TEKNIIKKA JA LIIKENNE

Tietotekniikka

Ohjelmistotekniikka

INSINÖÖRITYÖ

SVG-VEKTORIGRAFIIKAN SOVELTUVUUS SELAINPELEIHIN

Työn tekijä: Niko Korosuo
Työn ohjaaja: Miikka Mäki-Uuro
Työn ohjaaja: Erja Nikunen

Työ hyväksytty: __. __. 2009

Miikka Mäki-Uuro
lehtori

ALKULAUSE

Tämä insinöörityö tehtiin Metropolia Ammattikorkeakoululle kesän ja syksyn 2009 aikana. Kyseessä on ensikosketukseni SVG-tekniikkaan, jonka käyttömahdollisuuksien tutkiminen on ollut todella antoisaa.

Kiitän ohjaajia ja lukijaa kiinnostuksesta aihevalintaan sekä erityisesti Suomen Punaista Ristiä heidän päihdevalistusmateriaaliensa käyttömahdollisuudesta tässä työssä.

Järvenpäässä 3.11.2009

Niko Korosuo

Metropolia

TIIVISTELMÄ

Työn tekijä: Niko Korosuo

Työn nimi: SVG-vektorigrafiikan soveltuvuus selainpeleihin

Päivämäärä: 3.11.2009 **Sivumäärä:** 42 s.

Koulutusohjelma: Tietotekniikka

Ammatillinen suuntautuminen: Ohjelmistotekniikka

Työn ohjaaja: lehtori Miikka Mäki-Uuro

Työn ohjaaja: yliopettaja Erja Nikunen

Tämän työn tarkoituksena on selvittää SVG:n (Scalable Vector Graphics) sopivuus Internet-selaimella pelattavien tietokonepelien alustaksi. Tutkittavia näkökulmia ovat muun muassa tämänhetkinen selaintuki, grafiikan laatu, animointi, tuotantotavat, hiiren ja näppäimistön käyttömahdollisuudet sekä skriptien tarpeellisuus.

Aluksi työssä tutustutaan vektorigrafiikan luonteeseen ja sen etuihin bittikarttakuviin nähden. Samalla vilkaistaan muutamia jo olemassa olevia vektorigrafiikkamuotoja ja tehdään pikakurkistus rasterointiin. Sitten esitellään SVG yleisellä tasolla ja tehdään alustavia arvioita siitä, miksi se voisi soveltua käytettäväksi selainpeleissä. Tämän jälkeen syvennytään pelinäkökulmaan. Siinä otetaan selvää, miten peleissä vaadittavia ominaisuuksia saadaan toimimaan SVG:n kanssa. Lopuksi SVG:n soveltuvuus pelialustaksi todetaan itse tehdyllä pelillä. Esimerkkipeliksi toteutettiin Suomen Punaisen Ristin koulupäihdevalistusta tukeva seikkailupeli, jossa pelaaja joutuu tekemään valintoja. Pelin suunnittelu ja toteutus onnistuivat huolellisen ja jatkuvan testauksen ansiosta.

Tulosten perusteella monenlaisia SVG-pelejä voidaan tehdä suurelle yleisölle. Kerättyjen tietojen ja käytännön kokeilun avulla voidaan muodostaa arvio, mitä pelityyppejä SVG:llä voidaan toteuttaa. Rajoittavia tekijöitä havainnoidaan koko työn pituudelta.

Avainsanat: SVG, skaalautuva vektorigrafiikka, peliohjelmointi, web-teknologiat

Metropolia

ABSTRACT

Name: Niko Korosuo

Title: Feasibility of Scalable Vector Graphics in Web Browser Games

Date: 3.11.2009 **Number of pages:** 42

Department: Information Technology

Study Programme: Software Engineering

Instructor: Miikka Mäki-Uuro Senior Lecturer

Supervisor: Erja Nikunen Principal Lecturer

The objective of this study is to analyse feasibility of SVG (Scalable Vector Graphics) for computer games which are playable on web browsers. The main focus is related to practical issues such as current browser support, quality of graphics, animation, code production, input devices (mouse and keyboard) and necessity of scripts.

First the study takes a look into vector graphics in general and its advantages compared to bitmap graphics. It contains brief descriptions of some vector graphics technologies (other than SVG) and a quick look inside rasterizing. Then SVG and related technologies are introduced with considerations why SVG might be used in web browser games. After that follows the analysis how typical elements for computer games are implementable with SVG. Finally the feasibility of SVG in web browser games is demonstrated with a self-made example game. The game was an adventure game for supporting Red Cross's drug education in schools. The game design and implementation were successful due to careful and continuous testing.

The results show that at many kind of SVG games can be done for the wide public. With help of collected information and practical coding it is possible to make an estimation which game types apply for SVG. Limitation factors are noted across in the study.

Keywords: SVG, scalable vector graphics, game programming, web technologies

SISÄLLYS

LYHENTEET

API	*Application Programming Interface* (ohjelmointirajapinta). Tietokoneohjelman osien (funktioiden, metodien tai moduulien) esittelykokoelma, jonka kautta ohjelmoituja toimintoja voidaan käyttää.
CSS	*Cascading Style Sheets* (porrastetut tyyliarkit). W3C:n määrittelemä web-sivujen muotoilutekniikka.
DOM	*Document Object Model* (dokumenttioliomalli). W3C:n määrittelemä alusta- ja kieliriippumaton ohjelmointirajapinta, jonka XML- tai (X)HTML-puuhierarkia muodostaa. Käytetään pääsääntöisesti selainskriptien yhteydessä.
DTD	*Document Type Definition* (dokumenttityyppimääritys). Määrittelee muun muassa XML-, HTML- ja XHTML -hierarkioiden rakenteen.
EMF	*Enhanced MetaFile* (paranneltu metatiedosto). Microsoftin laatima vektorigrafiikkaa sisältävä tiedostomuoto, joka voidaan liittää kuvana osaksi Office-dokumenttia (WMF:stä edelleen kehitetty).
HTML	*HyperText Markup Language* (hypertekstinen merkkikieli). W3C:n määrittelemä julkaisukieli. Valtaosa web-sivuista käyttää tätä tekniikkaa.
MIME	*Multipurpose Internet Mail Extensions* (monikäyttöiset Internet-postin laajennukset). MIME on määritelty Internetin kehittämiseen keskittyvän IETF-yhteisön RFC-dokumentissa 2045 - 2049, joka tarjoaa mekanismit sähköisten viestien kuvaamiseen.
PDF	*Portable Document Format* (siirrettävä dokumenttimuoto). Adoben kehittämä digitaalinen tiedostomuoto, jolla dokumentit saadaan luotettavasti siirrettyä alustalta toiselle siinä muodossa, kuin ne on tarkoitettu esitettäväksi. PDF vastaa paperille tulostettua versiota, mutta tietokoneen ruudulta katseltuna.
SMIL	*Synchronized Multimedia Integration Language* (synkronoitu multimediaintegrointikieli). W3C:n määrittelemä tapa animoida XML-puuta reaaliajassa.
SPR	Suomen Punainen Risti. Vapaaehtoistyöhön perustuva kansalaisjärjestö.
SVG	*Scalable Vector Graphics* (skaalautuva vektorigrafiikka). W3C:n määrittelemä vektorigrafiikkamuoto.
URI	*Uniform Resource Identifier* (yhtenäinen resurssitunniste). Mekanismi, jolla Internetissä sijaitseva resurssi (esimerkiksi web-sivu tai tiedosto) voidaan yksilöidä.
URL	*Uniform Resource Locator* (yhtenäinen resurssisijainti). URI:n osajoukko, jonka avulla resurssin sijainti voidaan paikallistaa.
W3C	*World Wide Web Consortium*. Maailmanlaajuinen web-standardeja kehittävä konsortio.

WMF	*Windows MetaFile* (Windows-metatiedosto). Microsoftin laatima vektorigrafiikkaa sisältävä tiedostomuoto, joka voidaan liittää kuvana osaksi Office-dokumenttia.
XHTML	*eXtensible HyperText Markup Language* (laajentuva hypertekstinen merkkikieli). HTML-versio, joka täyttää XML:n muotoiluvaatimukset.
XML	*eXtensible Markup Language* (laajentuva merkkikieli). W3C:n määrittelemä joustava tekstipohjainen tiedostomuoto tiedon jäsentämiseen. SVG ja Microsoftin kehittämä XAML perustuvat tähän tekniikkaan.

1 JOHDANTO

Maailmanlaajuisen web-standardeja kehittävän konsortion, World Wide Web Consortiumin (W3C), määrittelemä SVG (Scalable Vector Graphics, skaalautuva vektorigrafiikka) on hitaasti tehnyt tuloaan Internet-selaimien integroiduksi ominaisuudeksi. SVG on vuonna 2001 standardoitu tekniikka, jolla voidaan tuottaa samankaltaista vektorigrafiikkaa kuin esimerkiksi Adobe Flashilla, Microsoft Silverlightilla tai Sun Javalla. SVG:tä tuotetaan ohjelmoijille tutulla tiedon jäsentämiseen käytettävällä XML-merkkikielellä (eXtensible Markup Language) ja sitä voidaan ohjata skripteillä. Tämä tutkimus selvittää SVG-grafiikan soveltuvuuden selainpeleissä käytettäväksi.

Tutkimusmenetelminä käytetään SVG:ssä vastaan tulevien käytännön asioiden huomioimista (kuten selaintuen kattavuus, animointi, pelin tallentaminen, syötelaitteet ja moninpeli verkon yli), niihin liittyvien ongelmien ratkaisemista ja SVG:n ominaisuuksien hyödyntämistä. Ennen näihin perehtymistä tutkimus aloitetaan johdatuksella vektorigrafiikan teoriaan ja SVG:n esittelyllä.

Tutkimuksen lopuksi SVG:n soveltuvuus selainpeleihin osoitetaan itse tehdyllä pelillä. Esimerkkipeliksi toteutettiin Suomen Punaisen Ristin koulupäihdevalistusta tukeva seikkailupeli, jossa pelaaja joutuu tekemään valintoja selvitäkseen eteenpäin kohti maalia. Pelin testaus on osa tutkimusta, ja se on dokumentoitu selostavassa muodossa. Testauksessa käsitellään grafiikan subjektiivista arviointia, staattista testausta ja suorituskykyä.

Tutkimuksessa tullaan keskittymään SVG-spesifikaation versioon 1.1, jonka World Wide Web Consortium (W3C) määritteli suositukseksi 14. tammikuuta 2003. Uudempaa stabiilia versiota ei tätä kirjoitettaessa ollut olemassa lukuun ottamatta mobiililaitteisiin suunnattua SVG Tiny 1.2:a.

Tutkimuksen ajankohtaisuus

SVG itsessään ei ole enää uusi asia. Spesifikaatio on säilynyt sellaisenaan jo yli kuuden vuoden ajan. Sen käyttö ei vain yleistynyt alkuvuosinaan, kenties siksi että sen kilpailijat olivat jo valmiiksi suosittuja, ja ehkä myös siksi, ettei kunnollista selaintukea ollut saatavilla oikein missään vaiheessa. Viime aikoina tällä saralla on kuitenkin tapahtunut muutoksia ja tapahtuu edelleen.

Kesäkuussa 2008 Mozilla julkaisi Firefox 3 -selaimen, joka (verrattuna edelliseen versioon) toi merkittäviä parannuksia SVG:n tuottamiseen etenkin grafiikkasuodattimien osalta (grafiikkasuodatin on grafiikkaan sovellettava lisätehoste, jonka avulla lopputuloksesta saadaan näyttävämpi). Pelinäkökulmasta tämä on hyvä asia, sillä selain on todella suosittu nykyisin (Refsnes Data 2009b; Net Applications 2009). Firefoxiin SVG-tuki tuli ensimmäisen kerran lokakuussa 2005 versioon 1.5 (SVG-parannuslista ja julkaisutiedot Mozillalta (2008; 2009)).

Lisäksi selainkilpailu on lisääntynyt aivan viime aikoina. Erittäin nopeat (Futuremarkin (2009) mittausten mukaan) Apple Safari- ja Google Chrome -selaimet julkaistiin Windows-käyttöjärjestelmälle vuonna 2008, ja molemmat osaavat tuottaa SVG-grafiikkaa ilman erillisiä selainlaajennuksia. SVG toimii niiden uusimmissa versioissa virheettömämmin kuin koskaan aikaisemmin. Parantunut selaintuki nostaa sen käyttöarvoa ja tekee kannattavaksi arvioida SVG:tä pelimahdollisuuksien näkökulmasta.

2 VEKTORIGRAFIIKKA YLEISESTI

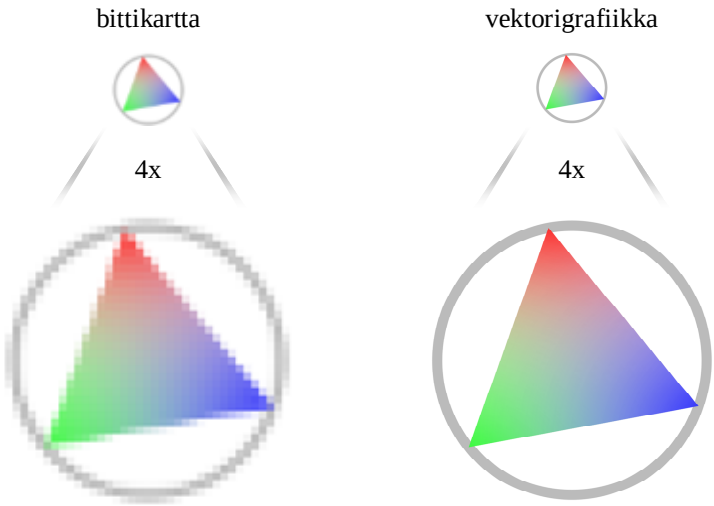
Vektorigrafiikka on kuvainformaatiota, jota koostetaan kaksiulotteiselle pinnalle (esimerkiksi tietokoneen näytölle tai paperille) koordinaatein ja matemaattisin kaavoin määritetyistä kuvioista. Vektorigrafiikkaa siis tuotetaan laskennallisesti. Tämän ansiosta vektorigrafiikkaa voidaan skaalata esitysnäkymässä loputtomiin niin, että lopputulos on aina terävä. (Microsoft 2007.)

2.1 Bittikartta vs. vektorigrafiikka

Perinteisesti kuvien esittämiseen on käytetty bittikarttoja. Bittikarttakuva koostuu eri värisistä neliön muotoisista pikseleistä, joita on kuvassa aina rajallinen määrä. Tällaista kuvaa suurennettaessa yksittäiset pikselit erottuvat, mikä ei välttämättä ole toivottava sivuvaikutus. Erilaisilla pehmennys- ja hahmontunnistusalgoritmeilla ongelmaa voidaan lieventää, mutta se ei poista sitä tosiasiaa, että bittikarttakuva sisältää vain rajallisen määrän tietoa. Siksi kuva on suurennettuna enemmän tai vähemmän suttuinen.

Tällainen rajoitus ei koske vektorigrafiikkakuvia, sillä ne luodaan aivan eri tavalla. Vektorigrafiikka toteutetaan geometrisista muodoista laskennallisesti ja siksi kuva on aina terävä koosta riippumatta. Vektorigrafiikka on omiaan esimerkiksi karttojen, teknisten piirustusten, kaavioiden ja logojen esittämi-

seen, koska kuviot ovat helposti määriteltävissä, yksinkertaisia ja kuvanlaatu ei kärsi suurennoksista. Tätä havainnollistaa kuva 1.



Kuva 1: Bittikarttakuvan ja vektorigrafiikkakuvan erojen esittely. Vektorigrafiikka on tarkka suurennettaessakin, koska se toteutetaan suoraan geometrisista muodoista.

Vektorigrafiikkaa ei suinkaan ole tarkoitettu bittikarttakuvien täydelliseksi korvaajaksi vaan lähinnä yksinkertaisten kuvioiden muodostamiseen. Bittikartta suoriutuu pääsääntöisesti paremmin digitaalisissa valokuvissa ja pienissä ikoneissa, ainakin piirtämiseen käytettävän laskenta-ajan ja tallennuskapasiteetin osalta (etenkin, kun bittikarttakuvia on tapana pakata).

Vektorigrafiikan toimivuus perustuu kuvauskieleen, joka voi periaatteessa muistuttaa luonnollista kieltä. Voidaan esimerkiksi komentaa tietokoneessa oleva vektorigrafiikan katseluohjelma muodostamaan viiva ohjeistamalla "aloita mustan viivan piirtäminen koordinaattipisteestä (1, 2) ja lopeta se pisteeseen (2, 5)". Tällöin täytyy vain varmistaa, että katseluohjelma ymmärtää oikein kyseisen toimintaohjeen. Käytännön syistä komennot eivät yleensä ole näin pitkiä, joten saman lopputuloksen aikaansaamiseksi komento voisi olla esimerkiksi "viiva: (1, 2) (2, 5) musta". Käytetystä kuvauskielestä riippumatta toimintaperiaate on kuitenkin sama: vektorigrafiikka saadaan visuaaliseen muotoon määrätynlaisia piirto-ohjeita ymmärtävän tulkin ja piirtämisen suorittavan rasterointiprosessin avulla (rasterointia käsitellään luvussa 2.3). Vektorigrafiikkatiedostot sisältävät siis kuvauskielelle sopivia kuvauksia eli piirto-ohjeita. SVG:n tapauksessa kuvauskieli on XML.

2.2 Vektorigrafiikkamuotoja

SVG:n standardoituessa yleisessä käytössä oli jo muita vakiintuneita vektorigrafiikkamuotoja. Kokonaiskuvan saamiseksi voidaan tässä vaiheessa tutustua niihin, sillä niiden kautta vektorigrafiikan luonne on yleensä tullut tutuksi ennen SVG:tä. Kunkin vektorigrafiikkamuodon pääsääntöiset käyttötarkoitukset ja ominaisuudet kuvaillaan lyhyesti. SVG:tä ei esitellä vielä tässä luvussa; tämä on vasta yleistietoa.

2.2.1 PostScript

PostScript on Adoben kehittämä grafiikan ja tekstin (paperi-)tulostamiseen suunniteltu sivunkuvauskieli. Grafiikka voi olla sekä vektorikuvaa että bittikarttaa.

PostScript tarjoaa tavan kuvailla kuvioita ja kuvia laiteriippumattomalla tavalla. Laiteriippumattomuus saavutetaan ottamatta kantaa kohdelaitteen ominaisuuksiin, jotta samaa kuvausta kyettäisiin käyttämään missä tahansa PostScript-tulostimessa ilman mitään muunnoksia (Weingartner 2006, luku "What is PostScript"). Tulostimen on siis oltava PostScript-yhteensopiva. Koska PostScript on kieli, se ei sellaisenaan sovellu näyttöruudulla esitettäväksi ilman sopivaa tulkkia. Adobe on laatinut dokumenttien esikatselua varten PDF-tiedostoformaatin (Portable Document Format), joka on käytännössä tulkattu PostScript-tiedosto (Evans 2009).

PostScript-tiedosto voi myös olla kapseloituna Encapsulated PostScript -tiedostomuotoon, joka mahdollistaa esimerkiksi rasteroidun esikatselukuvan liittämisen mukaan. Näin se voidaan helpommin liittää osaksi muita dokumentteja. (Adobe 2008a, s. 351.)

PostScript-vektorikuvia voidaan tuottaa tavallisella tekstieditorilla, mutta matalatasoinen koodi tekee siitä vaikealukuista ja hankalasti tuotettavaa. Korkeatasoisempi MetaPost-ohjelmointikieli ja siihen soveltuvat työkalut helpottavat jonkin verran. MetaPostista voidaan suoraan viedä vektorigrafiikkaa kolmeen eri tiedostomuotoon: PostScript, Encapsulated PostScript ja SVG (Hobby ja muut 2009, s. 4).

2.2.2 Flash

Flash on tällä hetkellä Internetin suosituin interaktiivisen vektorigrafiikan esitysmuoto. Useimmat selainpelit on toistaiseksi julkaistu Flash-muodossa. Nykyisellään Flash taitaa muun muassa vektorigrafiikan, bittikartan, tekstin, grafiikkasuodattimet, äänen, tiedostojen lataamisen ja tallentamisen, dynaamisen videovirran ja 3D-muunnokset (Adobe 2009b).

Flash-sovellusten tai -multimediaesitysten tuottamiseen vaaditaan maksullinen Adobe Flash -ohjelmisto. Ohjelmiston valttina on erittäin helppo animaatioiden toteutus. Sillä tarvitsee vain määrittää kuville, kuvioille tai ryhmille avainkuvat (keyframes) eli näiden alku- ja loppupisteet transformaatioasetuksineen (esimerkiksi kierto tai skaalaus). Flash-toisto-ohjelma laskee toistettaessa avainkuvien välille muodostuvat välikuvat reaaliajassa, minkä ansiosta käyttäjille julkaistavan Flash-tiedoston koko jää todella pieneksi.

Ruudulle piirrettävien elementtien käyttäytymistä voidaan lisäksi ohjata ja manipuloida oliopohjaisella ActionScript-ohjelmointikielellä. ActionScript perustuu samaan ECMAScript-standardiin ECMA-262 kuin selaimissa käytettävä JavaScript (Adobe 2009a).

Flash on suljettu tiedostoformaatti ja nykyään Adoben omistama (ennen Shockwaven). Flash-tiedosto vaatii selaimessa toimiakseen erillisen selainlaajennuksen, jota käyttäjän on päivitettävä aika ajoin. Päivitystarve perustuu uusiin edistyksellisiin ominaisuuksiin, jotka sovelluskehittäjät haluavat yleensä ottaa käyttöön.

2.2.3 Silverlight

Silverlight on verrattain uusi Microsoftin kehittämä käyttöjärjestelmä- ja alustariippumaton .NET-toteutus, joka mahdollistaa rikassisältöisten ja työpöytäsovellustasoisten web-sovellusten (RIA, Rich Internet Application) tuottamisen. Silverlightin pääpaino on interaktiivisessa multimediassa, mutta se kykenee vektorigrafiikkaankin.

Silverlight käyttää Microsoftin kehittämiä käyttöliittymän määritteleviä ja XML-merkkikieleen perustuvia XAML-tiedostoja (eXtensible Application Markup Language). Niistä voidaan viitata ulkoisiin luokkatiedostoihin, jotka on toteutettu Visual Basicilla, C#:lla, Rubylla tai Pythonilla. Julkaisukelpoinen tiedosto on xap-tiedosto, joka on käytännössä .NET assembly -koodatuista

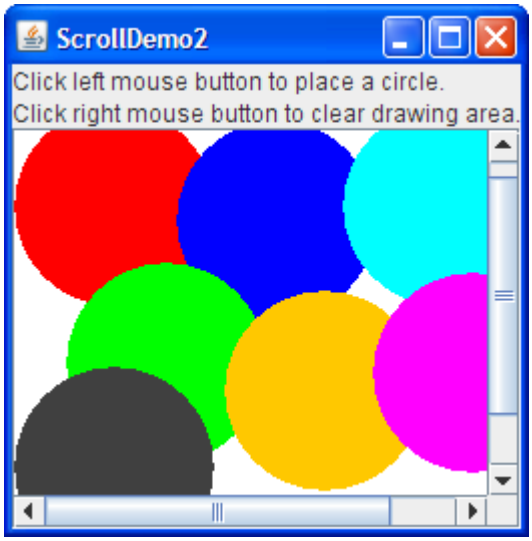
XAML-tiedostoista ja muista halutuista resursseista (kuten kuvista) koostuva zip-tiedosto. (Guthrie 2008.)

Silverlight vaatii selaimessa toimiakseen erillisen selainlaajennuksen. Uusien kehitettyjen ominaisuuksien hyödyntäminen edellyttää laajennuksen päivittämistä silloin tällöin.

2.2.4 Java

Java on Sun Microsystemsin kehittämä alustariippumaton korkean tason olio-ohjelmointikieli. Se sisältää vektorikuvien piirtämiseen valmiit API-kirjastot (Application Programming Interface) eli ohjelmointirajapinnat, joiden kautta piirtäminen voidaan suorittaa.

Piirtäminen tapahtuu Java 2D API:n määrittelemillä piirtokomennoilla. Java-ohjelma voi piirtää kuviot piirtopinnalle valmiiksi, tai sitten toimia interaktiivisesti esimerkiksi siten, että kuviot piirtyvät vasta hiiren klikkauksella (kuva 2). Java-ohjelmoijalla on vapaus tämän asian suhteen tehdä ohjelmansa käyttäytymään siten kuin hän haluaa.



Kuva 2: Esimerkki Java-sovelluksesta, johon käyttäjä voi piirtää suoraan (Sun 2008).

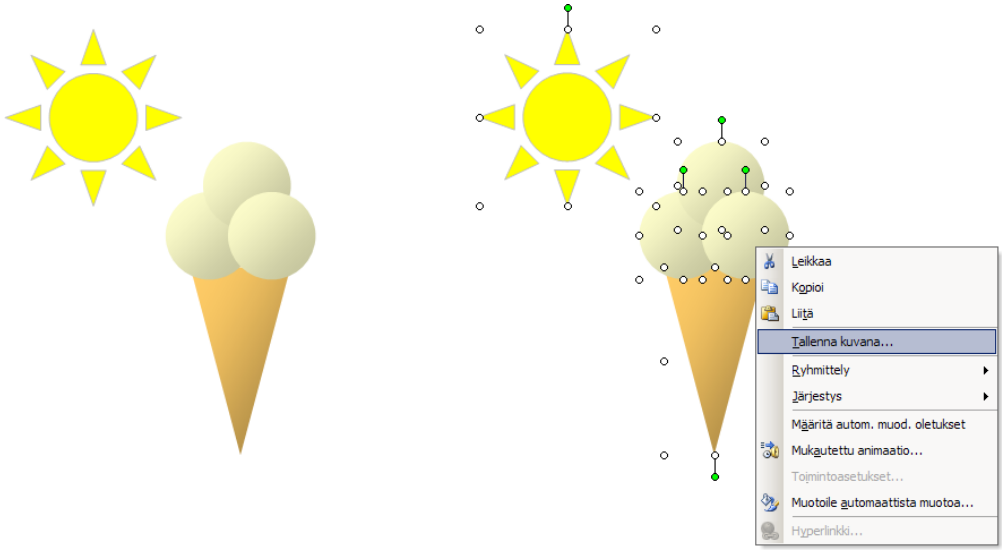
Selaimessa ajettuna Java-sovellus toimii applettina, joka voidaan liittää (X)HTML-merkkikielellä ((eXtensible) HyperText Markup Language) tehdylle web-sivulle melkein yhtä helposti kuin kuvakin. Java-sovellukset vaativat toimiakseen asennetun Java-ajoympäristön (JRE, Java Runtime Environment), jota käyttäjän on päivitettävä aika ajoin.

2.2.5 *Windows Metafile ja Enhanced Metafile*

Windows Metafile (WMF) ja siitä edelleen kehitetty Enhanced Metafile (EMF) ovat metatiedostoformaatteja, jotka sisältävät tietoa vektoreista ja bittikartoista (periaatteessa voivat kuitenkin sisältää mitä esitettävää dataa tahansa, kuten tekstiä). Näistä Enhanced Metafile on itse laajennettavissa ja siihen voidaan lisätä jopa toiminnallisuutta. (Microsoft 2007.)

WMF- ja EMF -tiedostojen sisällöt ovat katseltavissa sellaisenaan Windowsin kuvien ja faksin esikatseluohjelmalla (Windows XP tai uudempi). Lisäksi tiedostot voidaan liittää kuvana osaksi Word-, Excel- tai PowerPoint-dokumenttia. Tiedostomuodot eivät ole tuettuja selaimissa Windowsin omaa Internet Exploreria lukuun ottamatta.

Vektorikuvia voidaan tuottaa WMF- ja EMF -tiedostomuotoihin esimerkiksi PowerPoint-esitysgrafiikkaohjelmalla piirtämällä diaan halutut kuviot (kuva 3). Samalla periaatteella voidaan vientitoiminnon (englanniksi export) kautta tehdä SVG-kuvia OpenOffice-toimisto-ohjelmistoon kuuluvilla Draw-piirto-ohjelmalla tai Impress-esitysgrafiikkaohjelmalla.

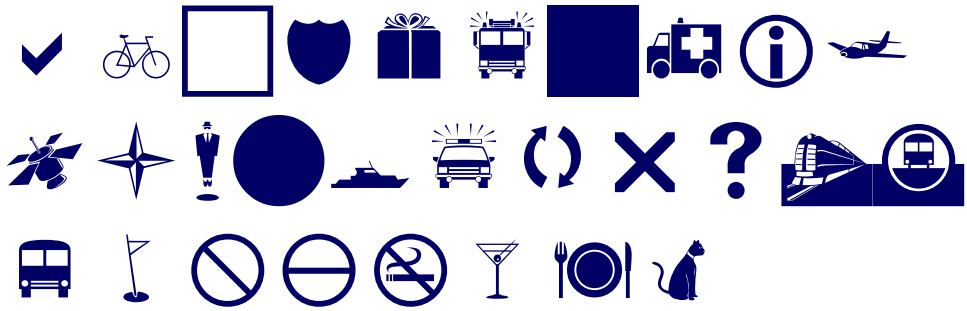


Kuva 3: Tämä herkullinen jäätelöannos on piirretty PowerPoint 2003:lla. Sen saa vietyä vektorigrafiikkatiedostoksi (WMF tai EMF) valitsemalla kaikki dian elementit, klikkaamalla hiiren kakkospainiketta valinta-alueen päällä (valikon esiin saamiseksi) ja valitsemalla Tallenna kuvana.

2.2.6 TrueType-fontit

TrueType-fontitkin ovat eräänlaisia vektorikuvia. Käyttöjärjestelmässä tai laiteajurissa oleva rasterointiohjelma piirtää kirjaimet päätelaitteeseen lukemalla fonttitiedostossa olevat kuvaukset (viivat ja kurvit), skaalaamalla ne pyydettyyn kokoon ja täyttämällä alueet jollakin sopivalla värillä. (Microsoft 1997b.)

Tällainen fonttityyppi mahdollistaa minkälaisen yksivärisen kuvion tahansa. Microsoft julkaisikin pelkästään kuvioita ja symboleja sisältäviä fontteja Windows 3.1 -käyttöjärjestelmän mukana vuonna 1992 (Microsoft 2009). Esimerkkejä tämänkaltaisista symboleista on kuvassa 4.



Kuva 4: Webdings-fontin kirjaimia (a-ö pienaakkosin).

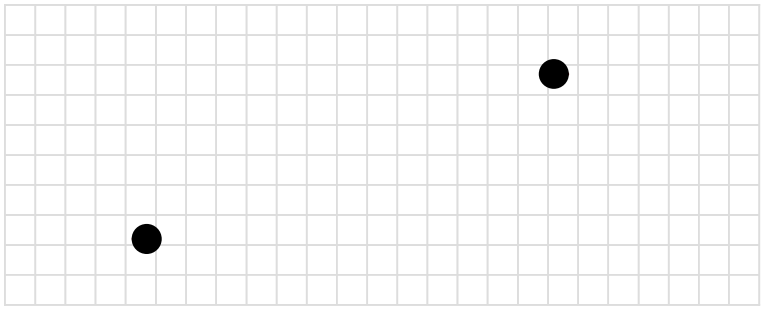
2.3 Rasterointi

Päätelaitteet, joihin vektorigrafiikka lopulta piirretään, yltävät vain rajalliseen tarkkuuteen. Näytöissä nämä maksimit ilmaistaan pikselimäärinä (vaaka- ja pystysuunnassa) ja tulostimissa pistemäärinä tuumaa kohden (DPI, dots per inch).

Ennen kuin vektorigrafiikkaa voidaan esittää tällaisissa laitteissa, se täytyy rasteroida (Microsoft 1997a; W3C 2003, luku 2.1) eli muuntaa geometriamuodot pikseleiksi bittikartalle (tai tulostimen tapauksessa pisteiksi paperitulostusta varten). Rasteroinnilla voidaan tarkoittaa myös painovärien hajautustekniikkaa (englanniksi halftone, "puolisävytys"), jota ei tässä käsitellä. Tässä asiayhteydessä rasterointia ei pidä myöskään sekoittaa pikselöintiin, jolla yleensä tarkoitetaan kuvan sumentamistekniikkaa (Adobe 2008b, s. 409).

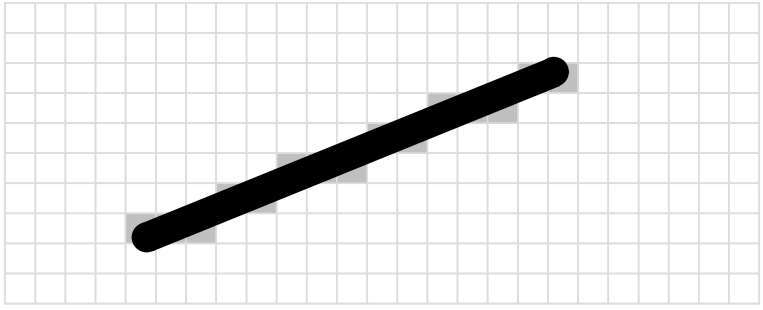
Seuraavassa kuvaillaan eräs tapa rasteroida vektorikuva. Käytetään havainnollistavana esimerkkinä mustaa viivaa, joka aiotaan piirtää pikseliruudukolle

(eli bittikartalle). Viivan muodostaminen aloitetaan määrittämällä alku- ja loppupisteen sijainnit (kuva 5).



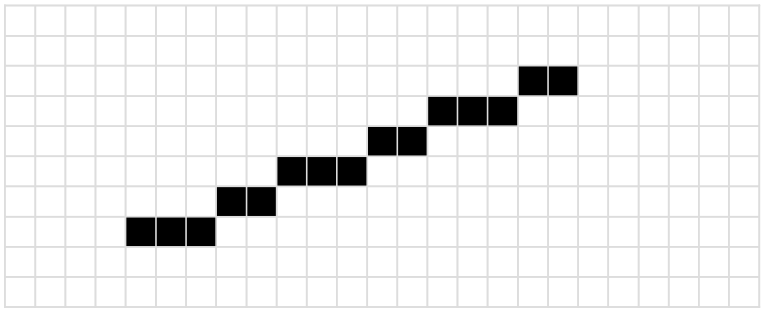
Kuva 5: Rasteroinnin vaihe 1. Viivan päät määrittelevät kaksi pistettä, joiden koordinaatit tiedetään. Olkoot ne kohdissa (4,7; 7,8) ja (18,2; 2,3), origon ollessa vasemmassa yläkulmassa. Nämä pisteet täytyy jotenkin yhdistää toisiinsa.

Rasterointiohjelman täytyy laskea pisteiden välinen ero vaaka- ja pystysuunnassa, jotta voitaisiin saada selville piirtokulma. Sen avulla voidaan tunnistaa, mitkä pikselit kuuluu värittää viivan värillä, jotta bittikartta vastaisi mahdollisimman tarkkaan annettua viivakuviota (kuva 6). Tietenkin joukossa on mukana myös niitä pikseleitä, joita viiva kattaa vain murto-osan verran. Lisätoimenpiteenä näihin pikseleihin voidaan tehdä osittainen väritys, mikäli halutaan, ettei lopputulos näytä sahalaitaiselta. Tällaista sahalaitaisuuden poistomenetelmää kutsutaan reunanpehmennykseksi tai antialiasoinniksi (englanniksi antialiasing tai anti-aliasing). (Foley ja muut 1994, s. 70 - 71 ja s. 119 - 121.)



Kuva 6: Rasteroinnin vaihe 2. Kun tiedetään, miten viiva etenee, väritettävät pikselit voidaan tunnistaa. Merkitään tilapäisesti harmaalla ne ruudukon alueet, joissa viivan pinta-ala peittää vähintään 50 % (antialiasointia ei sovelleta tässä). Viivan lopputulos siis piirtyisi väritettyihin ruutuihin viivan omalla värillä.

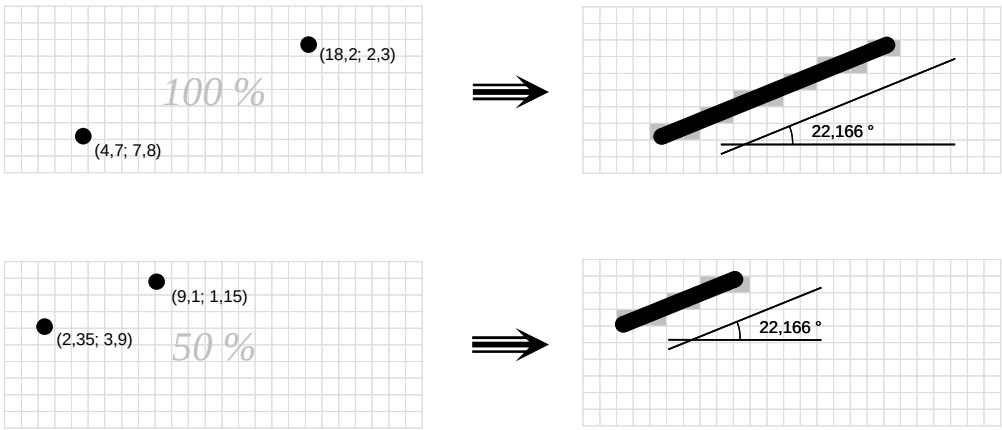
Kun väritettävät pikselit on tunnistettu, suoritetaan väritys. Lopputulos on konkreettista pikselidataa, joka voidaan ohjata esimerkiksi tietokoneen näytölle (kuva 7).



Kuva 7: Rasteroinnin vaihe 3. Lopputulos. Nyt vektoriviiva on muutettu pikselidataksi. Enää tarvitsee siirtää bittivirta kohdelaitteelle.

Jos vektorikuvaan halutaan tehdä muutoksia, vektorigrafiikkanäkökulmasta ainoastaan koordinaattipisteiden paikat muuttuvat (kuva 8). Juuri tämä tekee vektorigrafiikasta joustavan ratkaisun erilaisiin käyttötarkoituksiin.

Tässä on huomattava, että koko rasterointiprosessi joudutaan aina aloittamaan alusta jokaisen muutoksen jälkeen. Tämä tekee tiheää päivitysnopeutta vaativista sovelluksista erittäin raskaita heikkotehoisille laitteille. Adobe tarjoaa suorituskykyvinkkejä PDF-muodossa (2009c, s. 446 - 447) ja verkossa ensisijaisesti omalle Flash-ohjelmalleen, mutta ohjeita voidaan soveltaa monilta osin mihin tahansa vektorigrafiikkamuotoon.



Kuva 8: Skaalauksen vaikutus rasteroitavaan kuvaan. Vaikka pisteet sijaitsisivat esimerkiksi 50 % lähempänä tai kauempana origoa, viivan kulma pysyy samana, mikä tarkoittaa sitä, että vektorikuvaa voi väännellä loputtomasti ja lopputulos on aina tarkka. Viivan paksuutta ei huomioida tässä.

3 SVG:N ESITTELY

SVG on interaktiivisen ja dynaamisen kaksiulotteisen vektorigrafiikan esittämiseen suunniteltu avoin kuvauskieli, jonka kehityksestä ja standardoinnista vastaa maailmanlaajuinen W3C-konsortio. SVG-spesifikaatioon sisältyvä

grafiikkatarjonta on runsas ja monipuolinen. SVG 1.0:n ja 1.1:n ominaisuuksiin kuuluvat:

- perusmuodot (suorakulmio, ympyrä, viiva...)
- polygonit (viivoja, joita pitkin omia kuvioita piirretään) ja polut (viivoja ja käyriä)
- teksti
- (bittikartta-)kuvat
- piirrettävien elementtien ryhmittäminen
- väritys (tasaväri, liukuväri, toistuva kuvio, bittikarttakuva)
- transformaatiot (matriisioperaatioihin perustuvat muunnokset, kuten siirto, skaalaus, kierto, viistous tai peilaus, jotka voidaan kohdistaa yksittäiseen elementtiin tai ryhmään)
- CSS-tyylit (Cascading Style Sheets, web-sivuilla käytettävä muotoilutekniikka)
- leikkauspolut (menetelmä, jolla elementtiin asetetaan läpinäkyviä alueita), osittainen läpinäkyvyys ja maskit (kuvankäsittelystä tuttu toimenpide, jolla eriasteisia läpinäkyvyyksiä voidaan käyttää samanaikaisesti)
- grafiikkasuodattimet (grafiikan näyttävyyttä parantavat lisätehosteet)
- linkittäminen
- skriptien käyttömahdollisuus
- SMIL-animointi (Synchronized Multimedia Integration Language) (SMIL-animointia käsitellään luvussa 4.3.1).

Teknisesti SVG on XML-tekstitiedosto, joka on web-sivujen tekijöille ja ohjelmoijille jo valmiiksi tuttu tiedostomuoto. XML on avoin ja stabiili standardi (vuoden 1998 jälkeen XML 1.0 -spesifikaatioon on tehty pääsääntöisesti vain virhekorjauksia ja tarkennuksia) ja lähdekoodi on ihmiselle melko selkokielistä luettavaa. XML:lle on olemassa runsaasti ilmaisia työkaluja ja niiden puuttuessakin tiedostoja voidaan käsitellä millä tahansa tekstieditorilla. SVG:lle voidaan soveltaa samoja työkaluja kuin XML:lle ja lisäksi SVG:lle on valmistettu erityistyökaluja piirto-ohjelmien muodossa (ilmaisiakin). SVG:lle on varattu oma DTD (Document Type Definition), rakennemääritys, jota vasten XML-koodin kielioppi voidaan helposti tarkistaa automaattisesti. DTD-asetukset ovat saatavilla W3C:ltä (2007b).

SVG:tä ymmärtävät katseluohjelmat voivat tiedostossa annettujen kuvausten perusteella piirtää vektorigrafiikkaa ja liittää siihen bittikarttakuvia ja tekstiä.

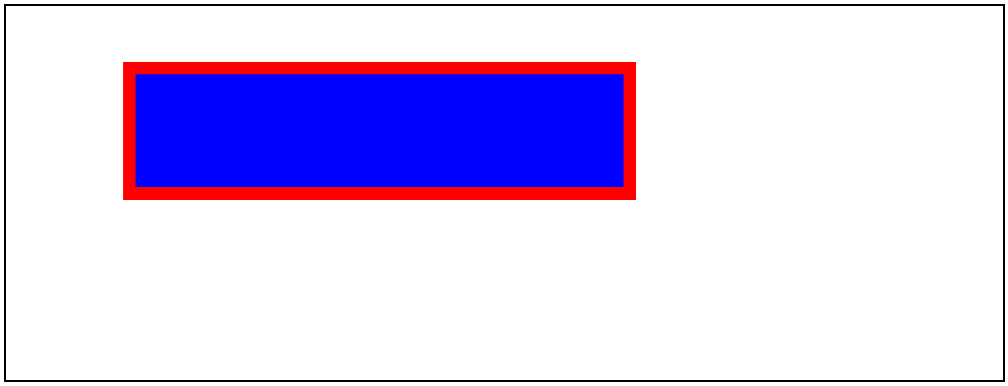
Kunkin piirrettävän elementin sijainti, mitat ja muut ominaisuudet asetetaan attribuuteilla. Näin SVG-katseluohjelma saa kaikki tarvittavat tiedot kuvioiden piirtämiseksi ruudulle. Web-sivuja aiemmin tehneille koodin tekniikassa ei ole mitään uutta, joten SVG on heille helposti lähestyttävä ja aloituskynnys matala. Eri SVG-kuvien välillä voidaan navigoida helposti linkittämällä elementit a-tagilla HTML-sivuille sijoitettavien kuvalinkkien tapaan.

SVG:ssä käytettävää koodia esitellään koodiesimerkissä 1. Siinä tuotetaan 400x100 pikselin kokoinen suorakulmio, joka sijoitetaan 800x300 pikselin kokoiselle näkymäalueelle. Näkymäalueen määrittää `svg`-juurielementti ja suorakulmion `rect`-elementti.

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<!-- Encoding-asetus on annettava XML-määrityksessä, jotta
skandinaaviset merkit tulostuvat oikein -->

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1"
 viewBox="0 0 800 300"
 xmlns="http://www.w3.org/2000/svg">

    <!-- Sininen suorakulmio punaisella reunuksella -->
    <rect width="400" height="100" x="100" y="50"
     fill="blue" stroke="red" stroke-width="10" />

</svg>
```

Koodiesimerkki 1: Kokonainen SVG-tiedosto, jossa ainoa piirrettävä elementti on suorakulmio.

Koodi tuottaa kuvassa 9 näkyvän lopputuloksen. SVG-kuvan koordinaatiston nollakohta (eli origo) sijaitsee vasemmassa ylänurkassa. X-koordinaatti kasvaa oikealle ja Y-koordinaatti alaspäin.



Kuva 9: Koodiesimerkin 1 tuottama suorakulmio.

SVG-kuville käytetään yleensä omaa tiedostopäätettä, joka on .svg. SVG:n MIME-tyypiksi (Multipurpose Internet Mail Extension) on sovittu `image/svg+xml`, eli lähtökohtaisesti se esitetään "itsenäisenä dokumenttina" (W3C 2003, luku 2.3) tai kuvana toisin tulkittuna.

SVG ei vaadi erillistä selainlaajennusta toimiakseen, jos selain tukee sitä suoraan. Koska SVG on avoin ja kansainvälisesti sovittu vektorigrafiikka-muoto, selaimen tai muun asiakasohjelman päivitystarve (SVG:n takia) ei ole läheskään niin suuri kuin sen kilpailijoilla.

3.1 Ohjelmoitavuus

SVG on ohjelmoitavissa selainskriptillä täsmälleen samalla tavalla kuin web-sivutkin, mikä mahdollistaa dynaamisen ja interaktiivisen ratkaisun monenlaisiin tarpeisiin. Tämä tekee SVG:stä paljon monipuolisemman kuin pelkän vektorigrafiikan erään esitysmuodon. Skriptikieleksi kelpaa ECMAScript, siihen rinnastettava JavaScript tai muu vastaava olio-ohjelmointikieli, jota selaimet yleisesti ymmärtävät.

Skriptikäyttöä varten selain luo SVG-kuvasta (X)HTML-sivujen tapaan oliomallin, joka toteuttaa DOM-rajapinnan (Document Object Model). Oliomalli koostuu solmuista (englanniksi node), jotka vastaavat rakenteeltaan ja sisällöltään selaimessa katseltavan dokumentin XML-puun elementtejä. Malli mahdollistaa dokumentin sisällön, rakenteen ja tyylin päivittämisen. Tällä tavalla dokumentin elementteihin ja attribuutteihin tarjotaan helppo ja suora pääsy ohjelmoijalle ja käyttäjälle lopputulos näkyy vain muuntuneena XML-puuna. (W3C 2009a; Refsnes Data 2009c.)

3.2 Soveltuvuus muiden tekniikoiden kanssa

SVG 1.1 soveltuu käytettäväksi osana muita XML-toteutuksia. Spesifikaatio kuvailee itseään modulaariseksi kieleksi, toisin kuin sen edeltäjänsä SVG 1.0. Versiossa 1.1 tämä toteutuu abstraktien moduulien kokoelmana siten, että moduuleja voidaan yhdistää toisiinsa tai käyttää muiden spesifikaatioiden, kuten XHTML:n kanssa (Nykänen 2007, s. 13; W3C 2003, luku 1.1.1).

Esimerkiksi tekstimoduuli pitää sisällään joukon erilaisia tekstielementtejä, näille ominaiset attribuutit ja lisäksi muita attribuutteja niistä moduuleista, joista se on riippuvainen. Kukin moduuli sijaitsee omassa DTD-

tiedostossaan, joten sen käyttö oman XML-toteutuksen kanssa tulee yksinkertaiseksi, vaikkei varsinaisesti olisikaan SVG-grafiikan kanssa missään tekemisissä.

3.3 Profiilit

Spesifikaation modulaarisuus mahdollistaa edellä mainitun lisäksi erilaisten profiilien käytön luettelemalla joukon SVG-moduuleja, joista profiilit koostuvat (W3C 2003, luku 1.1.3). Suosituksen mukaisia profiileja on kolme: Full, Basic ja Tiny. Full-profiili sisältää kaikki moduulit, Basic karsitun määrän moduuleja ja Tiny tiukasti rajatun, vain välttämättömän määrän moduuleja. Tällaiset rajatut profiilit mahdollistavat SVG:n käytön mobiililaitteissa, joissa on heikko suorituskyky. Tällaiset laitteet eivät tarvitse SVG:n kaikkia (raskasta laskentatehoa vaativia) ominaisuuksia.

Basic-profiili on muotoiltu sopivaksi kämmentietokoneisiin ja Tiny matkapuhelimiin. Basic:n merkittävimmät ominaisuuksien karsinnat Full-profiiliin nähden ovat:

- numeroarvot vain välillä -32767,9999 ja +32767,9999
- grafiikkasuodattimet osittain
- leikkauspolut osittain.

Tiny-profiili sisältää samat rajoitukset kuin Basic ja lisäksi karsii pois kokonaan käytöstä:

- liukuvärit
- skriptit (palautettiin SVG Tiny 1.2:een Mikro DOM:n myötä)
- CSS-tyylit (SVG-juurielementtiä lukuun ottamatta)
- grafiikkasuodattimet
- leikkauspolut
- osittainen läpinäkyvyys
- maskit
- sisäkkäiset SVG-kuvat.

Kaiken kaikkiaan SVG vaikuttaa jo tässä vaiheessa olevan monikäyttöinen ja kelpo väline interaktiivisten sovellusten toteuttamiseksi erilaisille kohderyhmille ja käyttötarkoituksille. Selainpelimahdollisuudet ovat lupaavia.

4 PELINÄKÖKULMA SVG:SSÄ

Seuraavassa on pohdintaa joistakin SVG-peleihin liittyvistä näkökulmista, joita tulisi ottaa huomioon. Ne auttavat sanelemaan, mitä SVG-peleissä on mahdollista toteuttaa ja mitä ei. Suurin osa sisällöstä käsittelee SVG:n ja skriptien välistä yhteistyötä.

4.1 Selaintuki

SVG-tuen kattavuus on yllättävän sovelluskohtaista. Luonnollisestikaan esimerkiksi kuvankäsittelyohjelmat eivät tue animointia, linkkejä tai skriptejä, mutta hajontaa löytyy ikävästi myös selainpuolelta. Läheskään kaikkia spesifikaation osa-alueita ei tueta.

Selaintuki asettaa rajoitukset sille, minkälaisia pelejä SVG:llä kannattaa tai ei kannata tehdä tällä hetkellä. Tuen puuttuessa pelille ei olisi pelaajia. Selaimien tuetuimmat SVG-ominaisuudet (tutkimuksen kirjoitushetkellä) on listattu taulukkoon 1. Taulukon täyttämiseen on käytetty apuna W3C:n testisarjaa (2006), mutta lisäksi Jeff Schilleriltä (2009) on saatavilla yksityiskohtainen (epävirallinen) SVG-selaintuen kattavuustaulukko, joka perustuu samaan testisarjaan.

Taulukko 1: Selaimissa tuetut SVG-ominaisuudet pääpiirteittäin. X = tuettu, ~ = vajavaisesti tuettu. Testipenkkiin valittiin kirjoitushetkellä uusimpien selaimien vakaat versiot (poikkeuksena Konqueror, josta kaksi eri versiota).

Selain / ominaisuus	**Amaya 11.1**	**Chrome 1.0.154.65**	**Firefox 3.0.11**	**Internet Explorer 8.0.7100.0**	**Konqueror 3.4.3**	**Konqueror 4.2.2**	**Opera 9.64**	**Safari 3.2.3**
Linkitys	X	X	X		X		X	X
Skriptit		X	X		X	~	X	X
Animointi (SMIL)	X				~		X	
Grafiikka-suodattimet			X				X	
Bittikartta-kuvat	X	X	X		X		X	X
Sisäkkäiset SVG-kuvat	X				X		X	
Fonttityylit		X	~		X		X	X
Leikkaus-polut		X	X		X	X	X	X

Markkinaosuudeltaan ehkä suurin selainperhe, Microsoft Internet Explorer, ei sellaisenaan tue SVG:tä lainkaan, vaan se tarvitsee jonkin selainlaajennuksen. Opera sen sijaan läpäisee koko 275 testin sarjan liki täydellisesti. Kaikkien muiden selainten suoritus jää keskinkertaiseksi. Esimerkiksi SMIL-

pohjaista animointia, joka sulautuu luontevaksi osaksi SVG:n XML-puuta, tuetaan kunnolla ainoastaan kahdessa selaimessa.

Tuetuimmiksi ominaisuuksiksi voidaan taulukon perusteella lukea linkitys, skriptit, bittikarttakuvat ja leikkauspolut. Näihin kaikkiin kerralla kykenevät Chrome, Firefox, Konqueror 3, Opera ja Safari. Ulkopuolelle jäävät Amaya, Internet Explorer ja Konqueror 4.

4.1.1 Internet Explorer ja SVG

Koska Internet Explorer ei ymmärrä lainkaan SVG-tiedostomuotoa, vaatii sen käyttö jonkin selainlaajennuksen. Saatavilla olevia selainlaajennuksia SVG:n esittämiseksi ovat nykyään:

- Adobe SVG Viewer
- Corel SVG Viewer
- GPAC Project on Advanced Content
- RENESIS SVG Player
- Ssrc SVG.

Näistä Adobe SVG Viewerin ja Corel SVG Viewerin elinkaaret ovat jo päättyneet. Kumpaakaan ei tarjota enää tuotekatalogeissa, mutta ovat hakukoneella vielä löydettävissä ainakin toistaiseksi.

Internet Explorerin SVG-puute on myös kierrettävissä käyttämällä jotakin jo olemassa olevaa selainlaajennusta ja valjastamalla se renderöimään SVG-tiedostoja. Esimerkiksi svgweb-projekti (tuntemattomat 2009) käyttää Flash-selainlaajennusta ja JavaScriptiä SVG:n piirtämiseksi. Vastaava voitaneen toteuttaa Javallakin.

4.1.2 Tukemattomien ominaisuuksien tunnistaminen ja vaihtoehtoisen sisällön näyttäminen

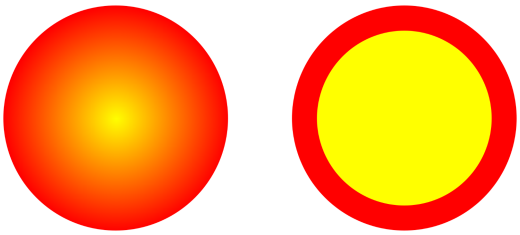
SVG:ssä voidaan tehdä ehdollista prosessointia, jolla automatisoidaan näytettävä sisältö sen mukaan, mitä spesifikaation osa-alueita SVG-katseluohjelma ilmoittaa tukevansa. Tämän avulla voidaan esimerkiksi upottaa karsittu mobiiliversio samaan SVG-tiedostoon. Jos jotkin ominaisuudet eivät jostain syystä ole mahdollista toteuttaa käyttäjällä, ehdollisen prosessoinnin avulla voidaan siirtyä vaihtoehtoiseen sisältöön automaattisesti.

Temppu onnistuu `requiredFeatures`- ja/tai `requiredExtensions` -attribuuteilla. Jos SVG-katseluohjelma ei tue näiden attribuuttien kautta ilmoitettuja vaatimuksia, sen tulee jättää koko elementti piirtämättä. Lisäksi tarvitaan `switch`-elementtiä. `switch`:llä tarjotaan useita vaihtoehtoja (ohjelmoijille tutulla switch-casea muistuttavalla tavalla) ja näistä tulee valituksi *ensimmäinen* elementti (ja kaikki sen lapsielementit), jonka vaatimukset SVG-katseluohjelma voi täyttää. Loput vaihtoehdot jätetään huomiotta. Koodiesimerkissä 2 ja kuvassa 10 havainnollistetaan `switch`-elementin ja `requiredFeatures`-attribuutin yhteiskäyttöä. Koodissa määritellään kaksi ympyrää, joista ensimmäisessä on liukuväritäyttö. Tämä liukuväritäyttöinen ympyrä piirretään, jos SVG-katselin väittää tukevansa `requiredFeatures`:ssa pyydettyä ominaisuutta.

```
<defs>
    <!-- Liukuvärin alustaminen -->
    <radialGradient id="Gradient01">
        <stop offset="0%" stop-color="yellow" />
        <stop offset="100%" stop-color="red" />
    </radialGradient>
</defs>

<switch>
    <!-- Vaihtoehto 1: liukuväritäyttöinen ympyrä -->
    <circle id="gradientCircle"
     requiredFeatures="http://www.w3.org/TR/SVG11/feature#Gradient"
     r="45" cx="50" cy="50"
     fill="url(#Gradient01)" />

    <!-- Vaihtoehto 2: tasaväritäyttöinen ympyrä -->
    <circle id="solidCircle"
     r="40" cx="50" cy="50"
     fill="yellow" stroke="red" stroke-width="10" />
</switch>
```

Koodiesimerkki 2: Tuettujen (tai tukemattomien) SVG-moduulien tunnistaminen switch-elementillä ja requiredFeatures-attribuutilla.

Koodin lopputulos on jompikumpi kuvassa 10 olevista vaihtoehdoista. Jos `requiredFeatures`-attribuutti puuttuisi, ensimmäinen `switch`:ssä esitetyistä vaihtoehtoista tulisi aina valituksi. Tällöin, jos liukuväriä ei tuettaisi SVG-katseluohjelmassa, seurauksena olisi pahimmillaan oletusväri eli musta ympyrä.



Kuva 10: Ehdollisen prosessoinnin seurauksena (koodiesimerkki 2) vasemmanpuoleinen ympyrä näytetään, jos SVG-katseluohjelma tukee liukuvärejä, oikeanpuoleinen muulloin.

Asetelma toimii kokonaan ilman skriptiä, mutta oikea käyttäytyminen tietysti vaatii, että `requiredFeatures`- ja `requiredExtensions` -attribuutit sekä `switch`-elementti ovat tuettuja. Jos SVG-katseluohjelma tukee skriptejä (kuten useimpien selaimien tapauksessa yleensä on), vaihtoehtoinen tapa on käyttää `hasFeature`-funktiota (koodiesimerkki 3).

```
<svg id="root" version="1.1"
 xmlns="http://www.w3.org/2000/svg"
 onload="testSupport()">

    <text id="info" font-family="Arial" font-size="12px" x="5"
     y="20">Selaimesi ei tue skriptejä tai ne ovat pois päältä</text>

    <script type="text/ecmascript"><![CDATA[
        function testSupport() {
            // Haetaan elementti XML-puusta
            var element = document.getElementById('info');

            // Tukeeko selain tätä moduulia?
            /* (versionumero on kaikissa moduuleissa käytettävän
               SVG-spesifikaation mukainen eli 1.1) */
            if( document.implementation.hasFeature(
            'http://www.w3.org/TR/SVG11/feature#CoreAttribute',
            '1.1') ) {
                // Päivitetään teksti tekstielementtiin
                element.firstChild.nodeValue = 'tukee';
            }
            else {
                // Päivitetään teksti tekstielementtiin
                element.firstChild.nodeValue = 'ei tue';
            }
        }
    ]]></script>

</svg>
```

Koodiesimerkki 3: Tuettujen ja tukemattomien SVG-moduulien tunnistaminen skriptin avulla.

Näillä tavoilla saadaan hallittu lopputulos riippumatta käyttäjällä olevista ohjelmistoista. `requiredFeatures`- ja `requiredExtensions`-attribuuteille sekä `hasFeature`-funktiolle kelvolliset merkkijonot löytyvät SVG 1.1 -spesifikaation (W3C 2003) liitteestä O.

4.2 Koodin uudelleenkäytettävyys

Ohjelmistotuotannolle on ominaista, että tuote pilkotaan pienempiin osiin ja sitä kehitetään pieninä osakokonaisuuksina. Tavallisesti osista halutaan samalla uudelleenkäytettäviä siten, että ne kelpaavat sellaisenaan tulevaisuudessakin ilman muokkauksia tai hyvin pienellä muokkausmäärällä. Käyttökelpoisia mahdollisuuksia on SVG:ssä vähänlaisesti.

Osakokonaisuuksia ajatellen olisi mukavaa, jos kerran piirrettyjä SVG-kuvia voisi hyödyntää muuallakin esimerkiksi upotettuna johonkin toiseen SVG-kuvaan. Spesifikaatio mahdollistaa sisäkkäisten SVG-kuvien käytön kahdella tavalla: `image`-tagilla tai `svg`-tagilla. Taulukossa 1 sisäkkäisiä SVG-kuvia koskevat testit suoritettiin `image`-tagilla. Sille mekanismille ei siis ole riittävää selaintukea, vaikka samaa tagia käytetään bittikarttakuvien esittämiseen menestyksekkäästi. Sisäkkäisen `svg`-tagin hyödyt puolestaan rajoittuvat lähinnä palvelinpäähän, sillä jos palvelimen tehtävänä on koostaa lähdekoodi eri SVG-tiedostoista, asiakasohjelman täytyy saada tämä kooste yhtenä kappaleena koodiesimerkissä 4 muotoillulla tavalla. Tämän käyttöönottoa vaikeuttaa myös se, että elementtien yksilöllisyys on taattava.

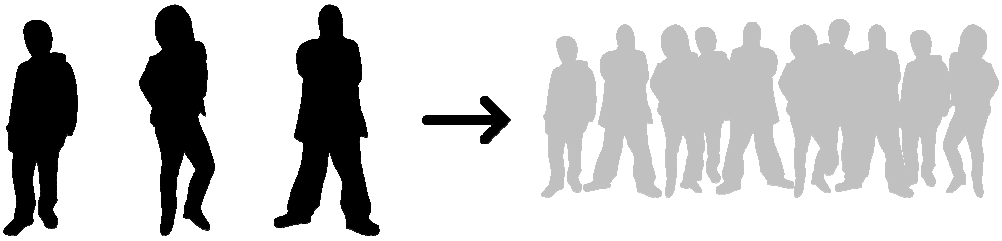
```
<svg ...>
    <rect id="box1" ... />
    <svg ...>
        <rect id="box2" ... />
    </svg>
</svg>
```

Koodiesimerkki 4: Sisäkkäisten SVG-tagien käyttö (esimerkin kannalta epäoleelliset attribuutit on korvattu pisteillä).

Parhaat osakokonaisuuksien hallintamahdollisuudet löytyvät ulkoisten tiedostojen lainailujen sijaan sisältäpäin. SVG tarjoaa `defs`-tagia useita kertoja tarvittaville elementeille. Sinne lisättyihin määrityksiin voidaan viitata niin monta kertaa kuin halutaan. Kun elementeille annetaan yksilölliset nimet (attribuutilla `id`), niistä tulee URI-resursseja (Uniform Resource Identifier). `defs`-tagia onkin jo käytetty koodiesimerkissä 2 liukuväriesimerkin muodossa.

Lisäksi `defs`-tagiin voidaan määrittää konkreettisia piirroksia, esimerkiksi lumihiutale, jota tarvitsee vain kutsua aina kun halutaan piirtää uusi lumihiutale. Piirtokelpoiset elementit kutsutaan `use`-tagilla, joka tarvitsee vain sijoittaa (ja haluttaessa muotoilla) `transform`-attribuutilla (kuva 11). Ryhmit-

televää g-tagia käyttämällä voidaan kutsua kerralla isompiakin osakokonaisuuksia. Mahdolliset muutokset alkuperäiselementtiin näkyvät välittömästi myös sitä hyödyntävissä klooneissa.



Kuva 11: Use-tagin avulla kerran piirrettyjä kuvioita voidaan uusiokäyttää ilman lähdekoodin copy-pastea. Tällä tavalla esimerkiksi muutamaa ihmishahmoa kloonaamalla saadaan tehtyä kokonainen väkijoukko minimaalisella koodimäärällä.

`use`-tagi saadaan viittaamaan haluttuun piirrokseen asettamalla sen URI-resurssi `xlink:href`-attribuutin arvoksi (katso attribuutin käyttöönotto sivulta 29). Periaatteessa nämä URI-resurssit voisivat sijaita myös suhteellisen tai absoluuttisen hakemistopolun takana, eli kokonaan erillisessä tiedostossa, mutta selaimet tukevat tällaista käyttötapaa yllättävän huonosti. Yhden tiedoston rajoissa voidaan silti huomata, että `defs`- ja `use`-tagien yhteiskäytöllä lähdekoodista saadaan ratkaisevan paljon ylläpidettävämpää, selkeämpää ja lyhyempää.

4.3 Animointi

SVG-grafiikkaa voidaan animoida kahdella tavalla: SMIL-kielellä tai skriptillä. Staattiseen, määrittelevään tapaan soveltuu SMIL, kun taas dynaamiset ratkaisut edellyttävät skriptiä. Molemmissa tapauksissa animointi tapahtuu käyttämällä attribuutteja.

4.3.1 SMIL

SMIL on W3C:n määrittelemä XML-pohjainen interaktiivisen multimedian esittämiseen tarkoitettu kieli. Sillä voidaan asettaa esimerkiksi XML-elementin kullekin attribuutille alku- ja loppuarvot sekä kesto näiden välillä siirtymiseen koodiesimerkin 5 tapaan.

```
<rect ...>
    <animate attributeName="width"
     from="10px"  to="100px" begin="0s" dur="10s" />
    <animate attributeName="height"
     from="100px" to="10px"  begin="0s" dur="10s" />
</rect>
```

Koodiesimerkki 5: Animointi SMIL:llä.

Koodiesimerkissä korkea ja kapea suorakulmio venytetään matalaksi ja leveäksi kymmenen sekunnin ajan. Edellä todetun kehnon selaintuen perusteella pelin sovelluslogiikkaa ei voida jättää SMIL-animoinnin varaan, eikä sitä ei ole muutenkaan suunniteltu siihen tarkoitukseen, koska elementin attribuuttiarvot eivät muutu:

> When an animation is running, it should not actually change the attribute values in the DOM. The animation runtime should maintain a *presentation value* for each animated attribute... (W3C 2008, luku 12.4.2.)

Käytetyistä selaimista ainoastaan Opera ja Amaya tukevat SMIL:iä tällä hetkellä. SMIL-animointia tulisi siis käyttää ennemmin ylimääräisenä koristeena tai tehostekeinona niitä selaimia varten, jotka erikseen sitä tukevat, mikäli animaatioita kummempaa toiminnallisuutta ei "peliin" haluta lisätä.

4.3.2 Skripti

Ohjelmointitaitoisille selainskripti on SMIL:iä paljon miellyttävämpi vaihtoehto. Se on tarkoitettu nimenomaan selaimessa näytettävän dokumentin manipulointiin, ja soveltuu SVG-kuville aivan yhtä hyvin kuin (X)HTML-sivuillekin. SMIL:stä poiketen attribuuttiarvot ovat aina ajantasaisia ja siten sovellettavissa suoraan selainpelin sovelluslogiikkaan. Tämän ansiosta skripti voi haluttaessa reagoida uudella tavalla, kun animaatio on edennyt tiettyyn pisteeseen. Skriptin mahdollisuudet ovat siis paljon laajemmat kuin SMILin.

> Every attribute and style sheet setting is accessible to scripting, and SVG offers a set of additional DOM interfaces to support efficient animation via scripting. As a result, virtually *any kind of animation can be achieved*. The timer facilities in scripting languages such as ECMAScript can be used to start up and control the animations. (W3C 2003, luku 19.1.)

Skriptillä animointi onnistunee helpoiten JavaScriptin `setTimeout`-funktiolla. Sillä viivästytetään skriptikoodin suorittamista. Funktiolla on kaksi pakollista parametria, joista ensimmäinen on suoritettava skriptikoodi merk-

kijonomuotoisena ja toinen viiveen kesto millisekunteina. Koodiesimerkin 6 funktiossa kasvatetaan elementin x-attribuutin arvoa 10 kertaa sekunnissa.

```
function liikuta(elementId) {
    // Luetaan animoitavan elementin nykyinen arvo
    var element = document.getElementById(elementId);
    var x_nyt = new Number( element.getAttribute('x') );

    var x_uusi = x_nyt + 5;

    // Asetetaan uusi arvo
    element.setAttribute('x', x_uusi);

    // Suoritetaan tämä funktio uudelleen 100 ms kuluttua
    setTimeout('liikuta(\''+elementId+'\')', 100);
}
```

Koodiesimerkki 6: Animointi skriptillä.

`liikuta`-funktion suoritus ei pysähdy `setTimeout`-funktion kohdalle odottamaan ajan umpeutumista, vaan jatkuu välittömästi. `setTimeout`-funktiokutsun jälkeen voidaan siis asettaa toinen `setTimeout`, minkä ansiosta useita elementtejä voidaan animoida yhtä aikaa tai tehdä muita rinnakkaistoimenpiteitä. Yllä olevassa esimerkissä `setTimeout`-pyyntö voisi siis aivan yhtä hyvin olla ensimmäinen komento koko `liikuta`-funktiossa.

`setTimeout`:n sijaan voidaan käyttää myös `setInterval`-funktiota, joka toistaa skriptikoodin itsenäisesti uudelleen. `setInterval` tarvitsee (ja täytyy) suorittaa kullekin toistoa kaipaavalle funktiolle *vain yhden ainoan kerran* koko ohjelmasuorituksen aikana. Tämä on hieman vaarallista, sillä pinoutumisen riski on ilmeinen (minkä seurauksena on yleensä selaimen jumiutuminen). Lisäksi käynnistettyä toistoa ei voida keskeyttää `if`-ehdolla eikä hallita millään muullakaan tavalla. Tällainen skriptisuoritus saadaan lakkaamaan siirtymällä selaimella muualle: esimerkiksi toiseen SVG-tiedostoon tai sulkemalla selainikkuna (tai välilehti).

4.4 Satunnaisuus

Jos SVG-peliin halutaan satunnaisuutta, ainoa tapa lisätä sitä on skripti. JavaScript tarjoaa `Math.random`-funktion, jota ei tarvitse alustaa siemenluvulla. Funktio palauttaa pseudosatunnaisen desimaaliluvun nollan ja yhden välillä. Tätä hyödyntämällä voidaan rakentaa esimerkiksi arpakuutio (koodiesimerkki 7).

```
function heitaNoppaa() {
    var r = Math.random();
    return parseInt(r*6 + 1);
}
```

Koodiesimerkki 7: Arpakuutio.

`parseInt`-funktiolla desimaaliluvusta saadaan kokonaisluku pyöristämättä. Muita muotoilufunktioita käytettäessä seurauksena on pyöristyminen, jolloin pienimmän ja suurimman sallitun luvun todennäköisyys on pienempi.

4.5 Äänet

SVG on alun perin tarkoitettu vektorigrafiikkaformaatiksi eikä Flashin kaltaiseksi kaikenkattavaksi audiovisuaaliseksi multimediaratkaisuksi. SVG 1.1 -spesifikaatio ei valitettavasti tarjoa mitään puitteita äänien toistamiseen, mikä on pelien kannalta harmillista. `audio`-elementti ilmestyi vasta SVG 1.2 -spesifikaatioon. Tarve äänille tuli siis vasta jälkeenpäin, joskin äänitukea ei tämän elementin kautta ole saatavilla vielä nykyselaimissakaan.

Luotettavin tapa sisällyttää äänet peliin lienee tehdä se SVG:n ulkopuolella esimerkiksi erillisessä ikkunassa ja käyttämällä siinä perinteistä (X)HTML-koodia. Etenkin taustamusiikin lisääminen tällä tavalla luulisi olevan melko vaivatonta. Tapahtumapohjaiset äänet voidaan käynnistää ainoastaan skriptin kautta.

4.6 Pelitilanteen tallentaminen ja lataaminen

Kokonaislaajuudeltaan useita tunteja kestävissä peleissä pelaajalle on tarpeellista, että hän kykenee tallentamaan pelitilanteensa, keskeyttämään sen ja lataamaan sen sitten myöhemmin uudelleen, jotta voi jatkaa siitä mihin viimeksi jäi. Selainpelissä tietojen tallentaminen onnistunee turvallisimmin ja kätevimmin evästeen avulla.

Evästeitä voidaan hallita palvelimella esimerkiksi yleiskäyttöisellä PHP-skriptikielellä (PHP: Hypertext Preprocessor) tai paikallisesti selainskriptillä. Hallintatavasta riippumatta itse eväste kuitenkin säilytetään aina käyttäjällä olevan selaimen välimuistissa. Tästä syystä käyttäjää tulee tiedottaa evästeiden käytöstä, jottei hän selaimen välimuistia tyhjentäessään hävittäisi tietämättään myös pelitallennuksiaan samalla. Evästeet ovat tekstitiedostoja, joten ne on helppo varmuuskopioida.

JavaScriptillä evästeitä käytetään `document.cookie`-olion kautta. Kukin eväste sisältää nimi-arvo-parin. Lisäksi se voi sisältää vanhentumisajan, hakemistopolun ja toimialueen, joiden puitteissa se on voimassa (vanhentumisaika saadaan helposti oikeaan muotoon Date-luokan `toGMTString()`- tai `toUTCString()` -funktioiden avulla). Eväste lisätään tai päivitetään merkkijonosijoituksella koodiesimerkin 8 tapaan. Teknisesti `document.cookie` ei kuitenkaan ole merkkijono, sijoitusoperaatio vain toimii kuvatulla tavalla. Olion sisältökin tarjotaan merkkijonona, tosin sisältäen vain nimi-arvo-parit. (Koch 2007.)

```
document.cookie = 'player=alien; expires=Thu, 2 Aug 2010 20:47:11
GMT; path=/; domain=.example.org';

document.cookie = 'level=2; expires=Thu, 2 Aug 2010 20:47:11 GMT;
path=/; domain=.example.org';

// Näytetään sisältö
alert(document.cookie);  // Saadaan: 'player=alien; level=2'
```

Koodiesimerkki 8: Kahden evästeen asettaminen ja niiden lukeminen.

4.7 Käyttäjäsyötteet

Käytettävissä olevia syötelaitteita ovat selainpeleissä käytännössä näppäimistö ja hiiri. Pelaamiseen varattuja kontrollereita, kuten ratteja tai pad-ohjaimia ei ole totuttu näkemään käytettävän Internet-selaimen kanssa, joten PC-perusvarustuksen ulkopuolisia "kuunneltavia" laitteita ei odoteta tulevan vastaan. Selaintuen ainakin oletetaan olevan erityislaitteille olematon.

Hiirellä käytettävät point 'n' click -tyyppiset pelit ovat kaikkein helpoimpia tehdä. Niissä toimintaideana on osoittaa hiirellä haluttua kohdetta ja klikata sitä, jolloin linkki tai skripti aktivoituu. Skriptin liittäminen onnistuu lisäämällä klikattavaan elementtiin `onclick`-attribuutti ja sen arvoksi skriptikomento, joka on yleensä funktiokutsu (koodiesimerkki 9).

```
<rect id="laatikko" x="0" y="0" width="100" height="100"
 fill="blue" onclick="katoa('laatikko')" />

<script type="text/ecmascript"><![CDATA[
    function katoa(elementId) {
        // Haetaan elementti XML-puusta
        var element = document.getElementById(elementId);

        // Piilotetaan elementti
        element.setAttribute('visibility', 'hidden');
    }
]]></script>
```

Koodiesimerkki 9: Hiirenklikkaukseen käytettävän katoa-esimerkkifunktion asettaminen ja sen toteutus.

Vastaavalla tavalla järjestetään muutkin hiiritapahtumat, kuten `onmouseover`, `onmouseout` ja `onmousemove` (näiden kuvaus myöhempänä). Näppäinten käsittely on sen sijaan hivenen hankalampaa.

Hiirelle on attribuutit valmiina, mutta näppäintapahtumat on rekisteröitävä erikseen (koodiesimerkki 10). Rekisteröinti voidaan suorittaa esimerkiksi heti SVG-tiedoston latautumisen jälkeen `onload`-attribuutin avulla. Itse rekisteröinti tapahtuu juurielementin `addEventListener`-funktiolla.

```
<svg id="root" onload="init()" ... >

    <script type="text/ecmascript"><![CDATA[
        function onKeyDown(evt) {
            /* Peruutetaan selaimen oletustapahtumat,
               kuten paluutoiminto backspace-näppäimellä
               ja uudelleen latautuminen F5-näppäimellä */
            evt.preventDefault();

            // Mitä painettiin?
            alert(evt.keyCode); // keyCode on kokonaisluku

            // switch-case...
        }

        function onKeyUp(evt) {
            evt.preventDefault();
            // switch-case...
        }

        function init() {
            // Rekisteröidään näppäinkuuntelijat
            var element = document.getElementById('root');
            element.addEventListener('keydown', onKeyDown, false);
            element.addEventListener('keyup', onKeyUp, false);
        }
    ]]></script>

</svg>
```

Koodiesimerkki 10: Näppäintapahtumien funktiototeutukset ja niiden rekisteröinti.

Rekisteröitävän tapahtuman nimi ei siis sisällä on-prefiksiä, vaan se kuuluu ainoastaan attribuuteille. Taulukossa 2 on kokoelma tapahtumista, jotka

voidaan rekisteröidä. Tapahtumien on tarkoitus käyttäytyä siten kuin ne on DOM Level 2 Event- ja DOM Level 3 Event -spesifikaatioissa määritelty (DOM Level 3 oli kirjoitushetkellä vielä vedosvaiheessa, mutta SVG Tiny 1.2 -suositus luottaa jo siihen). Koska nykyselaimet tukevat DOM-tapahtumia, näppäinsyötteet toimivat useimmissa tapauksissa, vaikkeivät varsinaisesti vielä SVG 1.2:a muuten tukisikaan.

Taulukko 2: Käytettävissä olevat syötelaitteita koskevat tapahtumat. Näppäintapahtumat ja hiiren rullatapahtuma suositeltiin vasta SVG 1.2:ssa.

Tapahtuma	Attribuuttinimi	Kuvaus	SVG	Alkuperä
DOMFocusIn	onfocusin	Elementissä on kohdistus (saavutettu esim. näppäimistön tabulaattorilla tai viemällä hiiren osoitin elementin päälle)	1.0	DOM Level 2 Events (W3C 2000)
DOMFocusOut	onfocusout	Kohdistus poistuu elementistä (saavutettu esim. näppäimistön tabulaattorilla tai viemällä hiiren osoitin pois päältä)	1.0	DOM Level 2 Events
DOMActivate	onactivate	Elementti on aktivoitu (saavutettu esim. Enter-näppäimellä tai klikkaamalla hiirtä)	1.0	DOM Level 2 Events
mousedown	onmousedown	Hiiren nappia on painettu	1.0	DOM Level 2 Events
mouseup	onmouseup	Hiiren nappi on vapautettu	1.0	DOM Level 2 Events
click	onclick	onmousedown + onmouseup	1.0	DOM Level 2 Events
mouseover	onmouseover	Hiiren osoitin on elementin päällä	1.0	DOM Level 2 Events
mousemove	onmousemove	Hiiren osoitin liikkuu elementin päällä	1.0	DOM Level 2 Events
mouseout	onmouseout	Hiiren osoitin poistuu elementin päältä	1.0	DOM Level 2 Events
mousewheel	-	Hiiren rullaa sovelletaan elementtiin	1.2	DOM Level 3 Events (W3C 2007a)
textInput	-	Elementtiin on syötetty kirjain tai muu merkki	1.2	DOM Level 3 Events
keydown	-	Näppäin on painettu alas	1.2	DOM Level 3 Events
keyup	-	Näppäin on vapautettu	1.2	DOM Level 3 Events

Mikäli vain suinkin mahdollista, esteettömyysnäkökulmasta ei tulisi suosia pelkästään hiirtä tai näppäimistöä. Tälle on varattu muun muassa `onactivate`-attribuutti, jolla elementti saadaan "valituksi" laiteriippumattomalla tavalla, mutta toiminto on harmillisen huonosti tuettu selaimissa. Parempi kompromissi on siis suosia hiirtä *ja* näppäimistöä.

4.8 Moninpeli

Ajatus sovelluksesta, joka on helposti ohjelmoitavissa ja jossa on Internet-ominaisuudet valmiina, kutkuttaa harrastajien mielikuvissa mahdollisuutta toteuttaa pieniä monen pelaajan verkkopelejä, jotka toimisivat eri tietokoneiden välillä. Selain on periaatetasolla juuri tällainen: helposti ohjelmoitava

valmis sovellus, jolla pääsee Internetiin. Skriptikieli takaa korkean tason ohjelmoitavuuden, laiteriippumattomuuden ja usein myös käyttöjärjestelmä- ja ohjelmistoriippumattomuuden. Kaikki kuulostaa olevan mahdollista.

Tiedossa ei kuitenkaan ole keinoja, joilla protokollatasoinen (TCP/IP, UDP) ohjelmointi onnistuisi pelkällä selainskriptillä, joka on tarkoitettu lähinnä (X)HTML/XML-dokumenttien puuhierarkioiden manipulointiin. Näin ollen suoraan eri pelaajakoneiden välinen kommunikointi tuskin onnistuu.

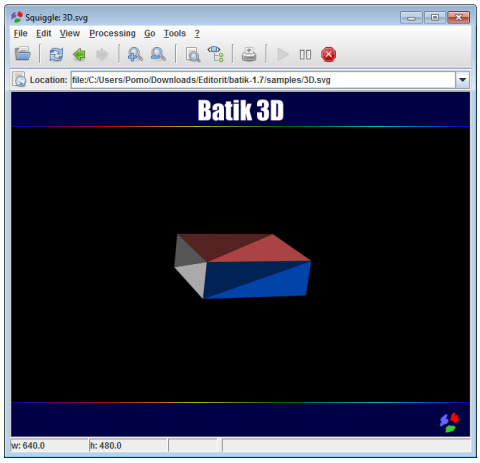
Periaatteessa SVG-tiedoston antaneelta palvelimelta voi kuitenkin pyytää "lisätietoja" AJAX-tekniikoilla (Asynchronous JavaScript and XML). AJAX:n avulla voidaan siirtää dataa selaimen (tai muun asiakasohjelman) ja palvelimen välillä ilman, että koko tiedosto täytyy ladata uudelleen. Menetelmä perustuu XMLHTTPRequest-olion, JavaScriptin ja (X)HTML/XML:n yhteiskäyttöön. (Tuntematon 2008; Refsnes Data 2009a.)

Internet-liikenteen välityksellä tapahtuva moninpeli toimisi siis epäsuorasti palvelinkoneen kautta esimerkiksi siten, että kaikki asiakaskoneet pyytävät säännöllisesti palvelimelta (esimerkiksi 10 kertaa sekunnissa) toisiin pelaajiin liittyvät tiedot. Asiakaskoneet puolestaan raportoivat omasta pelitilanteestaan palvelimelle säännöllisesti tai vain muutosten tapahtuessa. Huomautettakoon, ettei menetelmää ole kokeiltu käytännössä, joten idean onnistumisesta ei ole takeita.

Mikäli idea kuitenkin toimii käyttökelpoisena, verkon yli tapahtuva moninpeli kannattaa suunnitella ja toteuttaa niin, että se sietää kohtalaisesti viiveitä. Viivettä aiheuttavat muun muassa tietoliikenne (ping+siirtoaika noin 1 - 300 millisekuntia yhden pelaajan ja palvelimen välillä), skriptin käsittely selaimessa ja tulosten näyttäminen SVG-piirroksena (20 - 100 ms), palvelimen käsittelyaika (1 - 50 ms) ja mahdollisesti palvelimen kiintolevyn hakuviive (10 ms per tiedosto). Kovin intensiivistä räiskintäpeliä ei siis kannata yrittää toteuttaa, ellei se tapahdu aivan lähiverkossa ja riittävän tehokkailla koneilla. Tulosten näyttämisaikaa voi lyhentää käyttämällä yksinkertaisempia grafiikoita (katso luvun 2.3 Rasterointi viimeinen kappale). Tosissaan verkkopeliä optimoivan kannattaa tutustua muuhun kirjallisuuteen, esimerkiksi Juuso Savolaisen (2007) verkkopelisynkronointia käsittelevään insinöörityöhön.

4.9 Kolmiulotteisuus

SVG ei itsessään sisällä mitään kolmiulotteiseen grafiikkaan liittyvää, mutta sellainen on hallinnoitavissa skriptien avulla. Skriptejä tarvitaan 3D-koordinaattien projisoimiseksi kaksiulotteiselle pinnalle, joka SVG:n tapauksessa on yleensä tietokoneen näyttö. 3D-tilan käsittely tapahtuu siis skripteissä ja projisoitu lopputulos esitetään XML-puussa tavallisena 2D-kuvana (kuva 12).



Kuva 12: Batik-ohjelman (Apache 2009) mukana seuraa 3D-animaatioesimerkki.

Jokainen projisoitu monikulmio (esimerkiksi kolmio) piirretään 2D-pinnalle SVG:n `polygon`-elementillä. Sille tarvitsee vain määrittää täyttöväri ja pisteet, jotka vastaavat konkreettisia x- ja y-koordinaatteja näytettävässä kuvassa.

5 PELIN TOTEUTUS

SVG:n soveltuvuus selainpeleissä käytettäväksi osoitetaan itse tehdyllä pelillä. Esimerkkipeliksi toteutettiin Suomen Punaisen Ristin (SPR) koulupäihdevalistusta tukeva seikkailupeli, jossa pelaaja joutuu tekemään valintoja. Teknisesti peli on yksinkertainen hiirellä käytettävää point 'n' click -tyyppiä. SVG:n avulla peliä voidaan pelata kokoruututilassa (englanniksi fullscreen) siten, että näytön koko kapasiteetti saadaan käyttöön. Pelistä on tarkoitus tulla aiemmin käytetyn PowerPoint-version korvaaja.

5.1 Ennen aloitusta

Ennen pelikehityksen toimeenpanoa tulee kartoittaa, mitä pelin uudella versiolla yritetään tavoittaa. Tavoitteiden saavuttamiseksi täytyy valita oikeat tuotantotavat, sopiva arkkitehtuuri ja tarkoituksenmukaiset työkalut.

Ongelmat ja tavoitteet

Edellä kuvattu peli haluttiin PowerPoint-versiota tasokkaammaksi ja samalla mielellään myös Internet-käyttöiseksi. Käytettävä tekniikka oli vapaavalintainen, joten Flashin sijaan valittiin avoin, riippumattomampi ratkaisu, SVG. Point 'n' click -tyyppiset pelit on toki mahdollista toteuttaa myös kuvalinkkeihin perustuvilla staattisilla HTML-sivuilla. Selaimilla on kuitenkin kyky suurentaa SVG-kuvat automaattisesti täyttämään koko ruutu. Jos HTML-sivuilla, CSS-tyyleillä ja bittikarttakuvilla yritetään saavuttaa samaa efektiä, se on hidasta toteuttaa eivätkä kuvat siitä terävöidy. SVG on tällaiseen käyttöön paljon joustavampi vaihtoehto.

Pelin suunnittelu

Peli etenee sarjakuvamaisesti ruutu kerrallaan, joten suunnitteludokumentaation virkaa toimitti paperille käsin piirretyt pelin näyttöruudut. Peliin liittyy toki kartta, joka määrittelee, mistä näyttöruudusta seuraa mikäkin näyttöruutu, mutta SPR on jo suunnitellut kartan valmiiksi. Jäljelle jäi lähinnä näyttöruutujen tekninen toteutus ja niissä olevien tilannekuvien piirtäminen. Kukin näyttöruutu rakennettiin omaan SVG-tiedostoonsa ja niiden välillä pääsee liikkumaan linkkien avulla.

Linkkien (ja yleensäkin ulkoisten resurssien) toimiminen SVG-tiedostoissa edellyttää `xmlns:xlink="http://www.w3.org/1999/xlink"` -nimiavaruuden käyttöönottoa `svg`-tagissa. Toisin kuin HTML-sivuissa, viittaukset ulkoisiin resursseihin tehdään tämän vuoksi SVG-tiedostoissa `xlink:href`-attribuutilla.

Toteutusvaiheen alkupuolella ei vielä tiedetty, kuinka suuri skriptien osuus pelistä tulisi olemaan, sillä selainskriptien kirjoittamisesta ei juuri ollut aiempaa kokemusta. Pelin suunnitteluun otettiin sellainen lähestymistapa, että se voitaisiin toteuttaa täysin ilman skriptejä, mikäli olisi aivan pakko.

Työkalut

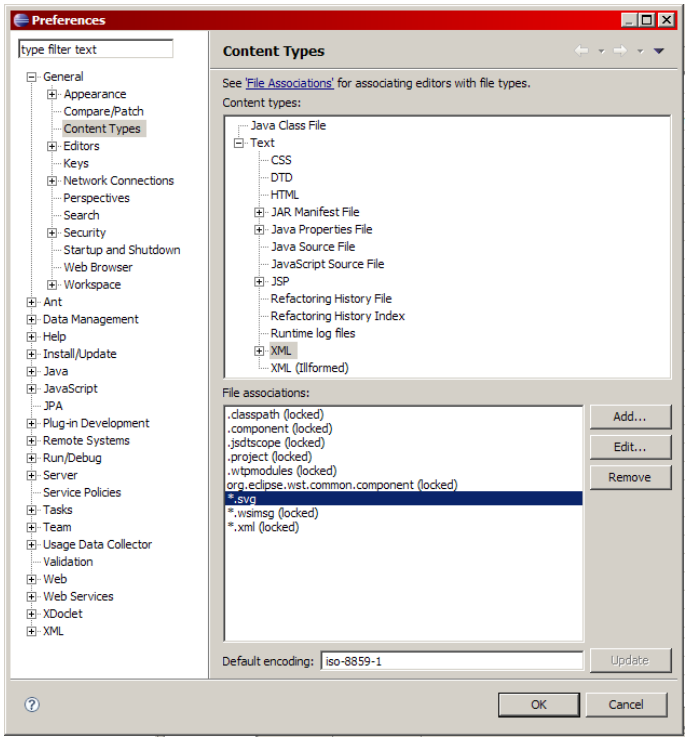
Työkalujen valintaan varattiin paljon aikaa. Moniin Internetistä saatavilla oleviin vaihtoehtoihin tutustuttiin, ominaisuudet punnittiin ja lopulta asetettiin vertailutaulukkoon. Tämän pohjalta SVG-kuvien päätuotantovälineeksi valittiin Inkscape-piirto-ohjelma monipuolisuuden ja tehokkaan käyttöliittymän vuoksi. Sen rinnalle otettiin Eclipse-ohjelmointiympäristö skriptien kirjoittamista, linkkien luontia ja muuta jälkikäsittelyä varten. Molemmat ovat ilmaisia avoimen lähdekoodin ohjelmistoja ja saatavilla Windowsille, Mac OS X:lle ja Linuxille. Hieman yllättäen Inkscape käyttää vanhempaa SVG-spesifikaation versiota 1.0, mutta se ei lopputulosta katsellessa näy missään. SVG 1.1 tarjoaa lähinnä parempaa modulaarisuutta, kuten jo alussa todettiin, joten se vaikuttaa vain joidenkin attribuuttien käyttöön.

5.2 Kehitysprosessi

Pelin kehitysprosessi oli melko suoraviivainen. Lyhyesti kuvaillen peli toteutettiin näyttöruutu kerrallaan siten, että Inkscapella piirrettiin kunkin kynävedoksen pohjalta vektorigrafiikkaversio, minkä jälkeen Eclipsellä siivottiin XML-koodia, lisättiin linkit ja kirjoitettiin skriptit. Pienen harjoittelun jälkeen tällainen Inkscapen ja Eclipsen välinen yhteistyö oli sujuvaa ja saumatonta.

Inkscape on tehty nimenomaan SVG-kuvien piirtovälineeksi, joten ilahduttavasti sen natiivitallennusmuoto on SVG jopa projektitiedostomuotoisena ("Inkscape-SVG"). Tämän ansiosta voidaan välittömästi tarkistaa selaimesta, että lopputulos on toivottu. Runsas ylimääräisten nimiavaruuksien käyttö kuitenkin tekee XML-koodista paikoin vaikealukuista. Tähän Inkscape tarjosi lääkkeeksi mahdollisuutta tallentaa piirros pelkistettynä SVG-tiedostona ("plain SVG"), mikä tekee tekstieditorilla jälkikäsittelyn miellyttävämmäksi ja varmemmaksi. Itse piirtäminen tapahtui Inkscapessa hiirellä polkuja muodostaen ja värittäen.

Eclipsen osalta SVG-tiedostot täytyi ensi töiksi saada jotenkin avautumaan lähdekoodimuodossa, sillä se ei tunnistanut SVG:tä suoraan XML-tyypiksi. SVG saatiin assosioitua Eclipsen XML-editoriin muokkaamalla asetuksia. Asetuksissa ohjelmalle kerrottiin SVG:n olevan XML-tyyppiä, jotta sitä voitaisiin editoida kuten XML:ää (kuva 13). Eclipse sallii XML:n muokkauksen sekä puuhierarkialla että lähdekooditasolla. Näistä jälkimmäinen oli mielekkäämpi vaihtoehto skriptien kirjoittamiselle.



Kuva 13: SVG-tiedostotyypin assosiointi Eclipsen XML-editoriin onnistuu näiden asetusten kautta. XML:n assosiaatiolistaan kuuluu lisätä ".svg".*

SVG-tiedostot muokattiin Eclipsessä JavaScript-projektina. Näin Eclipse osasi automaattisesti tarjota DOM-oliomallin ohjelmointirajapinnasta sopivat funktiot skriptejä varten ja huomauttaa ohjelmointivirheistä.

Aina, kun Inkscapella oli saatu piirros valmiiksi, se jälkikäsiteltiin Eclipsellä koodisiivouksella ja pelin vaatimien ominaisuuksien lisäämisellä. Inkscape tuotti jo valmiiksi varsin siistiä koodia. Tehty koodisiivous oli lähinnä piirto-vaiheessa käytettyjen apulinjojen poistamista, liukuvärien yhtenäistämistä, mittojen täsmentämistä (esimerkiksi 50,000372 → 50), skandinaavisten merkkien korjausta, kommentointia ja joidenkin attribuuttien manipulointia pelin tarpeisiin. Esimerkiksi Inkscapen asettamat fyysiset mitat `svg`-tagissa (`width` ja `height`) vaihdettiin `viewBox`-attribuuttiin, jotta kuva voisi skaalautua vapaasti ja täyttää koko sovellusikkunan automaattisesti (tai toimia muulla tavalla SVG-katselimen parhaaksi katsomallaan tavalla). `viewBox`in tehtävänä on rajata näkymäalue, kuten kuvassa 9 (s. 12) oli tehty.

Skriptit

Peli sisältää animaatioita, häivytyksiä, satunnaisuutta, hiiriefektejä ja muita skriptillä toteutettuja pieniä yksityiskohtia. Skriptikieleksi valittiin JavaScript.

Skriptikoodi oli hidasta toteuttaa, sillä niiden käyttäytymistä ei spesifioitu pelin suunnitteluvaiheessa. Onneksi ne kuitenkin toimivat pelissä pääsääntöisesti yhdenmukaisesti eri selainten välillä, joten koodin kirjoitus oli melko kivutonta.

Tyylit

Inkscape asetti kaikki muotoiluun liittyvät elementtien ominaisuudet CSS:stä tuttuun `style`-attribuuttiin (kuten fonttikoon ja täyttövärin) koodiesimerkin 11 mukaiseen tapaan.

```
style="font-family:Arial;font-size:32px;fill:black"
```

Koodiesimerkki 11: Elementin alustaminen CSS:llä.

Tämä tekee skripillä ohjaamisen hankalaksi, sillä muokkausta vaativiin yksittäisiin ominaisuuksiin ei näin päästä suoraan käsiksi, ellei ohjelmoija halua aivan välttämättä onkia merkkijonosta jokaista ominaisuutta ja sen arvoa erikseen. Yllä oleva esimerkki kuitenkin tuottaa saman lopputuloksen kuin erillisinä attribuutteina esitettynä alla olevassa koodiesimerkissä 12. Skriptien toimimiseksi tarvittiin siis vain poistaa tyylit ja korvata ne attribuuteilla.

```
font-family="Arial"
font-size="32px"
fill="black"
```

Koodiesimerkki 12: Elementin alustaminen attribuuteilla.

Lopputulos

Noin kuukauden verran kestäneen näyttöruutujen piirtämisen ja ohjelmoinnin jälkeen peli valmistui. Valmis peli koostuu 30 näyttöruudusta ja neljästä erillisestä skriptitiedostosta, joihin koottiin usein tarvittavat funktiot. Pelin grafiikka saatiin liukuvärien avulla riittävän laadukkaaksi, joten suodattimia ei tarvittu käyttää pelissä lainkaan. Animaatioiden kannalta ne olisivat sitä paitsi saattaneet tehdä pelistä yliraskaan vanhemmilla tietokoneilla suoritettavaksi (lisää luvussa 5.3.3 Suorituskyky).

Peli ei sisällä kovin paljon uudelleenkäytettäviä osia. Skriptifunktiot saatiin melko geneerisiksi, mutta itse piirroksista ainoastaan muutamaa kyettiin hyödyntämään useassa paikassa. Pelissä ei myöskään voi (tai ainakaan ole tarkoitus) palata takaisin aiempaan pelitilanteeseen, ellei selaimen paluupainikkeella huijaamista lasketa sellaiseksi. Seikkailupeligenressä on

epätavallista, ettei pelaaja pääse liikkumaan minkäänlaisella kartalla palatakseen paikkoihin, joissa hän on aiemmin käynyt. Tämä tosin ei ollut esimerkkipelin tavoite.

5.3 Testaus

Esimerkkipeli testattiin käsin suurimmalta osin. Testauksessa keskityttiin selainten väliseen yhdenmukaisuuteen, XML- ja skriptikoodin oikeellisuuteen (staattinen testaus) sekä suorituskykyyn. Kaikki testauksen tavoitteet saavutettiin riittävällä tarkkuudella.

5.3.1 Yhdenmukaisuus eri selainten välillä

Ensimmäinen SVG:ssä mielellään testattava asia on se, vastaako piirtojälki odotettua. Ainoa tapa tehdä tämä on arvioida piirroksen lopputulosta silmämääräisesti. Tuloksen luotettavuutta voidaan parantaa vertailemalla piirtojälkeä muilla selaimilla (tai erillisillä SVG-katseluohjelmilla), sillä eri ohjelmilla on taipumus tulkita spesifikaatiota vähän omalla tavallaan tai tehdä tarkoituksenmukaisia poikkeuksia. Lopputuloksen tarkastelu viidellä (tai useammalla) eri SVG-katselimella ei ole ollenkaan liioittelua, sillä se auttaa priorisoimaan löytyneitä ongelmia: jos sama epänormaali ilmiö löytyy useasta eri ohjelmasta, vika on todennäköisemmin omassa koodissa kuin käytetyssä SVG-katselimessa. Piirtojäljen yhdenmukaisuudesta eri selaimien välillä on pidetty erityisen tarkkaa huolta pelikehityksen alusta saakka.

Skripteihin kiinnitettiin myös jonkin verran huomiota, sillä niistä riippuu pelin toiminnallisuus. Kovin tarkkoja yksikkötestauksia ei kuitenkaan tehty, sillä funktiot eivät käsittele ulkopuolisia käyttäjäsyötteitä esimerkiksi lomakkeiden muodossa. Funktiot käsittelevät siis vain luotettavasta lähteestä olevaa dataa. Pelin ainoa interaktiivisuus on nappuloiden klikkailussa. Millään muulla tavalla pelaaja ei pääse vaikuttamaan skriptien käyttäytymiseen.

Pelin kehityksen alkupäässä testaustiheys selainten yhdenmukaisesta käyttäytymisestä, niin SVG-kuvien piirtämisessä kuin skripteissä, oli suurimmillaan. Näin saatiin riittävän aikaisessa vaiheessa hahmotettua, mikä toimii ja mikä ei, joten loppuvaiheessa korjattavien ongelmien määrä jäi pieneksi.

Pelin lopullisen version grafiikka ja skriptien käyttäytyminen todettiin riittävän yhdenmukaiseksi (vain kolme vähämerkityksellistä eroa jäi jäljelle) seuraavilla selaimilla/SVG-katseluohjelmilla:

- Batik 1.7
- Chrome 1.0.154.65
- Firefox 3.0.11
- Internet Explorer 8 RENESIS SVG Player -selainlaajennuksella
- Opera 9.64
- Safari 3.2.3.

5.3.2 Koodin oikeellisuus

Aina ei riitä, että koodi vain näyttää toimivan oikein, vaan sen tulee olla myös rakenteellisesti kunnossa. SVG-koodien eheysvaatimukseksi asetettiin seuraavat ehdot:

- linkit toisiin SVG-kuviin ovat ehjiä
- ei XML-kielioppivirheitä
- ei skriptivirheitä.

Linkkien eheys testattiin "surffaamalla" pelipuun kaikki vaihtoehdot läpi käsin. Urakka ei ollut suuren suuri, sillä peli toimii sarjakuvamaisesti näyttöruutu kerrallaan. Pelin kesto on lyhimmillään 7 ruutua ja pisimmillään 15 ruutua, mutta haarautuu monessa kohdassa. Pelin lopullisesta versiosta ei löytynyt "katvealueita".

XML-kielioppivirheet vältettiin Eclipsen reaaliaikaisen, sisäänrakennetun XML-validaattorin ansiosta. Lopputulos varmistettiin vielä W3C:n (2009b) merkkikielivalidaattoria vasten. Tulos oli kaikissa SVG-tiedostoissa validi. Koska SVG on teknisesti puhdasta XML:ää, voidaan tämän perusteella pelkästään XML-validaattoriin luottaa täysin, kunhan sille on annettu oikea DTD.

Skriptit ovat väljiä ohjelmointikieliä, joten niiden variaatioiden kirjavuuden vuoksi ei ole olemassa yhtä oikeaa validointitapaa. Eclipsekin kykenee puuttumaan lähinnä vain selviin syntaksivirheisiin. Tämän vuoksi skriptivirheet ilmenevät yleensä vasta suorituksen aikana. Avuksi tähän otettiin koodityylin ja -laadun tarkistava JSLint (Crockford 2009), jolloin yhdenmukaisuus eri selainten välillä on todennäköisesti parempi.

JSLint huomautti enimmäkseen animointiin käytetystä `setTimeout`-funktiosta. Muita toimivia animointikeinoja ei ollut tiedossa, joten tähän korjausehdotukseen ei voitu puuttua millään tavalla. Muita korjausehdotuksia oli

vain muutamia, mutta näitäkään korjauksia ei tehty, koska väljää kieltä käytettiin tahallisesti hyväksi. Esimerkiksi numeron sisältävä attribuuttiarvo voi olla joko kokonaisluku tai liukuluku. JSLintin ehdottamaa samaa arvoa ja *samaa tyyppiä* tutkivaa "==="-vertailuoperaattoria ei käytetty, koska ei voitu olettaa, kumpaa tyyppiä kyseinen numero on. "==="-operaattorin käytöstä olisi siis luultavasti ollut enemmän haittaa kuin hyötyä. Insinöörityön pääkohteena on kuitenkin SVG eikä JavaScript, joten skriptejä ei nähty tarpeelliseksi virittää viimeisen päälle. Nykyiset skriptit toimivat näennäisesti yhdenmukaisesti edellä luetelluilla SVG-katselinohjelmilla. JSLintin löytämät rakennevirheet (esimerkiksi funktion käyttö ennen sen määrittelyä) korjattiin pelin lopulliseen versioon.

5.3.3 Suorituskyky

Esimerkkipelin kehittämiseen käytetty kotitietokone on noin neljä vuotta vanha (prosessori AMD Athlon 64 3000+ (1,8 GHz, yksi ydin) ja RAM-muisti 1 Gt 400 MHz DDR). Suorituskykyvaatimukseksi asetettiin, ettei animaatiosessioissa sen koneen prosessorin käyttöaste saa nousta sataan prosenttiin kuin vain hetkellisesti, jolloin voidaan olla melko varmoja siitä, että peli toimii käyttökelpoisena myös jonkin verran sitä vanhemmissa tietokoneissa. Peliä ajettiin näytön tarkkuudella 1280x1024 (jotkut selaimet virtuaalikoneessa tarkkuudella 1024x768), selainikkuna suurennettuna.

Jos animaatiot ovat liian raskaita, liikerata ei ole sujuva, vaan epäsäännöllisen katkeileva (selain tai SVG-katseluohjelma jättää kuvaruutuja välistä) tai todella hidas (selain tai SVG-katseluohjelma *ei* jätä kuvaruutuja välistä). Animaatioiden kuormituksen hallitsemiseksi skriptifunktioihin lisättiin FPS-parametri (Frames Per Second), joka määrittelee animaation ruudunpäivitystiheyden. FPS-arvot vaihtelivat 10 - 30 välillä animaatioissa olevien elementtien monimutkaisuuden mukaan. Yksinkertaisissa ja yksivärisissä elementeissä voitiin käyttää suuria ruudunpäivitystiheyksiä, mutta monimutkaisissa ja liukuvärejä sisältävissä täytyi vastaavasti käyttää pienempiä arvoja. Tällä tavalla tasapainotellen voitiin saada katkeilematon animaatioelämys niiltä osin, joissa elementtien yksinkertaistaminen ei riittänyt tai ei olisi ollut mielekästä.

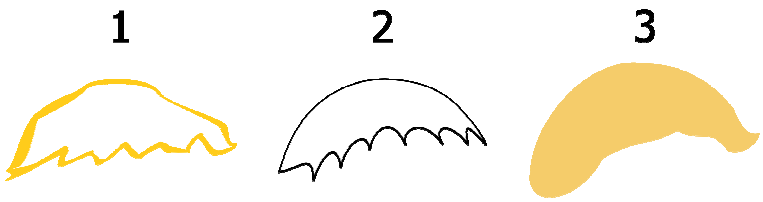
Suorituskykyvaatimus saavutettiin: animaatiot ja häivytykset olivat riittävän sulavia jopa virtuaalikoneessa ajettuna. Eri selaimien rasterointinopeudessa ei havaittu suuria eroja, joskin kokeiltu versio Safarista oli vaikeuksissa muu-

taman yksittäisen SVG-animaation kanssa. Monirivisen tekstin häivytys oli yllättävän raskasta kaikilla selaimilla, mutta tästäkin selvittiin pilkkomalla teksti pieniksi osakokonaisuuksiksi eli rajoittamalla rivien määrää kutakin tekstielementtiä kohden.

5.4 Kompromissit

Vaikka peliin sisältyvän piirtotyön määrä arvioitiin huolella, jouduttiin jo toteutusvaiheen alkupuolella pohtimaan yksinkertaistuksia aikataulun kuromiseksi. Osaltaan aikataulun venymiseen vaikuttivat vektorigrafiikan tuottamisen kokemattomuus, työkalujen (ja SVG:n) opettelu ja toimivuuden varmistaminen kaikille käytössä olleille selaimille. Konqueror 3 -selaimen tukemisesta jouduttiin luopumaan kokonaan, sillä ainuttakaan siinä vaivannutta ongelmaa ei saatu korjattua esimerkkipelistä. Muun muassa skriptin vaikutus ei aina päivittynyt näkyviin ja `tspan`-tekstirivitys poikkesi muista selaimista.

Yksinkertaistusmenetelminä käytettiin polkuja muodostavien solmujen määrän minimointia (esimerkki kuvassa 14) sekä liukuvärien säännöstelyä. Lisäksi pelihahmojen kasvot jätettiin ellipsin muotoisiksi. Kun piirrokset ovat yksinkertaisia, ne saadaan nopeammin valmiiksi.



Kuva 14: Pelihahmon eri hiusversioita. Version 1 piirtopolku koostuu 64:stä solmukohdasta, version 2 kahdestatoista ja version 3 seitsemästä. On selvää, että version 1 tai 2 teettäminen kaikkiin pelin 30 näyttöruutuun olisi ollut tarpeettoman työlästä.

Peliin piirretyissä elementeissä käytettiin häivytystä monin paikoin. Jos samaan elementtiin kohdistuvat `fadeIn`- ja `fadeOut`-skriptifunktiot joutuivat suoritukseen yhtä aikaa, ne kumosivat toisensa, minkä seurauksena häivytys keskeytyi. Tämä tilanne oli mahdollista tapauksissa, joissa suorituskyky ei ollut riittävä, eli ensimmäiseksi annettu häivytyskomento ei suoriutunut odotetussa ajassa. Häivytysfunktiot oli toteutettu siten, että ne eivät tarkista häivytyksen tilaa tai etenemistä. Tälle ei rakennettu erillistä "älykkyyttä", joten jääköön se jatkokehitysehdotukseksi.

6 YHTEENVETO

Tutkimuksessa on selvitetty, mitä vektorigrafiikka on, miten se toteutuu SVG:ssä, miten siitä tehdään Internet-selaimella pelattava tietokonepeli käytännössä ja mitä asioita siinä täytyy ottaa huomioon. Tutkimus painottui selvästi nykyisten selaimien suhtautumiseen SVG:een eli selaintukeen, minkä tarkoitus oli osoittaa tekniikan todellinen käyttökelpoisuus. Tämä vielä vahvistettiin toteutetulla esimerkkipelillä.

Esimerkkipelin interaktiiviset osat saatiin toteutettua linkeillä ja monipuolisilla skripteillä. Liukuväreillä grafiikka saatiin näyttämään hienolta sekä pienillä että suurilla tarkkuuksilla. Pelin kehityksen aikana tuli vastaan asioita, joita ei alussa otettu huomioon. Ensin piirtämisen hitaus pakotti yksinkertaistamaan ja vähentämään yksityiskohtien määrää. Sitten huomattiin, miten tarpeellinen skriptiominaisuus todella on, joten sen käyttöön liittyviä näkökulmia tutkittiin aiottua tarkemmin. Näin SVG:n lisäksi tuli automaattisesti opeteltua samalla JavaScript-kieltä ja vähän DOM-oliomallirajapintaa, vaikka se ei kuulunut alkuperäiseen suunnitelmaan. Pelistä saatiin täysin toimiva ja käyttökelpoinen. Testaus tehtiin huolellisesti ja myös suorituskykynäkökulmaan päästiin paneutumaan. Kaikki pelille asetetut tavoitteet saavutettiin.

Edistykselliset työkalut nopeuttavat työskentelyä olennaisesti. Vaikka esimerkkipelin piirtämisprosessi kuvattiinkin hitaaksi, erillisellä piirto-ohjelmalla pelin graafinen puoli saadaan kuitenkin hoidettua sellaisella nopeudella, johon pelkällä teksti- tai XML-editorilla ei ikinä ylletä. Myöskin SVG:ssä käytettäviä ohjelmointi- ja merkkikieliä ymmärtävä tekstieditori on korvaamaton apu. Värikoodaus ja automaattinen syntaksitarkistus paljastavat tehtyjä virheitä.

Oleellisimpia yleisvaatimuksia SVG-pelin tekemiseen ovat tietenkin ohjelmointitaidot ja XML-tekniikan perustuntemus. Muun muassa animointiin ja käyttäjäsyötteiden hallintaan annettiin joitakin vinkkejä, mutta muutoin pelin toimintalogiikka ja arkkitehtuuri riippuvat pelityypistä, joten yksityiskohtaisia yleisohjeita tietyntyyppisen SVG-pelin tekemiseen on vaikeaa antaa. Pelin tekeminen täytyy kokea itse.

6.1 SVG:n pelimahdollisuudet

Kun on saatu kartoitettua, mitä kaikkea SVG:llä ja sen kanssa käytettävillä tekniikoilla voidaan tehdä, pystytään tekemään jonkinlaisia arvioita siitä, mihin se pystyy ja minkälaisia pelejä sen avulla saadaan aikaan. SVG-selainpelien kirjon rajat asettaa ennemmin skriptit ja käytettävissä olevat laitteet kuin itse SVG, sillä pelien ohjelmoitava (ja yleensä myös animoitava) osuus toteutuu skripteillä. Skriptien osuus pelistä on yllättävän suuri, ja SVG:n vastuulle jää lähinnä tulosten näyttäminen. Ilman skriptejä SVG-pelit olisivat toivottomia XML-ominaisuuden vuoksi.

Tutkittujen ominaisuuksien valossa SVG:n pelimahdollisuudet ovat melkoisen laajat. Luetellaan yleisesti joukko joitakin tietokonepelien lajityyppejä:

- ammuntapelit
- autopelit, urheilupelit ja muut simulaatiot
- lautapelit ja ongelmanratkontapelit
- musiikkipelit
- seikkailupelit ja roolipelit
- strategiapelit
- rakentelupelit
- tappelupelit
- tasohyppelypelit.

Selaimen kanssa toimivia laitteistoja silmälläpitäen ei näy esteitä toteuttaa mitä tahansa näistä pelityypeistä lukuun ottamatta musiikkipelejä tai muita ääniominaisuuksia vaativia ratkaisuja (luku 4.5). Näyttää siltä, että SVG laajentaa sitä, mitä pelkällä selainskriptien ja kuvatiedostojen yhdistelmällä on jo aiemmin pystytty tekemään. Etenkin kolmiulotteisen toimintaympäristön mahdollisuus on asia, joka ei ennen SVG:tä ole ollut mahdollista (luku 4.9). Lisäksi geometriamuotojen suora käyttö, transformaatiot, kuvamanipulaatiot (suodattimilla) ja muut SVG:lle ominaiset piirteet synnyttävät helposti kokeilunhalua.

6.2 SVG:n tulevaisuus

SVG:n asema suhteessa sen kilpailijoihin on mielenkiintoinen. Ensimmäinen SVG-suositus julkaistiin melko pian sen jälkeen, kun vuonna 1999 julkaistu HTML 4 -suositus oli kypsymässä selaimissa, joten ajoitus oli sikäli hyvä, ett-

ei muuta vastaavaa sopivaa avointa standardia ollut tuolloin saatavilla. Suljetut vektorigrafiikkamuodot olivat kuitenkin jo tuolloin valmiita sekä yleisessä käytössä. Näiden selainlaajennusperiaatteen ansiosta yhdenmukaisuus eri selainten välillä on taattu, joten niille tulee jatkossakin löytymään oma kannattajajoukkonsa.

Vuosia kestäneen selaintukiparanemisen myötä SVG 1.1:n voidaan kuitenkin tutkittujen näkökulmien perusteella todeta kelpaavan liki kaikenlaisille selainpeleille. Useissa tapauksissa ei siis tarvita enää sitoutumista yhden yrityksen hallinnoimiin suljettuihin ratkaisuihin ja investoida niitä varten tehtyihin kalliisiin kehitysympäristöihin. Vaikka selaimet eivät katakaan kaikkia SVG-suosituksen piirteitä, sen avulla tehdyt pelit ovat käyttökelpoisia. Selainkilpailun ansiosta yksittäiset puutteet korjaantuvat ajan myötä, joten SVG-pelien yleistyminen lähitulevaisuudessa on melko todennäköistä. Tästä kertoo sekin, että SVG-spesifikaatiosta on tekeillä uusia versioita. Täysimittainen SVG 1.2 tulee sisältämään multimediaominaisuuksia, jotka puuttuivat aiemmista versioista. Mobiililaitteisiin soveltuva SVG Tiny 1.2-suositus onkin jo valmis. Sieltä saa esimakua tulevasta kehityssuunnasta. SVG ei vielä korvaa täysin kilpailijoitaan, mutta on jo nyt varsin lupaava vektorigrafiikan yleisratkaisu.

VIITELUETTELO

Adobe Systems Incorporated, 2008a. *ADOBE INDESIGN CS4 -tuotteen käyttö* [PDF-dokumentti]. Copyright 2008 [viitattu 17.7.2009]. Saatavilla: http://help.adobe.com/fi_FI/InDesign/6.0/indesign_cs4_help.pdf.

Adobe Systems Incorporated, 2008b. *ADOBE PHOTOSHOP CS4:n käyttö* [PDF-dokumentti]. Copyright 2008 [viitattu 24.6.2009]. Saatavilla: http://help.adobe.com/fi_FI/Photoshop/11.0/photoshop_cs4_help.pdf.

Adobe Systems Incorporated, 2009a. *ActionScript Technology Center* [verkkodokumentti]. Copyright 2009 [viitattu 19.7.2009]. Saatavilla: http://www.adobe.com/devnet/actionscript/.

Adobe Systems Incorporated, 2009b. *Flash Player 10: Features* [verkkodokumentti]. Copyright 2009 [viitattu 20.7.2009]. Saatavilla: http://www.adobe.com/products/flashplayer/features/.

Adobe Systems Incorporated, 2009c. *Using ADOBE FLASH CS4 PROFESSIONAL* [PDF-dokumentti]. Päivitetty 5.3.2009 [viitattu 21.7.2009]. Saatavilla: http://help.adobe.com/en_US/Flash/10.0_UsingFlash/flash_cs4_help.pdf.

The Apache Software Foundation, 2009. *Batik SVG Toolkit* [verkkodokumentti]. Päivitetty 19.3.2009 [viitattu 1.9.2009]. Saatavilla: http://xmlgraphics.apache.org/batik/.

Crockford, Douglas, 2009. *JSLint, The JavaScript Code Quality Tool* [verkkopalvelu]. Päivitetty 13.9.2009 [viitattu 18.9.2009]. Saatavilla: http://www.jslint.com/.

Evans, David, 2009. *PostScript vs. PDF* [verkkodokumentti]. Copyright 2009 Adobe Systems Incorporated [viitattu 1.11.2009]. Saatavilla: http://www.adobe.com/print/features/psvspdf/.

Foley, James D. ja muut, 1994. *Introduction to Computer Graphics*. Yhdysvallat: Addison-Wesley Publishing Company, Inc. ISBN: 0-201-60921-5

Futuremark Corporation, 2009. *Browser Statistics* [verkkodokumentti]. Copyright 2009 [viitattu 17.5.2009]. Saatavilla: http://service.futuremark.com/peacekeeper/browserStatistics.action.

Guthrie, Scott, 2008. *Silverlight Tutorial Part 1: Creating "Hello World" with Silverlight 2 and VS 2008* [verkkodokumentti]. Julkaistu 22.2.2008 [viitattu 6.7.2009]. Saatavilla: http://weblogs.asp.net/scottgu/pages/silverlight-tutorial-part-1-creating-quot-hello-world-quot-with-silverlight-2-and-vs-2008.aspx.

Hobby, John ja MetaPost-kehitystiimi, 2009. *A User's Manual for MetaPost* [PDF-dokumentti]. 25.6.2009 [viitattu 19.7.2009]. Saatavilla: http://www.tug.org/docs/metapost/mpman.pdf.

Koch, Peter-Paul, 2007. *Cookies* [verkkodokumentti]. 6.7.2007 [viitattu 11.10.2009]. Saatavilla: http://www.quirksmode.org/js/cookies.html.

Microsoft Corporation, 1997a. *TrueType fonts* [verkkodokumentti]. Päivitetty 30.6.1997 [viitattu 23.6.2009]. Saatavilla: http://www.microsoft.com/typography/TrueTypeFonts.mspx.

Microsoft Corporation, 1997b. *The TrueType rasterizer* [verkkodokumentti]. Päivitetty 30.6.1997 [viitattu 23.6.2009]. Saatavilla: http://www.microsoft.com/typography/TrueTypeRasterizer.mspx.

Microsoft Corporation, 2007. *Description of the guidelines for selecting the appropriate picture format in an Office program* [verkkodokumentti]. Viimeksi katsastettu 7.5.2007 [viitattu 18.6.2009]. Saatavilla: http://support.microsoft.com/kb/320314.

Microsoft Corporation, 2009. *The History of Microsoft - 1992* [verkkodokumentti]. 4.6.2009 [viitattu 23.6.2009]. Saatavilla: http://channel9.msdn.com/shows/History/The-History-of-Microsoft-1992/.

Mozilla Corporation, 2008. *SVG improvements in Firefox 3* [verkkodokumentti]. Laadittu 15.1.2008 [viitattu 8.4.2009]. Saatavilla: https://developer.mozilla.org/en/SVG_improvements_in_Firefox_3.

Mozilla Corporation, 2009. *Firefox Releases* [verkkodokumentti]. Copyright 1998 - 2009 [viitattu 8.4.2009]. Saatavilla: http://www.mozilla.com/en-US/firefox/releases/.

Net Applications Corporate, 2009. *Browser Market Share* [verkkodokumentti]. 28.6.2009 [viitattu 29.6.2009]. Saatavilla: http://marketshare.hitslink.com/report.aspx?qprid=0&qpmr=15&qpdt=1&qpct=0&qptimeframe=M.

Nykänen, Ossi, 2007. *SVG – Skaalautuva vektorigrafiikka*. Jyväskylä: Docendo, WSOY. 1. painos.
ISBN-13: 978-951-846-306-4
ISBN-10: 951-846-306-9

Refsnes Data, 2009a. *AJAX Introduction* [verkkodokumentti]. Copyright 1999 - 2009 [viitattu 1.10.2009]. Saatavilla: http://www.w3schools.com/Ajax/ajax_intro.asp.

Refsnes Data, 2009b. *Browser Statistics* [verkkodokumentti]. Copyright 1999 - 2009 [viitattu 6.4.2009]. Saatavilla: http://www.w3schools.com/browsers/browsers_stats.asp.

Refsnes Data, 2009c. *XML DOM Introduction* [verkkodokumentti]. Copyright 1999 - 2009 [viitattu 6.10.2009]. Saatavilla: http://www.w3schools.com/dom/dom_intro.asp.

Savolainen, Juuso, 2007. *Verkkopelin synkronointi*. Insinöörityö. Helsingin ammattikorkeakoulu Stadia. Tietotekniikan koulutusohjelma. Helsinki. 2007.

Schiller, Jeff, 2009. *SVG Support Tables* [verkkodokumentti]. Päivitetty 29.3.2009 [viitattu 6.4.2009]. Saatavilla: http://www.codedread.com/svg-support.php.

Sun Microsystems, 2008. *How to Use Scroll Panes* [verkkodokumentti]. Viimeksi päivitetty 14.2.2008 [viitattu 20.7.2009]. Saatavilla: http://java.sun.com/docs/books/tutorial/uiswing/components/scrollpane.html.

Tuntematon, 2008? todennäköinen vuosi (tekijä esiintyy nimimerkillä). *AJAX - Asynchronous JavaScript and XML* [verkkodokumentti] [viitattu 1.10.2009]. Saatavilla: http://www.ohjelmointiputka.net/opas.php?tunnus=ajax.

Tuntemattomat, 2009 (tekijät esiintyvät nimimerkeillä). *svgweb - Scalable Vector Graphics for Web Browsers using Flash* [verkkodokumentti]. Päivitetty 20.8.2009 [viitattu 5.9.2009]. Saatavilla: http://code.google.com/p/svgweb/.

World Wide Web Consortium, 2000. *Document Object Model (DOM) Level 2 Events Specification* [verkkodokumentti]. 13.11.2000 [viitattu 22.9.2009]. Saatavilla: http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/.

World Wide Web Consortium, 2003. *Scalable Vector Graphics (SVG) 1.1 Specification* [verkkodokumentti]. 14.1.2003 [viitattu 6.4.2009]. Saatavilla: http://www.w3.org/TR/2003/REC-SVG11-20030114/.

World Wide Web Consortium / The SVG Working Group, 2006. *W3C Scalable Vector Graphics (SVG) 1.1 Test Suite* [verkkodokumentti]. 13.12.2006 [viitattu 6.4.2009]. Saatavilla: http://www.w3.org/Graphics/SVG/Test/20061213/.

World Wide Web Consortium, 2007a. *Document Object Model (DOM) Level 3 Events Specification* [verkkodokumentti]. 21.12.2007 [viitattu 24.9.2009] Saatavilla: http://www.w3.org/TR/2007/WD-DOM-Level-3-Events-20071221/.

World Wide Web Consortium, 2007b. *Recommended DTDs to use in your Web document* [verkkodokumentti]. Päivitetty 26.4.2007 [viitattu 4.7.2009] Saatavilla: http://www.w3.org/QA/2002/04/valid-dtd-list.html.

World Wide Web Consortium, 2008. *Synchronized Multimedia Integration Language (SMIL 3.0)* [verkkodokumentti]. Päivitetty 1.12.2008 [viitattu 20.9.2009] Saatavilla: http://www.w3.org/TR/2008/REC-SMIL3-20081201/.

World Wide Web Consortium, 2009a. *Document Object Model (DOM)* [verkkodokumentti]. Päivitetty 6.1.2009 [viitattu 6.10.2009]. Saatavilla: http://www.w3.org/DOM/.

World Wide Web Consortium, 2009b. *The W3C Markup Validation Service* [verkkopalvelu]. Päivitetty 26.3.2009 [viitattu 18.9.2009]. Saatavilla: http://validator.w3.org/.

Weingartner, Peter, 2006. *A First Guide to PostScript* [verkkodokumentti]. Päivitetty 24.2.2006 [viitattu 16.7.2009]. Saatavilla: http://www.tailrecursive.org/postscript/postscript.html.