotinpituudesta, soittimen numerosta ja nuotin taajuudesta. Nämä välttämättömät tiedot annetaan juuri tässä järjestyksessä nuottidataan.

Ensimmäinen nuotin tarvitsema tavu määrittelee siis nuotinpituuden, joka voi saada arvon väliltä $00 – 1F (32 eri vaihtoehtoa)[40]. Nuotinpituuden määrittävän tavun kolme eniten merkitsevää bittiä toimivat seuraavilla tavoilla:

TAULUKKO 8. Kolmen ylimmän bitin toiminta nuotinpituuden ilmoittavassa tavussa

BITIN NRO - ASENTO	**TOIMINTA**
5 - 0	Nuotinpituuden loppuessa vaimenemisvaihe käynnistetään
5 - 1	Nuotti sammuu ilman vaimenemisvaihetta
6 - 0	-
6 - 1	Nuotti sidotaan edelliseen nuottiin (ei aluketta)
7 - 0	Seuraava tavu on nuotin taajuus
7 - 1	Seuraava tavu on soittimen nro tai portamento

[40] Monty On The Run –pelin pääteeman (SIDCD 4) tempoksi on rutiinissa annettu arvo $01. Ohjelma vähentää annetusta nuotinpituudesta luvun 1 joka toisella ruudun päivityskerralla (25 Hz PAL), joten pisin mahdollinen yksittäisen nuotin kestoaika ($1F) on 32/25 Hz = 1,28 s. Nuotteja voi tietenkin sitoa toisiinsa ilman aluketta, jolloin yhden äänen nuotinpituus voi olla käytännössä miten suuri tahansa.

Nuotinpituuden määrittävää tavua seuraa joko taajuuden määrittävä tavu tai instrumentin/portamenton määrittävä tavu - riippuen bitin 7 asennosta edellisessä tavussa. Jos bitti 7 on päällä ensimmäisessä tavussa, niin toinen tavu voi määritellä seuraavat ominaisuudet äänelle:

TAULUKKO 9. Instrumentti/portamento -tavun toiminta

BITIN NRO - ASENTO	**TOIMINTA**
7 - 0	Tavu on instrumentin numero
7 - 1	Tavu on portamenton muutosarvo (bitit 1 – 6)
0 - 0	Portamenton arvo on positiivinen
0 - 1	Portamenton arvo on negatiivinen

Koska bitti 7 on varattu määrittelemään tavun laatu, jää instrumenttinumeroille bitit 0 – 6, eli rutiinissa on mahdollisuus käyttää instrumenttinumeroja 0 – 127. Portamenton muutosarvon taas määrittelevät bitit 1 – 6 (arvot 1 –64), sillä bitti 0 on varattu liu'un suunnan määrittämiseen. Portamentotavun arvo lisätään tai vähennetään sellaisenaan nuotin kaksitavuisen taajuusarvon vähemmän merkitsevästä tavusta, kunnes nuotinpituus on kulunut loppuun.

Seuraava tavu määrittelee nuotin taajuuden. Taajuuden määrittävä tavu on siis joko toinen tai kolmas komentojonossa riippuen seitsemännen bitin asennosta ensimmäisessä, nuotinpituuden määrittävässä tavussa.

Musiikkirutiinin käyttämät taajuudet on määritelty erillisessä taulukossa tasavireistä järjestelmää vastaaviksi nuotinkorkeuksiksi (A=440Hz), kuten aikaisemmin käsitellyissä rutiineissakin. Matalin taajuus saadaan aikaiseksi käyttämällä nuottidatassa arvoa $00, ja ylin saadaan arvolla $5F. Taajuudet ulottuvat siis matalimmasta C–nuotista pianon koskettimistolla (subkontraoktaavi) 95 puolsävelaskelta ylöspäin C–ääneen, joka jää pianon koskettimiston ulkopuolelle (kuusiviivainen C). Tauko voidaan tuottaa millä tahansa arvolla, joka on suurempi kuin $5F.

Seuraava tavu komentojonossa tarkoittaisi joko uutta nuotinpituutta, tai komentoa siirtyä seuraavaan kuvioon raidalla (arvolla $FF).

Esimerkiksi komentojonon peräkkäiset arvot $89, $01 ja $36 tuottaisivat nuotin C-3 pituudella 9 käyttäen instrumenttia numero 1. Arvoilla $87, $83, $40 ja $FF ohjelma soittaisi nopeudella 2 alaspäin liukuvan nuotin E-3, jonka pituusarvo on 7. Tämän jälkeen ohjelma siirtyisi seuraavaan kuvioon samalla raidalla.

3.3.1.1 Instrumentit

Kukin instrumentti koostuu kahdeksan tavun suuruisesta taulukosta, joka määrittelee kaikki soittimen ominaisuudet.

TAULUKKO 10. Monty On The Run –musiikkirutiinin instrumenttitaulukko

TAVU	TOIMINTA
$00	Pulssin leveys, vähemmän merkitsevä tavu
$01	Pulssin leveys, enemmän merkitsevä tavu
$02	Aaltomuoto
$03	Aluke/heikentymisaika
$04	Sointivoimakkuus ja vaimenemisaika
$05	Vibraton syvyys
$06	Pulssipyyhkäisyn nopeus
$07 bitti 0 bitti 1 bitti 2	Efektitavu Rumpuääni Taajuuden hidas pudotus Oktaaviarpeggio

Jos aaltomuodoksi ($02) on valittu pulssiaalto, ilmaistaan pulssin leveys käyttämällä parametreja $00 ja $01. Tavut $03 ja $04 määrittelevät verhokäyrän arvot instrumentille. Muut ominaisuudet käsitellään omissa alaluvuissaan.

3.3.1.2 Vibrato

Vibratorutiini on varsin vaatimaton edellisiin ohjelmointirutiineihin verrattuna. Vibrato koostuu yksinkertaisesti nousevasta ja laskevasta kahdeksan askeleen jaksosta lukuarvojen 0 – 3 välillä (0,1,2,3,3,2,1,0). Soivan nuotin taajuuteen lisätään vibratotaajuus niin monta kertaa kuin kulloisenkin askeleen lukuarvo on. Lisättävän vibratotaajuuden arvo taas määräytyy instrumenttitaulukon parametrin $05 mukaan.

Vibraton nopeus määräytyy näyttöruudun virkistystajuuden mukaan kuten nuotinpituuskin, muutos tapahtuu jokaisella ruudun päivityskerralla (50 Hz). Kun nuotin kestoaikaa on jäljellä vähemmän kuin kahdeksan kierrosta, eli yhden vibratojakson pituus, vibrato kytkeytyy pois päältä. Tällä tavoin varmistetaan, että nuotti alkaa ja loppuu samalla taajuudella.

Vibratotaajuus ei ole absoluuttinen taajuusarvo, vaan suhteellinen arvo, joka riippuu soivan nuotin taajuudesta. Vibratorutiini kaksinkertaistaa ensin soivan nuotin taajuuden ja laskee tämän uuden taajuuden ja sitä puolsävelaskelta ylempänä olevan taajuuden erotuksen. Tämä erotus taas jaetaan kahdella *niin monta kertaa kuin parametrissa $05 annetun luvun arvo on* - parametrin $05 arvo toimii siis käänteisesti. Tästä seuraa, että suurin yksittäinen lisättävä vibratotaajuus on vain 1/8-sävelaskelta, jolloin vibraton amplitudi on suurimmillaankin vain 3/8–sävelaskelta.

Mainittakoon, että tulkintani vibraton amplitudin muodostumisesta *saattaa olla* virheellinen. Joka tapauksessa yksittäisen lisättävän taajuuden täytyy olla hyvin pieni, koska jaksossa on askelia vain kahdeksan, ja suurin mahdollinen askeleen sisältämä kerroin on kolme. Monty On The Run –kappaleessa käytettyjen vibratojen hyvin pienet amplitudit tukevat tätä käsitystäo. Jos lisättävä taajuus olisi kovin suuri, askeleet olisivat helposti kuultavissa erillisinä nuotteina, jolloin vaikutelma muuttuu vibratosta hyvin nopeaksi arpeggioksi.

3.3.1.3 Pulssipyyhkäisy

Parametrin $06 arvo määrittelee luvun, joka lisätään jokaisella pyyhkäisyn askeleella sellaisenaan pulssinleveyden määrittävän rekisterin enemmän merkitsevään tavuun. Askelten suoritusnopeus on sama kuin

vibratorutiinissakin. Saapuessaan ääriarvoihinsa (enemmän merkitsevän tavun arvot $0E tai $08) pulssipyyhkäisy muuttaa suuntaansa. Rutiinin käyttämät pulssinleveyden raja-arvot eivät ole suurin ja pienin, jotka SID pystyisi tuottamaan. Näiden todellisten äärirajojen käyttämistä kuitenkin vältetään (vrt. Driller-rutiini, 3.1.1.1, sivut 46-47), sillä käynti rajalla tuottaa ikävän “törähdyksen” ääneen tai ohuimmillaan ääni saattaa kadota hetkeksi kuuluvista.

3.3.1.4 Efektitavu

Instrumenttitaulukon kahdeksas tavu ($07) määrittelee mitä efektejä soitin käyttää. Jos bitti 0 on ylhäällä, soitin on rumpuääni. Rumpuefektiä käytettäessä soittin syttyy 1/50 sekunnin ajan kohina-aaltomuodolla, jonka jälkeen aaltomuoto vaihtuu instrumenttitaulukon parametrissa $02 määritellyksi. Jos aaltomuodolle on annettu arvo $00 instrumenttitaulukossa, käytetään kohina-aaltoa koko ajan[41]. Efekti vähentää samaan aikaan arvon $01 taajuusrekisterin enemmän merkitsevästä tavusta 50 kertaa sekunnissa.

Käytännössä rumpuäänen taajuus laskee niin nopeasti, ettei mitään yksittäistä taajuutta ole mahdollista erottaa. Kohina-aaltomuoto rumpuäänien alukkeessa simuloi kohtuullisen hyvin lyömäsoittimien aluketta. Rumpuefektin lisäksi verhokäyrän aluke- ja heikentymisaika kannattaa asettaa hyvin pieniksi, mikä vastaa perkussiivisten soitinten alukkeiden ominaisuuksia.

Bitin numero 1 ollessa päällä instrumentilla soitetut äänet liukuvat aina hitaasti alaspäin. Tämä negatiivinen portamento on toteutettu vähentämällä arvo $01 taajuuden määrittävän rekisterin enemmän merkitsevästä tavusta joka toisella näyttöruudun päivityskerralla (25 Hz).

Bitin 2 ollessa ylhäällä instrumentilla soitetut äänet soivat joka toisella ruudun päivityskerralla oktaavia ylempää. Efekti on siis ikäänkuin hyvin yksinkertainen oktaaviarpeggio, jolla soittimen ääni saadaan paksumman kuuloiseksi.

[41] Monty On The Run –kappaleessa Hubbard käyttää pulssiaaltoa yhdistettynä rumpuefektiin, ja pelkkää kohina-aaltomuotoa imitoidessaan muita rumpusetin osia.

3.3.2 Yhteenveto Monty On The Run -musiikkirutiinista

Rob Hubbardin käyttämä musiikkirutiini on tutkielmassa käsitellyistä ohjelmointirutiineista kaikkein suoraviivaisin ja pienikokoisin (vain n. 1 000 tavua), mutta silti se sisältää käytännössä kaikki samat ääniominaisuudet kuin muutkin analysoidut ohjelmointirutiinit; pulssipyyhkäisy, portamento, legato, vibrato, oktaaviarpeggio ja rumpuäänet. Tosin toiminnaltaan nämä äänenominaisuuksia muokkaavat rutiinit ovat hyvin yksinkertaisia eikä efektien/instrumenttien ominaisuuksien muuntelu onnistu suoraan nuottidatassa annettavilla komennoilla.

Joka tapauksessa ne tuottavat hyvin käyttökelpoisia äänenvärejä ja kestävät kulutusta mainiosti, vaikka kalpenevatkin vertailussa esimerkiksi Matt Grayn ja Martin Galwayn käyttämiin modulaatiorutiineihin – varsinkin rumpuäänet toimivat hämmästyttävän hyvin yksinkertaisesta koodausperiaatteestaan huolimatta (vrt. Galwayn rumpuääniin; SIDCD 4). Samoin portamenton tuottaminen onnistuu Hubbardin rutiinilla hyvin helposti verrattuna Galwayn käyttämään varsin monimutkaiseen taajuusmodulaatiorutiiniin.

Instrumentit koostuvat ainoastaan kahdeksan tavun kokoisista parametritaulukoista, mutta näillä tavuilla pystytään määrittelemään kaikki samat äänenvärit kuin monimutkaisella Galwayn rutiinillakin lukuunottamatta monipuolisempia taajuusmodulaatioita.

Mielenkiintoista on, että Hubbard ei ole sisällyttänyt suodinta kyseiseen musiikkirutiiniinsa lainkaan[42]. Ainoa syy tälle voisi käytännössä olla suotimen epäluotettavuus – ohjelmointitaidon tai muistitilan puutteesta tuskin oli kyse. Toinen syy voisi olla hektinen tahti, jolla uusia pelejä tuotiin markkinoille vuosina 1985-87, jolloin kaikkien ohjelman ominaisuuksien viimeistelyyn ei yksinkertaisesti jäänyt aikaa. Samoin instrumenttitaulukoista ei löydy mahdollisuutta lisätä ääneen synkronisointi- tai kehämodulaatioefektiä. Kuitenkin jo samana vuonna säveltämissään pelimusiikeissa Hubbard käyttää suodinta, kehämodulaatiota sekä muita todella omaperäisiä efektejä hyvinkin

[42] Monty On The Run –kappaleen alun urkupisteäänessä on kuitenkin kuultavissa asteittain muuttuva suodinefekti. Todennäköisesti kyseessä on pelin ääniefektejä varten luotu ominaisuus (tai pulssipyyhkäisy), joka ei siis ole osa varsinaista musiikkirutiinia, eikä siten myöskään kuulu analyysien aihepiiriin (kts. 1.4, s. 13).

paljon (esim. Crazy Comets; Martech 1985 (SIDCD 7), Formula 1 Simulator; Mastertronic 1985, The Master Of Magic; MAD/Mastertronic 1985).

Mitä todennäköisimmin Hubbard on sijoittanut suotimen sekä monia muita uusia efektejä ja äänenvärejä juuri instrumenttitaulukon efektitavuun $07, jotka on voinut kutsua käyttöön kytkemällä tietyt bitit ylös efektitavusta. Uudet ominaisuudet on voinut siten ohjelmoida itsenäisiksi aliohjelmiksi muuttamatta varsinaista musiikkirutiinia mitenkään.

Hubbardin rutiini koostuu siis hyvin harkituista, helposti yhdisteltävistä palasista, jotka tuottavat monipuolisia äänenvärejä. Koko ohjelman toimintaperiaate on hyvin raiturimainen ja selkeä. Rutiinin vahvuudet ovatkin ennen kaikkea tehokkuudessa ja suoritusnopeudessa. Parhaimmillaan Hubbard ohjelmoi jopa puolessa vuorokaudessa kaikki yksittäisen pelin (esim. Commando; Elite 1985) sisältämät musiikit (MacKenzie 1997).

3.4 Chris Hülsbeck

Chris Hülsbeckin henkilö- ja työhistoriaa koskeva tieto on kerätty hänelle tehdystä haastattelusta (Interview With Chris Hülsbeck 2005), *Wikipedia–verkkotietosanakirjan artikkelista* (Chris Hülsbeck 2005) *sekä hänen omalta Internet-kotisivultaan* (Hülsbeck, C. 2005).

Saksalaisen Chris Hülsbeckin musiikkirutiinin valitseminen lähempään tarkasteluun tutkielmassani oli käytännössä välttämätöntä. Hänen ohjelmoimallansa MusicMaster–raiturilla (versiosta 1.0 eteenpäin SoundMonitor) oli epäilemättä valtava vaikutus musiikkiohjelmien käyttöjärjestelmien muokkautumiseen 1980-luvun lopulla, vaikka samaa raituriperiaatetta käyttävä Real-Time Composer -ohjelma Fairlight CMI II –musiikkityöasemalle olikin ilmestynyt jo vuonna 1982 - Fairlight Computer Music Instrument II –työasema maksoi nimittäin ilmestyessään n. 25 000 $, kun SoundMonitor–ohjelman taas pystyi hankkimaan näpyttelemällä lähdekoodin muutaman Saksan markan arvoisesta 64’er–tietokonelehdestä Commodore 64 –koneen muistiin (Fairlight 2005 & Holmes, G. 2005).

SoundMonitor–ohjelma sisältää graafisen käyttöliittymän, jota on helppo käyttää suoraan tietokoneen näppäimistöltä ja sisälsi jopa mahdollisuuden

tallentaa SID–ääntä reaaliaikaisesti. Chrisin koodaama ohjelma aloitti aallon vastaavien tracker –ohjelmien tuotannossa, joista ensimmäinen kaupallinen versio ilmestyi Amiga 500 –kotitietokoneelle vuonna 1987 Karsten Obarskin ohjelmoimana. Varsinainen kulta-aika näille helppokäyttöisille, nopeille ja suhteellisen monipuolisille musiikkiohjelmille oli 80- ja 90-lukujen taite.

Chris Hülsbeck syntyi musikaaliseen perheeseen ja aloitti pianonsoiton jo viiden vuoden iässä. Teini-iässä hän sai ensimmäisen rumpusettinsä, MIDI-syntetisaattorin ja Commodore 64 -kotitietokoneen, jolla hän työsti ensimmäisen pelimusiikkinsa ystävänsä ohjelmoimaan peliin. Vuonna 1986, ollessaan 18-vuotias, Chris lähetti C64-sävellyksensä “Shades” (SIDCD 8) saksalaisen 64’er–tietokonelehden sävellyskilpailuun. Hänen kappaleensa voitti kilpailun, ja jo seuraavan kuukauden lehdessä esiteltiin hänen käyttämänsä musiikkieditori SoundMonitor.

Näiden julkisuudessa vietettyjen hetkien jälkeen hän pääsi töihin Rainbow Arts –peliyhtiöön ja on siitä lähtien työskennellyt monille eri yhtiöille tietokoneella sävelletyn musiikin ja peliohjelmoinnin parissa. Tällä hetkellä hän tuottaa musiikkia Factor 5 –firman tuotteisiin. Hän on ollut perustajäsen myös kahdessa yrityksessä, KAIKO ja synSONIQ, joiden kautta hän on julkaissut omaa musiikkiaan yhteensä yhdentoista albumin ja kahden singlejulkaisun verran.

Elokuussa vuonna 2004 järjestetyn GC Games Convention –tapahtuman yhteydessä The FILMharmonic Orchestra Prague esitti Chris Hülsbeckin Turrican–peleihin säveltämiä musiikkeja Andy Brickin johdolla. Chris Hülsbeckin tuotannosta ja taustoista löytyy paljon tietoa lukuisilta Internet–sivustoilta, joihin kannattaa hakeutua hänen virallisten kotisivujensa kautta osoitteesta www.huelsbeck.com.

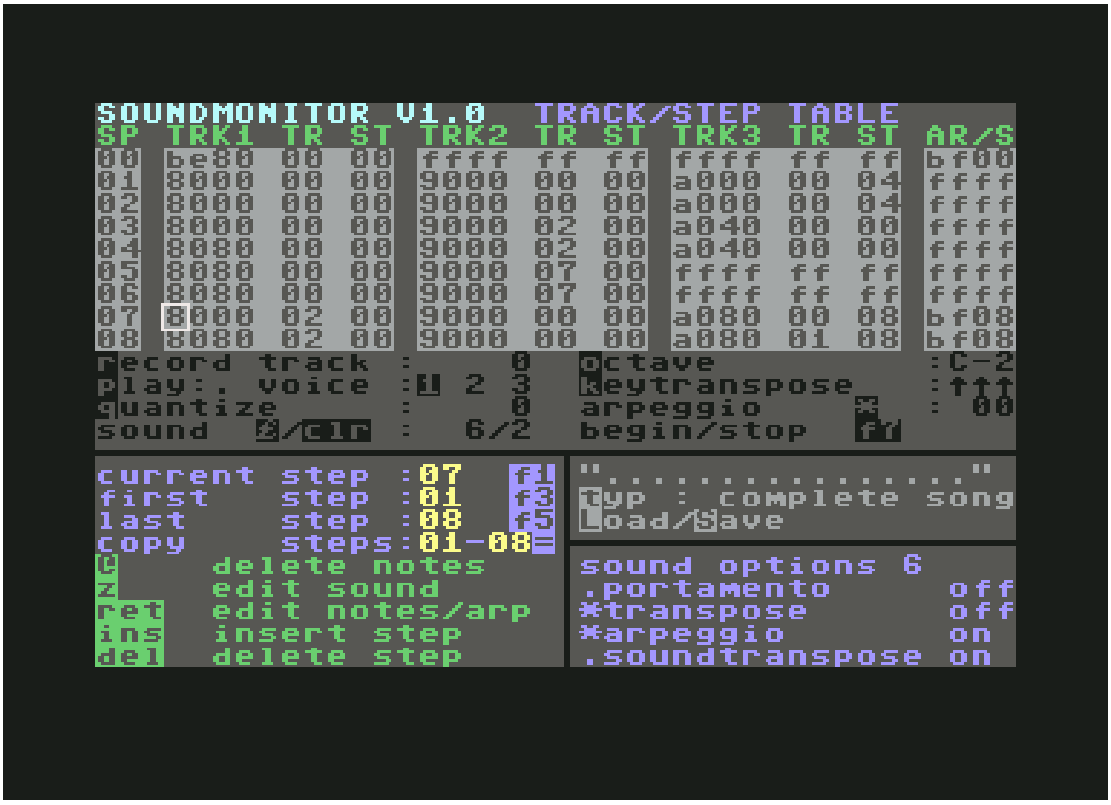
3.4.1 SoundMonitor

Chris Hülsbeck on itse antanut ohjeet SoundMonitor–ohjelman käyttämiseen lähdekoodin julkaisemisen yhteydessä 64’er–lehdessä (Hülsbeck, C. 1986). Ohjeet löytyvät uudelleen julkaistuna Internet-osoitteesta www.softwolves.pp.se/idoc/read/soundmonitor_de. *Tämän lisäksi olen itse käyttänyt ohjelmaa ja kokeillut sen mahdollisuuksia käytännössä.*

SoundMonitor–ohjelman analyysi ei ole rakennettu tietystä pelistä irroitetun musiikkikoodin pohjalta kuten edelliset analyysit. Tämän takia en viittaa mihinkään tiettyyn pelimusiikkiin kyseisessä luvussa. Liitteenä olevalta cd-levyltä löytyy kuitenkin esimerkki Chris Hülsbeckin sävellystaidoista (SIDCD 9: Grand Monster Slam [Rainbow Arts, 1989]).

SoundMonitor–musiikkieditori koostuu neljästä erilaisesta toimintaruudusta; raituri-, arpeggio/synkronisointi-, tahti- ja instrumenttiruutu. Raituriruudun kautta käsitellään kappaleen rakennetta, soittimia ja nuottitietoa kokonaisuutena. Tahtiruudussa määritellään tahtien sisältämät nuotit sekä nuottien käyttämät soittimet ja efektit (arpeggio, portamento). Instrumenttiruudun asetuksilla määritellään soittimien ominaisuudet ja arpeggio/synkronisointiruudussa kappaleen perusasetukset (tempo, tahtin pituus, yleisvoimakkuus) sekä arpeggiotaulukot. Kukin toimintaruutu käsitellään erikseen omassa alaluvussaan.

3.4.1.1 Raituriruutu



KUVIO 12. SoundMonitor–ohjelman raituriruutu

Raituriruutu on jaettu viiteen sarakkeeseen, joista vasemmanpuoleisin (SP = STEP; eng., askel) kertoo askelnumeron, eli soitettavan tahdin järjestysnumeron. TRK 1, TRK 2 ja TRK 3 (TRK = TRACK; eng., raita) tarkoittavat kappaleen ääniraitoja - yksi raita vastaa yhden oskillaattorin toimintaa. Jokainen raita koostuu tahdeista (askelista), joiden sisältö määräytyy TRK–tekstin alla olevan nelinumeroisen heksadesimaaliluvun mukaan – kyseinen luku ilmoittaa sen muistipaikan osoitteen, joka sisältää tahdin ensimmäisen iskun.

Jokainen raidan sisältämä tahti voidaan transponoida muuttamalla otsikon TR (TRANSPOSE; eng., transponoida) alla olevaa heksadesimaalista lukuarvoa. Arvot $01 – 7F transponoivat *kaikkia tahdin sisältämiä säveliä* ylöspäin puolsävelaskeleen välein, ja arvot $FF – 80 vastaavasti alaspäin puolsävelaskeleen välein ($01 = +1, $02 = +2, $FF = -1, $FE = -2 jne). Samoin tahdissa käytettyihin instrumentteihin viittaavia numeroita voidaan ”transponoida” ylöspäin (ST = SOUNDTRANSPOSE; eng., äänen transponointi) - yksi luku lisää ST–sarakkeen arvossa vaihtaa tahdissa käytetyt instrumentit yhtä numeroa suuremmiksi[43].

AR/S–tekstin (ARPEGGIO/SYNCHRONISATION; eng., arpeggio/synkronisointi) alapuolella oleva sarake näyttää nelinumeroisen muistipaikan osoitteen, joka sisältää joko tiedot kappaleen perusasetuksista tai kyseisessä tahdissa käytetyn arpeggiotaulukon. AR/S–sarakkeen kautta pääsee erilliseen ruutuun, jonka kautta perusasetuksia ja arpeggiotaulukoita voi muunnella. Arpeggio/synkronisointiruutu käsitellään omassa alaluvussaan 3.4.1.2.

Raitojen alapuolella olevassa valikko-osiossa määritellään reaaliaikaisessa soitossa ja nauhoituksessa käytettävät asetukset. Nuottien syöttämiseen reaaliaikaisesti käytetään tietokoneen näppäimistöä, joka on ohjelmoitu jäljittelemään pianon koskettimiston asettelua. Ohjelma sijoittaa käynnistyessään tempon sisältävän kohotahdin muistipaikkaan $BE80. Käytettäessä reaaliaikaista nauhoitusta kannattaa kohotahti sijoittaa nauhoitettavan osion ensimmäiseen askeleeseen, jolloin tietokone laskee kappaleen tempon ennen varsinaisen nauhoituksen alkamista.

---

[43] Tahtiruudussa (kuvio 14) määritellään jokaiselle nuotille soitin arvoilla $01 – 0F (oikeanpuoleinen numero efektitavusta, kts. 3.4.1.3), mutta soittimia voidaan luoda yhteensä $FF kappaletta. SOUND TRANSPOSE –ominaisuus on siis välttämätön, jotta instrumentteja väliltä $10 – FF voidaan käyttää.

record track – valitsee raidan, jolle äänitetään (1 – 3)

play voice – valitsee raidan/oskillaattorin, jota käytetään koskettimistolta soitettaessa (ei koske äänitystä)

quantize – määrittelee kvantisoinnin nuottiarvon, eli äänitetyn rytmin pienimmän mahdollisen aika-arvon (0 – ei kvantisointia, 1 – 16-osa, 2 – 8-osa, 3 – neljäsosa)

sound – vasemmanpuoleinen numero luvussa määrittelee mitä efektejä instrumentti käyttää ja oikeanpuoleinen numero ilmoittaa käytettävän instrumentin numeron

octave – valitsee näppäimistön oktaavialan (C-0 on subkontraoktaavi)

keytranspose – transponoi reaaliaikaisessa soitossa/äänityksessä käytettävää näppäimistöä puolsävelaskeleen välein

arpeggio – valitaan äänessä käytettävä arpeggiotaulukko ($08 = arpeggio nro 1, $10 = arpeggio nro 2, $18 = arpeggio nro 3 jne.)

Ruudun vasemmassa alalaidassa olevassa osiossa määritellään muunneltavien askelten jakso (FIRST STEP = jakson ensimmäinen askel, LAST STEP = jakson viimeinen askel, CURRENT STEP = askel, jolle ohjelma on sillä hetkellä pysäytetty). Esimerkiksi jotakin tiettyä kappaleen kohtaa voidaan kuunnella erikseen ilmoittamalla halutun osan ensimmäinen ja viimeinen askel tässä valikossa. Jos toiston keskeyttää, tallentuu viimeksi soitetun tahdin numero CURRENT STEP –kohtaan, jolloin kuuntelutilaan palattaessa ohjelma jatkaa toistoa keskeytetystä kohdasta.

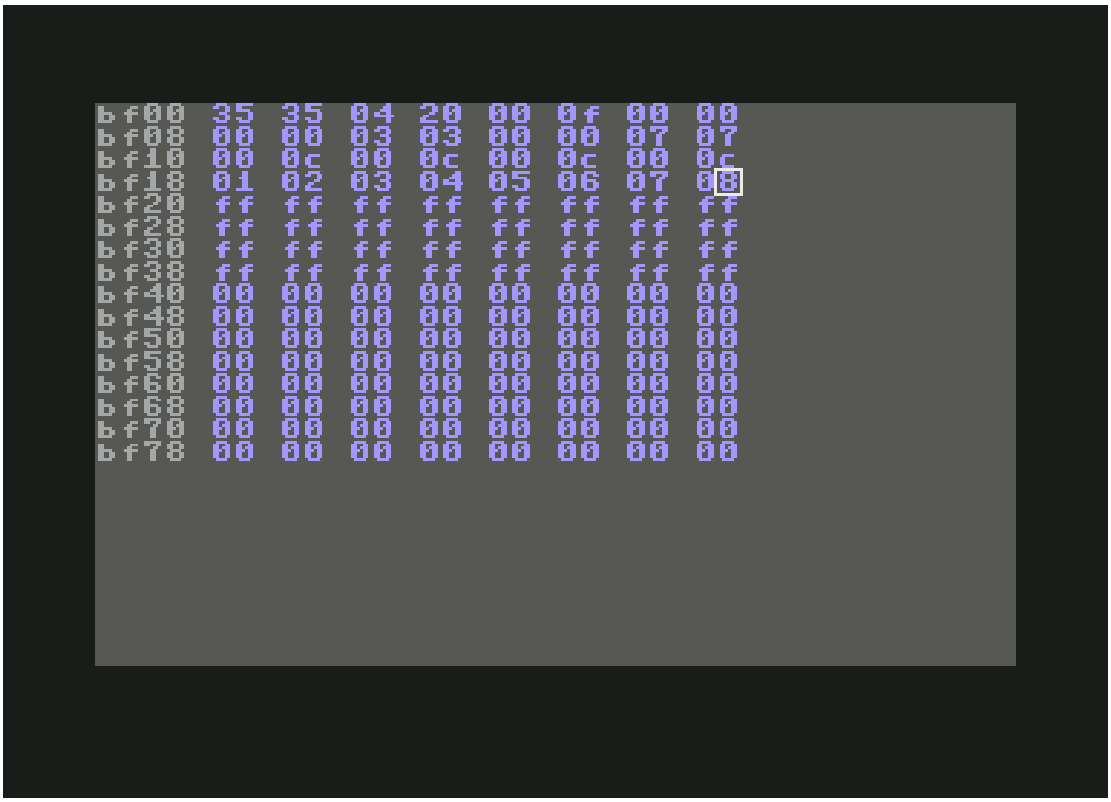
COPY STEPS –asetuksella määritellään niiden askelten jakso, jonka tiedot kopioidaan toiseen kohtaan kappaleessa. Askeleista voidaan kopioida joko kaikki tieto, tai ainoastaan nuottitieto, soittimet tai arpeggiot. Käskyillä INS/DEL STEP (INSERT/DELETE; eng., lisää/poista) voidaan lisätä ja

poistaa askelia kappaleesta, ja käsky DELETE NOTES poistaa valitun tahdin sisältämän nuottitiedon, mutta pitää askeleen/tahdin paikallaan kappaleessa.

Raituriruudun oikean alalaidan ylempi osio on varattu nuottitiedon tallentamista ja lataamista varten. TYP–kohdassa (DATA TYPE; eng., tiedon laatu) valitaan mitä tietoa kappaleesta tallennetaan tai kappaleeseen ladataan; koko kappale musiikkirutiinin kanssa (COMPLETE SONG), musiikkidata ilman rutiinia (STEPS ONLY) tai halutut instrumentit väliltä $00 – FF (SOUND NUMBER).

Aivan oikeassa alakulmassa oleva SOUND OPTIONS -osio sisältää laskurin, jota käytetään apuna efektitavun oikean arvon määrittämisessä. SOUND OPTIONS -osio käsitellään alaluvussa 3.4.1.3.

3.4.1.2 Arpeggio/synkronisointiruutu



KUVIO 13. SoundMonitor–ohjelman arpeggio/synkronisointiruutu

Vasemmanpuoleisin sarake arpeggio/synkronisointiruudussa ilmoittaa muistipaikan osoitteen, josta lähtien kyseisen rivin kahdeksan heksadesimaalista lukua on tallennettu tietokoneen muistiin. Ruudun ensimmäinen rivi määrittelee kappaleen perusasetukset, joita voi muunnella

väliaikaisesti myös kesken kappaleen. Käytännössä kaikki muut rivit ovat ensimmäistä lukuunottamatta arpeggiotaulukoita.

Perusasetukset määrittävän rivin tavut toimivat vasemmalta lukien seuraavin tavoin:

tavut 0 & 1 – kappaleen nopeuden määrittävä kaksitavuinen luku[44]. Tavu 0 on vähemmän merkitsevä ja tavu 1 enemmän merkitsevä – enemmän merkitsevä tavu voi käyttää arvoja $35 – 50, vähemmän merkitsevä arvoja $00 - FF. Mitä suurempi tavujen arvo, sitä hitaampi tempo. Tavulla 0 kappaleen tempo voidaan hienosäätää kohdalleen esimerkiksi jonkin toisen laitteen kanssa (levysoitin, videonauhuri)

tavu 2 – erittäin karkea tempon määrittely, vain arvot $00 – 04 ovat käytössä

tavu 3 - tahdin pituus, eli suurin mahdollinen nuottien määrä tahdissa ($00 – FF). Tahti tarvitsee jokaista nuottitavua kohden tavun, joka määrittelee käytetyn instrumentin sekä äänessä käytetyt efektit. Tästä johtuen yksi tahti vaatii kaksi kertaa tavussa numero 3 ilmoitetun määrän muistitilaa[45]

tavu 4 - jälkisointiefekti kappaleen lopussa ($00 – FF). Mitä suuremman arvon tavu saa, sen pidempään kappaleen viimeisten nuottien vaimeneminen kestää. Chris suosittelee käytettäväksi arvoja väliltä $10 - 30

tavu 5 - yleisvoimakkuus ($00 - 0F)

tavut 6 & 7 – ei käytössä

---

[44] Kappaleen tempon määrittely perustuu C64:sen sisäisen CIA–ajastimen (Complex Interface Adapter) arvoon, ei siis näyttöruudun päivitysnopeuteen. Tavuissa 0 & 1 annettu luku ilmoittaa ajastimelle kierrosten määrän kappaleen lyhimmän iskun sisällä.

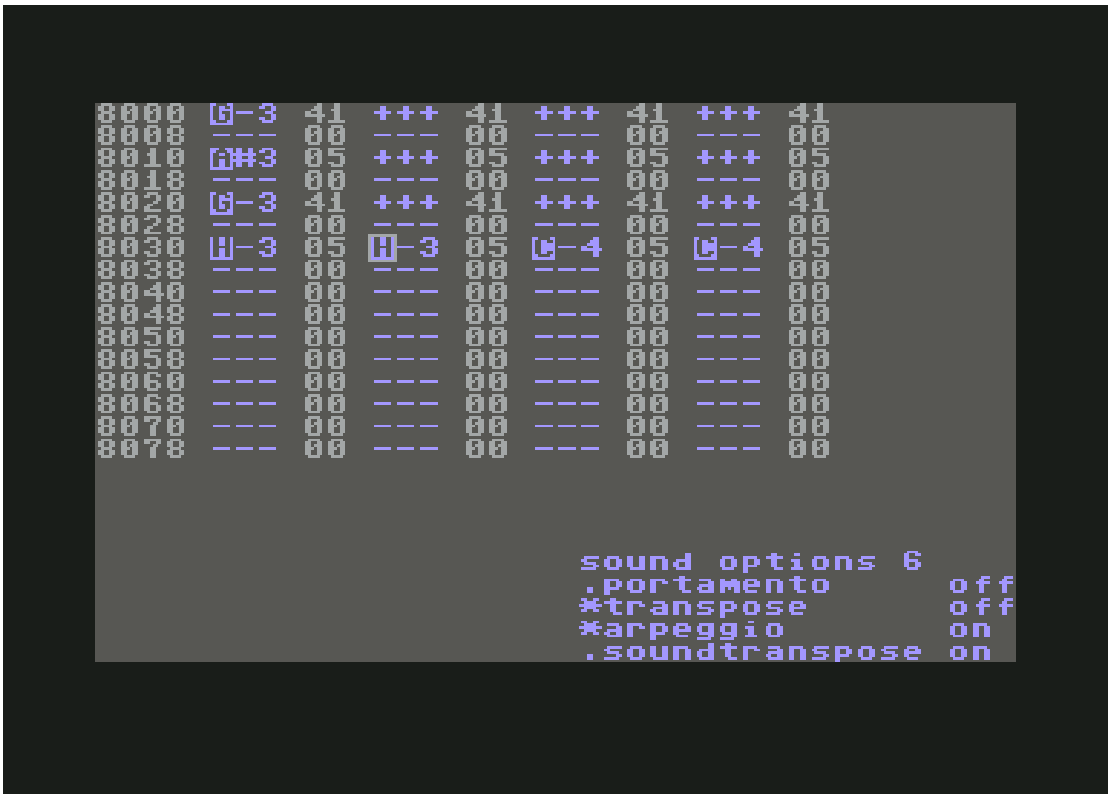
[45] Esimerkiksi nuottien määrän ollessa $20/tahti (32 kpl), tarvitaan lisäksi 32 kappaletta efekti-/instrumenttitavuja, jolloin yksi tahti vie kokonaisuudessaan muistitilaa 64 tavua. Seuraava tahti alkaa siis tietokoneen muistipaikasta, joka on 64:n pykälän päässä edellisen tahdin ensimmäisen nuotin muistipaikasta.

Arpeggiot muodostetaan transponoimalla tahtiruudussa (kuvio 14) annettua nuottia kahdeksalla eri arvolla, eli yhdellä synkronisointi/arpeggioruudun (kuvio 13) rivillä. Arvot $00 – 7F tuottavat positiivisen transponoinnin puolsävelaskeleittain ja arvot $FF – 80 negatiivisen. Arpeggiokuvio soi kerran läpi yhtä nuottiruudun tavua kohden - arpeggion nopeus on siis suhteessa kappaleen tempoon.

Yhden tahdin aikana on mahdollista käyttää vain yhtä arpeggiokuviota, jonka osoite kirjoitetaan raituriruudun AR/S–sarakkeeseen halutun tahdin/askeleen kohdalle. Tahtiruudussa (kuvio 14) jokaisen äänen perässä on tavu, jonka vasemmanpuoleinen numero määrittelee, mitä efektejä äänessä on käytössä. Arpeggioefekti kytketään päälle tahtiruudussa (kuvio 14) efektitavun bitillä 6.

Jos arpeggioefekti ei ole käytössä millään raidalla, ohjelma tulkitsee AR/S–sarakkeessa olevan osoitteen viittaavan uuteen perusasetustavuun. Tällaiset kesken kappaleen tehdyt perusasetusten muutokset ovat kuitenkin voimassa vain niiden askelten kohdalla, joiden AR/S–sarakkeessa kyseinen uusien asetusten osoite on. Tämän takia arpeggiota ei voi käyttää tahdeissa, joiden perusasetukset poikkeavat kappaleen alussa määritellyistä perusasetuksista.

3.4.1.3 Tahtiruutu



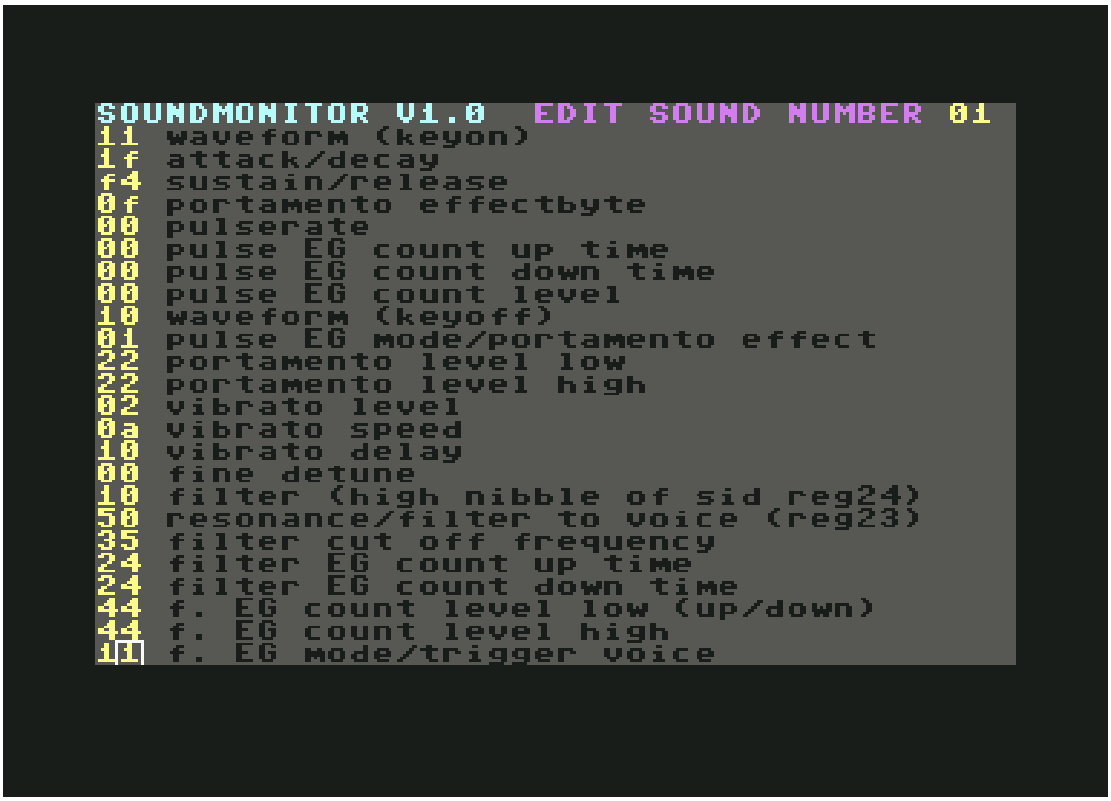
KUVIO 14. SoundMonitor –ohjelman tahtiruutu

Tahtiruudun vasemmanpuoleisin nelinumeroinen heksadesimaaliluku ilmoittaa rivin ensimmäisen nuotin muistipaikan numeron. Käytettävissä olevat sävelet ulottuvat subkontraoktaavista viisiviivaiseen oktaaviin (C-0 - H-7). # vastaa tahtiruudussa ylennysmerkkiä, --- aloittaa edellisen nuotin vaimenemisvaiheen ja +++ jatkaa edellisen nuotin sointivaihetta. Kaksinumeroinen heksadesimaaliluku jokaisen nuotin perässä on efektitavu, joka ilmaisee nuotissa käytetyt efektit sekä instrumentin numeron kyseiselle sävelelle.

SOUND OPTIONS -osio toimii laskurina efektitavun arvoa määriteltäessä. Jokaista efektiä vastaa yksi bitti efektitavussa. Kytkettäessä halutut efektit päälle (teksti ON/OFF) laskuri laskee samalla bittien muodostaman heksadesimaalisen luvun, joka vastaa haluttuja efektejä. Tämä luku kirjataan nuotin perään efektitavun *vasemmanpuoleiseksi* numeroksi, jolloin halutut efektit kytkeytyvät päälle kyseisen nuotin kohdalla. Oikeanpuoleinen numero ilmaisee nuotissa käytetyn instrumentin numeron.

Laskuri on ainoastaan apuväline oikean luvun saamiseksi efektitavuun – laskurilla ei voi suoraan muuttaa mitään nuotin/äänen ominaisuuksista. SOUND OPTIONS –osion asetukset vaikuttavat kylläkin *reaaliaikaisesti käytetyn soittimen asetuksiin.*

3.4.1.4 Instrumenttiruutu



KUVIO 15. SoundMonitor–ohjelman instrumenttiruutu

Instrumenttiruudussa määritellään kaikki instrumenttien ominaisuudet. Oikean yläkulman heksadesimaalinen luku (EDIT SOUND NUMBER) ilmaisee muokattavan instrumentin numeron. Instrumentti numero $00 on varattu metronomin käyttöön.

Seuraava listaa kuvaa jokaisen instrumenttitavun ominaisuudet:

waveform (key on) – vasemmanpuoleinen numero määrittelee aaltomuodon nuotin syttyessä. Oikean puoleisen numeron bitit toimivat

päälle kytkettäessä seuraavasti; bitti 0 – verhokäyrän käynnistys, bitti 1 – synkronisointi, bitti 2 – kehämodulaatio, bitti 3 - testausbitti[46]

attack/decay – aluke/heikentymisaika

sustain/release – sointivoimakkuus/vaimenemisaika

portamento effectbyte – portamenton syvyys, eli taajuudenlisäyksen/-vähennyksen raja-arvo, jonka ylittäessään portamentoefekti palaa alkuperäiseen taajuuteen

pulserate – pulssiaaltomuodon pulssinleveys,vähemmän merkitsevä tavu ($00 – FF)

pulse EG count up time – kuinka kauan pulssinleveyttä nostetaan pulssipyyhkäisyssä

pulse EG count down time – kuinka kauan pulssinleveyttä kavennetaan pulssipyyhkäisyssä

pulse EG count level – pulssinleveyden muutosarvo pulssipyyhkäisyssä, vähemmän merkitsevä tavu

waveform (key off) – aaltomuoto vaimenemisvaiheessa

pulse EG mode/portamento effect – vasemmanpuoleisen numeron ollessa 1 pulssipyyhkäisyefekti toistuu jatkuvasti ja numerolla 0 pulssipyyhkäisy tapahtuu vain kerran äänen verhokäyrän vaiheiden aikana. Oikeanpuoleisen numeron bitit toimivat päällä ollessaan seuraavasti: bitti 0 – portamento ylöspäin, bitti 1 – portamento alaspäin, bitti 2 – portamento tapahtuu vain kerran verhokäyrän aikana. Portamento alkaa alusta, jos taajuus on noussut/laskenut ylitse

[46] Testausbitillä voidaan sammuttaa oskillaattorin ulostulo haluttaessa. Tätä toimintoa hyödynnetään luotaessa monimutkaisia aaltomuotoja perusaaltomuodoista. Käytännössä tämä tapahtuu katkaisemalla ääni ja aloittamalla se uudestaan toisella (tai samalla) aaltomuodolla erittäin nopealla vaihteluvälillä.

neljännessä tavussa (PORTAMENTO EFFECTBYTE) määritellyn arvon. Tavallinen, sävelestä toiseen etenevä portamento saadaan antamalla oikeanpuoleiselle numerolle arvo 0

portamento level low – portamenton nopeus, vähemmän merkitsevä tavu ($00 – FF)

portamento level high – portamenton nopeus, enemmän merkitsevä tavu ($00 – FF)

vibrato level – vibraton syvyys

vibrato speed – vibraton nopeus ($00 - hyvin nopea, $7F - hyvin hidas). Bitti 7 määrittelee vibraton lähtösuunnan (0 – alaspäin, 1 – ylöspäin)

vibrato delay – viive ennen vibraton alkua

fine detune – soittimen vireen hienosäätö ($00 – A = 440 Hz)

filter – vasemmanpuoleinen numero määrittelee suotimen laadun: bitti 0 – alipäästö, bitti 1 – kaistapäästö, bitti 2 ylipäästö. Jos soittimessa ei käytetä suodinta, asetetaan tavun arvoksi $FF. Tämä estää suotimen ”naksumisen” päälle ja pois kytkettäessä

resonance/filter to voice – vasemmanpuoleinen numero vastaa resonanssin arvoa (16 voimakkuusvaihtoehtoa). Oikeanpuoleinen numero määrittelee suodinta käyttävät oskillaattorit (bitit 0 – 2 vastaavat oskillaattoreita 1 – 3)

filter cut off frequency – suotimen leikkaustaajuus ($00 – FF)

filter EG count up time – kuinka kauan leikkaustaajuutta korotetaan suodinpyyhkäisyssä

filter EG count down time – kuinka kauan leikkaustaajuutta pudotetaan suodinpyyhkäisyssä

filter EG count level low (up/down) – suodinpyyhkäisyn muutosarvo, vähemmän merkitsevä tavu. Vasemmanpuoleinen numero määrittelee kuinka paljon leikkaustaajuus nousee ja oikeanpuoleinen määrittelee paljonko taajuus laskee suodinpyyhkäisyssä

filter EG count level high – suodinpyyhkäisyn muutosarvo, enemmän merkitsevä tavu ($00 – FF)

filter EG mode/trigger voice – vasemmanpuoleisen numeron biti toimivat seuraavasti: bitti 0 – suodinpyyhkäisy toistuu jatkuvasti bitin ollessa päällä, bitti 1 – pyyhkäisyn suunta; bitin ollessa pois päältä suunta on ylöspäin ja päällä ollessa alaspäin. Oikeanpuoleinen numero määrittelee mitkä oskillaattorit käyttävät suodinpyyhkäisyä (bitit 0 – 2 vastaavat oskillaattoreita 1 – 3)

3.4.2 Yhteenveto SoundMonitor-ohjelmasta

Chris Hülsbeckin SoundMonitor–ohjelma on ehdottomasti helppokäyttöisin tutkielmassa käsitellyistä ohjelmointirutiineista; mm. graafinen käyttöliittymä, reaaliaikainen soitto/nauhoitus tietokoneen näppäimistöä käyttäen, tekstiselvennykset jokaisen instrumenttiparametrin kohdalla sekä mahdollisuus muunnella soitinten ja arpeggioiden parametreja *kappaleen soidessa* erottuvat selkeästi edukseen vertailtaessa rutiineja keskenään. Ainoastaan soittimien valinta SOUND TRANSPOSE –komennolla on hankalasti toteutettu verrattuna ohjelman muihin varsin käyttäjäystävällisiin ominaisuuksiin.

Kappaleen rakenteen, soitinten ja efektien muuntelu on vaivatonta ja tämän lisäksi muutosten vaikutukset musiikkiin ovat välittömästi kuultavissa, mikä mahdollistaa hyvin harkittujen sovitusten ja musikaalisten ratkaisujen tekemisen. Samoin reaaliaikainen nauhoittaminen mahdollistaa

luonnollisemman tavan käsitellä nuottitietoa kuin pelkkä numeroiden sijoittelu muistipaikkoihin voi ikinä olla.

Yllä mainituilla ominaisuuksilla on myös varjopuolensa varsinkin pelimusiikkia sävellettäessä. Musiikkidata vaatii aina seurakseen säveltämisessä käytetyn ohjelmointirutiinin, jonka tulisi mielellään olla mahdollisimman pieni, jotta varsinaiselle nuottitiedolle ja itse pelillekin jäisi tilaa Commodore 64:sen varsin rajallisessa muistitilassa. Kaikki SoundMonitorin käyttöä helpottavat ominaisuudet kuitenkin kuluttavat varsin paljon muistitilaa ja heikentävät ohjelman soveltuvuutta pelimusiikkien säveltämiseen.

SoundMonitor ei ole joustava ohjelma toisin kuin konekielen välityksellä käytettävät Rob Hubbardin ja Martin Galwayn musiikkirutiinit. Heidän rutiininsa mahdollistavat täysin uusien ja mitä erilaisimpien efektien hyödyntämisen rutiinin osina, joiden kutsuminen onnistuu yksinkertaisilla musiikkirutiinin käskyillä joko suoraan nuottidatassa (Galway) tai kytkemällä ohjelmoitua efektiä vastaavan bitin päälle instrumentin efektitavusta (Hubbard). SoundMonitor–ohjelmassa ei voi myöskään käyttää sample–ääniä, vaan ainoastaan instrumenttiruudussa määriteltäviä soittimia[47].

SoundMonitor –raiturin portamento-, arpeggio- ja vibratoefektit ovat varsin käyttökelpoisia ja kohtuullisen monipuolisesti muunneltavissa – esimerkiksi portamentoefektiä voi käyttää erikoisiin taajuusmodulaatioihin kytkemällä efektin alkamaan aina alusta portamenton raja-arvon ylittyessä, nostamalla portamenton rajataajuutta hyvin korkeaksi/matalaksi sekä asettamalla liu'un hyvin nopeaksi.

Kappaleen tempon voi määritellä hyvin tarkasti kolmen tavun avulla, kokonaisia tahteja voi transponoida ainoastaan yhtä tavua muuttamalla ja jopa yksittäisen soittimen virettä voi vaihdella instrumenttiruudun tavulla FINE DETUNE. Mielenkiintoisia ratkaisuja ovat myös tahdin pituuden määritteleminen yhdellä tavulla (Grayn ja Hubbardin rutiineissa nuottikuvioiden pituutta ei ole millään tavoin ennaltaa määritelty) sekä mahdollisuus lisätä erityinen jälkisointi kappaleen loppuun.

---

[47] SoundMonitor–ohjelman päivitetyt versiot (SoundMonitor 1.1 [1986], RockMonitor II[1987]) sisältävät mahdollisuuden käyttää näytteitä. SoundMonitor 1.1 -ohjelman parannuksista vastaa Daniel Kahlin ja SoundMonitor–raituriin pohjautuvan RockMonitor II –ohjelman ovat tehneet Oscar Giesen ja Marco Swagerman Dutch USA Team -hakkeriryhmästä (Tognon, S. 2005).

Kappaleen perusasetuksia voi muunnella myös kesken kappaleen, mikä mahdollistaa temponvaihdokset, sekä tahdinpituuden ja siten myös iskujen alijakojen muuttamisen. Arpeggioita ei kuitenkaan voi käyttää perusasetuksia muutettaessa, sillä molemmat käyttävät samaa raituriruudun AR/S–saraketta.

4 TULOSTEN POHDINTAA

Totesin jo tutkielman tekemisen alkuvaiheessa neljän eri säveltäjän konekielisten ohjelmointirutiinin purkamisen ja selittämisen olevan varsin hidasta ja työlästä puuhaa tietokonealan ammattilaisillekin - puhumattakaan allekirjoittaneesta, joka ei ollut ennen tutkielman aloittamista ikinä ohjelmoinut Commodore 64 -kotitietokoneella. En halunnut kuitenkaan hylätä valitsemaani aihetta, joten otin suoraan yhteyttä C64:lle 1980-luvulla pelimusiikkeja tehneisiin säveltäjiin sekä ohjelmoijiin, jotka voisivat konekielen asiantuntemuksellaan auttaa valitsemieni säveltäjien ohjelmointirutiinien purkamisessa.

Kerroin sähköpostiviesteissäni tutkielmani taustoista, tavoitteista sekä ongelmakohdista ja tiedustelin, josko viestin saajalla olisi mahdollisuus auttaa minua välillä ylivoimaisiltakin tuntuneiden esteiden yli. Eräs vastanneista oli englantilainen, erittäin kokenut pelimusiikkien säveltäjä Ben Daglish (Hades Nebula; Nexus 1987, Jack The Nipper; Gremlin Graphics 1986, The Last Ninja; System 3 1987). Hänellä ei valitettavasti ollut aikaa auttaa tutkielman analyysiosiossa, mutta toivotti menestystä työssä ja kertoi oman näkemyksensä tutkielman ydinkysymyksestä, onko käytetyillä ohjelmointirutiineilla vaikutusta Commodore 64 –kotitietokoneen pelimusiikkien säveltämiseen ja sovittamiseen:

> *“Interesting project. I think you'll probably find it's more to do with how the tunes were structured, and the styles of the composers themselves rather than the specific 'make a noise' routines.”*
>
> *“Mielenkiintoinen projekti. Luulen, että tulet todennäköisesti huomaamaan asian koskevan ennemmin säveltäjien omaa tyyliä sekä tapaa, jolla kappaleet on koostettu, kuin vain tiettyjä ‘ääntelyrutiineja’.”*
>
> (Daglish, B. 28.2.2005)

Näistä sanoista huolimatta päätin jatkaa nimenomaan ohjelmointirutiinien tutkimista, ja analyysien edistyessä Project: Galway –levyjen toinen kokoaja Alistair “Boz” Bowness antoikin tutkielmastani keskustellessamme oman, hieman Benin lausunnosta poikkeavan mielipiteensä:

> *“What Ben said is very interesting... although I don't agree with him entirely! I think that Rob, Ben, Matt and Hülbeck had very similar routines, but Galway's was completely different, as you have found out on your "travels" in the music-routine world!”*
>
> *“Se mitä Ben sanoi, on hyvin mielenkiintoista… vaikka en ole täysin samaa mieltä! Omasta mielestäni Rob (Hubbard), Ben (Daglish), Matt (Gray) ja (Chris) Hül(s)beck käyttivät hyvin samanlaisia rutiineja, mutta (Martin) Galwayn rutiini oli täysin erilainen, kuten olet matkoillasi musiikkirutiinien maailmassa huomannut!”*
>
> (Bowness, A. 2-4/2005)

Selvitettyäni tässä tutkielmassa Matt Grayn, Martin Galwayn, Rob Hubbardin ja Chris Hülsbeckin käyttämien musiikkirutiinien käyttöominaisuuksia ja toimintaperiaatteita, sekä kuunneltuani heidän säveltämäänsä musiikkia vuosien ajan, voin huoletta yhtyä Alistairin mielipiteeseen – instrumentin hallinnassa käytetyillä tekniikoilla todella on vaikutuksensa myös instrumentilla sävellettyyn musiikkiin.

4.1 Matt Gray

Musiikkitiedon jakaminen raiturimaisesti kuvioihin mahdollistaa vaivattoman kappaleen osioiden toiston, mutta tällöin kuvioiden sisältöä ei voida muunnella toistosta toiseen. Tästä johtuen Grayn kappaleet sisältävät monesti identtisiin kuvioiden toistoon ilman variaatioita. Suotimen jättäminen pois musiikkirutiinista hankaloittaa elävien instrumenttivärien luontia ja

vähentää merkittävästi mahdollisten erilaisten soitinten määrää. Samoin kehämodulaatio- ja synkronisointiefektien käyttämättä jättäminen aiheuttavat nimenomaan erilaisten äänenvärien puuttumisen Driller–musiikkirutiinilla tehdyistä kappaleista. Tosin Matt kompensoi suotimen, kehämoduloinnin ja synkronisointiefektin puuttumista käyttämällä itse ohjelmoimiaan efektejä luodessaan erilaisia instrumentteja.

Matt Grayn rutiini kykenee hyödyntämään hyvin SID–mikrosirun ominaisuuksia rumpuäänien tuottamisessa (myös jännitevuotoa käyttämällä), mikä myös kuuluu hänen tuotannossaan - käytännössä jokainen hänen säveltämänsä pelimusiikki sisältää rummut. Rutiinilla on mahdollista luoda myös yleisimpiä polyrytmejä (2:3, 4:3) mutta metristen modulaatioiden vaivaton tekeminen vaatisi erillisen konekielisen aliohjelman liittämistä rutiiniin. Tempo määritellään aina kappaleen alussa, joten toinen vaihtoehto olisi aloittaa uusi kappale moduloitavasta kohdasta. Molemmissa tapauksissa syntyvä viive olisi kuitenkin todennäköisesti liian suuri, jotta modulaatiota voisi järkevästi käyttää.

Matt käytti joissain sävellyksissään myös MusicMaster ja RockMonitor–editoreita (Tognon, S. 2002), joten kaikki kappaleet eivät ole vertailtavissa keskenään analysoitaessa ohjelmointitekniikoiden vaikutusta hänen sävellyksiinsä. Kyseiset editorit ilmestyivät kuitenkin ennen Driller–rutiinin käyttöönottoa, joten on oletettavaa, että hän ohjelmoi oman rutiininsa vastaamaan omia tarpeitaan pelimusiikkien säveltämisessä.

4.2 Martin Galway

Poikkeuksellinen instrumenttien ohjelmointitekniikka mahdollistaa soitinten äänenvärien hienovaraisen muuntelemisen missä vaiheessa äänen sointia tahansa. Martinin ohjelmoimat taajuus-, pulssi- ja suodinmodulaatiot ovat erittäin monipuolisia ja mahdollistavat hyvin eläväisten, paksujen ja voimakkaiden lead -äänten muodostamisen. Martinin sävellyksiä ja sovituksia leimaakin hyvin vahva melodisuus ja laulettavuus.

Musiikkirutiinissa on mahdollisuus käyttää “kylmäkäynnistystä” jokaiselle nuotille, jolloin SID–sirun verhokäyrän epätarkkuusongelmat voidaan kiertää. Tämä ominaisuus korostuu käytettäessä monipuolisesti

verhokäyrän eri asetuksia, mikä taas on edellytys elävien melodisten elementtien luomiselle.

Itse asiassa tutkielmassa käsiteltävä Arkanoid–musiikkirutiini poikkeaa rumpuäänien käytön osalta Martinin aiempien sävellysten melodisesta luonteesta suurestikin. Hän itse koki rumpuäänien imitoimisen olevan hänen heikkoutensa ohjelmoijana, eikä hän niitä pahemmin ennen vuotta 1987 sävellyksissään käyttänytkään. Arkanoid–kappaleessa vaatimattomat rumpuäänet ovat kuitenkin jatkuvassa käytössä, vaikka ne vaativatkin oman aliohjelmansa lisäksi kokonaisen kappaleen raidan itselleen, jolloin ainoastaan kaksi oskillaattoria jää muiden instrumenttien käyttöön.

Rumpuäänien ohjelmointitekniikan lisäksi Martin Galwayn sävellystyyliin vaikutti epäilemättä myös kehämodulaatio- ja synkronisointiefektien jättäminen pois musiikkirutiinista. Näiden efektien käyttö olisi mahdollistanut tavallisuudesta poikkeavien äänimaailmojen käytön kontrastina vahvalle melodiselle tulkinnalle.

4.3 Rob Hubbard

Musiikkirutiinin suoraviivaisuus ja sen tuottamien äänenvärien yksinkertaisuus (esim. kolmivaiheinen vibrato) siirtävät painopisteen Hubbardin pelimusiikeissa melodisten elementtien käytöstä rytmiikkaa korostavaan suuntaan. Myös suotimen jättäminen pois rutiinista vähentää radikaalisti melodiassa käytettävien äänenvärien määrää. Rob paikkaa rutiininsa puutteita melodiaäänten käsittelyssä käyttämällä sekä melodisina että perkussiivisina tehokeinoina omalaatuisia aliohjelmilla toteutettuja efektejä. Tutkielmassa käsitelty Monty On The Run –musiikkirutiini sisältää kuitenkin valitettavan vähän erilaisia efektejä (myös kehämodulointi- ja synkronisointiefektit puuttuvat), joita Rob käyttää myöhemmissä sävellyksissään hyvinkin paljon.

Hubbardin säveltämis- ja sovittamistyyliä ovat epäilemättä ohjailleet myös musiikkirutiinin erittäin hyvin toimivat rumpuäänet. Rutiinin erittäin pieni koko ja sujuva toiminta mahdollistavat sekä pitkien sävellysten liittämisen pelien yhteyteen, että instrumenttiäänien nopean vuorottelun yhdellä raidalla - hyvin usein Rob hyödyntääkin tätä mahdollisuutta mm. varaamalla yhden

raidan kappaleistaan rummuille ja bassokuviolle, ja käyttää kahta muuta oskillaattoria omalaatuisiin efekteihin, harmonioihin ja melodiaääniin.

Kappaleiden rakenteen suhteen rutiinin raiturimainen luonne ohjaa käyttäjäänsä helposti toistamaan tiettyjä kuvioita ja osia kappaleesta muuntelemattomina kuten Matt Grayn Driller-rutiinikin.

4.4 Chris Hülsbeck

Mahdollisuus reaaliaikaiseen musiikkidatan käsittelyyn SoundMonitor–ohjelmassa vaikuttaa epäilemättä sävellysten ja sovitusten luonteeseen – ideoita on helppo kokeilla, kuunnella ja muunnella tietokoneen näppäimistöltä ohjelman pyörittäessä samalla taustalla haluttua kappaleen osaa. Pianon koskettimiston jäljittely tietokoneen näppäimistöllä voi taas johtaa “pianistisiin” ratkaisuihin sävellyksissä.

Erilaisten äänenvärien sekä rumpuäänien luominen onnistuu hyvin ohjelman monipuolisilla efekti- ja modulaatioparametreilla. Varsin omalaatuinen ominaisuus on soittimien hienoviritys FINE DETUNE –arvolla. Sample–äänien lisääminen SoundMonitor–ohjelmalle tehtyihin musiikkeihin ei ole mahdollista, mutta myöhemmissä versioissaan rutiinista Chris lisäsi myös tämän ominaisuuden ohjelmaan.

Tempon ja tahdin pituuden muuttaminen kesken kappaleen mahdollistaa monimutkaisten polyrytmien ja metristen modulaatioiden käyttämisen, sekä kappaleen iskujen alijaon muuttamisen - näitä ominaisuuksia en ole tosin kuullut Chrisin kertaakaan käyttävän.

Ohjelmassa on mahdollista kopioida valitut tahdit uuteen paikkaan joko käyttämällä tahtien muistipaikkojen osoitenumeroita tai kopioimalla tahdeista yksitellen pelkän nuottitiedon kokonaan uuteen tahtiin. Tällä tavoin kappaleen osien toistoihin voi lisätä helposti myös variaatioita ja siten elävöittää musiikkia.

SoundMonitor–ohjelman ainoat heikot puolet säveltämisessä ovat kehämodulointi- ja synkronisointiefektien puuttuminen sekä kyvyttömyys kutsua käyttöön erillisiä konekielisiä aliohjelmia esimerkiksi omien efektien tekoon. SoundMonitor on käsitellyistä rutiineista kaikkein neutraalein ja myös helppokäyttöisin graafisen käyttöliittymänsä ansiosta – molemmat seikat

epäilemättä vaikuttivat ohjelman saamaan suosioon tietokoneharrastajien keskuudessa.

4.5 Lopuksi

Jokainen tutkielmassa käsitelty ohjelmointirutiini heijastaa tekijöidensä taitoa ohjelmointitekniikoiden hallinnassa ja ymmärrystä musiikin lainalaisuuksista niin käytännössä kuin teoriassakin. Huolimatta ohjelmointinäytteiden ammattimaisuudesta ja korkeasta tasosta, niiden väliltä löytyy selkeitä eroavaisuuksia, jotka muokkaavat ohjelmointirutiinin käyttäjän lähestymistapaa sävellys- ja sovitustyöhönsä. Tämän seikan huomioiminen korostaa teknisten valmiuksien merkitystä musiikin luovissa prosesseissa (sävellys, sovitus, improvisaatio), ja mielestäni vastaavan tutkielman tekeminen sekä akustisten soittimien, että ilman tietokonetta ohjailtavien sähköisten musiikki-instrumenttien parissa olisikin enemmän kuin tervetullut.

Uusia soittotekniikoita ei pidä arvioida musiikista irrallisina elementteinä tai soittajan ”kikkailuna”, vaan nähdä ne mahdollisuutena syventää muusikon ilmaisuvoimaa ja yksilöllisyyttä, sekä parantaa muusikon valmiuksia työelämässä - varsinkin oltaessa kosketuksissa erilaisten musiikin tyylilajien kanssa.

Lopuksi haluan antaa suunvuoron Martin Galwaylle ja suuresti arvostetulle demomuusikko Frank Seemanille, joka tunnetaan Atari -skenessä paremmin nimellä Tao (Frank on jäsen edelleen aktiivisessa Cream-koodaajaryhmittymässä, jonka kotisivut löytyvät Internet–osoitteesta http://www.creamhq.de). Myös heidän kommenteistaan käy ilmi, että instrumenttitekniikan hallitsemisella todellakin on vaikutuksensa sävellysten luonteeseen sekä säveltäjän tekemien musikaalisten ratkaisujen muodostumiseen.

> *“Rob Hubbard, David Whittaker, Ben Daglish, Chris Huelsbeck, Martin Galway and the Maniacs of Noise were living gods in the C64 music-scene. (…) Why are these guys still mentioned, when we're talking about computer musix? It can't be the soundprocessors they were*

programming. The YM-2149 was pure shit, Paula and SID, great chips at their time, can easily be emulated by the new generation of soundchips (even by STE). It was their fantasy in composing music and getting the best out of their possibilities. If you love computer music (…), you can identify each of these guys as author, just by listening to the first notes of a tune. It were their unique techniques to create music, what let them stand out of the mass: Check out for yourself and listen to the details in the musics." (Tao of Cream)

"Rob Hubbard, David Whittaker, Ben Daglish, Chris Huelsbeck (Hülsbeck), Martin Galway ja Maniacs of Noise olivat C64 –skenen eläviä jumalia. (…) Miksi heidät yhä mainitaan tietokonemusiikista puhuttaessa? Se ei voi johtua heidän ohjelmoimistaan ääniprosessoreista. YM-2149 (Atari ST) oli täysi susi, Paula (Amiga) ja SID, mahtavia mikrosiruja aikanaan, ovat helposti jäljiteltävissä uuden sukupolven mikrosiruilla (jopa [Atari] STE:llä). Kyse on heidän mielikuvituksestaan musiikin säveltämisessä ja sirujen mahdollisuuksien hyödyntämisestä parhaimmalla mahdollisella tavalla. Jos rakastat tietokonemusiikkia (…), pystyt tunnistamaan näistä säveltäjistä jokaisen pelkästään kuuntelemalla kappaleen ensimmäisiä nuotteja. Heidän ainutlaatuiset tekniikkansa musiikin luomisessa nosti heidät esiin massasta: totea tämä itse kuuntelemalla yksityiskohtia heidän musiikissaan." (Tao of Cream)

(Seeman, F. 2005)

Carr: *“Is there a SID tune that wasn’t your own that you would have liked to have composed yourself?”*

Galway: *“Absolutely, they’re usually Rob’s!!! But I suppose I was simply wanting ot use his program to get all that percussion. I wouldn’t have minded having a go with his software and doing music in my style with his percussion. Perhaps we should have swapped drivers for a game just to see what we could have come up with.”*

Carr: *“Onko olemassa SID –kappaletta, joka ei ole omasi, mutta jonka olisit itse halunnut säveltää?”*

Galway (Martin): *Ehdottomasti, ne olivat yleensä Robin (Hubbard)!!! Mutta oletan, että halusin vain käyttää hänen ohjelmaansa saadakseni kaikki ne lyömäsoittimet. En olisi pistänyt pahakseni, jos olisin päässyt kokeilemaan säveltämistä omalla tyylilläni hänen ohjelmallaan ja lyömäsoittimillaan. Kenties meidän olisi pitänyt vaihtaa ohjaimia pelin ajaksi, ihan vain nähdäksemme mitä me olisimme saaneet aikaiseksi.”*

(Carr, N. & Abbott, C. (toim.) 2001)

5 LÄHTEET

Bowness, A. 2003. Project: Galway. WWW-dokumentti. Binary Zone Interactive. http://www.binaryzone.co.uk/.

Bowness, A. 2-4/2005. Sähköpostikeskustelu Petri Kentalan kanssa.

Butterfield, J. 1984. Machine Language For The Commodore 64 And Other Commodore Computers. Brady Communications Company Inc.

Carr, N. & Abbott, C. (toim.) 2001. An Interview With Martin Galway. WWW-dokumentti. Remix64. Sivuston perustaja ja ylläpitäjä on Markus Klein. http://remix64.phatsites.de/index.php?interview_martin_galway.

Carr, N. 2001a. An Interview With FeekZoid. WWW-dokumentti. Remix64. Sivuston perustaja ja ylläpitäjä on Markus Klein. http://remix64.phatsites.de/index.php?interview_feekzoid.

Carr, N. 2001b. An Interview With Chris Abbott. WWW-dokumentti. Remix64. Sivuston perustaja ja ylläpitäjä on Markus Klein. http://remix64.phatsites.de/?load=interview_nr_1.

Chris Hülsbeck 2005. WWW-dokumentti. Wikimedia Foundation Inc. de.wikipedia.org/wiki/Chris_H%C3%BClsbeck

Commodore 64 Programmer's Reference Guide 1983. Commodore Business Machines, Inc. & Howard W. Sams & Co., Inc.

Commodore Computer History 2005. WWW-dokumentti. Up & Running Technologies Incorporated. http://www.commodore.ca.

Daglish, B. 2/2005. Sähköpostikeskustelu Petri Kentalan kanssa.

Fairlight CMI 2005. WWW-dokumentti. Wikimedia Foundation Inc. http://www.answers.com/main/ntquery?method=4&dsid=2222&dekey=Fairlight+CMI&gwp=8&curtab=2222_1.

Harbron, R., Harsfalvi, L. & Judd, S. 2001. The C64 Digi. C=Hacking verkkolehti nro 20. http://www.ffd2.com/fridge/chacking/c=hacking20.txt.

Haste Töne? 1989. Haste Töne? - Soundprogrammierer Jochen Hippel im Interview. Aktueller Software Market (ASM) Special 6/1989.

Holmes, G. 2005. The Fairlight CMI. WWW-dokumentti. The Holmes Page. Sivujen tekijänoikeuksia hallinnoi GH Services. http://ghservices.com/gregh/fairligh/.

Huhtamo, E. & Kangas, S. (toim.) 2002. Mariosofia: Elektronisten pelien kulttuuri. Tampere: Oy Yliopistokustannus University Press Finland.

Hülsbeck, C. 2005. WWW-dokumentti. Website Of Chris Hülsbeck. http://www.huelsbeck.com.

Hülsbeck, C. 64'er Ausgabe 10/Oktober 1986, Listing des Monats. 64’er 10/1986.

HVSC 2005. High Voltage SID Collection. WWW-dokumentti. http://www.hvsc.c64.org/.

Interview With Chris Hülsbeck 2005. WWW-dokumentti. Music4Games Inc. www.music4games.net/f_chrishuelsbeck_starwars_rebelstrike.html

Interview With Martin Galway 2005. WWW-dokumentti. C64hq. www.c64hq.com/interviews/galway.html.

Interview With Rob Hubbard. Happy Computer 7/1986.

Inventors of the Modern Computer 2005. WWW-dokumentti. About Inc. http://inventors.about.com/library/weekly/aa043099.htm.

Judd, S. 1996. SID Primer: The Working Man's Guide to SID. Verkkolehti Discovery nro 2. http://www.ffd2.com/fridge/discovery/issue2.gz

Klose, T. 2005. MIDIbox SID. WWW-dokumentti. MIDIbox –tuotteiden ekijänoikeudet omistaa Thorsten Klose. http://www.uCApps.de.

Kristensen, A. 2005. Sound And Music. WWW-dokumentti. http://home20.inet.tele.dk/domix/cli/C64/sound64.htm.

MacKenzie, J. 1996. Interview With Martin Galway. Commodore Zone nrot 6-7.

MacKenzie, J. 1997. Interview With Rob Hubbard. Commodore Zone 10.

MacSweeney, A. 1993. Rob Hubbard’s Music: Disassembled, Commented and Explained by Anthony McSweeney. C=Hacking verkkojulkaisu nro 5. http://www.ffd2.com/fridge/chacking/c=hacking5.txt.

MOS Technology SID 2005. WWW-dokumentti. Wikimedia Foundation Inc. http://en.wikipedia.org/wiki/MOS_Technology_SID.

Most Important Chips. BYTE 9/1995.

Mäkela, M. & Alstrup, A. Examination of SID noise waveform. WWW-dokumentti. http://unusedino.de/ec64/technical/misc/sid6581/sidtech5.html.

Newport, S. 2001. The Guinness Book Of World Records. Lontoo: HIT entertainment.

Nokian modeemimainos. Mikrobitti 4/1987.

Nyqvist-Shannon Sampling Theorem 2005. WWW-dokumentti. Wikimedia Foundation Inc. http://en.wikipedia.org/wiki/Nyquist-Shannon_sampling_theorem.

Oppermann, J. 2004. SID MOS 6581. WWW-dokumentti. http://www.joogn.de/sid.html.

Programming – Introduction 2005. WWW-dokumentti. http://old.c64.ch/programming/intro.php.

Rapo, P. 14-vuotiaana yritysjohtajaksi. MikroBitti 8/1986.

Sánchez, C. 2003. An Interview With Martin Galway. WWW-dokumentti. Lemon Commodore 64. Sivuja ylläpitää Kim Lemon. http://www.lemon64.com/index.php?mainurl=http%3A//www.lemon64.com/interviews/martin_galway.php.

Schembri, T. & Boisseau, O. 2005. Commodore 64. WWW-dokumentti. Old-computers.com. New York Internet. http://www.old-computers.com/museum/computer.asp?st=1&c=98.

Seeman, F. 2005. Jochen Hippel – Intro. WWW-dokumentti. The Thalion Source. Sivustoa ylläpitää Gerald Müller-Bruhnke. http://home.wtal.de/gmb/Hippel/hippelm.htm.

The Complete Works Of Rob Hubbard 2005. WWW-dokumentti. The Complete Works Of Rob Hubbard 2001-2005. http://www.robhubbard.co.uk.

Tognon, S. 2002. Matt Gray's Driller Music Routine. SIDin–verkkolehti, nro 2. http://digilander.libero.it/ice00/tsid/sidin2.

Tognon, S. 2003. Martin Galway's Arkanoid Music Routine. SIDin–verkkolehti, nro 4. http://digilander.libero.it/ice00/tsid/sidin4.

Tognon, S. 2005. High Voltage Music Engine Collection. WWW-dokumentti. http://utenti.lycos.it/ice00/HVMEC/CONTROL/.

Tulip Computers Sells Commodore To Yeahronimo Media Ventures Inc. 2004. WWW-versio Commodore International BV -yhtiön lehdistötiedotteesta.

http://www.commodoreworld.com/site/DesktopDefault.aspx?tabindex=4&tabid=702&bix=4&bid=5&itemIDS=18&ModID=2.

Uutisia. Mikrobitti 8/1986.

Wallich, P. Design Case History: The Commodore 64. IEEE Spectrum 3/1985.

Varga, A. 1996. Interview with Bob Yannes. WWW-dokumentti. http://stud4.tuwien.ac.at/~e9426444/yannes.html.

Windisch, S. 1983. Grafiikka ja äänitehosteet: CBM-64. Förlagsgruppen.

Yannes, R. 1983. US4677890: Sound interface circuit. Patenttihakemus.

LIITTEET

LIITE 1.

CD-levy

LIITE 2.

CD-levyn (liite 1) sisällysluettelo:

1. ESIMCD 1 - Kolmioaalto
2. ESIMCD 2 - Sahalaita-aalto
3. ESIMCD 3 - Pulssiaalto
4. ESIMCD 4 - Kohina-aalto
5. ESIMCD 5 - ”Pulssipyyhkäisy”
6. ESIMCD 6 - Alipäästösuodin
7. ESIMCD 7 - Kaistapäästösuodin
8. ESIMCD 8 - Ylipäästösuodin
9. ESIMCD 9 - Taustakohina (SID 6581)
10. ESIMCD 10 - Suodinvertailu SID 6581 (Klose, T. 2005)
11. ESIMCD 11 - Suodinvertailu SID 8580 (Klose, T. 2005)
12. SIDCD 1 - Maze Mania (Hewson, 1989)
13. SIDCD 2 - Driller (Incentive, 1988)
14. SIDCD 3 - Yie Ar Kung Fu (Imagine, 1985)
15. SIDCD 4 - Arkanoid (Imagine, 1987)
16. SIDCD 5 - Ocean Loader 2 (Ocean, 1985)
17. SIDCD 6 - Monty On The Run (Gremlin Graphics, 1985)
18. SIDCD 7 - Crazy Comets (Martech, 1985)
19. SIDCD 8 - Shades (Chris Hülsbeck, 1986)
20. SIDCD 9 - Grand Monster Slam (Rainbow Arts, 1989)