Ville Helppolainen

Vektorigrafiikat ja niiden käyttö videopeleissä

Tietotekniikan kandidaatintutkielma

13. joulukuuta 2018

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Ville Helppolainen

Yhteystiedot: vivakank@student.jyu.fi

Työn nimi: Vektorigrafiikat ja niiden käyttö videopeleissä

Title in English: Vector graphics and their usage in videogames

Työ: Kandidaatintutkielma

Sivumäärä: 22+0

Tiivistelmä: Tässä tutkielmassa pohditaan, minkä syiden takia vektorigrafiikoiden käyttö ei ole yleistynyt videopeleissä. Lisäksi esitellään joitain vektorigrafiikoita käyttäviä pelejä. Huolimatta vektorigrafiikoiden eduista bittikarttagrafiikoihin verrattuna, on niiden käytössä monia ongelmia, kuten huonompi suorituskyky, puuttuva pelimoottorituki ja taiteellinen rajoittuneisuus. Näitä ongelmia pyritään vähentämään parantamalla vektorigrafiikoiden suorituskykyä ja tarjoamalla työkaluja, joiden avulla vektorigrafiikoita on helpompi käyttää.

Avainsanat: vektorigrafiikat, videopelit, videopeligrafiikat, SVG

Abstract: This study discusses reasons for the lack of adoption of vector graphics in videogames. In addition, some games that do use vector graphics are presented. Despite the benefits of vector graphics over raster graphics, their usage has many problems such as performance, the lack of game engine support and artistic limitations. Vector graphics are constantly developed in attempt to overcome these problems by increasing the performance and offering tools for more accessible use.

Keywords: vector graphics, videogames, videogame graphics, SVG

Kuviot

Sisältö

1 Johdanto

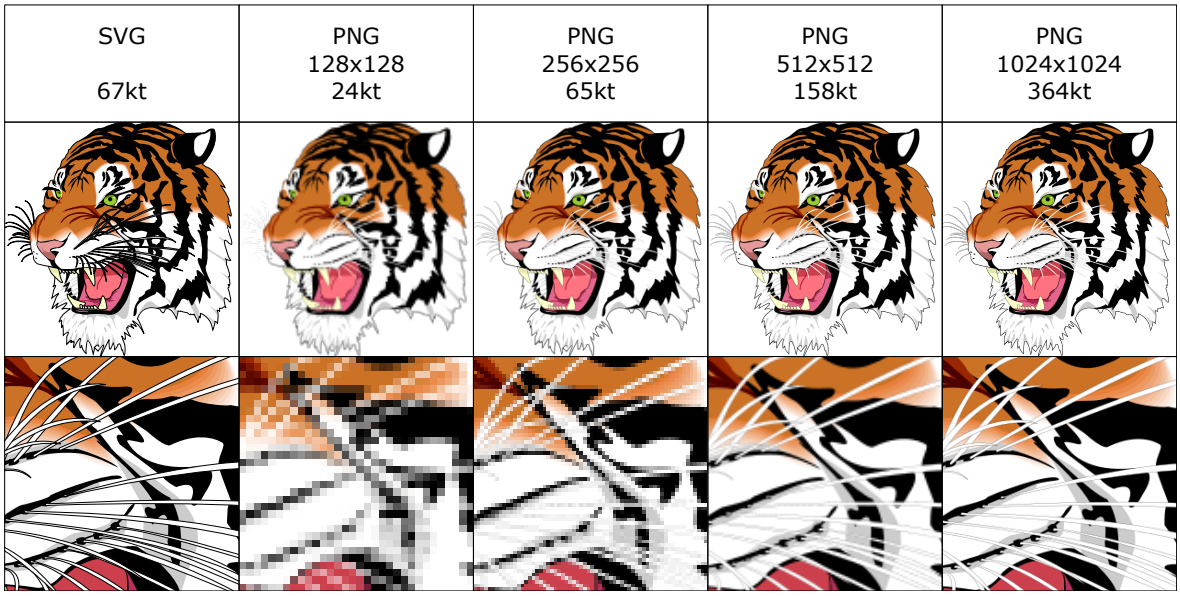
Videopelejä pelataan nykyään monilla erikokoisilla ja -resoluutioisilla näytöillä. Esimerkiksi Bethesda Softworksin julkaisema Fallout Shelter -peli on saatavilla älypuhelimille, tietokoneille ja pelikonsoleille (Bethesda Softworks 2018). Käyttäessä perinteistä bittikarttagrafiikkaa kehittäjien täytyy tehdä kompromisseja kohdentaessaan pelejään eri tarkkuuksisille näytöille. Vaihtoehtoja ovat kuvan venyttäminen eri kokoihin, mikä tuottaa epätarkkoja tai pikselöityneitä kuvia, tai monen eri kokoisen kuvan tekeminen, mikä lisää työmäärä ja lopullisen pelin tiedostokokoa. (Drapeau 2016.)

Vektorigrafiikka on yksi ratkaisu näihin ongelmiin. Se sallii kuvan venyttämisen eri kokoiseksi ilman laadun kärsimistä. Vektorigrafiikkatiedosto, kuten SVG, on usein paljon pienempi kooltaan kuin bittikarttatiedosto, kuten PNG (Parsons 2017; Spuy 2010). Kuviosta 1 voi huomata, että jos kuvan haluaa esittää skaalattuna alimman rivin kokoiseksi, vasta 512x512 pikselin versiossa on tarpeeksi resoluutiota, eikä se näytä epäselvältä, toisin kuin sitä pienemmät versiot. 512x512 pikselin version tiedostokoko on kuitenkin yli kaksinkertainen suhteessa alkuperäisen SVG-kuvan tiedostokokoon. Jos kehittäjä haluaa pelin tukevan kaikkia kuviossa mainittuja resoluutioita, tulee yhteenlasketuksi tiedostokooksi 612 kilotavua, yhdeksän kertaa alkuperäisen SVG-kuvan tiedostokoko.

SVG-tiedosto tukee myös animointia, ja yhdessä tiedostossa voi olla monta animaatiota (SVG Working Group 2011). Tällä tavalla esimerkiksi pelihahmolle saa samaan tiedostoon toimeton-, juoksu- ja hyppyanimaation. Kilgard ja Bolz (2012) mukaan vektorigrafiikan renderöinti on perinteisesti ollut hidasta, eikä sitä ole voitu suorittaa tarpeeksi nopeasti, jotta se olisi kannattavaa. Nykyiset teknologiat ja laitteistot kuitenkin mahdollistavat vektorikuvien esittämisen huomattavasti aiempaa nopeammin. Lisäksi vektorikuvia on mahdollista esittää 3D-gafiikan seassa. (Kilgard ja Bolz 2012.) Kysymykseksi jääkin, miksei vektorigrafiikan käyttö videopeleissä ole yleistynyt, vaikka teknologia sen sallii. Mitä rajoitteita vektorigrafiikoiden käytössä on? Miten vektorigrafiikoita on käytetty aiemmin videopeleissä ja miten niitä käytetään nykyään?

Tässä kandidaatintutkielmassa tutkitaan vektorigrafiikoiden käyttöä videopeleissä kirjalli-

suuskatsauksen metodein. Vektorigrafiikoiden etuja ja haittoja verrataan bittikarttagrafiikoihin 2D-pelien kontekstissa, mutta myös 3D-objektien teksturointia sivuten. Lisäksi tehdään katsaus peleissä käytetyn vektorigrafiikan historiasta ja nykytilasta.



Kuvio 1: SVG-kuvan ja erikokoisten PNG-kuvien vertailua. Ylemmällä rivillä on alkuperäinen kuva ja alemmalla suurennos yksityiskohdasta.

2 Vektorigrafiikka

2.1 Vektorigrafiikat ja niiden käyttö

Vektorigrafiikat ovat tietynlaisia tietokoneella generoituja kuvia. Toisin kuin bittikarttagrafiikat, jotka sisältävät jokaisen pikselin väriarvot, vektorigrafiikat sisältävät tietoa pisteistä ja niiden väliin tulevista suorista ja käyristä viivoista sekä muista muodoista (Parsons 2017; Spuy 2010). Erilaista tietosisältöä ovat esimerkiksi viivan väri, paksuus, katkonaisuus ja käyttäytyminen kulmissa (terävät, pyöreät tai tylpät kulmat). Viivan sisäpuolen voi täyttää yhtenäisellä värillä, liukuvärillä tai kuviolla, ja sitä voi myös leikata toisella viivalla. (SVG Working Group 2011.) Koska vektorigrafiikan pisteitä ei ole sidottu pikseleihin vaan omaan koordinaatistoonsa, voidaan sitä skaalata minkä kokoiseksi tahansa ilman, että kuvanlaatu kärsii tai tiedostokoko muuttuu (Spuy 2010).

Käytettyjä vektorigrafiikan tiedostomuotoja ovat esimerkiksi Encapsulated PostScript (EPS), Flash (SWF), Adobe Illustrator (AI) ja Scalable Vector Graphics (SVG) (Parsons 2017). Tämä tutkielma keskittyy vektorigrafiikoiden tarkasteluun SVG-tiedostomuodon pohjalta, sillä se on W3C:n standardi vektorigrafiikoille. SVG:n spesifikaatio on myös kattavampi kuin esimerkiksi EPS:n (Adobe Systems 1992) ja SWF:n (Adobe Systems 2012). SVG-tiedostomuoto tukee edellä mainittujen ominaisuuksien lisäksi muun muassa bittikarttakuvien upottamista, tekstien lisäämistä ja animointia. (SVG Working Group 2011.)

Vektorigrafiikoilla on useita käyttökohteita. Parsons (2017) esittelee käyttökohteiksi muun muassa logot, tekniset piirustukset ja kaaviokuvat. Ganacim ym. (2014) lisäävät käyttökohteisiin esimerkiksi fontit, kartat, käyttöliittymät ja pelit. Seuraavissa kappaleissa esitellään perusteluita miksi vektorigrafiikat soveltuvat hyvin näihin käyttötarkoituksiin.

Yhdestä vektorikuvasta on helppo luoda erikokoisia versioita (Parsons 2017). Esimerkiksi yritysten logot voivat esiintyä monessa eri koossa pienistä käyntikorteista suuriin tienvarsimainoksiin. On siis tärkeää, että logosta on olemassa useita eri resoluutioisia versioita, tai yksi resoluutioriippumaton versio, vektorikuva. Samasta syystä kuin logojen, teksteissä käytettyjen fonttien tulee olla hyvin skaalautuvia. Nykyisin suurimmassa osassa fontteja kirjai-

mia esittävät kuvat ovatkin juuri vektorikuvia, mikä sallii tekstin esittämisen eri fonttikoilla (Parsons 2017). Vektorikuvaa on myös helpompi muokata kuin bittikarttakuvaa, sillä sen elementit ovat itsenäisiä objekteja (Parsons 2017). Jos yritys haluaa vaihtaa esimerkiksi värejä, fontteja tai elementtien sijoittelua logossaan, onnistuu se vektorikuvan kanssa helposti.

Vektorigrafiikoilla voidaan esittää laadukkaita interaktiivisia kuvaajia (Lang ja Nolan 2012). Artikkelissaan Lang ja Nolan esittelevät R-ohjelmointikielellä tekemäänsä työkalua, jolla pystyy luomaan datasta erinäköisiä SVG-kuvia. Kuvaan pystyy SVG-tiedoston ominaisuuksien johdosta lisäämään interaktiivisia toiminnallisuuksia. Esimerkiksi elementtejä saa animoitua, kun niitä osoitetaan tai klikataan hiirellä, ja liukusäätimestä voidaan säätää kuvassa esitettävää sisältöä. (Lang ja Nolan 2012.) Myös Googlen Charts API muodostaa SVG-muotoisia interaktiivisia kaavioita sille syötetystä datasta (Google 2017).

Tietokoneen käyttöliittymät voivat olla toteutettu vektorigrafiikkana. Microsoftin Windows Presentation Foundationia (WPF) käyttävät ohjelmat ovat vektoripohjaisia. WPF lupaa, että kehittäjän ei tarvitse murehtia päätelaitteen monitorin koosta tai resoluutiosta, sillä se hoitaa skaalaamisen sopivan kokoiseksi. (Chappell 2006.) Unixin työpöytäympäristö KDE 4:n grafiikat ovat vektorigrafiikoita. Työpöytä ja siihen liittyvät elementit ovat joukkoja SVG-kuvia ja niitä muokkaamalla voidaan säätää värejä ja muotoja. Kaikki ikonit ovat SVG-kuvia, mutta reaaliaikaisen renderöinnin sijasta ikonista pyydetään tietynkokoista versiota, joka rasteroidaan eli renderöidään bittikarttakuvaksi välimuistiin. Tällä tavalla parannetaan suorituskykyä, jota vektorikuvan jatkuva uudelleenrenderöinti heikentäisi. (Rusin 2008.)

Vektorigrafiikoita käytetään peleissä eri tavoin. Harvey (2013) kertoo artikkelissaan, kuinka Guacamelee! -pelin eri vaiheissa käytettiin vektorigrafiikoita. Ensimmäiset pelin luonnokset olivat Flashilla tehtyjä vektorianimaatioita. Myös pelin lopulliset grafiikat tehtiin Flashilla, mikä takasi sulavat animaatiot ja säästi muistia. (Harvey 2013.) Ronaghan (2012) tekemässä haastattelussa Brittany Aubert kertoo Scribblenauts Unlimited -pelin grafiikoiden olevan vektorikuvia, jotta objekteja voisi venyttää ilman kuvanlaadun kärsimistä. Stoic (2013) puolestaan kertoo artikkelissan Banner Saga -pelin animointiprosessista. Animoitavat liikkeet kuvattiin ihmisten näyttelemänä ja rotoskoopattiin Flashilla sarjaksi vektorikuvia. Sen jälkeen Flash-animaatiot rasteroitiin joukoksi bittikarttakuvia. (Stoic 2013.)

2.2 Vektorigrafiikan ja bittikarttagrafiikan eduista ja haitoista

Vektorigrafiikasta puhuttaessa tulee väistämättä esiin sen erot bittikarttagrafiikkaan. Usein vektorigrafiikka määritelläänkin juuri sillä, mitenkä se eroaa bittikarttagrafiikasta (ks. luku 2.1). Kuten aiemmin on jo ilmennyt, vektorigrafiikan olennaisin ero ja etu bittikarttagrafiikkaan verrattuna on sen resoluutioriippumattomuus. Vektorigrafiikalla on monia muitakin etuja bittikarttagrafiikkaan verrattuna, mutta myös monia haittoja. Tässä luvussa eritellään näitä etuja ja haittoja.

Koska vektorigrafiikka koostuu yksittäisistä elementeistä, käyrien muodostamista muodoista, on sitä helppo muokata (Parsons 2017). Bittigrafiikat ovat taas niin sanotusti tyhmiä, ne tietävät vain kuvan mitat ja jokaisen yksittäisen pikselin värin, mutta eivät mitään kuvan esittämistä muodoista (Spuy 2010). Esimerkiksi viivalla rajatun alueen värin muuttaminen bittikarttakuvassa on koettu usean internetkeskustelun perusteella ongelmalliseksi (Jessicaw1988 2016; Edemardil 2012; Mohamd 2012). Vektorikuvan kanssa alueen värin vaihtaminen on kuitenkin helppoa, sillä tarvitsee vain määritellä kyseessä olevan elementin täyttöväri. Elementtien liikuttelu onnistuu vektorikuvassa helposti vaihtamalla pisteiden sijainteja, ja taustalla oleva elementti tulee silloin luonnollisesti näkyviin. Sen sijaan bittikarttakuvassa pitää ensin valita kaikki liikutettavat pikselit, jotka sitten liikkuessaan jättävät aukon kuvaan. (Parsons 2017.)

Vektorigrafiikan elementtipohjaisuus mahdollistaa sen helpon animoimisen. Elementtien ominaisuuksia kuten sijaintia ja väriä pystyy muuttamaan ajan suhteen ohjelmallisesti, jolloin animoijan ei tarvitse määrittää kuin muutama avainkuva (keyframe) ja tietokone luo niiden väliin jäävät kuvat automaattisesti. Bittikartta-animaatiota tehdessä pitää valita käytettävä kehysnopeus ja piirtää sen mukainen määrä yksittäisiä kuvia. (Parsons 2017.) Jos bittikartta-animaation kehysnopeuden haluaakin esimerkiksi kaksinkertaistaa, tulee myös työmäärä kaksinkertaiseksi. Sen sijaan vektorianimaatio ei välitä ollenkaan kehysnopeudesta, vain animaation kesto tiedetään ja kehysnopeus määräytyy käytettävän laitteiston perusteella. Tehokas tietokone voi siis piirtää monta sataa kuvaa sekunnissa, mutta pienitehoinen kone piirtää vain muutamia kuvia.

Bittikarttagrafiikan suurin etu onkin juuri suorituskyvyssä. Vektorigrafiikkakuvan renderöin-

ti vaatii monia matemaattisia laskutoimituksia, jotta käyriä kuvaavasta datasta saadaan piirrettyä pikseleistä koostuva kuva. Bittikarttakuvaan onkin tallennettu pikseleistä koostuva kuva jo valmiiksi, joten ylimääräistä laskemista ei tule ollenkaan. (Spuy 2010.) Uusien nopeampien algoritmien kehittäminen vektorigrafiikan renderöimiseen on mielenkiintoinen tutkimuksen kohde. Esimerkiksi Nehab ja Hoppe (2007) esittelevät algoritmia, joka nopeuttaa vektorikuvasta mielivaltaisesti otetun osan renderöimistä. Bittikarttakuvasta on tehokasta ottaa mielivaltainen osa, sillä kaikkien pikselien väri tiedetään jo valmiiksi. Sen sijaan vektorikuvasta mielivaltaisen osan ottaminen on epätehokasta, koska koko kuvan kaikki käyrät on joka tapauksessa pakko laskea. Nehabin ja Hoppen algoritmi tarjoaa ratkaisun tällaiseen ongelmaan. (Nehab ja Hoppe 2007.) Kilgard ja Bolz (2012) tarjoavat tehokkaan tavan hyödyntää näytönohjainta perinteisesti prosessorilla tapahtuvaan vektorigrafiikan renderöimiseen.

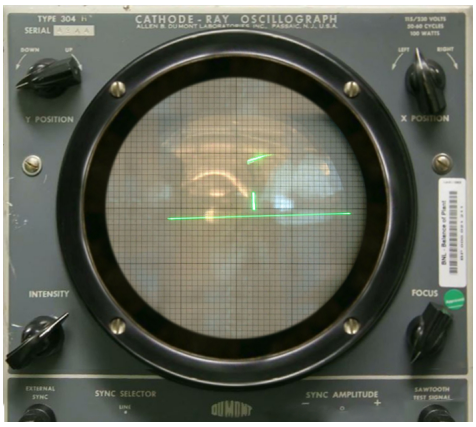
Vektorikuvan tiedostokokoon suhde bittikarttakuvan tiedostokokoon riippuu kuvan monimutkaisuudesta ja resoluutiosta. Vektorikuvan tiedostokokoon vaikuttaa kuvan monimutkaisuus: mitä enemmän viivoja sitä suurempi tiedosto. Bittikarttakuvan kokoon vaikuttaa sen resoluutio: mitä suurempi kuva sitä suurempi tiedosto. Käyttökohteissa, joissa kuva on yksinkertainen, on vektorikuvan tiedostokoko pienempi. Jos esitetään monimutkaista kuvaa kuten valokuvaa, on bittikarttakuvan tiedostokoko pienempi. Tiedostokoon puolesta vektorikuva on parempi valinta kuin bittikarttakuva yksinkertaiseen grafiikkaan kuten logoon, mutta monimutkaiseen grafiikkaan, jossa on paljon yksityiskohtia, on bittikarttakuva soveltuvampi. (Bryant ja Jones 2012.)

3 Vektorigrafiikat videopeleissä

3.1 Vektorigrafiikan historia peleissä

Vektorigrafiikoilla on ollut merkittävä osa videopelien historiassa. Ensimmäiset graafiset pelit käyttivät näyttöinään oskilloskooppeja ja satunnaisskannausnäyttöjä (random scan display), laitteita, jotka pikseleiden sijasta esittävät viivoja eli vektoreita. Tällaisia näyttöjä kutsutaan vektorinäytöiksi. Vektorinäytöillä oli monia etuja rasterinäyttöihin verrattuna, kuten elementtien helppo pyörittäminen ja koon muuttaminen. (Montfort ja Bogost 2009.)

William Higinbothanin vuonna 1958 kehittämä Tennis for Two (ks. kuvio 2a) on yksi ensimmäisistä graafisista videopeleistä (Donovan 2010). Tennis for Twossa on vain muutama yksinkertainen elementti, viivat esittämässä maata, verkkoa ja tennismailoja sekä piste esittämässä palloa, joten se soveltuu hyvin oskilloskoopilla pelattavaksi. Steve Russellin kumppaneineen vuosina 1961 ja 1962 kehittämä Spacewar! (ks. kuvio 2b) toimi alun perin Massachusetts Institute of Technologyn (MIT) Digital Equipment Corporationilta (DEC) saadulla PDP-1 -tietokoneella. Pelin suuren suosion seurauksena sitä levitettiin toisiin yliopistoihin ja siitä tehtiin versioita eri tietokoneille, tehden siitä ensimmäisen videopelin, joka on levinnyt yhden yliopiston ulkopuolelle. (Donovan 2010; Rutter ja Bryce 2006.)



(a) Tennis for Two (Brookhaven National Laboratory 2013)

(b) Spacewar! (Clayton 2007)

Kuvio 2: Restauroidut versiot varhaisista videopeleistä.

Tennis for Two ja Spacewar! olivat uraa uurtavia videopelejä aikana, jolloin tietokoneet maksoivat satojatuhansia dollareita ja olivat vain vakavaa tieteellistä tai kaupallista käyttöä varten. Higinbothan ja Russell ynnä muut kuitenkin osoittivat, että tietokoneista oli muuhunkin: paitsi että turhanpäiväisten pelien ja sovellusten tekeminen oli hauskaa, sillä oli myös käytännön hyötyä. Spacewar! käytti lähes jokaista PDP-1 -tietokoneen käskyä, joten DEC alkoi käyttää peliä testatessaan ja esitellessään uusien koneidensa toimintaa (Kossow 2008). 1970-luvulla videopelejä alettiin kehittää vakavasti ja tuomaan suuren yleisön saataville. 1971 julkaistiin ensimmäiset kolikkopelit Galaxy Game ja Computer Space, jotka molemmat olivat saaneet innoituksensa Spacewar! :sta. Seuraavana vuonna Magnavox julkaisi ensimmäisen kotikonsolin Magnavox Odysseyn. Apple Computer, Commodore ja Tandy julkaisivat vuonna 1977 ensimmäiset massatuotetut kotitietokoneet, joista suosituin oli Apple Computerin pelaamiseen keskittynyt Apple II. (Donovan 2010.)

Ensimmäiset kolikkopelikoneet, pelikonsolit ja kotitietokoneet käyttivät rasterinäyttöjä, muun muassa sen takia, että vektorinäytöt olivat liian kalliita näihin tarkoituksiin. Larry Rosenthal kuitenkin onnistui rakentamaan tarpeeksi halvan vektorinäytön pelihalleihin, ja vuonna 1977 julkaistiin ensimmäinen vektorinäytöllinen kolikkopeli Space Wars, joka oli Galaxy Gamen ja Computer Spacen tavoin saanut innoituksensa Spacewar! :sta. Vektorinäytön grafiikat olivat huomattavasti terävemmät verrattuna sen aikaisiin pieniresoluutioisiin rasterinäyttöihin, ja tämä saivat pelinkehittäjät innostumaan; syntyivät klassikkopelit Lunar Lander (1979) ja Asteroids (1979), sekä ensimmäiset 3D-pelit Tailgunner (1979) ja Battlezone (1980). Vektorinäyttöjen suosio ei kuitenkaan kestänyt kovin kauan, kun rasterinäytöt kehittyivät vuosikymmenten vaihteessa tarkemmiksi ja värikkäämmiksi. Vuoden 1983 videopelilaman myötä pelihallit eivät enää tilanneet vektoripelejä, eivätkä vektorinäytöt pysyneet rasterinäyttöjen kehityksen perässä. (Donovan 2010.)

Smith Engineeringin kehittämä, 1982 julkaistu Vectrex on ensimmäinen ja ainoa vektorinäytöllinen kotikonsoli. Vectrex menestyi melko hyvin, mutta sen menestys ei kestänyt kuin reilun vuoden. Videopelilaman takia sen valmistus lopetettiin jo 1984. Myöhemmin Vectrex on kuitenkin saanut melko merkittävän aseman retropeliharrastajien keskuudessa, ja se on suosittu alusta kotitekoisten pelien (homebrew games) tekemiseen. (Wolf 2012.)

3.2 Vektorigrafiikan nykytila peleissä

Vektorinäyttöjä ei enää ole laajasti saatavilla, mutta monet pelit matkivat niiden tyyliä. Geometry Wars: Retro Evolved (2005) ja Groov (2009) ovat hyviä esimerkkejä moderneista peleistä, jotka ovat saaneet inspiraationsa vanhoista vektoripeleistä. (Donovan 2010.) On kuitenkin epäselvää, käyttävätkö nämä pelit todellisuudessa vektorigrafiikoita vai bittikarttakuvia, sillä niiden kehittämisestä on hyvin vähän tietoa saatavilla. Sama tiedon puute koskee myös muita pelejä, jotka tyyliltään näyttävät siltä, että voisivat käyttää vektorigrafiikoita. Pelkästään peliä tarkastelemalla on hankala sanoa, käyttääkö se vektori- vai bittikarttagrafiikoita, sillä bittikarttakuvat voivat olla rasteroituja vektorikuvia, näyttäen tietyssä resoluutiossa täysin samalta. Tässä luvussa esitellään joitain videopelejä, jotka käyttävät tiettävästä vektorigrafiikoita ja joiden kehittämisestä löytyy dokumentaatiota.

Yksi näkyvimmin vektorigrafiikoita hyödyntävistä peleistä on Ian Synderin vuonna 2014 julkaisema The Floor is Jelly (ks. kuvio 3a). Uniikissa Flashilla ohjelmoidussa tasohyppelypelissä seikkaillaan maailmassa, joka on nimensä mukaisesti tehty hyytelöstä. Hyppiminen ja seiniin törmääminen saa aikaan hytkyvän efektin, joka on mahdollistettu käyttämällä vektorigrafiikoita. (Synder 2014.) Blogissaan Synder (2011) kertoo pelimaailman muodostuvan sarjasta pisteitä ja niiden sisään jäävästä väritetystä alueesta. Hyytelöefekti on toteutettu niin, että pisteitä liikutetaan iskun seurauksena aallon tavoin, ja lopulta ne palaavat takaisin lähtöpaikoillensa (Synder 2011). Pelin (Synder 2014) tiedostojen tarkastelu osoittaa, että pelimaailman lisäksi myös muut pelin grafiikat, kuten pelihahmot ja kasvillisuus, ovat vektorikuvia.

Guacamelee! on DrinkBox Studiosin 2013 julkaisema meksikolaisesta painista inspiroitunut toimintatasohyppelypeli. Guacamelee! on rakennettu DrinkBoxin omalla pelimoottorilla DrinkBox Enginellä, joka on ohjelmoitu C++:lla. Pelihahmojen grafiikat on tehty ja animoitu resoluutioriippumattomasti Flashilla, minkä jälkeen Flash-tiedostot on tuotu DrinkBox Engineen. Vektorikuvat ja -animaatiot mahdollistavat hahmojen skaalaamisen eri kokoisiksi ja 60 fps -animaation. (Harvey 2013.) Hahmoista poiketen pelin tasot ja taustat on tehty kerrostetuista 3D-kappaleista, mikä tuo peliin syvyyden vaikutelmaa (Quijano 2014).

5Antsin kehittämä ja Rovio Starsin vuonna 2013 julkaisema Tiny Thief on Robin Hood -

sävytteinen point-and-click -seikkailupeli Androidille, iOS:lle, Windowsille, OS X:lle ja Wii U:lle (Diener 2013; Steam 2013; Abylight 2015). 5Antsilla oli kehityksen aikana vaikeuksia Flashilla toteutettujen vektorianimaatioiden ja Adobe AIR -ajoympäristön kanssa. Tiny Thiefin yksi pääelementti oli sen käyttämät 3000 animaatiota, mutta Adobe AIRin rajoitteiden takia vektorigrafiikat eivät toimineet tarpeeksi tehokkaasti joillain laitteilla. Ongelma ratkesi vaihtamalla Adobe AIR Scaleform Mobile SDK:hon, jossa ei ollut samoja rajoitteita. (Diener 2013; Eccentric Engine 2014.) Tiny Thiefin kehittäminen on sittemmin lopetettu ja se on poistettu sovelluskaupoista (Rovio 2016).

Vaikka 3D-grafiikat pohjautuvatkin vektorigrafiikoihin (Parsons 2017), käytetään niiden teksturoimiseen bittikarttagrafiikoita (Wang ym. 2010). Bittikarttagrafiikoiden käyttö on ongelmallista, sillä läheltä tarkasteltaessa niistä tulee epäselviä, tai vaihtoehtoisesti ne pitää tallentaa erittäin suurikokoisina tiedostoina. Tämän ehkäisemiseksi on kehitetty tekniikoita, jotka hyödyntävät vektorigrafiikoita 3D-kappaleiden teksturoinnissa. (Green 2007.) Eräs tällaista tekniikkaa hyödyntävä peli on Valve Corporationin vuonna 2007 julkaisema Team Fortress 2 (ks. kuvio 3b). Pelissä esiintyvät tekstuurit on tallennettu etäisyyskenttinä (distance field), jotka ovat bittikarttakuvina koodattuja esityksiä vektorikuvista. Bittikarttakuvia ei esitetä käyttäjälle sellaisenaan, vaan niiden avulla lasketaan kuvan rajat, ja näistä rajoista muodostuvat vektorikuvat esitetään käyttäjälle. Tämä tekniikka mahdollistaa vektorigrafiikoiden esittämisen lähes yhtä tehokkaasti kuin bittikarttagrafiikoiden, mutta se aiheuttaa lievää epätarkkuutta lopulliseen kuvaan. (Green 2007.) 3D-kappaleiden tekstuurien esitys vektorikuvina saattaa tulla erityisen oleellisiksi virtuaalitodellisuutta hyödyntävissä peleissä, joissa venyneiden pikseleiden näkeminen rikkoo immersion tunnetta.

(a) The Floor is Jelly (Synder 2014)

(b) Team Fortress 2 (Valve Corporation 2007)

Kuvio 3: Vektorigrafiikoiden käyttö nykyaikaisissa videopeleissä.

3.3 Ongelmia vektorigrafiikan käytössä

Vektorigrafiikoiden käytössä on ongelmia, joiden takia ne eivät ole laajasti yleistyneet videopeleissä. Yksi suurimmista ongelmista on suorituskyky. Vaikka koko ajan kehitetään uusia tehokkaampia algoritmeja vektorigrafiikoiden renderöimiseen (esim. Kilgard ja Bolz 2012), bittikarttagrafiikoiden käyttäminen on huomattavasti tehokkaampaa kuin vektorigrafiikoiden (ks. luku 2.2). Tästä syystä kehittäjien on helpompi valita vektorigrafiikoiden sijasta bittikarttagrafiikat peleihinsä. Vektorigrafiikoiden käyttö aiheuttaa kehittäjille ylimääräisiä ongelmia suorituskyvyn kanssa, kuten 5Antsin tapauksessa (ks. luku 3.2).

Usein kehittäjillä ei ole mahdollisuutta vaikuttaa siihen, minkä tyyppisiä grafiikoita he voivat käyttää peleissään. Pelien pohjalla toimii aina pelimoottori, joka määrittelee millaisia ominaisuuksia pelissä voi olla. Pelimoottorista riippuu esimerkiksi se, millaisia grafiikkatyyppejä tuetaan. TIGAn (2014) tekemän tutkimuksen mukaan 53 prosenttia vastanneista pelinkehittäjistä ei käytä itse tekemiään vaan kolmannen osapuolen pelimoottoreita. Unity 3D Engine ja Unreal Engine olivat kyselyn perusteella kaksi eniten käytettyä kolmannen osapuolen pelimoottoria: vastaajista 62 prosenttia käytti Unitya ja 12 prosenttia Unreal Enginea (TIGA 2014). Unityn ja Unreal Enginen käyttöoppaista selviää, etteivät ne oletusarvoisesti tue vektorigrafiikkatiedostojen käyttöä (Unity Technologies 2018; Epic Games 2018), joten monen kehittäjän on tyydyttävä bittikarttagrafiikoihin.

TIGAn kyselyn perusteella kolmannen osapuolen pelimoottorien määrä olisi tulevaisuudessa yhä nousussa. Kyselyyn vastanneista vain 35 prosenttia aikoi jatkossa käyttää itse tekemiään pelimoottoreita. (TIGA 2014.) Tällainen kehitys siirtää vastuun tuetuista grafiikkatyypeistä yhä enemmän pois yksittäisten pelikehittäjien käsistä pelimoottorien kehittäjien käsiin. Onkin havaittavissa, että kiinnostus vektorigrafiikoita kohtaan olisi kasvamassa sekä pelimoottoreiden käyttäjien että niiden kehittäjien keskuudessa. Unityyn ja Unreal Engineen on kumpaankin saatavilla epävirallisia SVG-tiedostoja tukevia lisäosia, jotka eivät kuitenkaan tue kaikkia tiedoston ominaisuuksia (Stehlik 2017; Cultrarius 2017). Lisäksi heinäkuussa 2018 julkaistuun Unityn versioon 2018.2 sisältyy testiversio SVG-paketista, jonka avulla voidaan käyttää SVG-tiedostoja sekä luoda ja muokata vektorikuvia ohjelmallisesti. Unity ei renderöi SVG-kuvia perinteisesti, vaan pilkkoo kuvan automaattisesti sopivaksi määräksi helposti renderöitäviä kolmioita. (Krogh-Jacobsen 2018.)

Bittikarttagrafiikoiden käyttö vektorigrafiikkojen sijasta voi myös olla taiteellinen valinta. Vektorigrafiikat sopivat hyvin selkeiden muotojen esittämiseen, mutta tavoiteltaessa esimerkiksi realistista tyyliä bittikarttakuvat ovat parempi vaihtoehto (Parsons 2017). Realistisen tyylisten tai muuten monimutkaisten vektorikuvien tiedostokoko voi olla valtavan suuri (Bryant ja Jones 2012), ja niiden renderöimiseen tarvittava laskentateho kohtuuton. Monimutkaisen grafiikan tapauksessa vektorigrafiikoiden käyttö ei ole suositeltavaa (Bryant ja Jones 2012). NeoGAF (2015) -keskustelufoorumilla käydyn keskustelun perusteella monien mielestä vektorigrafiikat ovat yksinkertaisesti rumia, ja ne yhdistetään huonolaatuisiin ilmaisiin nettipeleihin, jotka on tehty Flashilla ja käyttävät vektorigrafiikoita. Vektorigrafiikkoihin tuntuukin liittyvän monia harhaluuloja ja ennakkoluuloja.

Vektorikuvien tekemisen työnkulku on erilainen kuin bittikarttakuvien, sillä piirtämisen ja maalaamisen sijasta määritelläänkin pisteiden sijainteja ja ominaisuuksia. Tästä syystä vektorigrafiikoiden tekeminen voi olla vaikeaa niille, jotka ovat tottuneet tekemään videopeligrafiikoita bittikarttagrafiikkoina. Ne puolestaan, jotka ovat tottuneet tekemään vektorigrafiikoita, ovat usein tottuneet tekemään graafisia esityksiä, kuten logoja, pelihahmojen sijasta. Nykyaikaiset vektorigrafiikkaohjelmat, kuten Adobe Systemsin (2018) Adobe Illustrator ovat pyrkineet madaltamaan vektorigrafiikoiden tekemisen kynnystä mahdollisimman paljon. Adobe Illustrator tarjoaa monia bittikarttagrafiikkaohjelmista tuttuja ominaisuuksia, kuten erilaisia siveltimiä, joiden avulla vektorikuvaan voi lisätä viivoja ikään kuin maalaten. (Adobe Systems 2018.)

4 Yhteenveto

Vektorigrafiikoiden käytön videopeleissä voi jakaa kahteen aikakauteen: ennen ja jälkeen vuoden 1983 videopelilaman. Ennen lamaa kukoistivat niin sanotut oikeat vektoripelit, eli pelit, joita pelattiin vektorinäytöillä. Laman jälkeen siirryttiin käyttämään rasterinäyttöjä eivätkä vektorigrafiikat enää täyttäneet yleisön vaatimuksia. Lähes kaikki pelit 80-luvulta lähtien käyttävätkin bittikarttagrafiikoita. Vektorigrafiikoiden renderöinti rasterinäytölle sopivaksi vaatii paljon laskentatehoa, jota varsinkaan 80-luvun tietokoneissa ei ollut. Teknologian kehittymisen myötä vektorigrafiikoita on voitu hyödyntää paremmin, mutta vieläkään niistä ei ole tullut yleisesti käytettyjä videopeleissä. Edes viimeisen kymmenen vuoden ajalta on vaikea löytää vektorigrafiikoita käyttäviä pelejä, mikä on osaltaan hankaloittanut tämän tutkielman tekoa.

Tehokkaampien algoritmien ja laitteistojen kehittäminen antaa mahdollisuuden tehdä pelejä, joissa on enemmän ja monimutkaisempia vektorikuvia. Tekniikan kehittyminen parantaa kehittäjien kykyä käyttää peleissään vektorikuvia graafisesta tyylistä riippumatta. Olemassa olevien ominaisuuksien parantamisen lisäksi vektorigrafiikoihin kehitetään myös uusia ominaisuuksia. Esimerkiksi Orzan ym. (2008) esittävät täysin uudenlaista vektorigrafiikkaelementtiä, diffuusiokäyrää. Käyrien välinen alue täytetään käyrien värejä vastaavilla liukuväreillä, ja näin saadaan luotua erittäin monimutkaisia vektorikuvia vain muutamalla käyrällä. (Orzan ym. 2008.) Uudet teknologiat mahdollistavat vektorigrafiikoiden käytön sellaisissa graafisissa tyyleissä, joissa ne eivät ole ennen toimineet.

Teknisesti on siis täysin mahdollista käyttää vektorigrafiikoita videopeleissä, mutta käytännössä siinä on vielä monia ongelmia. Puuttuvan pelimoottorituen ja virheellisten ennakkoluulojen takia käytetään mieluummin bittikarttagrafiikat harkitsemattakaan vektorigrafiikoita. Unityn SVG-paketin olemassaolo kuitenkin kertoo vektorigrafiikoille olevan niin paljon kysyntää, että tällaisen paketin tekeminen koetaan liiketoiminnallisesti kannattavana. Vektorifiikoiden näkyvyyden lisääntyminen pelinkehittäjien keskuudessa lisää vektorigrafiikoiden käyttöä ja samalla huomataan, etteivät vektorigrafiikat automaattisesti tarkoita alkeellisia ja yksinkertaisia grafiikoita. Kehittyneiden teknologioiden ansiosta on mahdollista tehdä edistyneitä ja monimuotoisia vektorikuvia, joita voi silti renderöidä reaaliaikaisesti.

Aiheeseen liittyvä tieteellinen tutkimus on vähäistä ja tutkielman myötä herää monia tutkimuskysymyksiä tulevaisuuteen. Olisi mielenkiintoista selvittää vektorigrafiikoiden käytön osuus kaikissa videopeleissä. Entä miten vektorigrafiikoiden käyttö vaikuttaa pelin kehittämiseen käytettyyn aikaan ja budjettiin, tai lopullisen pelin tiedostokokoon ja suorituskykyyn? Tätä voitaisiin jatkossa tutkia esimerkiksi toteuttamalla samanlainen peli käyttäen sekä bittikarttagrafiikoita että vektorigrafiikoita ja vertailemalla käytettyjä työmääriä ja pelien suorituskykyjä. Tieteellinen tutkimus voisi tuoda lisää näkyvyyttä vektorigrafiikoiden käytölle videopeleissä ja mahdollisesti vähentää niihin liittyviä ennakkoluuloja.

Lähteet

Abylight. 2015. *Nintendo, 5Ants and Abylight working together to take forward the Wii U version of Tiny Thief.* Luettu 18.11.2018, marraskuu. http://abylight.com/nintendo-5ants-and-abylight-working-together-on-wii-u-version-of-tiny-thief/.

Adobe Systems. 1992. *Encapsulated PostScript File Format Specification,* toukokuu.

———. 2012. *SWF File Format Specification Version 19.*

———. 2018. *Adobe Illustrator,* lokakuu.

Bethesda Softworks. 2018. *Fallout Shelter.* Luettu 29.10.2018. https://www.fallout shelter.com/.

Brookhaven National Laboratory. 2013. *Tennis For Two on a DuMont Lab Oscilloscope Type 304-A,* elokuu. https://commons.wikimedia.org/wiki/File:Tennis_For_Two_on_a_DuMont_Lab_Oscilloscope_Type_304-A.jpg.

Bryant, Jay, ja Mike Jones. 2012. *Pro HTML5 performance.*

Chappell, David. 2006. *Introducing Windows Presentation Foundation.* Luettu 4.11.2018, syyskuu. https://msdn.microsoft.com/en-us/library/aa663364.aspx.

Clayton, Nik. 2007. *SpaceWar!,* syyskuu. https://www.flickr.com/photos/nikclayton/1394377691.

Cultrarius. 2017. *SVG Importer Plugin.* Luettu 18.11.2018. https://www.unrealengine.com/marketplace/svg-importer-plugin.

Diener, Matthew. 2013. *The value of a capable accomplice: The making of Tiny Thief.* Luettu 18.11.2018, syyskuu. https://www.pocketgamer.biz/feature/53474/the-value-of-a-capable-accomplice-the-making-of-tiny-thief/.

Donovan, Tristan. 2010. *Replay: The History of Video Games.*

Drapeau, Martin. 2016. *Scaling your Mobile Game to Any Device Size.* Luettu 30.10.2018, kesäkuu. https://medium.com/@martindrapeau/scaling-your-mobile-game-to-any-device-size-4d12dd79cad6.

Eccentric Engine. 2014. *Tiny Thief Brought The Innocence Back To Gaming*. Luettu 18.11.2018, toukokuu. https://www.redbull.com/in-en/tiny-thief-brought-the-innocence-back-to-gaming.

Edemardil. 2012. *Bucket Fill Outline Issue.* Luettu 4.11.2018, marraskuu. https://forums.getpaint.net/topic/25548-bucket-fill-outline-issue/.

Epic Games. 2018. *Texture Import Guide.* Luettu 18.11.2018. https://docs.unrealengine.com/Engine/Content/Types/Textures/Importing.

Ganacim, Francisco, Rodolfo Lima, Luiz de Figueiredo ja Diego Nehab. 2014. *Massively-parallel vector graphics,* marraskuu. doi:10.1145/2661229.2661274. http://dl.acm.org/citation.cfm?id=2661274.

Google. 2017. *Using Google Charts.* Luettu 3.11.2018, helmikuu.

Green, Chris. 2007. *Improved alpha-tested magnification for vector textures and special effects,* elokuu. doi:10.1145/1281500.1281665. http://dl.acm.org/citation.cfm?id=1281665.

Harvey, Chris. 2013. *Postmortem: DrinkBox Studios' Guacamelee!* Luettu 2.11.2018, syyskuu. http://www.gamasutra.com/view/feature/200658/postmortem_drinkbox_studios_.php.

Jessicaw1988. 2016. *How to use the paint bucket without getting white edges around the paint?* Luettu 4.11.2018, syyskuu. https://forums.adobe.com/message/8986548#8986548.

Kilgard, Mark, ja Jeff Bolz. 2012. *GPU-accelerated path rendering,* marraskuu. doi:10.1145/2366145.2366191. http://dl.acm.org/citation.cfm?id=2366191.

Kossow, Al. 2008. *Oral History of Steve Russell,* elokuu.

Krogh-Jacobsen, Thomas. 2018. *2018.2 is now available.* Luettu 30.11.2018, heinäkuu. https://blogs.unity3d.com/2018/07/10/2018-2-is-now-available/.

Lang, Duncan Temple, ja Deborah Nolan. 2012. *Interactive and Animated Scalable Vector Graphics and R Data Displays,* tammikuu. http://econpapers.repec.org/article/jssjstsof/46_3ai01.htm.

Mohamd, Saif. 2012. *White pixels remains when using fill tool?* Luettu 4.11.2018, lokakuu. http://gimpchat.com/viewtopic.php?f=8&t=5567#p69683.

Montfort, Nick, ja Ian Bogost. 2009. *Random and Raster: Display Technologies and the Development of Videogames,* heinäkuu. doi:10.1109/MAHC.2009.50. https://ieeexplore.ieee.org/document/5223984.

Nehab, Diego, ja Hugues Hoppe. 2007. *Texel Programs for Random-Access Antialiased Vector Graphics,* heinäkuu.

NeoGAF. 2015. *Why don't 2D games use vector art?* Luettu 18.11.2018, helmikuu. https://www.neogaf.com/threads/why-dont-2d-games-use-vector-art.988508/.

Orzan, Alexandrina, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot ja David Salesin. 2008. *Diffusion Curves: A Vector Representation for Smooth-Shaded Images,* elokuu. doi:10.1145/1399504.1360691. https://hal.inria.fr/inria-00274768.

Parsons, June Jamrich. 2017. *New Perspectives on Computer Concepts 2018,* heinäkuu.

Quijano, Augusto. 2014. *The Art of Making Guacamelee! - From Folklore to Finish.*

Ronaghan, Neal. 2012. *Scribblenauts Unlimited Wii U Interview with 5TH Cell.* Luettu 2.11.2018, syyskuu. http://www.nintendoworldreport.com/interview/31785/scribblenauts-unlimited-wii-u-interview-with-5th-cell.

Rovio. 2016. *Why Isn't Tiny Thief Available In App Stores Anymore?* Luettu 18.11.2018.

Rusin, Zack. 2008. *SVG in KDE.* Luettu 4.11.2018. http://www.svgopen.org/2008/papers/104-SVG_in_KDE/.

Rutter, Jason, ja Jo Bryce. 2006. *Understanding digital games,* huhtikuu.

Spuy, Rex Van der. 2010. *AdvancED game design with Flash.*

Steam. 2013. *Tiny Thief on Steam.* Luettu 18.11.2018. https://web.archive.org/web/20131219041921/https://store.steampowered.com/app/257080.

Stehlik, Jaroslav. 2017. *SVG Importer.* Luettu 18.11.2018. https://assetstore.unity.com/packages/tools/sprite-management/svg-importer-38258.

Stoic. 2013. *Animation process.* Luettu 30.10.2018. https://stoicstudio.com/animation-process/.

SVG Working Group. 2011. *Scalable Vector Graphics (SVG) 1.1 (Second Edition),* elokuu. https://www.w3.org/TR/SVG11/.

Synder, Ian. 2011. *What technology/library do you use to create the "jelly effect"?* Luettu 16.11.2018, marraskuu. http://thefloorisjelly.tumblr.com/post/13477420108/what-technologylibrary-do-you-use-to-create-the.

———. 2014. *The Floor is Jelly,* tammikuu.

TIGA. 2014. *What game engines do you currently use?* Luettu 18.11.2018, elokuu. http://tiga.org/news/uk-developers-show-renewed-interest-in-consoles.

Unity Technologies. 2018. *Unity - Manual: Importing Textures.* Luettu 18.11.2018. https://docs.unity3d.com/Manual/ImportingTextures.html.

Valve Corporation. 2007. *Team Fortress 2,* lokakuu.

Wang, Lvdi, Kun Zhou, Yizhou Yu ja Baining Guo. 2010. *Vector solid textures,* heinäkuu. doi:10.1145/1778765.1778823. http://dl.acm.org/citation.cfm?id=1778823.

Wolf, Mark J. P. 2012. *Encyclopedia of video games: The Culture, Technology, and Art of Gaming,* elokuu.