

Erik Sissala

# 3D-kaupungin suunnittelu ja toteutus osana WebGL-multimediasovellusta

---

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinööriytyö

17.5.2015

Tekijä Otsikko  Sivumäärä Aika	Erik Sissala 3D-kaupungin suunnittelu ja toteutus osana WebGL-multimediasovellusta  42 sivua 26.4.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaajat	Toimitusjohtaja Oskari Heikel Lehtori Antti Laiho
<p>Insinöörityön tarkoituksena oli suunnitella ja toteuttaa 3D-kaupunki osana interaktiivista multimediasovellusta, joka hyödyntää WebGL-tekniikkaa. WebGL-tekniikan avulla verkkoselaimiin saadaan 3D-grafiikkaa ilman ylimääräisiä liitännäisiä.</p> <p>Insinöörityön toimeksiantaja oli suomalainen hissiyhtiö. Työ lähti liikkeelle toimeksiantajan tarpeesta saada esittelytilaansa enemmän interaktiivisuutta, mielenkiintoista sisältöä ja modernia teknologiaa. Entinen esittelytilan sisältö oli koostunut pääasiassa vain staattisista kuvista, teksteistä ja itsenäisesti toistuvista videoista ja animaatioista.</p> <p>Sovelluksen käyttöliittymänä ja kotivalikkona toimiva 3D-kaupunki mallinnettiin hyödyntäen toimeksiantajalta saatuja toiveita ja referenssimateriaaleja. Kaupunki koostui erityyppisistä rakennuksista, joihin lisättiin toimeksiantajan tuotteita ja palveluja. 3D-kaupungin toteutus sisälsi mallintamista, animointia ja optimointia. Multimediasovelluksen avulla tuotteet ja palvelut saatiin isolle videoseinälle virtuaalisesti ja näyttävästi.</p> <p>Työn tuloksena syntyi interaktiivinen 3D-kaupunki osana multimediasovellusta. Sovelluksen ensimmäinen versio asennettiin toimeksiantajan näyttelytilaan tammikuussa 2015. Kahden ensimmäisen kuukauden käyttökokemukset olivat erittäin positiivisia, ja käyttöliittymänä toimivaa 3D-kaupunkia pidettiin helppokäyttöisenä ja visuaalisesti näyttävänä. Tulevaisuudessa sovellusta on tarkoitus viedä yhtiön muiden maiden esittelytiloihin sekä ottaa messu- ja asiakaskäyttöön. Sisällön tuottaminen sovellukseen jatkuu vielä tulevina vuosina.</p>	
Avainsanat	WebGL, 3D, multimediasovellus, mallintaminen, animaatio, optimointi

Author Title	Erik Sissala Design and implementation of 3D city for multimedia application
Number of Pages Date	42 pages 26 April 2015
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructors	Oskari Heikel, CEO Antti Laiho, Senior Lecturer
<p>The purpose of this final year project was to design and implement a 3D city as a part of an interactive multimedia application by using WebGL that is a web technology that enables 3D graphics in the browser without having to install additional software.</p> <p>This study was made for a Finnish elevator and escalator company. The demand of the company was to get more interactivity, engaging content and modern technology in their showroom, since the showroom's former content consisted of static pictures, texts, self-running videos and animations.</p> <p>The 3D city that is also used as a user interface and home menu of the multimedia application was modeled in Autodesk 3ds Max 2015. Wishes and reference materials were received from the client. The modeled 3D city consisted of different types of buildings that include the client's products and services. The implementation of the 3D city was based on modeling, animating and optimization. Thanks to the multimedia application, products and services could be presented virtually in an impressive way.</p> <p>As a result, an interactive 3D city as a part of an interactive multimedia application was created. The first version of the application was installed in the showroom in January 2015. User experiences during the first two months have been very positive. Also the 3D city that works as a user interface has been customer friendly and visually attractive. In the future the main aim is to export the application into the company's showrooms and events in other countries.</p>	
Keywords	WebGL, 3D, multimedia application, modeling, animating, optimization

# Sisällys

## Lyhenteet

1	Johdanto	1
2	WebGL-ohjelmointirajapinta	2
2.1	WebGL:n perusteet	2
2.2	WebGL:n mahdollisuudet ja haasteet	5
2.3	WebGL:n tukemat selaimet	7
2.4	WebGL-apukirjastot	9
	Three.js	9
	X3DOM	11
2.5	WebGL:n tulevaisuudennäkymät	12
3	Multimediasovelluksen suunnittelu WebGL-tekniikalla	14
3.1	Toimeksiantaja	14
3.2	Virtual Wall -konsepti	14
4	3D-kaupungin toteutus	21
4.1	Kaupungin mallintaminen	22
4.2	Mallin optimointi	29
4.3	Hissituotteiden animointi	32
5	Multimediasovelluksen lopputulos ja tulevaisuus	36
6	Yhteenveto	38
	Lähteet	40

## Lyhenteet

3D-grafiikka	Kolmiulotteinen grafiikka, joka on sisäisesti mallinnettu kolmen ulottuvuuden suhteen.
WebGL	Web Graphics Library. Ohjelmointirajapinta, jonka avulla saadaan näytettyä 3D-grafiikkaa verkkoselaimessa ilman ylimääräisiä liitännäisiä.
OpenGL	Open Graphics Library. Rojaltivapaa ja alustariippumaton 2D- ja 3D-grafiikkaan erikoistunut ohjelmointirajapinta.
HTML5	Hypertext Markup Language. HTML-merkintäkielen uusi versio.
GLSL	OpenGL Shading Language. Väriarjostuskieli, jonka avulla grafiikkalaitteiden laskennat eli renderöinti suoritetaan.
X3DOM	WebGL-apukirjasto, jonka avulla saadaan interaktiivista 3D:tä verkkoselaimeen.
X3D	Extensible 3D Graphics. XML-pohjainen tiedostomuoto 3D-grafiikan näyttämiseen tietokoneessa.

## 1 Johdanto

Elämme kolmiulotteisessa maailmassa, jossa näkemämme digitaalinen sisältö sisältää nykyään entistä enemmän 3D-grafiikkaa. Interaktiivista 3D-grafiikkaa käytetään puolestaan paljon erilaisissa peleissä ja sovelluksissa. Aikaisemmin 3D-tekniikat ovat olleet hyvin laiteriippuvaisia. Vuonna 2011 ilmestynyt WebGL-tekniikka on kuitenkin poistanut tämän ongelman, ja sen avulla saadaan luotua interaktiivista 3D-grafiikkaa verkkoselaimiin ilman ylimääräisiä liitännäisiä. WebGL on alustariippumaton, ja siksi sen käyttö onnistuu eri päätelaitteilta. Katselamiseen tarvitaan ainoastaan WebGL:ää tukeva selain, joita ovat nykyään kaikki suosituimmat pöytätietokoneissa olevat selaimet. Interaktiivisuudella tarkoitetaan käyttäjän mahdollisuutta hallita näytöllä esiintyvää grafiikkaa esimerkiksi hiiren tai näppäimistön avulla. WebGL on varsin uusi tekniikka 3D-maailmassa, joten se esittellään tässä opinnäytetyössä.

Insinööriyön tavoitteena on suunnitella ja toteuttaa 3D-kaupunki osana interaktiivista multimediasovellusta hyödyntäen WebGL-tekniikkaa. Kehitettävä multimediasovellus kulkee nimellä Virtual Wall -konsepti, jossa videoseinän käyttöliittymä tulee olemaan erillisessä kosketusnäyttöpaneelissa. Kosketusnäytön avulla käyttäjä hallitsee multimediasovellusta, ja samalla se toimii interaktiivisesti videoseinän kanssa. Raportin alkupuolella perehdytään WebGL-tekniikan perusteisiin ja toimintatapoihin sekä käydään läpi 3D-grafiikan muodostuminen verkkoselaimelle. WebGL:ää voidaan pitää matalan tason ohjelmointirajapinta, joka vaatii paljon koodia ja osaamista. WebGL-sovelluksien toteuttamisen helpottamiseksi ja aloittelijan käyttökynnyksen pienentämiseksi WebGL:lle on kehitetty useita erilaisia JavaScript-apukirjastoja, jotka sisältävät erilaisia toimintoja ja ominaisuuksia. Työssä esitellään tarkemmin kaksi apukirjastoa, Three.js ja X3DOM.

Insinööriyön toimeksiantaja on suomalainen hissiyhtiö KONE. Työn tarkoituksena on tuoda toimeksiantajan näyttelytilaan enemmän interaktiivisuutta, tuotteiden ja palveluiden esittelyä. Tämä toteutetaan WebGL-multimediasovelluksena, johon mallinnetaan 3D-kaupunki. 3D-kaupunki toimii sovelluksen käyttöliittymän kotivalikkona. Pääajatuk-sena on tuoda kuvista ja automaattisesti toistuvista animaatioista koostuvaan esittelytilaan lisää ilmettä ja toiminnallisuutta.

## 2 WebGL-ohjelmointirajapinta

### 2.1 WebGL:n perusteet

Ihmiset liikkuvat, näkevät ja kokevat kaiken kolmiulotteisena. Digitaalinen sisältö, jota näemme päivittäin esimerkiksi televisiossa ja internetissä, sisältää jatkuvasti enemmän kolmiulotteista sisältöä. Interaktiivista 3D-grafiikkaa käytetään puolestaan paljon erilaisissa peleissä ja sovelluksissa. Tällöin sovelluksessa käyttäjä on vuorovaikutuksessa näytölle piirrettävän kuvan kanssa.

WebGL eli Web Graphics Library on ohjelmointirajapinta, jolla saadaan luotua 3D-grafiikkaa verkkoselaimeen. Samalla WebGL pystyy suorittamaan graafisesti raskaita sovelluksia ilman mitään ylimääräisiä liitännäisiä. WebGL on Khronos Groupin julkaisemaan OpenGL ES 2.0 -teknologiaan perustuva JavaScriptin kautta käytettävä ratkaisu 3D-grafiikan näyttämiseen suoraan selaimessa hyödyntäen HTML5:n canvas-elementtiä. [1; 2.]

WebGL:n edeltäjänä toiminut OpenGL ES on rojaltivapaa ja alustariippumaton 2D- ja 3D-grafiikkaan erikoistunut ohjelmointirajapinta. Alustariippumattomalla tarkoitetaan, että sen ominaisuudet eivät ole riippuvaisia tietyistä laitteista tai käyttöjärjestelmistä. Tavalliseen OpenGL-rajapintaan verrattuna WebGL on karsitumpi ja sen ominaisuutena on varmistaa suorituskyky ja joustavuus. OpenGL ES on tarkoitettu pelikonsoleille, kodinkoneille, älypuhelimille ja ajoneuvoille eli erilaisille sulautetuille järjestelmille. Selaimissa yleisemmin kuitenkin esiintyy käsite WebGL. WebGL on ominaisuuksiltaan hyvin samanlainen OpenGL ES 2.0 -rajapinnan kanssa. Tämä tarkoittaa sitä, että WebGL-sovellusta on mahdollista näyttää monissa eri päätelaitteissa, kuten pöytätietokoneissa. WebGL:n ja OpenGL ES 2.0:n ero onkin juuri siinä, että OpenGL ES 2.0 ei toimi pöytätietokoneiden selaimissa, toisin kuin WebGL. [2; 3.]

OpenGL sai siis uuden elämän verkkoselaimissa WebGL:n muodossa, joka julkaistiin vuonna 2011. Lopputuloksena oli 3D-grafiikkaa selaimessa piirtävä elementti, joka ei tarvitse näyttämiseen erikseen asennettavia lisäosia. WebGL:n avulla voidaan selaimiin lisätä esimerkiksi interaktiivisia musiikkivideoita, pelejä, taidetta, 3D-ympäristöjä tai 3D-mallinnetuja objekteja tai luoda fyysisiä simulaatioita. [1.]

WebGL toimii HTML5:n canvas-elementissä, ja se määrittää paikan, johon 3D-grafiikka piirretään. Canvas on HTML5:n uusi ominaisuus, joka mahdollistaa 3D-grafiikan piirtämisen suoraan selaimen piirtokäskyjen avulla. Lisäksi se käyttää Document Object Model -ohjelmointirajapinnan ominaisuuksia, kuten esimerkiksi JavaScriptia. Document Object Model -ohjelmointirajapinnan ansiosta sivuihin saadaan lisättyä toiminnallisuutta ja interaktiivisuutta. Vaikka WebGL ei ole virallisesti HTML5:n ominaisuus, siitä huolimatta se toimii kaikissa HTML5-kieltä tukevissa selaimissa. [4; 5, s. 51.]

WebGL-sovelluksien kehittämiseen tarvitaan ainoastaan WebGL-ohjelmointirajapintaa tukeva selain ja tekstieditori, eli WebGL-ohjelmointiin riittävät samat välineet kuin normaaliin Web-ohjelmointiin. Lisäksi sovelluksen kehittämisessä voidaan käyttää hyväksi omaa palvelinta ja 3D-mallinnusohjelmia omien mallien ja elementtien luomiseen.

Verkkoselaimet käyttävät toimiakseen laitteiden omaa grafiikkalaitteistoa. Koska WebGL on alustariippumaton, sovellusta voidaan katsoa reaaliaikaisesti tietokoneilla, tableteilla ja älypuhelimilla. Ainoina kriteereinä ovat tarvittava grafiikkalaitteisto ja WebGL:ää tukeva verkkoselain. Tunnetuimmat verkkoselaimet tukevat WebGL:ää suoraan, joten erillisiä liitännäisiä ei tarvita. Koska mobiililaitteet käyttävät selaimia, toimii WebGL myös älypuhelimissa. Koska WebGL on alustariippumaton, voi selainta vaihtaa, jos selain ei tue WebGL:ää. Se on huomattavasti helpompaa kuin käyttöjärjestelmän uudelleen asentaminen tai halvempaa kuin uuden tietokoneen ostaminen. [6; 7.]

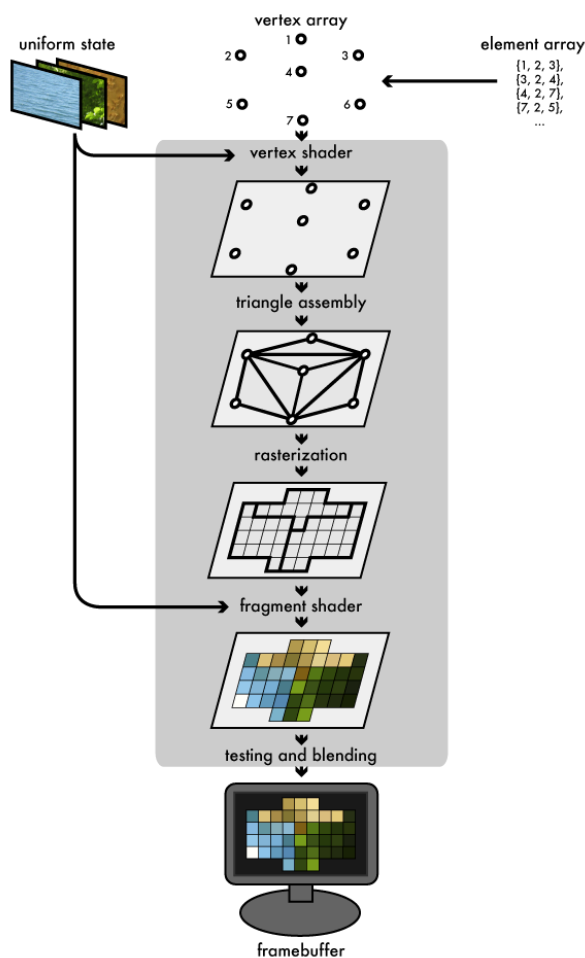
WebGL:ää voidaan pitää hyvin matalan tason ohjelmointirajapintana eli vaativana kieleenä käyttää. Suoraan WebGL:n päälle kehitettävien sovellusten rakentaminen on hyvin työlästä ja vaatii paljon osaamista. WebGL vaatii myös paljon koodia. Yksinkertaisenkin toiminto voi tarvita jopa satoja rivejä koodia toimiakseen. Tämän ongelman helpottamiseksi ja aloittelijan käyttökynnyksen pienentämiseksi WebGL:lle on kehitetty useita erilaisia JavaScript-apukirjastoja, jotka sisältävät eri toimintoja. Kirjastot on luotu helpottamaan käyttäjää samalla tavalla kuin JQueryn kaltaiset kirjastot internetsivustojen tekoon. Kirjastojen tehtävänä on auttaa yksinkertaisten ja yleisempien toimintojen toteuttamisessa. [1; 3; 6.]

WebGL käyttää GLSL- eli OpenGL Shading Language -värivarjostusta, jonka avulla grafiikkalaitteiden laskennat eli renderöinti suoritetaan. Värivarjostuskieli perustuu C-kieleen, minkä avulla kehittäjiä ei tarvitse ohjelmoida grafiikkaa laitekohtaisesti. Tässä tapauksessa OpenGL-rajapintaa tukeva näytönohjain tekee kaiken tarvittavan työn.



GLSL suorittaa laskemisen suoraan näytönohjaimesta, minkä takia näytönohjain pystyy laskemaan toiminnot huomattavasti nopeammin kuin JavaScript. Tämän ansiosta sivusta voidaan tehdä huomattavasti nopeampi. WebGL-sovelluksessa kaikki grafiikan piirtäminen tehdään pääosin tämän värivarjostuksen kautta. Ensiksi lasketaan pikselin paikka ruudulla vertex-varjostimella ja fragment-varjostin laskee pikselin värin. Varjostimien tehtävänä on laskea, miten WebGL-sovellukseen piirrettävien objektien muodot näytetään riippuen niille annetuista pinnoista, materiaaleista, valaistuksista ja katselukulmista. [1; 9, s. 51.]

3D-grafiikka, jota WebGL piirtää, muodostuu kolmioista. JavaScript luo vertekstitaulukot, jotka sisältävät tietoja jokaisesta verteksistä, kuten niiden sijainnin 3D-maailmassa, tekstuurin, värin ja valaistuksen vaikutuksen. Tällaiset tiedot WebGL saa erikseen ladata 3D-tiedostosta, tai ne voidaan luoda aivan alusta. [6.] Tarkempi grafiikan piirtämisprosessi esitetään kuvassa 1.



Kuva 1. 3D-grafiikan piirtämisprosessi [9].

Kuvan shadereilla tarkoitetaan komentosarjaa tai ohjelman pätkää, jolla luodaan kappaleelle pinnoite. Shader voi muodostaa esimerkiksi peilin kaltaisen heijastuksen tai lasin kaltaisen läpinäkyvyyden. Shaderiä voidaan siis pitää kappaleen pinnoitteena, mikä määrittää valon vaikutuksen kappaleeseen. Nämä kaikki määritellään shaderille syötettävillä arvoilla, jotka sitten grafiikkapiirillä tulkitaan näkyväksi kuvaksi. [7.]

JavaScriptin luotua verteksitaulukot tieto lähetetään grafiikkaprosessorille, joka syöttää tiedot verteksipuskuriin. Grafiikkaprosessori alkaa työstää tietoa ja ajaa sitä verteksivarjostimien läpi. Verteksivarjostin laskee verteksille niiden uuden sijainnin, värit ja tekstuurit näytölle. [6.]

Tämän prosessin jälkeen grafiikkaprosessori luo vertekseistä kolmioita, jotka lähetetään eteenpäin rasteroijaan. Rasteroijan tehtävänä on tehdä kolmioista pikselin kokoisia fragmentteja, jotka muodostavat yhtenäisen pinnan. Rasteroijan jälkeen fragmentit lähetetään fragmenttivarjostimen läpi, joka luo jokaiselle pikselille syvyys- ja väriarvot. Viimeisenä vaiheena pikselit tallennetaan kehyspuskuriin, joka piirtää eli renderöi grafiikan näytölle. Renderöinniksi kutsutaan prosessia, joka tuottaa kolmiulotteisesta kappaleesta kaksiulotteisen kuvan. [6.]

WebGL-sovelluksen rakenteeseen kuuluu vähintään kahdeksan erilaista osaa. Kaikki lähtee liikkeelle canvas-elementin luomisesta. Tämän jälkeen määritellään canvas-elementin ympäristö WebGL-rajapinnaksi, määritetään viewport WebGL-sovellukseen, luodaan elementeille vähintään yksi puskuri, määritellään vähintään yksi matriisi elementtien pyörittämistä ja sijoittamista varten, luodaan vähintään yksi värivarjostin piirtämisalgoritmia varten, yhdistetään elementtien tieto varjostimien kanssa ja viimeisenä suoritetaan renderöinti eli grafiikan piirtäminen. Grafiikan latautuminen näytölle saattaa hieman kestää riippuen siitä, kuinka raskas sovellus on. [8, s. 10.]

## 2.2 WebGL:n mahdollisuudet ja haasteet

WebGL on varsin uusi tekniikka, jolla on kilpailijoita markkinoilla. Niitä ovat esimerkiksi Flash, Silverlight ja Microsoftin DirectX. Uusi tekniikka tuo mukanaan monia mahdollisuuksia, mutta myös haasteita.

WebGL:n kutsuminen vaatii huomattavasti kilpailijoita vähemmän. WebGL-sovelluksen kutsumiseen tarvitaan ainoastaan kolme riviä koodia, jotka ovat canvas-elementin luonti, sen lisääminen dokumenttiin sekä halutun asiayhteyden määrittelemisen WebGL-rajapinnaksi. Näin vältetään erilaisten tiedostojen luomiselta, kirjastojen lisäämiseltä tai apuvälineiden erikseen kutsulta. [10.]

WebGL:n etu on rajapinnan tarjoaminen laitteen näytönohjaimen käyttöön. Erillisen näytönohjaimen suorituskyky 3D-grafiikan luomisessa on huomattavasti parempaa kuin pääsuorittimen eli prosessorin (CPU). WebGL käyttää 3D-grafiikan reaaliaikaiseen renderöintiin näytönohjaimen tehoa. Näytönohjaimen käyttö tuo etuja, mutta myös samalla rajoitteita. Etuna voidaan pitää suurta laskentatehoa, jonka ansiosta verkkoselaimissa toimivat sovellukset voidaan tehdä entistä toiminnallisemmiksi ja monipuolisemmiksi. Tämä voi myös asettaa sovellukselle rajoitteita silloin, kun tietokoneen näytönohjaimen tehokkuus ei ole riittävä. Tällöin laite ei pysty näyttämään sovellusta riittävän tehokkaasti tai pahimmassa tapauksessa sovellusta ei pystytä esittämään ollenkaan. Ongelmasta ollaan kuitenkin pääsemässä eroon. Laitekehittäjät, kuten Intel ja Nvidia, ovat huomioineet asian laitekehityksessään, ja suurimmassa osassa uusista laitteista on nykyään tarpeeksi tehokkaat grafiikkalaitteet ja näytönohjaimet WebGL-sovellusten näyttämiseen. [11.]

Näytönohjaimien tuki ja kehittyminen on tärkeää WebGL:n kehitymisessä ja laajentumisessa. WebGL onkin nykyään yksi testatuimmista rajapinnoista, mikä auttaa myös tuen saamiseen. WebGL-tekniikan testaaminen on käyttäjälle hyvin yksinkertaista. Testatakseen käyttäjän tarvitsee mennä selaimella haluttuun testiosoitteeseen ja klikata testin aloittamista. Testaaja välttyy erilaisten ajurien ja lisäosien asentamiselta.

WebGL on laajennettavissa samalla tavoin kuin OpenGL eli ilman suuria versiopäivityksiä. Näin esimerkiksi kehittäjän ei tarvitse ladata versiopäivityksiä, vaan se voi ottaa uusia ominaisuuksia käyttöön välittömästi niiden ilmestyessä. WebGL onkin tällä hetkellä ainoa 3D-teknologia selainympäristössä, mikä mahdollistaa uusien ominaisuuksien välittömän käyttöönoton. [10.]

WebGL:n suuri etu on apukirjastojen olemassa olo. Sovelluskehittäjien ei tarvitse apukirjastojen ansiosta koskea välttämättä lainkaan WebGL-rajapinnan koodiin. Helppous ja suoraviivaisuus on yksi suurimmista eduista, joita apukirjastot tarjoavat kehittäjille ja käyttäjille.

Haasteena WebGL-sovellusta suorittaessa on canvas-elementti, joka piirtää näytölle jatkuvasti kuvaa, vaikka mikään ei liikkuisi. Tällöin näytönohjain työstää sivua jatkuvasti. WebGL-sovellusta näyttävän sivuston ollessa auki muut käytössä olevat ohjelmat saattavat toimia hitaammin kuin tavallisesti. Tämän vuoksi on suositeltavaa sulkea sovellusta pyörittävä sivu aina, kun sovellusta ei käytetä. [1.]

Turvallisuus on noussut yhdeksi kysymykseksi sovelluskehittäjien keskuudessa. Yhtenä huolenaiheena on ollut hakkereiden pääsy käsiksi näytönohjaimiin, jotka mahdollistavat suuren laskentatehon. Näytönohjaimia ei ole suunniteltu kestämaan hyökkäyksiä. Hyökkäyksen kohteeksi joutunut näytönohjain voidaan ylikuormittaa tietynlaisilla käsilyillä, jolloin näytönohjain saadaan jumiin ja mahdollisesti jopa käyttäjärjestelmä kaatumaan. Esimerkiksi Microsoft on arvotellut voimakkaasti WebGL:n turvallisuuspuitteita, minkä vuoksi se oli pitkään tukematta WebGL-tekniikkaa. [12.]

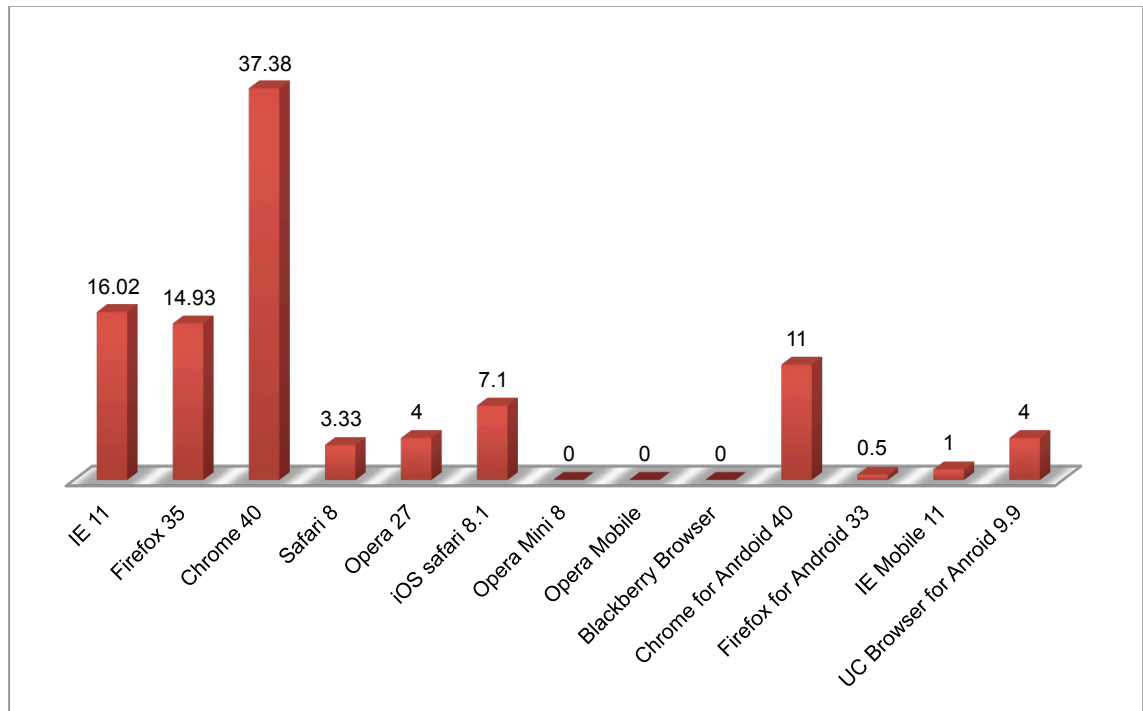
Turvallisuusongelmia on ilmennyt verkkoselaimissa, joissa WebGL on tuettu. WebGL-sovelluksien näyttämiseksi selaimista on täytynyt avata useita turvallisuusmääräyksiä, jotka tavallisesti on hyvä pitää suljettuina. Eri selainten valmistajat ovat kuitenkin pyrkineet jatkuvasti korjaamaan turvallisuuspuitteita ja tekemään tietoturvapäivityksiä mahdollisten hyökkäysten estämiseksi. [13; 14.]

### 2.3 WebGL:n tukemat selaimet

WebGL-rajapinnan tulevaisuus verkkoselainten 3D:n tuottamisen standardina näyttää hyvältä, mutta haasteita on vielä jäljellä. Niistä mahdollisesti suurin on saada kaikkien verkkoselaimien tuki WebGL-rajapinnalle. WebGL tarjoaa kuitenkin kattavan valikoidun toimintoja ja mahdollisuuksia, joiden avulla se on vahvoilla.

Internet Explorer oli pitkään ainoa pöytäkoneselain, joka ei tukenut WebGL-tekniikkaa. Tällä hetkellä WebGL:lle on kuitenkin tuki kaikissa suosituimmista selaimissa, esimerkiksi Google Chromessa, Mozilla Firefoxissa, Microsoft Internet Explorerissa, Apple Safarissa ja Operassa.

Kuvasta 2 nähdään, kuinka selainten suosio on jakautunut WebGL-sovelluksia käytettäessä. Kuvassa esitetään myös selainversio, jolla sovelluksia yleisimmin käytetään. Google Chromen versio 40 on selvästi yleisin ja luotettavin selain WebGL-sovelluksia käytettäessä lähes 40 prosentin suosiollaan. [15.]



Kuva 2. Selainten suosio prosentteina WebGL-sovellusta käytettäessä [15].

Syynä Microsoftin tukemattomuuteen on ollut OpenGL-rajapinnan kilpailu Microsoftin DirectX-rajapinnan kanssa. Microsoftin kehittämä Windows-käyttöjärjestelmille suunnattu DirectX tarjoaa suoran yhteyden näytönohjaimiin, äänilaitteisiin ja ohjainlaitteisiin. Se on yleistynyt Microsoftin käyttöjärjestelmissä ja Xbox-pelikonsoleissa. Microsoftin mukaan WebGL on ollut myös turvallisuusaukko. Laitteistotason rajapintojen avaaminen suoraan selaimelle ja sitä kautta verkkoselaimelle altistaa käyttäjät ja selaimet isolle määrälle ongelmia. Haitallinen sivusto voi syöttää näytönohjaimeen koodia, joka kaa-taa laiteohjaimen tai jopa koko koneen. Joko Microsoftin tietoturvatutkijat ovat muutta-neet mielensä tai tekniikka nähdään riittävän hyödyllisenä käytettäväksi riskeistä huolimatta, sillä Internet Explorerin versio 11 on saanut tuen WebGL-rajapinnalle. [12; 16; 17.]

## 2.4 WebGL-apukirjastot

WebGL-rajapinnan ohjelmointi voi olla hidasta ja hankalaa, jos kaiken joutuu tekemään alusta alkaen itse. Tämän vuoksi on hyvä harkita apukirjastojen käyttöä, jotka on luotu helpottamaan ja nopeuttamaan WebGL-sovellusten rakentamista. Apukirjastoissa esimerkiksi JavaScript-tiedostot sisältävät valmiita olioita ja funktioita perustoimintojen luomiseen. [18.]

Yksinkertaisen laatikon luominen ilman apukirjastoa voi vaatia yli sata riviä JavaScript- ja värivarjostinkoodia, mutta apukirjastojen avulla siihen tarvitaan vain muutama rivi [19].

WebGL:lle on olemassa paljon eri ohjelmistokehyksiä ja apukirjastoja helpottamaan WebGL-sovelluksen toteuttamista. Jokaisella apukirjastolla on hieman erilaiset toteuttamis- ja lähestymistapansa aiheeseen, mutta kaikki pohjautuvat lopulta WebGL-ohjelmointirajapintaan. Yhteistä kaikille apukirjastoille ovat JavaScript ja WebGL:n tarjoamat toiminnallisuudet, joihin apukirjastot perustuvat. [20.]

On hyvä valita omaan tarkoitukseen sopiva apukirjasto. Monet apukirjastot on suunnattu pelejä tai multimediaa varten. Apukirjastojen sisällön laajuus voi vaihdella hyvin paljon. Jotkut sisältävät pelkän pohjan 3D-ympäristölle, kun taas jotkut jopa puolivalmiin pelin tai fysiikkamoottorin.

Lisäksi on yleiseen sovelluskehitykseen suunnattuja apukirjastoja, kuten Three.js, joka esitellään tarkemmin seuraavassa.

### Three.js

Three.js on yksi WebGL:n apukirjastoista, ja se tarjoaa yksinkertaistettuja tapoja rakentaa ja hallita objekteja, materiaaleja, valaistusta, kameroita ja animaatioita. Three.js on samalla yksi suosituimmista apukirjastoista, joka perustuu JavaScriptiin. Sen renderöijinä toimivat Canvas ja SVG. Ne mahdollistavat 3D-mallien tuonnin verkkoselaimeen, ja erilaisten JavaScript-käskyjen avulla saadaan sovellukseen interaktiivisuutta ja toiminnallisuutta. Three.js:n suuri vahvuus on sen helppokäyttöisyys ja tehokkuus. [20.]

Three.js on avoimen lähdekoodin kirjasto, jonka kehittämiseen on osallistunut yli 100 henkilöä. Sen dokumentaatio on yhteisön vastuulla, minkä vuoksi dokumentaatio on osittain puutteellista ja laatu hyvin vaihtelevaa. Onneksi lähes kaikista ominaisuuksista on kuitenkin hyviä esimerkkejä opetteluun ja oman sovelluksen hyödyntämiseen. [21.]

Three.js on keskittynyt pääasiassa grafiikan luomiseen 3D-objekteista varjostimiin ja tehosteisiin. Siihen on kuulunut lisäosana myös näppäimistön käsittelyä ja törmäyksen tunnistusta. Three.js-sovellus koostuu yksinkertaisimmillaan kolmesta eri osasta, yleisten muuttujien määrittelystä, asetusten määrittelyfunktioista ja animaatiofunktioista. Sovellus tarvitsee myös vähintään kolme elementtiä toimiakseen. Nämä ovat scene, camera ja renderer. Mainitut elementit ovat kaikki Three.js-apukirjaston olioita. Scene-olio on paikka, jonne kaikki sovelluksessa esiintyvät 3D-mallit lisätään. Camera-olio on kamera, jonka avulla nähdään scene-oliolle luodut 3D-mallit. Kameralle kerrotaan esimerkiksi kameralle näkyvä leveys asteina, kuvasuhde, etäisyys lähimmästä ja kaukaisimmasta näkyvästä 3D-kappaleesta. Lopuksi render-olio piirtää ruudulle näkyviin kaikki scene-oliolla olevat 3D-mallit ja objektit. Render luo määritellyn kokoisen canvas-elementin HTML-dokumenttiin. [22; 23.]

Kuvassa 3 esitetään yhdysvaltalaisen urheiluvaatevalmistajan Under Armourin Three.js-apukirjaston avulla tehty WebGL-visualisointi ”I Will What I Want” -interaktiivinen sovellus (<http://gisele.underarmour.com>).



Kuva 3. Yhdysvaltalaisen urheiluvaatevalmistajan Under Armourin Three.js:llä tekemä WebGL-sovellus verkkosivuillaan [24].

Sovelluksessa käyttäjät pystyvät lähettämään kommentteja ja hallitsemaan sivulla näytettävää videota. Kommentit ilmestyvät interaktiivisesti videossa näkyville seinille.

## X3DOM

X3DOM on Three.js:n tavoin yksi avoimen lähdekoodin WebGL-apukirjastoista, ja sen avulla interaktiivista 3D-sisältöä saadaan julkaistua suoraan verkkoselaimessa. X3D on standardoitu XML-formaatti 3D-materiaaleille, ja X3DOM on JavaScript-kirjasto, jonka avulla X3D-tiedostoja voidaan käntää WebGL:n ymmärtämään muotoon. [25.]

3D-sovelluksen reaaliaikainen renderöinti tehdään WebGL:n avulla, joka sulautetaan suoraan X3DOM-ratkaisuun. Tämän ansiosta verkkoselaimeen ei tarvitse asentaa erillisiä liitännäisiä. Sen tarkoituksena on mahdollistaa 3D-elementtien käyttöä suoraan DOM:n puurakenteessa. [25.]

Koska 3D-objektit ovat osa tavallista DOM-rakennetta, niihin pääsee käsiksi kuten HTML-sivun elementteihin. X3DOM-apukirjaston avulla tehty sovellus koostuu tavallisista HTML5-elementeistä, joiden sisässä ovat head- ja body-elementit. HTML5-elementtien lisäksi tulee X3D-elementtejä eli X3D-noodeja. Jotta X3DOM pystyisi toimimaan, se tarvitsee oman JavaScript- ja CSS-tiedoston, jotka voidaan ladata X3DOM:n verkkosivulta. [26.]

X3D-elementit sijoitetaan body-elementin sisään, mikä aloitetaan X3D-noodin luomisella. Tämä määrittää verkkosivulle alueen, joka sisältää kaikki tiedot 3D-sisältöä varten. Tästä nähdään esimerkki kuvassa 4. Lisäämällä X3D-noodiin attribuutteja voidaan määrittää sisällön alueen leveys ja korkeus. [27.]



```

***x3dom.html
<!DOCTYPE html> <html>
<head>
  <link rel="stylesheet" type="text/css" href="http://www.x3dom.org/download/x3dom.css" />
  <script type="text/javascript" src="http://www.x3dom.org/download/x3dom.js"></script> </head>
<body>
  <x3d width="700px" height="600px">
    <scene>
      <viewpoint position="0 0 10"></viewpoint>
      <shape>
        <appearance>
          <material diffuseColor="red"></material>
        </appearance>
        <box></box>
      </shape>
    </scene>
  </x3d>
</body>
</html>

```

Kuva 4. Esimerkkikoodi yksinkertaisen laatikon luomisesta X3DOM-apukirjastolla.

Scene-noodilla ilmoitetaan tulevasta 3D-sisällöstä ja viewpoint-noodilla katselukulma. Viewpoint-noodi toimii kamerana, ja se voidaan asettaa haluttuun kohtaan position-attribuutin avulla. Tämän jälkeen luodaan 3D-objektin geometria shape-noodilla ja materiaali appearance-noodilla. Shape-noodin sisään tulee kaikki tieto esitettävästä 3D-objektista. Yksinkertainen objekti voidaan luoda kirjoittamalla halutun muodon nimi ja sulkemalla sen tagi. Kuvassa 4 esimerkkinä on käytetty laatikkoa. Yleensä WebGL-sovellukset sisältävät kuitenkin laatikkoa monimutkaisempia malleja. Tämän vuoksi X3DOM tarjoaa myös 3D-mallinnusohjelmille, kuten esimerkiksi Autodesk 3ds Max -ohjelmalle, InstantExport-liitännäistä, jonka avulla voidaan tuoda 3D-malleja ulos oikeassa muodossa X3DOMia varten. [28; 29.]

## 2.5 WebGL:n tulevaisuudennäkymät

WebGL on ollut olemassa vuodesta 2011 lähtien, ja sen tavoitteena oli tulla standardiksi 3D-grafiikan tuottamisessa verkkoselaimeen. Microsoft ei kuitenkaan lämmennyt tekniikalle ja ilmoitti jo alkuvaiheessa jättävänsä tuen pois Internet Explorer -selaimestaan. Tämä oli paha takaisku WebGL:n laajentumiselle, koska Internet Explorer on vielä suosituin isojen yritysten käyttämä verkkoselain. [30.]

WebGL-tekniikka ei ole täydellinen, koska sillä on vielä ongelmia ratkottavana. WebGL tarjoaa kuitenkin hyvän mahdollisuuden luoda interaktiivista sisältöä, pelejä, esityksiä, sovelluksia sekä työkaluja, jotka voidaan näyttää suoraan selaimessa. Internetin kehi-

tyksen myötä WebGL tarjoaa HTML5:n avulla hyvän tavan tuoda verkkosivuille entistä näyttävämpää ja hienompaa grafiikkaa, jota aikaisemmin ei ole nähty.

WebGL:n onneksi Microsoft muutti mielensä ja aloitti tekniikan tukemisen Internet Explorer 11 -selaimessaan. Tämä voi olla tulevaisuudessa huomattava edistysaskel WebGL:n todelliselle leviämiselle ja yrityksiä mielenkiinnon heräämiselle tekniikkaa kohtaan. WebGL:llä on kaikki mahdollisuudet päästä tavoitteeseensa tulella HTML5:n standardiksi interaktiivisen 3D-grafiikan tuottamiselle verkkoselaimessa. [30.]

Aika näyttää, tarvitaanko tulevaisuudessa enää pelikonsoleita vai voidaanko esimerkiksi konsoli- ja tietokonepelit pelata suoraan selainten kautta. Tätä vauhtia arkinen internetin selaaminen näyttää tulevaisuudessa hyvin erilaiselta, kuin mihin olemme tottuneet.

### 3 Multimediasovelluksen suunnittelu WebGL-tekniikalla

#### 3.1 Toimeksiantaja

Opinnäytetyön toimeksiantaja on suomalainen hissiyhtiö KONE. Vuonna 1910 perustettu KONE kuuluu maailman johtaviin hissi- ja liukuporrasvalmistajiin. Yritys tarjoaa asiakkailleen edistyksellisiä hissejä, liukuportaita, automaattiovia ja ratkaisuja parantamaan käyttäjäkokemusta rakennuksissa liikkumiseen. KONEen tarjonta ei rajoitu pelkästään edellä mainittuihin asioihin, vaan se tarjoaa myös huolto-, korjaus- ja modernisointiratkaisuja ympäri maailman. KONEen tavoitteena on ihmisten liikkuminen rakennuksissa sujuvasti, turvallisesti, mukavasti ja viivytyksettä. Maailmanlaajuisesti KONEella on yli 400 000 asiakasta ja 1 000 toimipistettä. KONEen uusin innovaatio on vuonna 2013 julkaistu KONE UltraRope, hiilikuidusta valmistettu hissien nostoköysi, joka avaa uudenlaisia mahdollisuuksia tornitalojen ja pilvenpiirtäjien rakentamiseen. UltraRope-köyden avulla hissi voi matkata jopa kilometrin korkeuteen ja olla samalla luotettava ja energiatehokas. Perinteisesti hissiköyden oma paino on rajoittanut nostokorkeuden 500 metriin. Innovaation seurauksena KONE voitti tilauksen vuonna 2018 valmistuvaan maailman korkeimpaan rakennukseen Kingdom Toweriin. Saudi-Arabiaan valmistuvan pilvenpiirtäjän arvioidaan nousevan yli kilometrin korkeuteen. [31.]

#### 3.2 Virtual Wall -konsepti

Virtual Wall -konseptin nimellä kulkevan multimediasovelluksen suunnittelu lähti liikkeelle toimeksiantajan tarpeesta saada esittelytilaansa enemmän interaktiivisuutta, tuotteiden ja palveluiden esittelyä, jotain uutta, joka jäisi vierailijan mieleen. Olemassa olevan esittelytilan sisältö oli koostunut pääasiassa vain staattisista kuvista, teksteistä sekä itsenäisesti pyörivistä videoista ja animaatioista (kuva 5).



Kuva 5. Nykyisen esittelytilan seinä, jossa staattinen kuva ja vieressä automaattisesti toimiva animaatio. Tämän seinän paikalle rakennetaan uusi videoseinä.

KONEen esittelytila People Flow Center sijaitsee Hyvinkäällä, jossa käy paljon vierailijoita aina asiakkaista opastettuihin opiskelijaryhmiin. People Flow Centerissä on näyttelytilan lisäksi myös asiakas- ja kokoustiloja. Pääajatuksena oli tuoda esittelytilaan lisää ilmettä ja toiminnallisuutta. Vastaavia esittelytiloja hieman pienemmässä mittakaavassa on KONEen myyntiyhtiöissä ympäri maailmaa.

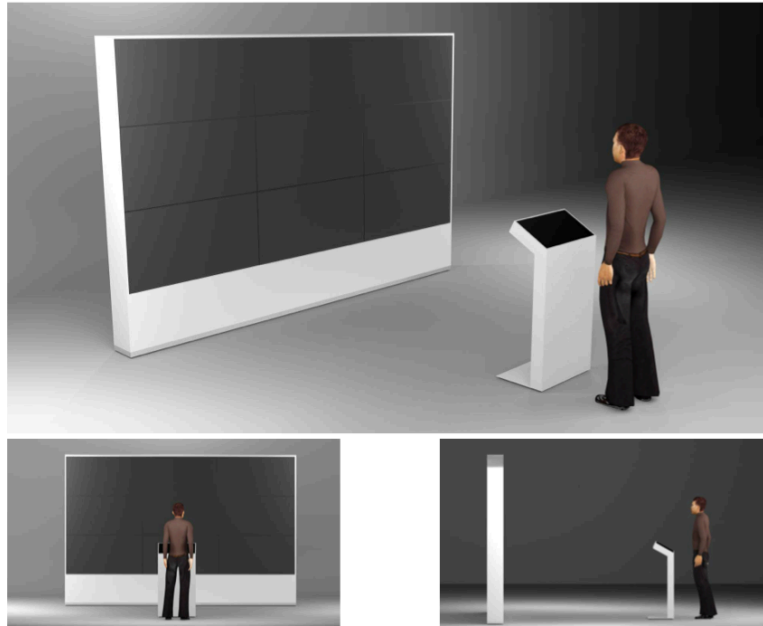
Multimediasovelluksen suunnittelussa käytettiin referenssinä Audi AG:n käyttämää Audi City -konseptia. Audin esittelytilassa autoja ei enää fyysisesti viedä esittelytilaan, vaan niitä esitellään virtuaalisesti huonetta ympäröiviltä videoseiniltä (kuva 6).



Kuva 6. Audi City -konseptin virtuaalinen esittelytila, joka toimi referenssinä [32].

Monet KONEen tuotteista, kuten hissit, liukoportaat ja automaattiovet, ovat erittäin isoja oikeassa mittakaavassa, joten niiden tuominen fyysisesti esittelytilaan on mahdotonta. Multimediasovelluksen avulla tuotteet saataisiin isolle videoseinälle virtuaalisesti. Myös monet tuotteet ovat hyvin teknisiä, joten niiden valokuvaaminen on vaikeaa tai jopa mahdotonta. Animaatioiden ja 3D-mallien avulla voidaan havainnollistaa mekaanisten laitteiden toimintaa helposti.

Kehitettävän virtuaaliseen käyttöliittymään tulisi olemaan erillisessä kosketusnäytössä, josta multimediasovellusta hallitaan. Kosketusnäyttö toimii interaktiivisena käyttöliittymänä ison videoseinän kanssa (kuva 7). Videoseinä haluttiin mahdollisimman isoksi, jotta yrityksen tuotteet saataisiin esiteltä virtuaalisesti mahdollisimman hyvin ja esimerkiksi hissien korien sisustuksia voitaisiin esitellä isolta videoseinältä näyttävästi ja lähes luonnollisessa koossa. Sovelluksen käyttöliittymän suunnittelussa painotettiin helppokäyttöisyyttä. Näyttelytilassa käy paljon opastamattomia vierailijoita ja opiskelijaryhmiä, joten sovelluksesta haluttiin niin helppokäyttöinen, että sitä osaisi käyttää kuka tahansa ilman erillistä opastusta. Lisäksi sen piti mahdollistaa myös monimuotoiset esitykset opastetuille asiakasryhmille.



Kuva 7. Suunnitelma, miltä iso videoseinä ja kosketusnäyttöpaneeli näyttäisi asennettuna [32].

Käyttöliittymän ei haluttu myöskään muistuttavan perinteistä internetkäyttöliittymää, vaan haluttiin mallinnetun 3D-kaupungin toimivan sovelluksen käyttöliittymän kotivalikona. 3D-kaupunki sopii kotivalikon näkymäksi, koska useimmat KONEen tuotteet ja palvelut liittyvät kiinteästi rakennuksiin ja niissä tapahtuvien liikennevirtojen hallintaan. Kaupungissa on helppo esittää erilaiset asiakas- ja rakennussegmentit sekä näyttää tuotteet luonnollisissa käyttöympäristöissään.

Kaupunkia ohjattaisiin kosketusnäytöltä vetämällä sormea vaakasuorassa, jolloin kaupungista saataisiin helposti näkymä eri puolilta ja näin rakennuksiin sijoitetut tuotteet sopivasti näkyviin. Kaupungin päälle rakennettaisiin navigointilinkit, joita koskettaessa sovellus siirtyisi kyseiseen osioon ja sen sisältöön. Sisällöistä palattaisiin kaupunkinäkymään ennen siirtymistä uuteen osioon. Sovelluksen käyttöliittymän tuli olla myös KONEen käyttämän visuaalisen konseptin mukainen; se kulkee nimellä BlueMan-konsepti (kuva 8). Tyypillistä konseptille on, että KONEen tuotteet ja palvelut visualisoidaan osaksi kuvitteellista rakennusympäristöä, jossa rakennukset kuvataan yksinkertaisina harmaansävyisinä ympäristöinä. Tuotteita, palveluita ja ihmisiä tuodaan esille ja korostetaan sinisellä värillä.

Tätä visuaalista konseptia käytetään muun muassa yrityksen printti- ja markkinointiviestinnässä. Valaistuksella ja varjoilla saadaan valkoisesta eri harmaansävyjä luo-

maan rakennuksiin syvyyttä ja pinnanvaihteluja. Visuaalinen konsepti sopii myös hyvin virtuaaliseen multimediasovelluksen toteutukseen, koska tiedetään, että lähtökohtaisesti kaupunkimallin polygonien määrä joudutaan minimoimaan riittävän suorituskyvyn saavuttamiseksi.

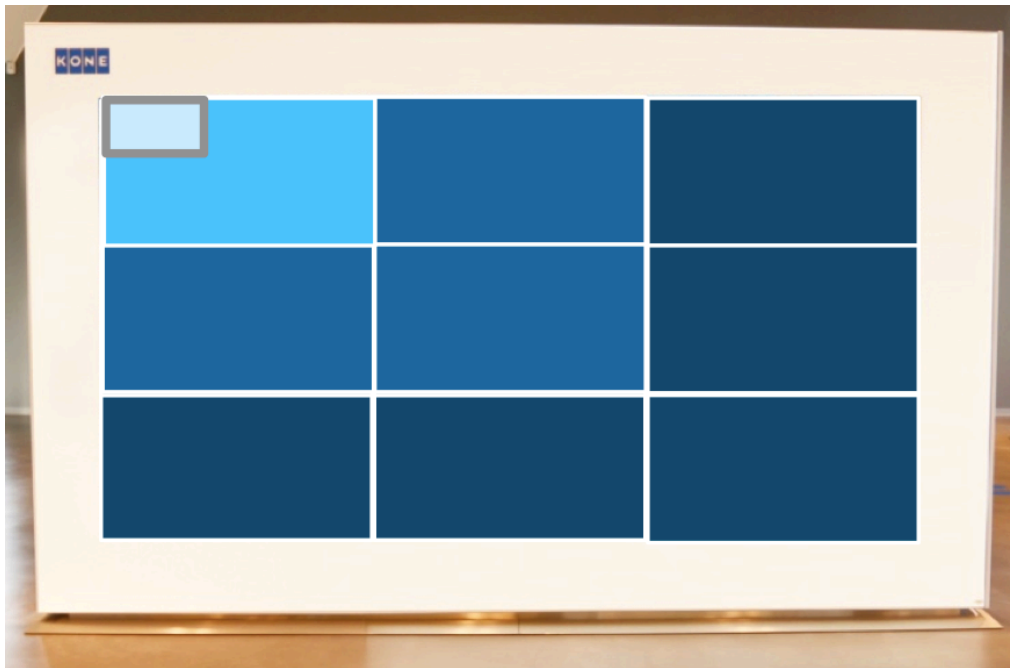


Kuva 8. Esimerkki KONEen BlueMan-konseptista. Kuvaan on lisätty myös suunnitellut navigointilinkit visualisoimaan, miltä sovelluksen käyttöliittymä näyttäisi [32].

Sovelluksen sisältö koostuu neljästä eri osiosta. Ensimmäisenä osiona on KONE Story, joka toimii KONE yhtiön yleisesittelynä. Tämä ilmestyy, kun käyttäjä ensimmäisen kerran käyttää sovellusta. Sovelluksen toinen osio esittelee yhtiön tuote- ja palvelutarjonnan. Osiossa käydään läpi KONEen hissejä, liukuportaita, ovia ja modernisointiratkaisuja. Sisällön kolmantena osiona ovat referenssit, jossa esitellään KONEen referenssirakennuksia eli rakennuksia, jotka sisältävät KONEen ratkaisuja ympäri maailmaa. Neljännessä osiossa on tarkoitus esitellä työkaluja ja suunnittelun apuvälineitä, joita KONE tarjoaa asiakkailleen. Sovelluksesta rakennetaan ensimmäisenä 3D-kaupunkia ja lähdetään keräämään KONEen materiaalipankeista mahdollisimman paljon materiaalia, jota sovelluksessa voitaisiin käyttää. Sovelluksen sisältö tulee olemaan hyvin visuaalinen ja tekstin määrä pyritään pitämään mahdollisimman vähissä. Sovellus tulee sisältämään 3D-kaupungin lisäksi, kuvia, videoita, lyhyitä tekstipätkiä sekä 2D- ja 3D-animaatioita. Sovelluksen sisällön rakentaminen on pitkän ajan suunnitelma. Sovelluksen ensimmäisen version valmistuttua sisältöä ja toimintoja aiotaan vielä kehittää.

tää tulevina vuosina lisää. Esimerkiksi KONEen referenssejä lisätään jatkossa sovellukseen.

Sovelluksen toteutusta suunniteltaessa ensimmäiseksi selvitettiin tekniikka, jolla olisi mahdollista ohjata 3D-malleja reaaliaikaisesti. Yhtenä mahdollisuutena harkittiin WebGL-tekniikkaa, jolla 3D-mallia saadaan ohjattua reaaliaikaisesti verkkoselaimessa. Toisena mahdollisuutena tutkittiin Unity3D-pelimoottorin käyttöä, joka pystyisi reaaliaikaisesti renderöimään 3D:tä. Työn kuluessa vaatimus sisällön esittämiseksi erilaisilla päätelaitteilla tuli vahvasti esille, joten tämän perusteella tehtiin lopullinen päätös WebGL-tekniikan käytöstä. Tämä mahdollistaisi sovelluksen esittämisen millä tahansa päätelaitteella, jonka kuvasuhde olisi 16:9, käyttäen verkkoselainta. Sovelluksen skaalautuvuus esitetään kuvassa 9.



Kuva 9. Multimediasovelluksen skaalautuvuus eri päätelaitteille.

Helppo skaalautuvuus mahdollistaisi sovelluksen viemisen helposti messuille, asiakastilaisuuksiin ja erikokoisiin näyttelytiloihin ympäri maailmaa. Tavoitteeksi asetettiin, että myyntimies voisi käyttää sovellusta tabletillaan ja se voitaisiin tarjoilla suoraan asiakkaille KONEen verkkosivujen kautta.

Viimeiseksi täytyi selvittää, millä WebGL:n apukirjastolla suunnitellun multimediasovelluksen toteuttaminen olisi mahdollista ja mikä sopisi parhaiten käyttötarkoitukseen.



Apukirjastoista sopivimmaksi osoittautui Three.js, jolla sovellusta lähdettiin toteuttamaan. Three.js:n valinta käytettäväksi apukirjastoksi perustui sen esimerkkisovellusten runsauteen ja visuaaliseen näyttävyys.

Sovelluksen toteuttamisessa käytettiin selvää työnjakoa. Yrityksen ohjelmoija hoitaisi kaikki WebGL-sovellukseen liittyvät ohjelmoinnit. Tuotantojohtaja hoitaisi toimeksiantajan kanssa sovelluksen visuaaliset sisällöt ja projektijohtaja olisi toimeksiantajan kanssa tiiviissä yhteydessä ja he pitäisivät yhdessä huolen sovelluksen sisällöstä ja aikataulusta. Vastuullani ovat 3D-kaupungin suunnittelu, mallintaminen, animointi ja optimointi. 3D-kaupungin toteutukseen paneudutaan tarkemmin seuraavassa luvussa.

## 4 3D-kaupungin toteutus

Suunnitteluvaiheen valmistuttua oli kaupungin mallintamisen aika. Mallintamiseen on olemassa useita 3D-ohjelmia, joista käytettiin Autodesk 3ds Max 2015 -ohjelmaa. 3ds Maxin valinta mallintamisohjelmaksi oli helppoa, koska ohjelmasta oli eniten kokemusta. Ohjelmassa on myös kattavat työkalut polygonimallintamiseen, ja siihen on saatavissa myös erillisiä liitännäisiä ja skriptejä esimerkiksi optimoinnin helpottamiseen.

3D-kaupungin toteutukseen saatiin hyvin vapaat kädet. Toimeksiantajalta saatiin sen aikaisemmassa käytössä olleita kuvituskuvia (kuva 10), jotka toimivat referenssimateriaalina ja suunnannäyttäjinä, mutta muuten luovuuden käyttäminen oli sallittua. Referenssimateriaalit ovat yleensä yksi tärkeimmistä asioista 3D-mallintamisen aloittamisessa. Referenssimateriaali kuvaa sitä, mihin tulevalla 3D-visualisoinnilla ja mallintamisella pyritään pääsemään.



Kuva 10. Referenssikuva, joka toimii kaupungin mallintamisen suunnannäyttäjänä [32].

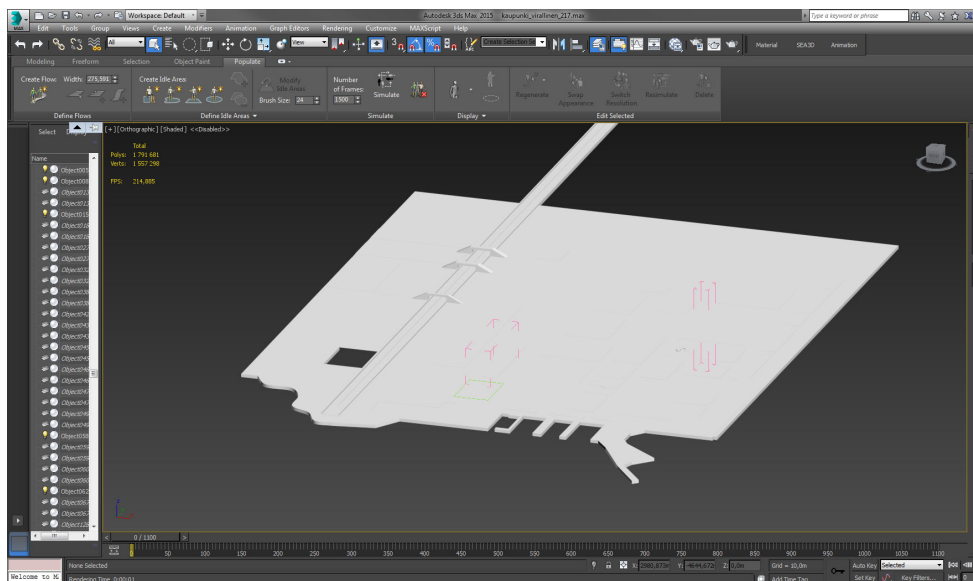
Koska 3D-kaupunki toimii reaaliaikaisesti renderöitävässä multimediasovelluksessa, tulee 3D-mallien verteksien ja polygonien lukumäärän olla mahdollisimman alhainen. Tällöin paras tekniikka mallintamiseen on Low Polygon -mallintaminen. Tämä tarkoittaa mallintamistapaa, jossa 3D-malli koostuu mahdollisimman alhaisesta määrästä polygoneja. Mallintamista aloitettaessa on hyvä miettiä mallinnettavan kappaleen tärkeyttä kokonaiskuvaan nähden. Tärkeimmissä malleissa, esimerkiksi lähellä kameraa olevien

mallien toteuttamisessa, voidaan käyttää enemmän yksityiskohtia verrattuna kauempana oleviin malleihin, jotka voivat sisältää huomattavasti vähemmän yksityiskohtia. Näin saadaan käyttäjän huomio kiinnittymään enemmän yksityiskohtia sisältäviin malleihin.

#### 4.1 Kaupungin mallintaminen

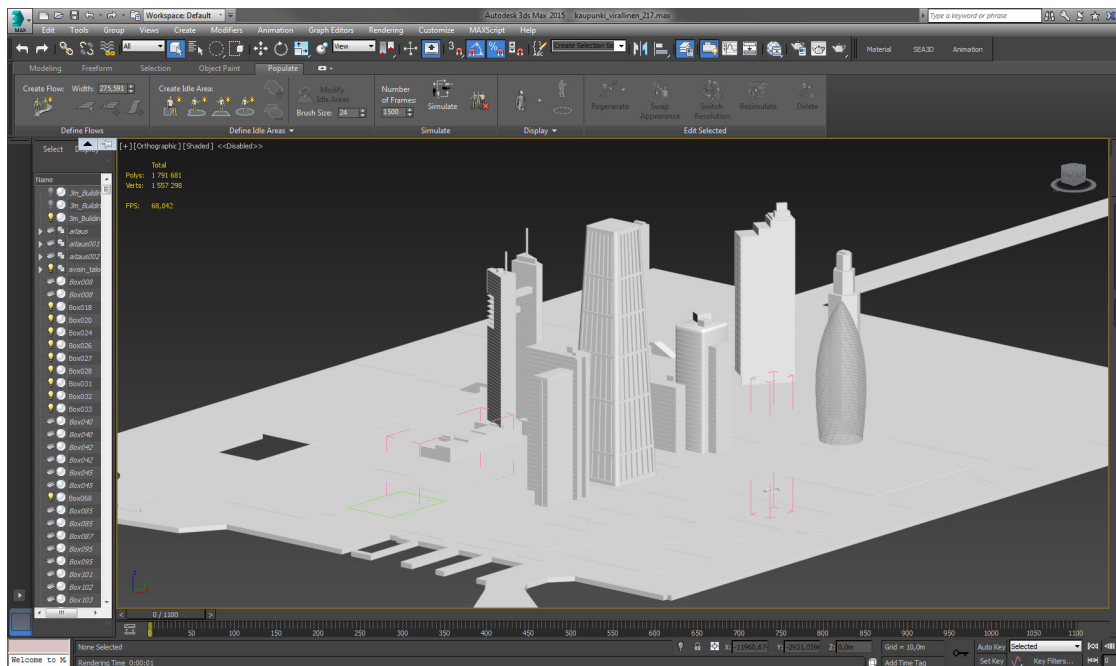
Sovelluksen käyttöliittymäksi luotiin 3D-kaupunki, joka koostui useista eri elementeistä ja rakennuksista: kaupungin pohjasta, korkeista ja matalista rakennuksista, hotellista, kauppakeskuksesta, lentokentästä, juna-asemasta, toimistotaloista, taustakaupungista, rantaviivasta, maastosta, satamasta ja vedestä. Lisäksi animaatioita varten mallinnettiin hissejä, liukuportaita, ovia, autoja ja ihmisiä. Ennen varsinaista mallinnuksen aloittamista tehtiin muutamia testejä, joilla kokeiltiin mallien siirtymistä 3D-ohjelmasta sovellukseen. Tarkastettiin myös 3ds Max 2015 -ohjelman asetukset kohdalleen, jotta mitta-kaavat olisivat realistiset rakennuksien korkeuksien suhteen.

Varsinainen mallinnus alkoi pohjan rakentamisesta kaupungille. Sen päälle sijoitettiin rakennukset. Pohja saatiin parhaiten tehtyä käyttämällä Line-työkalua ja piirtämällä halutun muotoinen alue (kuva 11). Tämän jälkeen pohjalle haluttiin paksuutta, joka saatiin Extrude-työkalulla. Pohjan päälle tehtiin myös samalla tekniikalla kortteleita, jotka muodostavat kaupunkiin kävely- ja autoteitä. Teille oli tarkoitus lisätä myöhemmin liikennettä animoimalla autoja ja ihmisiä.



Kuva 11. Varsinaisen keskustan pohja luotuna Autodesk 3ds Max 2015 -ohjelmalla.

Varsinaiset rakennukset alettiin mallintaa pohjan jälkeen. Toimeksiantajalta tuli toive, että kaupunki sisältäisi mahdollisimman paljon erityyppisiä rakennuksia, jotka sisältävät yleensä KONEen tuotteita, kuten hissejä, liukuportaita ja automaattioivia. Yleisesti rakennusten mallinnus aloitettiin luomalla laatikko tai sylinteri. Lisää yksityiskohtia, kuten ikkunoita ja kattoja, saatiin muokkaamalla laatikoita. Laatikko täytyi ensiksi muuttaa muokattavaan polygonimuotoon, jotta yksittäisiin vertekseihin ja polygoneihin päästiin käsiksi. Ikkunoiden mallintamisessa rakennuksiin täytyi seinien polygoneihin lisätä viivojen määrää pysty- ja vaakasuorassa Connect-työkalulla. Tämän jälkeen valittiin rakennuksen sivujen kaikki polygonit ja käytettiin Inset-työkalua, joilla saatiin ikkunoiden kehyksille leveys. Viimeiseksi ikkunoiden polygoneille täytyi antaa syvyysarvo, joka saatiin Extrude-työkalulla. Näin ikkunoita saatiin sisennettyä rakennuksen muusta pinnasta ja lopputuloksena saatiin toimistomaisia rakennuksia (kuva 12).

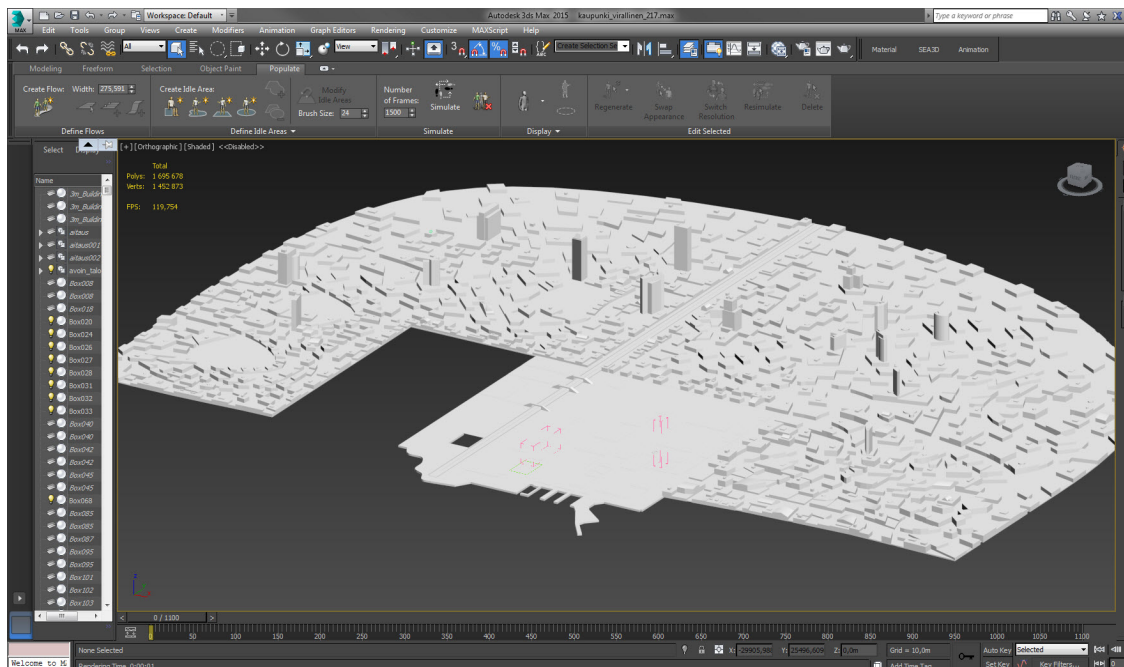


Kuva 12. Korkeita rakennuksia mallinnettuna Autodesk 3ds Max 2015 -ohjelmassa.

Suurin osa kaupungin rakennuksista tehtiin perinteiseen tapaan mallintamalla ne kuvi- en pohjalta. Tässä hyvänä referenssinä toimivat toimeksiantajalta saadut kuvituskuvat, jossa näkyi myös 3D:llä tehtyjä Low Polygon -malleja. Nämä rakennukset muodostavat kaupungin keskustan, johon käyttäjän päähuomion halutaan kiinnittyvän.

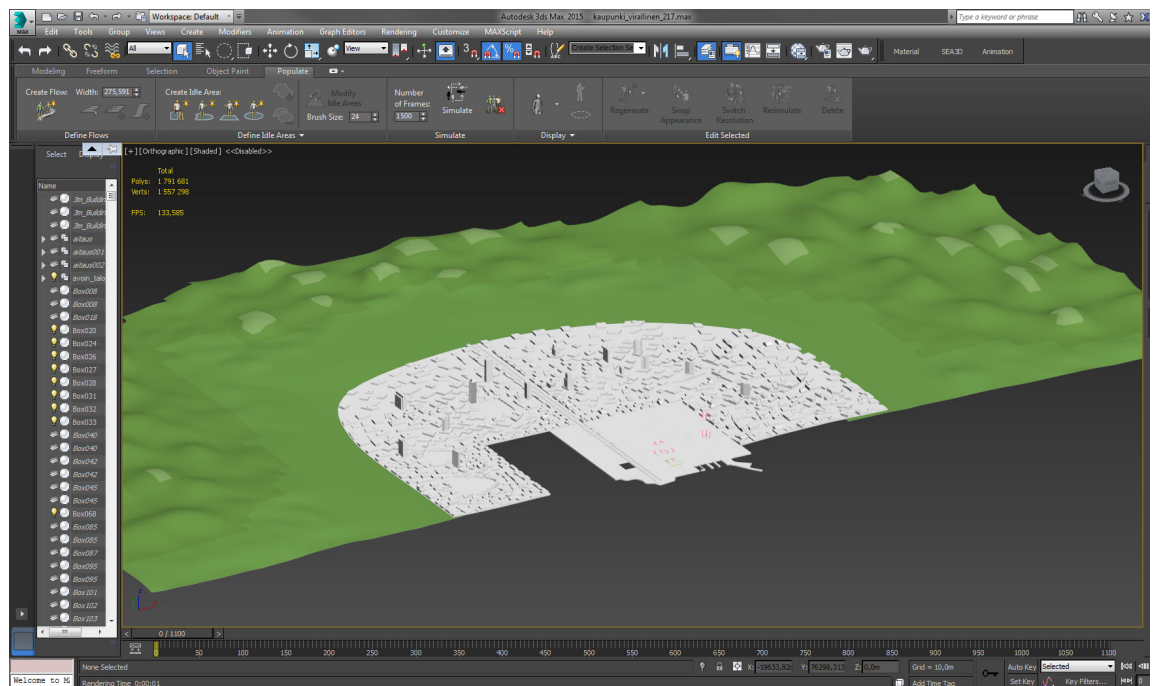
Varsinaisen keskustan hahmotuttua kaupunkiin lähdettiin lisäämään elementtejä, kuten taustakaupunkia, vuoristoa, rantaviivaa ja vettä. Näillä lisäyksillä kaupunki saisi realisti-

semman ympäristön. Taustakaupunkia suunniteltaessa löydettiin 3ds Max -ohjelmaan asennettava liitännäinen nimeltä Greeble, jolla saa tehtyä ”massakaupunkia”. Greeblen vahvuuksia on vähäinen polygonien käyttö. Liitännäinen otettiin käyttöön ja kokeiltiin, olisiko se sopiva tarkoitukseen. Ajatuksena oli, että taustakaupungin rakennukset eivät sisältäisi yksityiskohtia, toisin kuin keskustan rakennukset. Tällöin käyttäjän huomio kiinnittyä paremmin keskustaan, mutta kuitenkin kauempana näkyisi talojen geometriaa. Greeble-liitännäisellä ei kuitenkaan saatu toivottua lopputulosta. Rakennukset muodostuivat liian tiheästi, ja niiden väliin haluttiin saada enemmän tilaa, koska kyseessä ei ollut ydinkeskusta. Tämän vuoksi taustakaupungin toteutus päätettiin mallintaa ilman erillisiä liitännäisiä. Taustakaupungin mallintamisessa käytettiin aluksi samaa tekniikkaa kuin pohjan mallintamisessa. Ensin piirrettiin Line-työkalulla taustakaupungin alue, jolle määritettiin Extrude-työkalulla paksuus eli korkeus. Tämän jälkeen taustakaupungin pohja muutettiin muokattavaksi polygoniksi. Alueelle ruvettiin lisäämään verteksejä ja pilkkomaan polygoneja pienempiin osiin rakennuksia varten Cut-työkalulla. Kun polygonit olivat taustarakennuksille oikean kokoisia, alueelta valittiin polygoneja sieltä täältä, ja niille määritettiin korkeus Extrude-työkalun avulla (kuva 13). Näin saatiin taustakaupunkiin hieman erimuotoisia taloja melko vähäisellä polygonien määrällä.



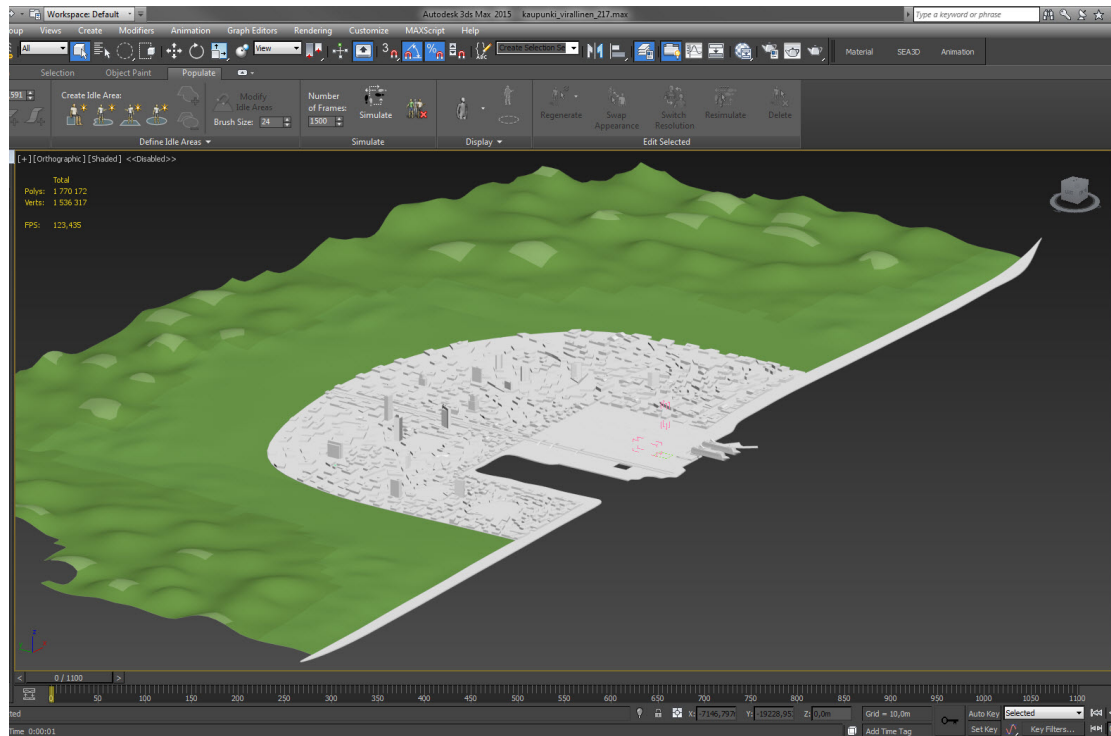
Kuva 13. Taustakaupunki mallinnettuna Autodesk 3ds Max 2015 -ohjelmassa.

Kaupunkiin haluttiin taustakaupungin lisäksi hieman enemmän realistista ympäristöä. Asutuksen harvennuttua päätettiin taustakaupungin jälkeen lisätä maastoa kuvaamaan luontoa asuinalueen päätyttyä (kuva 14). Maasto saatiin tehtyä lisäämällä taso kaupungin reunalle. Korkeuseroja tasolle saatiin ohjelman Push- ja Pull-työkaluilla. Työkaluilla saadaan laskettua tai nostettua verteksejä tason pinnasta ja halutulta alueelta pehmeästi. Tason päälle lisättiin Smooth-modifikaattori tasoittamaan pintaeroja, jotta kukkuloista saatiin pehmeitä ja sulavia. Maastoon tehtiin lopuksi myös hieman värieroja valitsemalla halutun alueen polygonit ja määrittämällä materiaalivalikosta niille väri. Lopputuloksena saatiin maastoon ja kukkuloihin pientä sävyvaihtelua.



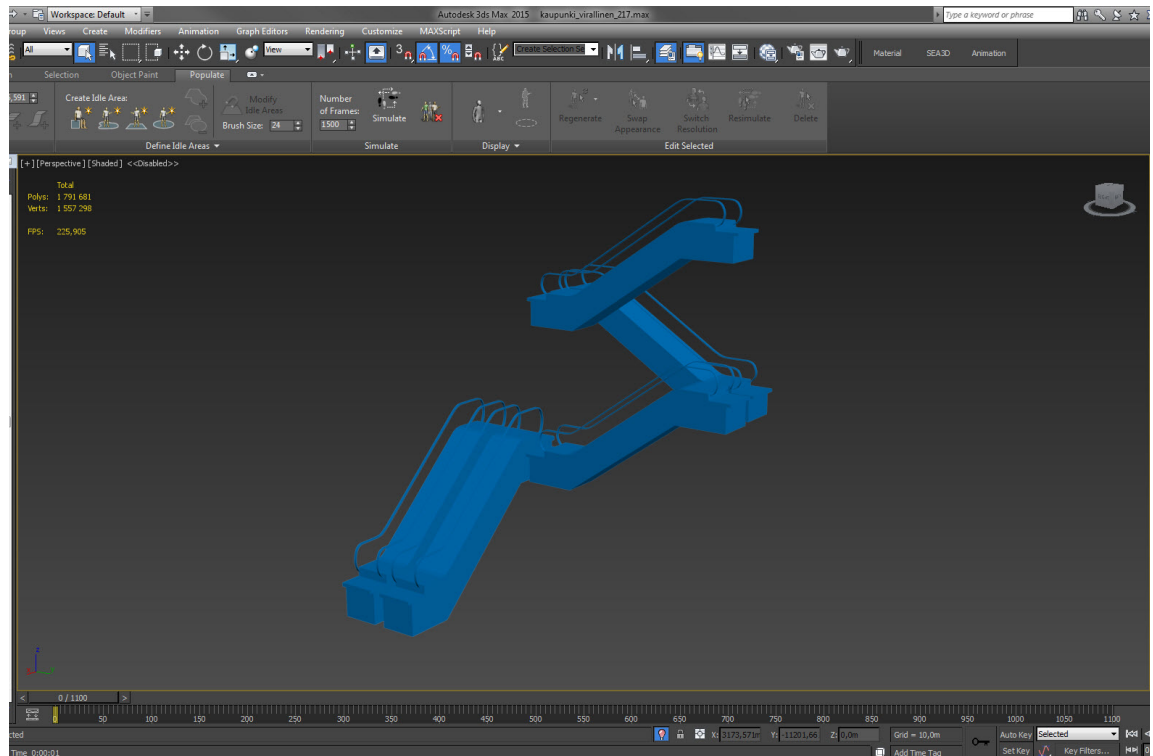
Kuva 14. Maasto mallinnettuna taustakaupungin ympärille Autodesk 3ds Max 2015 -ohjelmassa.

Kaupungin toisella puolella oli maastoa ja taustakaupunkia, joten toiselle puolelle haluttiin saada myös elävöittävää ympäristöä. Kaupunkiin päätettiin rakentaa laivasatama ja rantaviiva, jotka yhdistyvät mereen (kuva 15). Laivat sopivat hyvin konseptiin, koska myös laivahissit kuuluvat KONEen tuotetarjontaan. Näin saatiin kaupungin jokaisella puolella mielenkiintoista katsottavaa ympäröimään keskusta-alueita. Veden ja rantaviivan mallintamisessa käytettiin samaa tekniikkaa kuin pohjan mallintamisessa eli Line- ja Extrude-työkaluja.



Kuva 15. Satama ja rantaviiva mallinnettuna Autodesk 3ds Max 2015 -ohjelmassa.

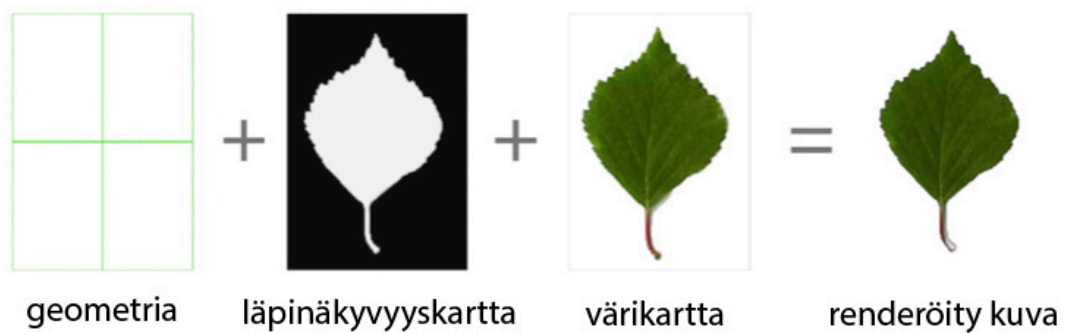
Kun suurin osa kaupungista ja sen ympäristöstä oli mallinnettu, oli aika mallintaa kaupunkiin KONEen tuotteita. Mallinnettavia tuotteita olivat hissit, liukuportaat, liukuovet ja automaattiovet. Osa kaupungin rakennuksista tarvitsi myös aukileikkaamista, jotta hissit ja liukuportaat saataisiin rakennuksissa näkyviin. Rakennusten aukileikkaaminen saatiin toteutettua 3ds Maxin Boolean-työkalulla. Hissien korit mallinnettiin laatikkoina ja sylintereinä, joihin tehtiin Line-työkalulla viivat esittämään hissien köyttä. Hissien ylä- ja alapäähän mallinnettiin myös hissien vetopyörät, joiden avulla köydet liikkuvat. Liukuportaiden ja ovien mallinnukset tehtiin suoraan toimeksiantajalta saatuun referenssikuvien pohjalta. Kuvien avulla pystyttiin liukuportaiden ja ovien mallintaminen toteuttamaan Low Polygon -tekniikalla (kuva 16).



Kuva 16. Liukuportaat mallinnettuna Autodesk 3ds Max 2015 -ohjelmassa.

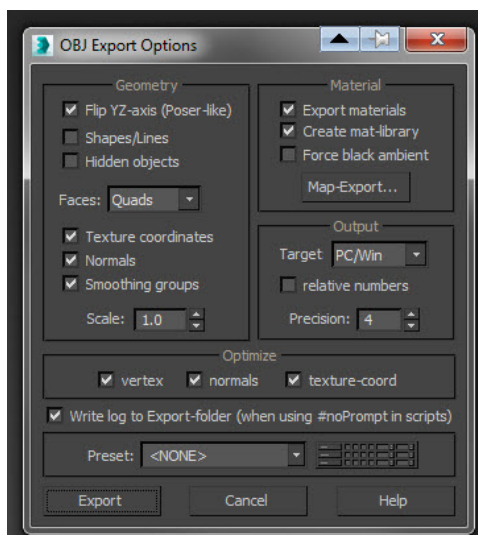
3D-mallinnuksessa ei jokaista pientä yksityiskohtaa kannata mallintaa geometrisesti. Mikäli näin tehdään, 3D-mallin polygonien määrä saattaa nousta hyvin suureksi. Tämän vuoksi puiden mallintamisessa käytettiin hyödyksi läpinäkyvyyskarttaa. Tämä vaikuttaa kappaleen läpinäkyvyyteen, ja sitä voidaan käyttää tekstuurina asioiden näyttämiseen tai piilottamiseen. Läpinäkyvyyskartta koostuu kahdesta väristä, jotka ovat musta ja valkoinen. Musta tarkoittaa täysin läpinäkyvää ja valkoinen läpinäkymätöntä eli täysin näkyvää. Kokonaisen puun mallintamisen sijaan käytettiin läpinäkyvyyskarttaa, jossa puun sisäosa on valkoista eli näkyvää ja puun ulkopuolinen osa mustaa, joka piilottaa geometrian ylimääräiset osat. Näin renderöitäessä saatiin kuva, joka näyttää mallinnetulta puulta, vaikka oikeasti siinä on vain neljä polygonia sisältävä taso. Läpinäkyvyyskartan vaikutus renderöitäessä näkyy kuvassa 17. Tämäntyyppiset mallit ovat todella kevyitä, eivätkä ne juuri kasvata tiedostokokoa, vaikka niitä kopioitiin kaupunkiin useita satoja. [33.]





Kuva 17. Läpinäkyvyyskartan vaikutus lopputulokseen [33].

Kaupungin mallinnuksen valmistuttua viimeisenä toimenpiteenä oli 3D-mallien ulosvienti 3ds Max 2015 -ohjelmasta WebGL:ää varten. Paikallaan olevien eli staattisten 3D-mallien ulosvienti tehtiin valitsemalla .obj eli Wavefront OBJ geometry ulosvietävien mallien tiedostomuodoksi. Kun tiedostomuoto oli valittu ja hyväksytty, aukesi kuvassa 18 esitetty valikko. Kuvassa esitellään viemisasiasetukset, joilla paikallaan olevien mallien ulosvienti onnistui. Valikosta voidaan valita asetukset, jotka malliin halutaan sisällyttää. Asetukset olivat perusarvoissaan, eikä niitä tarvinnut muuttaa ulosvientiä varten. Ulosviennin onnistuttua tiedostot luovutettiin ohjelmoijalle, joka konvertoi .obj-tiedostot .json-tiedostoiksi. Viimeiseksi tiedostot lisättiin ja ladattiin WebGL-sovelluksen tiedostoon.



Kuva 18. Wavefront OBJ geometry -tiedostomuodon ulosvientivalikko Autodesk 3ds Max 2015 -ohjelmassa.

## 4.2 Mallin optimointi

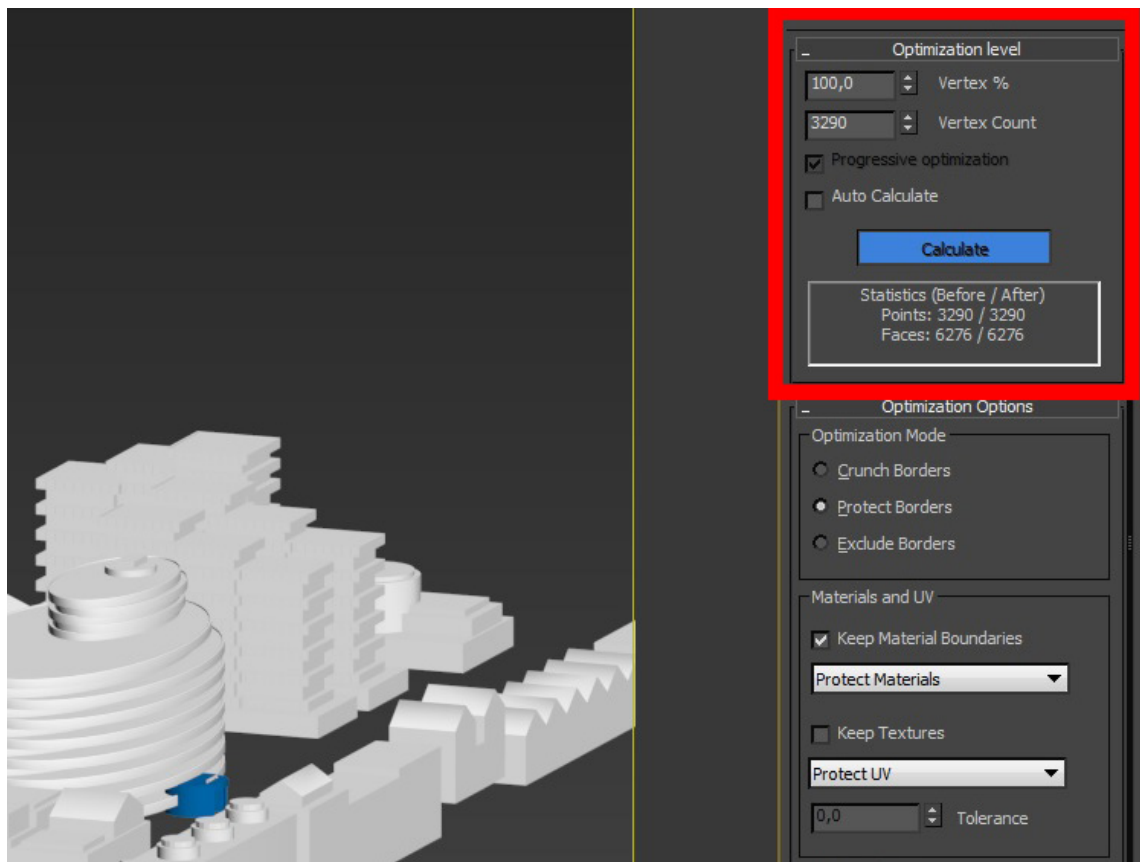
Reaaliaikainen renderöinti vaatii optimoimattoman 3D-mallin näyttämiseen tietokoneelta hyvin paljon, esimerkiksi tehokasta näytönohjainta ja riittävää määrää muistia. Tietokoneen tehontarve riippuu käsiteltävien 3D-mallien polygonien ja verteksien lukumäärästä. Mitä enemmän polygoneja mallissa esiintyy, sitä suurempi tehontarve tietokoneella on. [34.]

Kuvien esittämää lukumäärää kuvataan ja mitataan sekunteina. Sekunnissa vaihtuvien kuvien määrää kutsutaan kuvataajuudeksi eli frames per second (FPS). Tämä mittaa, kuinka monta kertaa sekunnissa kuva piirtyy ruudulle. Videoiden FPS on kiinteä, yleensä 25 kuvaa sekunnissa, mutta peleissä ja sovelluksissa luku saattaa vaihdella. Jotta liike olisi sulavaa (ei nykivää), kuvataajuuden tulisi olla vähintään 15 kuvaa sekunnissa. Riittävän suuren kuvataajuuden ylläpitäminen vaatii koneelta suurta laskentatehoa. Tehontarve kasvaa sitä mukaa, kuin kuvien määrää kasvatetaan sekuntia kohden.

Yleisesti 3D-malleista kannattaa karsia kaikki huono geometria pois. Huonoksi geometriaksi voidaan laskea geometria, joka aiheuttaa ongelmia geometrian muodostamisessa reaaliaikaisessa renderöinnissä. Huonoksi geometriaksi lasketaan myös yli neljä verteksiä sisältävät polygonit sekä liian raskaat geometriat eli mallit, jotka sisältävät liian paljon polygoneja verrattuna esitettyihin yksityiskohtiin. Lisäksi huonoa geometriaa voi olla päällekkäin oleva geometria tai liian lähellä toisiaan olevat verteksit. Reaaliaikaisessa renderöinnissä nämä saattavat aiheuttaa mallien välkkymistä. [35; 36.]

3D-mallien yleisenä optimointikeinona pidetään tekstuurien käyttöä. Tässä sovelluksessa tekstuureja käytettiin kuitenkin ainoastaan puiden mallintamiseen. Kaupungin rakennusten optimointiin polygonien ja verteksien vähentämiseksi käytettiin erillistä kolmannen osapuolen liitännäistä nimeltään Polygon Cruncher. Liitännäinen asennettiin 3ds Maxin liitännäisten kansioon, minkä jälkeen se näkyi 3ds Max -ohjelmassa. Polygon Cruncher -liitännäisen avulla rakennuksista saatiin vähennettyä verteksien lukumäärää automatisoidusti. Kun malli oli valittu, liitännäinen näytti sen verteksien lukumäärän, jota voitiin laskea prosentuaalisesti tai kappalemääräisesti (kuva 19). Prosenttiarvoa pystyi muuttamaan 100:n ja 0:n välillä, jossa 100 tarkoitti, ettei yhtäkään verteksiä ole poistettu tai yhdistetty. Prosenttiarvoa pienentämällä mallin verteksien lukumäärä vastaavasti pieneni. Verteksien lukumäärää vähennettäessä nähtiin reaaliaikaisesti optimoinnin vaikutus objektin geometriaan. Käytännössä verteksien lukumää-

rää voitiin vähentää niin paljon, kunnes objektissa ilmeni liian suuria geometrisia muutoksia tai objekti alkoi hajota.

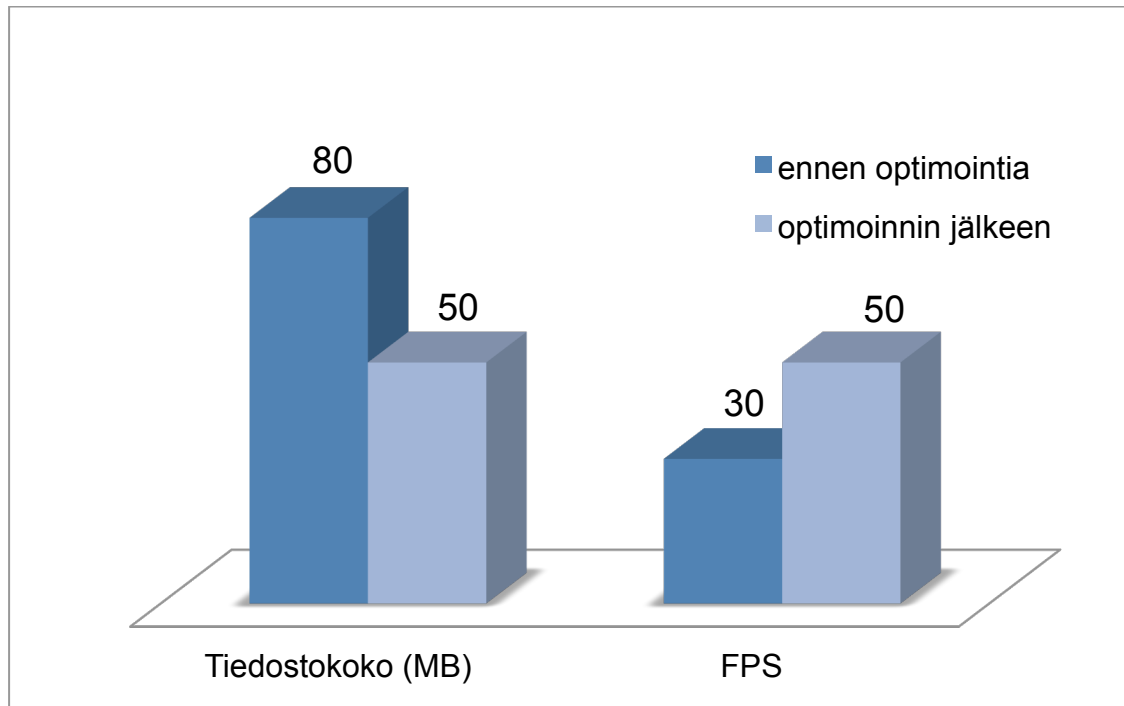


Kuva 19. Optimointiin käytetyn Polygon Cruncher -liitännäisen valikko Autodesk 3ds Max 2015 -ohjelmassa.

Liitännäisen käytön lisäksi kaupungista poistettiin kaikki ylimääräiset yksityiskohdat, joita ihmissilmä ei pystynyt erottamaan tai näkemään lopullisessa visualisoinnissa. Myös joitakin sellaisia objekteja poistettiin, joiden merkitys lopulliseen visualisointiin oli vähäistä.

Optimoinnin tavoitteena oli verteksien lukumäärän pienentäminen ja samalla sovelluksen tiedostokokojen pienentäminen. Tämän tarkoituksena oli mahdollistaa maksimaalinen suorituskyky sovelluksen näyttämiseen. Sovelluksen kuvataajuus on yleensä hyvin verrannollinen verteksien kokonaislukumäärään ja sovelluksen tiedostojen kokoon käytettävissä olevan tietokoneen tehon lisäksi. Mitä enemmän sovellus sisältää verteksejä, sitä isompi yleensä sovelluksen tiedostokoko on. Iso tiedostokoko puolestaan tarkoittaa yleensä alhaisempaa kuvataajuutta. Ennen optimoinnin aloittamista sovelluksen

tiedostokooksi mitattiin noin 80 megatavua ja verteksien lukumääräksi noin 1 700 000. Tällöin kuvataajuus oli noin 30. Kun kaupungin verteksien- ja polygonien määrää vähennettiin optimoinnin avulla, tiedostokoko saatiin pienennettyä 50 megatavuun ja verteksien lukumäärä oli enää noin 1 000 000. Optimoinnin jälkeen kuvataajuus nousi noin 50:een (kuva 20).



Kuva 20. Kuvataajuuden ja tiedostokoon muutos ennen optimointia ja sen jälkeen.

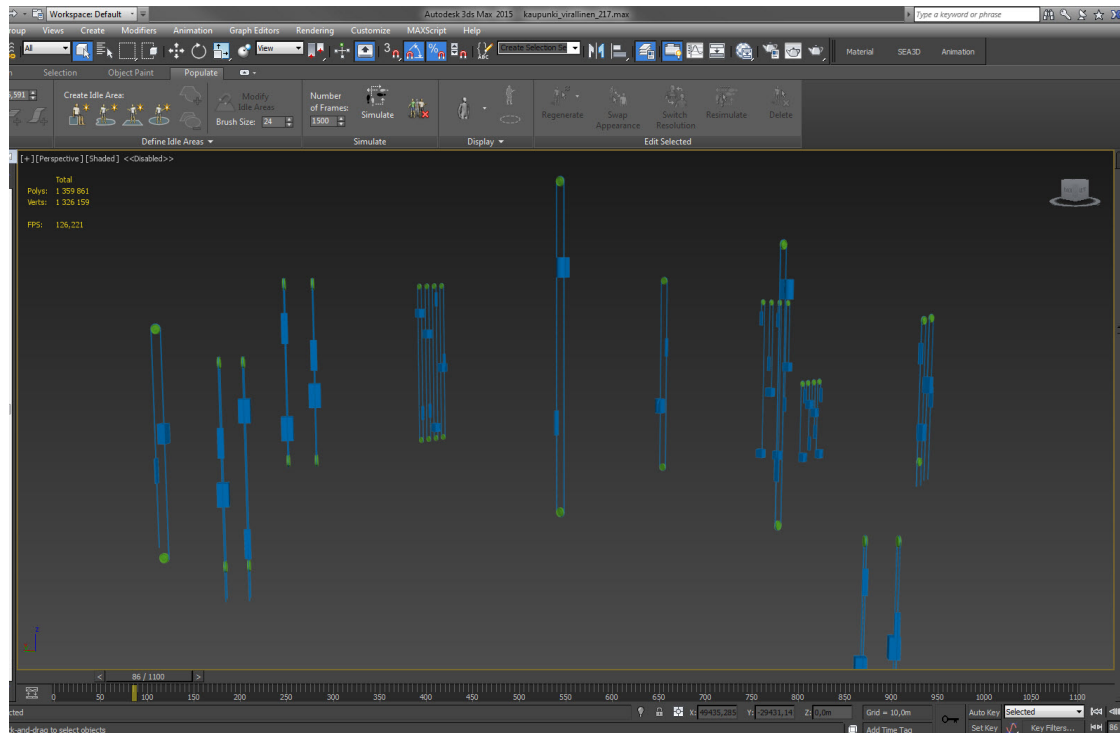
Liian matala tai heittelehtivä kuvataajuus vie sovelluksen liikkeestä luonnollisuuden ja voi häiritä käyttäjän vuorovaikutusta sovellusta käytettäessä. Sovelluksen on pystyttävä tuottamaan 30–60 kuvaa sekunnissa, jotta se käyttäjän mielestä toimisi sulavasti ja luonnollisesti. Tietokoneen suorituskyvyltä vaaditaan paljon, koska sen täytyy käsitellä samaan aikaan sovelluksen geometria, valaistus, tekstuurit ja toiminnallisuus.

### 4.3 Hissituotteiden animointi

3D-mallia halutaan saada usein elävöitettyä, koska paikallaan oleva malli voidaan kokea tylsäksi. Tässäkin tapauksessa sovellukseen haluttiin animaatioiden avulla liikettä käyttäjän interaktiivisuuden lisäksi. Animaatiot voidaan toteuttaa monella tavalla, joten nekin on hyvä suunnitella etukäteen. Tiettyjä mallien osia tai koko mallia voidaan animoida käsin tai osittain automaattisesti. Animointi tehdään aikajanalla keyframien eli avainruutujen avulla.

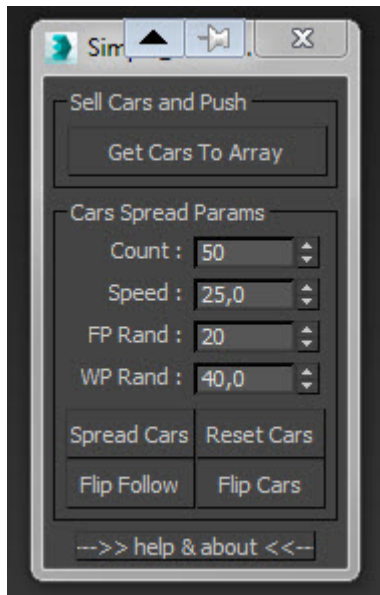
Animaatioiden toteutus jätettiin optimoinnin kanssa viimeiseksi, koska kaupunki sai hieman muutoksia projektin aikana toimeksiantajan kommenttien seurauksena. Animaatiot oli helpointa tehdä siinä vaiheessa, kun kaupungin mallinnus oli muuten valmis. Myös yleisesti animaatiot kannattaa tehdä 3D-malleille siinä vaiheessa, kun niiden geometria on lähes valmis tai niihin ei tule kovin isoja muutoksia. Koska kyseessä on reaaliaikasovellus, käyttäjä hallinnoi kameraa ja sovelluksen kuvakulmaa pyörittämällä kaupunkia. Tämän lisäksi sovellukseen haluttiin itsestään toistuvia animaatioita. Toimeksiantajan toiveesta kaupunkiin tehtiin animaatioita ihmisiin, hisseihin, oviin ja autoihin. Reaaliaikainen sovellus toi myös haasteita animaatioiden suunnitteluun. Animaatioista täytyi saada jatkuvia, jotta käyttäjä ei huomaisi, milloin animaatio alkaa ja päättyy.

Seuraavaksi käsitellään animaatioiden toteutusta reaaliaikaista renderöintiä varten. Ensimmäiseksi kaupungissa haluttiin hissien liikkuvan ja ovien aukeavan tietyissä rakennuksissa. Aikajanalla nämä animoidaan keyframeja eli avainruutuja käyttämällä. Avainruutu merkittiin aloituskohtaan, minkä jälkeen objektia eli tässä tapauksessa hissiä liikutettiin haluttuun kohtaan, johon lisättiin uusi avainruutu. Näin saatiin hissiin ensimmäinen liike. Koska hissit eivät oikeassa maailmassa liiku ainoastaan ylimmästä kerroksesta alimpaan kerrokseen, tuli animaatioon lisätä myös pysähdyksiä avainruutujen avulla. Niiden avulla hissien liikkeestä saatiin realistinen. Animaation jatkuva toistuvuus ilman selvää aloitus- ja loppumiskohtaa saatiin, kun hissien sijainti 3D-maailmassa määritettiin täsmälleen samaan kohtaan ensimmäisessä ja viimeisessä ruudussa. Hisseille luotiin hieman toisistaan poikkeavia animaatioita, jotta niiden animaatiot eivät olisi identtisiä. Hissien animaatioiden kestoksi määritettiin 1100 ruutua (kuva 21).



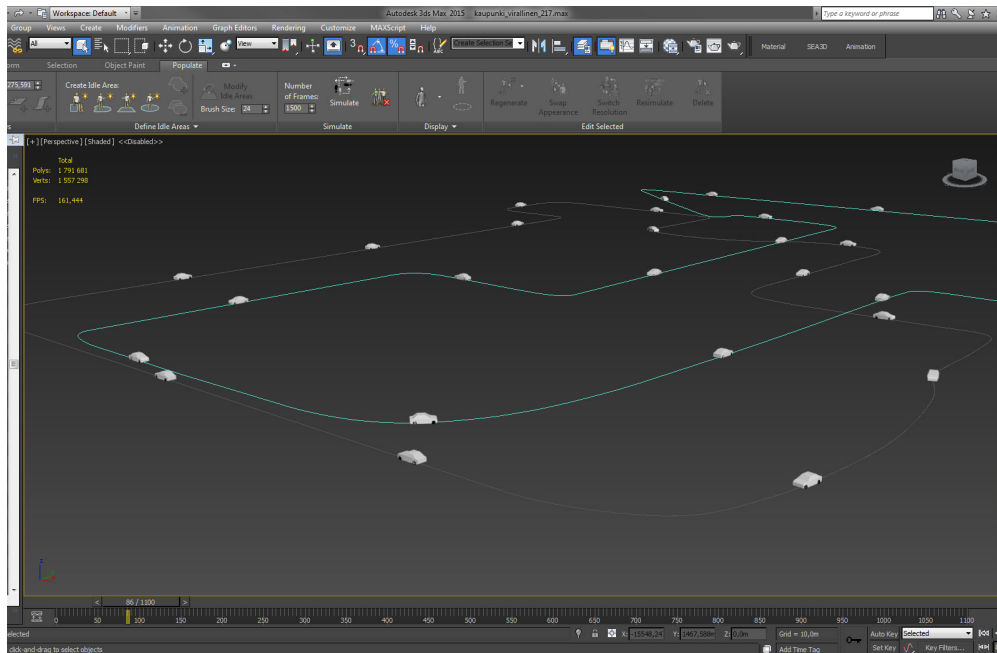
Kuva 21. Hissien animointi Autodesk 3ds Max 2015 -ohjelmassa.

Ihmisten animoinnissa kokeiltiin ensiksi 3ds Max 2015 -ohjelmasta valmiina löytyvää Populate People -työkalua. Työkalu ei kuitenkaan sopinut käyttötarkoitukseen, koska animaatioiden jatkuvuus oli mahdotonta ilman, että ihmiset ”syntyivät” uudestaan animaation loputtua. Tämän vuoksi työkalu hylättiin ja etsittiin vaihtoehtoinen tapa toteuttaa ihmisten animointi. Lopulta ihmisten ja autojen animoinnissa päädyttiin käyttämään ulkopuolista skriptiä nimeltä Simple Traffic. Skripti asennettiin 3ds Max 2015 -ohjelman kansioon, minkä jälkeen se käynnistettiin ohjelman yläreunassa olevasta MAXScript-valikosta. Kun skripti oli valittu kansioista, ohjelmaan aukesi kuvan 22 mukainen asetussvalikko.



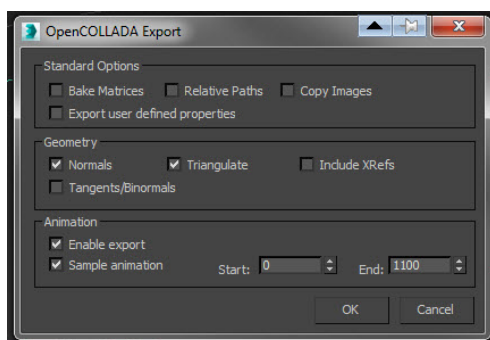
Kuva 22. Simple Traffic -skriptin asetusvalikko.

Skriptiä käytettäessä ensimmäiseksi luotiin polku, joka saatiin tehtyä Line-työkalulla. Tämän jälkeen ennalta määritettyyn polkuun liitettiin mallit, joiden haluttiin kiertävän polkua pitkin. Autot saatiin polulle valitsemalla ne ensiksi ja sen jälkeen valitsemalla asetusvalikosta kohta Get Cars to Array. Tämän jälkeen valittiin vielä polku ja asetusvalikosta kohta Spread Cars. Lopuksi asetuksien avulla pystyttiin määrittämään autojen lukumäärä, autojen tiheys leveys- ja pituussuunnassa sekä autojen nopeus polkua kierrettäessä. Autojen animaation parhaaksi pituudeksi osoittautui 616 ruutua ja ihmisten animaation 735 ruutua. Jos animaatiot olisi toteutettu siten, että autot tai ihmiset kiertäisivät määritellyn reitin kerran kokonaan ja palaisivat samaan kohtaan kuin olivat animaation ensimmäisessä ruudussa, olisi tiedostokoko kasvanut liian isoksi avainruutujen runsaan määrän takia. Tavoitteena oli saada kaikki tiedostokoot mahdollisimman pieniksi, joten päädyttiin määrittelemään semmoinen määrä ruutuja, jolloin toinen auto tai ihminen olisi täsmälleen samassa kohdassa kuin edellinen oli animaation ensimmäisessä ruudussa (kuva 23). Näin animaatioissa saatiin ylläpidettyä liikkeiden jatkuvuus. Autojen ja ihmisten animointi toteutettiin täsmälleen samalla tavalla.



Kuva 23. Autojen animointi Simple Traffic -skriptin avulla Autodesk 3dsMax 2015 -ohjelmassa.

Animaatioiden valmistuttua viimeisenä tehtävänä oli niiden ulosvienti sovellusta varten. Animaatioiden vieminen WebGL-sovellukseen vaati animaatioiden viemistä ulos .dae- eli OpenCollada-tiedostomuodossa. Animaatioiden tallentaminen OpenCollada-muotoon onnistui valitsemalla export-valikosta .dae-tiedostopäätte. Kun tiedostomuoto oli valittu, aukesi kuvan 24 mukainen valikko. Valikosta voitiin valita asetukset, jotka animaation haluttiin sisältävän. Animaatioihin valittiin asetuksista kohdat normals, triangulate, enable export ja sample animation.



Kuva 24. OpenCollada-tiedostomuodon ulosviennin asetusvalikko.

Lopuksi määritettiin animaatioiden alkamis- ja päättymisajankohdat avainruutujen muodossa. Viimeisenä toimenpiteenä oli valmiiden tiedostojen luovuttaminen ohjelmajalle.



## 5 Multim mediasovelluksen lopputulos ja tulevaisuus

Multim mediasovelluksen suunnittelu ja toteutus aloitettiin kesällä 2014, jonka aikana käynnistettiin 3D-kaupungin mallintaminen. 3D-kaupungin mallintamiseen, animointiin ja optimointiin kului yhteensä noin 450 työtuntia. Multim mediasovelluksen ensimmäinen versio valmistui tammikuun 2015 aikana, jolloin se asennettiin Hyvinkään People Flow Centeriin (kuva 25).



Kuva 25. Multim mediasovellus asennettuna KONEen People Flow Centeriin Hyvinkäälle.

Kahden ensimmäisen kuukauden käyttökokemukset ovat olleet erittäin myönteisiä. Sovelluksen käytettävyys on osoittautunut hyväksi. Satunnaisia käyttäjiä haastateltaessa kaikki ovat poikkeuksetta kehuneet käyttöliittymää helppokäyttöiseksi. Asennettu kosketusnäyttöpaneeli näkyy kuvassa 26. Myös KONEen tytäryhtiöissä sovellus on herättänyt voimakasta mielenkiintoa, ja on oletettavaa, että ainakin muutama maahan sovellus vietäisiin vuoden 2015 aikana ja laajemmassa mittakaavassa vuoden 2016 aikana, kun järjestelmän vaatimat alkuinvestoinnit on saatu budjetoitua osana tavallista vuotuista budjetointiprosessia.



Kuva 26. Kosketusnäyttöpaneeli, josta käyttäjä hallitsee sovellusta.

Sovelluksen istuntojen määrä kahden ensimmäisen kuukauden aikana on ollut noin 400 istuntoa kuukaudessa. Yhden istunnon keskimääräinen pituus on ollut 330 sekuntia. Sovelluksen sivuista suosituimmaksi ovat osoittautuneet 3D-kaupunki, hissit ja referenssit.

Sovelluksen skaalautuvuusprosessi on käynnistetty, ja tavoitteena on saada vuoden 2015 aikana kannettavalle tietokoneelle ja tabletille suunnitellut prototyypit testikäyttöön. Myös KONEen yleisesittely on vielä työn alla. Multimediasovelluksen neljäs osa, työkalut ja suunnittelun apuvälineet, on vielä suunnitteluvaiheessa, ja ensimmäiset toteutukset tehdään vuoden 2016 aikana.

Tulevaisuuden harkinnassa on, että KONEen pääreferensseistä tehtäisiin 3D-visualisointeja, joissa rakennusten sisällä olisi mahdollista liikkua. Tämä mahdollistaisi ihmisliikenteen seuraamisen ja KONEen tuotteiden kokemisen virtuaalisesti. Potentiaalinen ehdokas tähän olisi maailman korkeimmaksi rakennukseksi tuleva Kingdom Tower, joka voitaisiin näin esittää virtuaalisesti 3D-ympäristössä.

## 6 Yhteenveto

Insinöörintyön tarkoituksena oli perehtyä WebGL-ohjelmointirajapintaan sekä suunnitella ja toteuttaa 3D-kaupunki interaktiiviseen multimediasovellukseen, joka hyödyntää WebGL-tekniikkaa. WebGL-tekniikka mahdollistaa 3D-grafiikan näyttämisen verkkoselaimessa ilman ylimääräisiä liitännäisiä tai lisäosia. WebGL-tekniikan suosio on viime vuosina kasvanut merkittävästi, mistä yhtenä osoituksena on Microsoftin päätös tukea viimein WebGL-tekniikkaa Internet Explorer 11 -selaimessaan. Tämän myötä kaikki suosituimmat selaimet ovat saaneet tuen WebGL-tekniikalle, mikä auttaa varmasti tekniikan leviämisessä. WebGL:n avulla voidaan selaimiin lisätä esimerkiksi interaktiivisia musiikkivideoita, pelejä, sovelluksia, taidetta, 3D-ympäristöjä tai 3D-mallinnettuja objekteja tai luoda fysikaalisia simulaatioita.

Opinnäytetyö lähti liikkeelle toimeksiantajan tarpeesta saada esittelytilaansa enemmän interaktiivisuutta, tuotteiden ja palveluiden esittelyä. Olemassa olevan esittelytilan sisältö oli koostunut pääasiassa vain staattisista kuvista, teksteistä ja itsenäisesti toistuvista videoista ja animaatioista. Tähän ratkaisuna suunniteltiin ja toteutettiin 3D-kaupunki, joka on osana Virtual Wall -nimellä kulkevaa multimediasovellusta, jossa hyödynnetään WebGL-tekniikkaa. Virtuaalisen videoseinän käyttöliittymää hallitaan erillisestä kosketusnäyttöpaneelistä. Sovelluksen käyttöliittymäksi muodostui 3D-mallinnettu kaupunki, joka toimii samalla sovelluksen kotivalikkona. Kaupunkia ohjataan vetämällä sormella vaakasuorassa, ja kaupungin päälle rakennettujen navigointilinkkien avulla käyttäjä pääsee siirtymään eri osioihin ja sisältöihin. 3D-kaupunki sopii kotivalikon näkymäksi, koska useimmat toimeksiantajan tuotteista ja palveluista liittyvät kiinteästi rakennuksiin ja niissä tapahtuvien liikennevirtojen hallintaan. Kaupungissa on helppo myös esittää erilaiset asiakas- ja rakennussegmentit sekä näyttää tuotteet luonnollisissa käyttöympäristöissään.

3D-kaupungin suunnittelu ja toteutus lähti liikkeelle toimeksiantajan toiveista ja referenssimateriaaleista, mutta muuten visuaaliseen toteutukseen annettiin hyvin vapaat kädet. Projektin edetessä muutoksia tehtiin toimeksiantajan toiveiden mukaisesti, mikä ansiosta päästiin haluttuun lopputulokseen. 3D-kaupungin toteutus koostui kaupungin ja rakennusten mallintamisesta, animoinnista ja optimoinnista. Kaupungin mallintaminen osoittautui työläämmäksi, kuin osasi kuvitella ennen projektin alkamista.

Interaktiivisen multimediasovelluksen ensimmäinen versio valmistui tammikuussa 2015, jolloin se asennettiin toimeksiantajan esittelytilaan. Kahden ensimmäisen kuukauden käyttökokemukset ovat olleet erittäin myönteisiä, ja käyttöliittymänä toimivaa 3D-kaupunkia on pidetty helppokäyttöisenä ja visuaalisesti näyttävänä. Tulevaisuudessa sovellusta on tarkoitus viedä yhtiön muiden maiden esittelytiloihin sekä ottaa messu- ja asiakaskäyttöön. Sisällön tuottaminen multimediasovellukseen on pitkän ajan tähtäin, ja sitä jatketaan tulevaisuudessa.

Projekti oli mielenkiintoinen, mutta haastava. 3D-kaupungin valmistukseen kului yhteensä noin 450 tuntia, minkä aikana tuli opittua paljon uutta.

## Lähteet

- 1 WebGL: Frequently Asked Questions. 2012. Verkkodokumentti. Learning WebGL. <[http://learningwebgl.com/cookbook/index.php/WebGL:\\_Frequently\\_Asked\\_Questions](http://learningwebgl.com/cookbook/index.php/WebGL:_Frequently_Asked_Questions)>. 31.1.2012. Luettu 13.1.2015.
- 2 WebGLRaportti. 2010. Verkkodokumentti. Webhierarkia. <<https://code.google.com/p/webhierarkia/wiki/WebGL>>. 22.4.2010. Luettu 20.1.2015.
- 3 OpenGL ES 2.0 for the Web. 2012. Verkkodokumentti. Khronos. <[https://www.khronos.org/webgl/wiki/Getting\\_Started](https://www.khronos.org/webgl/wiki/Getting_Started)>. Luettu 20.1.2015.
- 4 Document Object Model (DOM). 2005. Verkkodokumentti. W3C. <<http://www.w3.org/DOM/>>. 19.1.2005. Luettu 20.1.2015.
- 5 Harris, Andy. 2011. HTML5 for Dummies Quick Reference. United States of America: Wiley Publishing.
- 6 Caballero, Luz. 2011. An introduction to WebGL. Verkkodokumentti. Dev.Opera. <<http://dev.opera.com/articles/view/an-introduction-to-webgl>>. 13.10.2011. Luettu 5.3.2015.
- 7 WEBGL-3D pelejä selaimiin. 2012. Verkkodokumentti. Animaattori. <[http://animaattori.com/?page\\_id=614](http://animaattori.com/?page_id=614)>. 2012. Luettu 2.2.2015.
- 8 Parisi, Tony. 2012. WebGL: Up and Running. United States of America: O'Reilly Media.
- 9 Caballero, Luz. 2011. An introduction to WebGL – Part 1. Verkkodokumentti. Dev.Opera. <<http://dev.opera.com/articles/view/an-introduction-to-webgl-part-1/>>. 13.10.2011. Luettu 5.3.2015.
- 10 Boesch, Florian. 2013. Why you should use WebGL. Verkkodokumentti. Codeflow. <<http://codeflow.org/entries/2013/feb/02/why-you-should-use-webgl/>>. 2.2.2013. Luettu 14.2.2015.
- 11 Bridge, Henry. 2011. GPU acceleration + old drivers = :( Verkkodokumentti. Chromium Blog. <<http://blog.chromium.org/2011/03/gpu-acceleration-old-drivers.html>>. 1.3.2011. Luettu 19.2.2015.
- 12 Welch, Chris. 2013. Microsoft reportedly bringing WebGL support to Internet Explorer 11. Verkkodokumentti. The Verge. <<http://www.theverge.com/2013/3/30/4165204/microsoft-bringing-webgl-support-internet-explorer-11-windows-blue>>. 1.3.2013. Luettu 15.3.2015.

- 13 WebGL Considered Harmful. 2011. Verkkodokumentti. Microsoft TechNet. <<http://blogs.technet.com/b/srd/archive/2011/06/16/webgl-considered-harmful.aspx>>. 16.6.2011. Luettu 19.3.2015.
- 14 WebGL – More WebGL Security Flaws. 2011. Verkkodokumentti. Context. <<http://www.contextis.com/resources/blog/webgl-more-webgl-security-flaws/>>. 16.6.2011. Luettu 28.3.2012.
- 15 WebGL – 3D Canvas graphics. 2011. Verkkodokumentti. Caniuse. <<http://www.contextis.com/resources/blog/webgl-more-webgl-security-flaws/>>. 1.3.2011. Luettu 28.3.2015.
- 16 Ramadan, Rido. 2012. A Brief History of OpenGL vs Microsoft DirectX. Verkkodokumentti. Games, Applications, Driving, and Beyond... <<https://personanonymouse.wordpress.com/2012/07/12/a-brief-history-of-opengl-vs-microsoft-directx/>>. 12.7.2012. Luettu 15.3.2015.
- 17 Laitila, Teemu. 2011. WebGL ei tule Internet Exploreriin – Liikaa tietoturvaongelmia. Verkkodokumentti. Afterdawn. <[http://fin.afterdawn.com/uutiset/artikkeli.cfm/2011/06/17/webgl\\_ei\\_tule\\_internet\\_exploreriin\\_-\\_liikaa\\_tietoturvaongelmia](http://fin.afterdawn.com/uutiset/artikkeli.cfm/2011/06/17/webgl_ei_tule_internet_exploreriin_-_liikaa_tietoturvaongelmia)>. 17.6.2011. Luettu 15.3.2015.
- 18 Ohjelmointi 1. 2010. Verkkodokumentti. Jyväskylän yliopisto <<http://kurssit.it.jyu.fi/ITKP102/moniste/html/moniste.html>>. 7.9.2010. Luettu 29.3.2015.
- 19 HTML5 Game Engines and Frameworks. 2013. Verkkodokumentti. Techslides. <<http://techslides.com/html5-game-engines-and-frameworks>>. 2013. Luettu 15.3.2015.
- 20 User Contributions. 2014. Verkkodokumentti. Khronos. <[https://www.khronos.org/webgl/wiki/User\\_Contributions](https://www.khronos.org/webgl/wiki/User_Contributions)>. 8.8.2014. Luettu 11.2.2015.
- 21 Three.js contributors. 2013. Verkkodokumentti. Mrdoob. <<https://github.com/mrdoob/three.js/graphs/contributors>>. 27.1.2013. Luettu 29.3.2015.
- 22 Three.js. 2013. Verkkodokumentti. Mrdoob. <<https://github.com/mrdoob/three.js/>>. Luettu 29.3.2015.
- 23 Creating-a-scene.html. 2014. Verkkodokumentti. Mrdoob. <<https://github.com/mrdoob/three.js/blob/master/docs/manual/introduction/Creating-a-scene.html>>. 30.12.2014. Luettu 29.3.2015.
- 24 I will what I want. 2014. Verkkodokumentti. Under Armour. <<http://gisele.underarmour.com/>>. 2014. Luettu 3.4.2015.

- 25 FAQ. 2013. Verkkodokumentti. X3DOM. <[http://www.x3dom.org/?page\\_id=122](http://www.x3dom.org/?page_id=122)>. 2013. Luettu 12.1.2015.
- 26 Background: What is X3DOM, and what can it do for me?. 2013. Verkkodokumentti. X3DOM. <<http://doc.x3dom.org/gettingStarted/background/index.html>>. 2013. Luettu 12.2.2015.
- 27 Basic X3D Concepts: Nodes, Components and Profiles. 2013. Verkkodokumentti. X3DOM. <<http://doc.x3dom.org/gettingStarted/basicX3D/index.html>>. 2013. Luettu 12.2.2015.
- 28 Welcome to X3DOM. 2013. Verkkodokumentti. X3DOM. <<http://x3dom.org/docs-old/#welcome-to-x3dom>>. 2013. Luettu 12.2.2015.
- 29 3ds Max Export. 2013. Verkkodokumentti. X3DOM. <[http://x3dom.org/docs-old/tutorial/max\\_export.html](http://x3dom.org/docs-old/tutorial/max_export.html)>. 2013. Luettu 12.2.2015.
- 30 WebGL today. Just a hype or future of our industry. 2014. Verkkodokumentti. Merixstudio. <<http://www.merixstudio.com/blog/webgl-hype-future-digital-interactive/>>. 2014. Luettu 16.3.2015.
- 31 KONE in brief. 2013. Verkkodokumentti. KONE. <<http://www.kone.com/en/company/in-brief/>>. 2013. Luettu 16.12.2014.
- 32 Sissala, Mikko. Markkinointipäällikkö. KONE Oy. Espoo. 14.2.2015
- 33 Klemettinen, Niko. 2012. Teollisuuden 3D-mallin optimointi reaaliaikarenderöintiin. Insinööriyö. Jyväskylän Ammattikorkeakoulu
- 34 System requirements for Autodesk 3ds Max and Autodesk 3ds Max Design 2012. 2012. Verkkodokumentti. Autodesk. <<http://knowledge.autodesk.com/support/3ds-max/troubleshooting/caas/sfdcarticles/sfdcarticles/System-requirements-for-Autodesk-3ds-Max-and-Autodesk-3ds-Max-Design-2012.html>>. 2012. Luettu 16.12.2014.
- 35 Optimazing graphics performance. 2013. Verkkodokumentti. Unity 3D. <<http://docs.unity3d.com/Documentation/Manual/OptimizingGraphicsPerformance.html>>. 2013. Luettu 1.3.2015.
- 36 How to: Reduce Polygon Count in 3DS Max Easily. 2012 Verkkodokumentti. My Creative Daddy. <<http://mycreatedaddy.com/how-to-reduce-polygon-count-in-3ds-max-easily/>>. Luettu 27.2.2015.

