Joel Pöllänen

Rakennusten 3D-mallinnus ja mallien käyttäminen WebGL-sovelluksessa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

10.5.2015

Metropolia

Tekijä Otsikko Sivumäärä Aika	Joel Pöllänen Rakennusten 3D-mallinnus ja mallien käyttäminen WebGL-sovelluksessa 55 sivua + 2 liitettä 10.5.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaajat	Amanuenssi Petteri Kummala Yliopettaja Harri Airaksinen

Insinöörityön tavoitteena oli tutkia kolmiulotteisen rakennusmallinnuksen tekniikoita ja käytänteitä. Työssä selvitettiin, miten olemassa oleva rakennus voidaan mallintaa kolmiulotteiseksi kappaleeksi ja mitä kaikkea tulee ottaa huomioon, jotta sitä pystyttäisiin hyödyntämään selaimella toimivassa web-sovelluksessa. Pyrkimyksenä oli, että arkkitehtuurimuseolle mallinnettaisiin vesitorni käyttäen annettuja pohjapiirroksia ja kuvamateriaalia. Vesitorni oli tämän jälkeen upotettava WebGL-sovellukseen, jossa sitä pystyisi pyörittelemään ja tutkimaan.

Vesitorni mallinnettiin CAD-ohjelmalla käyttäen pohjapiirroksista jopa puolen millimetrin tarkkuudella mitattuja etäisyyksiä. Kuva-aineistoa hyödynnettiin vaikeasti hahmotettaviin kohtiin, ja torni pyrittiin mallintamaan mahdollisimman tarkasti vastaamaan todellisuutta. Tässä oli huolehdittava myös mallin geometrian tarkkuudesta, joka ei saanut olla liian suuri. Vesitornia varten luotiin materiaalit, joiden avulla lopullinen malli saatiin näyttämään realistisemmalta. Malli tuotiin CAD-ohjelmasta ulos sekä kokonaisena että leikattuna kahtia, jotta lopullisessa sovelluksessa niiden välillä pystyisi vaihtamaan. WebGL-sovellus luotiin käyttäen erityistä grafiikkakirjastoa, ja se upotettiin vesitornista kertovaan sivustoon.

Insinöörityössä saavutettiin kaikki asetetut tavoitteet, vaikka se kestikin ajallisesti suunniteltua pidempään. Lopputuloksena syntyi WebGL-sovellus sivustoineen, ja se liitetään osaksi arkkitehtuurimuseon omaa sivustoa. Vesitornin mallia pystyy pyörittämään, liikuttamaan ja suurentamaan hiirtä käyttäen, ja mallia pystytään vaihtamaan kokonaisen ja läpileikatun välillä.

Työn teko osoittautui monimutkaiseksi, mutta mielenkiintoiseksi prosessiksi, jonka tulokset hyödyttävät arkkitehtuurimuseota mahdollisissa vastaavissa projekteissa.

Avainsanat	WebGL, 3D, HTML5, Three.js, rakennusmallinnus

Author	Joel Pöllänen
Title	3D modeling of buildings and using the models in a WebGL application
Number of Pages	55 pages + 2 appendices
Date	10 May 2015
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructors	Petteri Kummala, Museum Assistant Harri Airaksinen, Principal Lecturer

The purpose of this thesis is to study the techniques and practices used in architectural 3D modeling of buildings. The goal was to investigate how an existing building can be modeled and what should be considered when using the model in a web application. The starting point of the project was that a museum of architecture had ordered a web application where the model of a water tower could be rotated and examined.

The water tower was modeled in a CAD software using old blueprints and photographs as the only reference material. The drawings were measured with an accuracy of half a millimeter. The photos were used when the geometry of the building was hard to understand. The tower was modeled to resemble the actual water tower as precisely as possible. However, it was important to pay attention to the resolution of the model so that it would not be too heavy for the application. All the materials were made as realistic as possible. The final tower models were exported from the CAD software, and a WebGL application was created using an additional graphics library. The final application was embedded in a web site with additional information about the water tower.

All set goals were met during this project and the results are clear. The final product is a visual WebGL application that can be found on the web site of the museum of architecture as its own subsite in the future. The model in the application can be rotated, shifted and zoomed in and out with a mouse and it is possible to switch between a full view and a sectional one.

The whole task took longer than planned, in total 140 hours. It was a complex but interesting process, and the model will be helpful in perceiving the structure of the tower, especially after its deconstruction. The museum of architecture will use the results in their possible future projects as this task gathered information on the required resources and how the development process should be carried out as a whole.

Keywords	WebGL, 3D, HTML5, Three.js, building modeling

Sisällys

Lyhenteet

WebGL	Web Graphics Library. JavaScript-ohjelmointirajapinta, jonka avulla pystytään renderöimään 2D- ja 3D-grafiikkaa moderneihin selaimiin.
CAD	Computer-aided Design. Tietokoneavusteinen suunnittelu, jota käytetään etenkin arkkitehtuurissa ja tuotesuunnittelussa.
BIM	Building Information Model. Rakennuksen tietomalli, joka sisältään rakennuksen geometrian lisäksi tietoa sen kaikista elementeistä ja koko elinkaaresta.
XML	Extensible Markup Language. Merkintäkielen standardi, jolla tiedon merkitys on luettavissa muun tiedon keskellä.
OBJ	Wavefront Object file. Yksinkertainen tiedostomuoto kolmiulotteisen geometrian pakkaukseen.
DWF	Design Web Format. Autodeskin kehittämä tiedostomuoto CAD-mallin tehokkaaseen pakkaamiseen.
GI	Global illumination. Joukko 3D-grafiikassa käytettyjä algoritmeja, joilla pystytään luomaan realistisempia valaistuksia, muun muassa ottaen huomioon valonsäteiden kimpoamisen.
IES	Illuminating Engineering Society. Tiedostomuoto valon profiilin pakkaamiseen ja jakeluun.
HTML	Hypertext Markup Language. Yleisesti verkkosivujen koodauksessa käytetty merkintäkieli, jonka tarkoituksena on kuvata hyperlinkkejä sisältävää tekstiä.
CSS	Cascading Style Sheets. Tyyliohje, jolla WWW-dokumentin ulkonäköä voidaan muokata.

Ajax	Asynchronous JavaScript And XML. Joukko web-tekniikoita, joilla tietoa voidaan vuorovaikutteisesti vaihdella eri palvelimien välillä, ilman sivujen uudelleenlatausta.
SVG	Scalable Vector Graphics. Kuvauskieli, jonka tarkoituksena on tuottaa verkkosivuille 2D-vektorigrafiikkaa käyttäen XML-muotoista tietoa.
JSON	JavaScript Object Notation. Yksinkertainen ja avoin tiedostomuoto, ja kevyempi vaihtoehto XML-muodolle.
DOM	Document Object Model. Dokumenttioliomalli on tapa esittää HTML-, XHTML- tai XML-dokumentin rakenne, jossa kaikki elementit kuvataan omina olioina.

1 Johdanto

Insinöörityössä perehdytään rakennuksen mallinnuksen tekniikoihin. Tavoitteena on tutkia pienoismallien rakentamista perinteisesti käsin ja esittää mallintamisen nykyaikaisia digitaalisia tekniikoita monipuolisin esimerkein. Työssä myös käsitellään 3D-mallintamista yleisesti, mallinnusohjelmia ja muita huomioon otettavia asioita liittyen rakennuksen mittaamiseen ja mallinnukseen.

Valmista rakennuksen kolmiulotteista mallia pystytään esittämään monella eri tavalla. Näistä selaimessa toimiva WebGL (Web Graphics Library) -sovellus on hyvä ja toimiva vaihtoehto, ja tässä työssä pyritään selvittämään, minkälainen prosessi sovelluksen teko on. WebGL mahdollistaa interaktiivisten 3D-sovellusten käyttämisen nykyaikaisilla selaimilla, joihin ei tarvitse asentaa ylimääräisiä liitännäisiä. Sovelluksen tekoa helpottaa Three.js-grafiikkakirjasto, joka kuitenkin vaatii käyttäjältä ohjelmoinnin osaamista.

Insinöörityössä tehdään mallinnus Lauttasaaren vesitornista käyttäen aineistona vain pohjapiirroksia ja valokuvia. Tarkoitus on tutkia myös, miten malli viedään WebGL-sovellukseen, jossa käyttäjä voi sitä tutkia. Lauttasaaren vesitorni on Länsi-Helsingissa Lauttasaaren kaupunginosassa sijaitseva yli 50 vuotta vanha vesitorni, jolle annettiin purkupäätös vuonna 2013. Sovelluksen tilaaja on Suomen Arkkitehtuurimuseo, joka toivoo vesitornista web-sovellusta, josta pystyisi tutkimaan tornin rakennetta ja ulkonäköä. Sovellus olisi hyödyllinen varsinkin tornin purkamisen jälkeen, kun sitä ei pääsisi enää paikan päälle katsomaan.

Suomen Arkkitehtuurimuseo on arkkitehtuuriin keskittyvä erikoismuseo Helsingin keskustassa. Se perustettiin vuonna 1956, ja se on maailman toiseksi vanhin arkkitehtuurimuseo. Sen mittavaan kokoelmaan kuuluu pienoismalleja, valokuvia, piirroksia ja kirjasto, ja ne painottuvat 1900-luvun jälkeiseen arkkitehtuuriin. Se on myös ollut yhteistyössä ympäri maailmaa järjestettävissä näyttelyissä ja muissa hankkeissa. Arkkitehtuurimuseo toimii Kaartinkaupungissa uusrenessanssityylisessä rakennuksessa, joka on vuodelta 1899 ja jo itsessään arkkitehtonisesti mielenkiintoinen. Vesitornille tehtävä sivusto liitetään myöhemmin osaksi Arkkitehtuurimuseon omaa sivustoa. [1.]

Valitsin aiheen sen takia, että WebGL ja muut web-tekniikat kiinnostavat ja minulla on niistä aikaisempaa kokemusta. Lauttasaaren vesitorni on myös rakennuksena kiinnos-

tava, ja senaikaista historiaa Helsingistä on mukava tutkia. Työn tekeminen myös laajentaa omaa tietämystäni 3D-mallinnuksesta ja web-sovellusten teosta. Työ pyrkii kartoittamaan suositeltavia käytänteitä mallintamisessa ja selvittämään ratkaisuja vastaantuleviin ongelmiin. Tarkoitus on, että tästä olisi hyötyä vastaavanlaista projektia tekeville ja sellaisille, joilla ei ole paljon kokemusta kolmiulotteisesta mallintamisesta ja Three.js-sovellusten teosta. Insinöörityöraportissa en kuitenkaan opeta 3D-mallintamista yksityiskohtaisesti, sillä työkalut ja toimintatavat vaihtelevat joka ohjelmassa.

2 Rakennusten mallintamisen tekniikat

2.1 Perinteinen rakennusmallinnus

Rakennusten mallintaminen on ollut aina osa arkkitehtuuria ja rakennusten suunnittelua. Tapahtuipa se sitten käsin tai tietokoneella, se on aina ollut välttämätön hyöty rakennusten ulkonäön ja niiden rakenteellisten kokonaisuuksien hahmottamiseen. Myös rakennusten sovittaminen ympäristöön on tärkeää, ja se on onnistunut parhaiten erillisten mallien avulla. Malleja voidaan myös rakentaa tai mallintaa jostain purettavista tai jo puretuista rakennuksista. Historiallisesta kohteesta voidaan myös rekonstruktoida malli, jotta saadaan käsitys siitä, miltä se on joskus näyttänyt.

Vaikka aikaisimmat löydetyt lähteet arkkitehtuurissa käytetyistä pienoismalleista ovat Roomasta ensimmäiseltä vuosisadalta jKr., on rakennuksista tehtyjä esineitä löydetty jo vuodelta 4600 eKr., vaikkakin niitä käytettiin pääosin vain koriste-esineinä. Fyysisen mallin rakentaminen on ollut paras tapa tuota rakennus eloon. Alun perin kun malleja käytettiin arkkitehtuurissa, se oli ainut tapa saada arkkitehdin visio viestitettyä rakentajille (ks. kuva 1). Malli toimi myös varmistuksena sille, että rakennettava rakennus toimisi muotonsa puolesta myös käytännössä. [2; 3.]

Kuva 1. Firenzessä sijaitsevan Santa Maria del Fiore -katedraalin kupolin pienoismalli 1400-luvulta [3].

Vasta myöhemmin arkkitehtuuripienoismalleja alettiin arvostaa niiden näyttävyyden takia, ja vaikka malleista oli edelleen hyötyä arkkitehdeille rakennusta suunniteltaessa ja rakennettaessa, oli mallien rakentamisen päätarkoitus muuttunut. Nykyään malli ensisijaisesti myy rakennuksen suunnitelmaa asiakkaille ja mahdollisille sijoittajille. Mallin avulla asiakas tai sijoittaja pystyy hahmottamaan rakennuksen ulkonäköä ja muotoa paremmin ja auttamaan sijoituspäätöksen teossa. Rakennuksen ulkonäöstä ja mahdollisista muutoksista pystytään myös helpommin keskustelemaan arkkitehdin kanssa. [2; 3.]

2.2 Rakennusmallinnus nykyään

Ennen tietokoneiden kehittymistä pienoismallien lisäksi myös pohjapiirrokset tehtiin aina käsin. 1900-luvun puolivälin jälkeen, kun tekniikka alkoi kehittyä, voitiin tietokoneita ensimmäistä kertaa hyödyntää myös pohjapiirrosten tekoon. Tämä tapahtui ensimmäisillä CAD (Computer-aided Design) -ohjelmilla, joissa algoritmeilla pystyi luomaan yksinkertaisia viivoja ja muotoja näytölle. Tietokoneiden käyttö ei ollut kuitenkaan vielä kovin suosittua tässä vaiheessa, ja kynä ja paperi pysyivät suosiossa vielä pitkään. 3D-mallien luominen uusilla CAD-ohjelmilla alkoi tulla mahdolliseksi vasta 1980-luvun jälkeen, jolloin tietokoneiden suosiokin alkoi nousta. 1980- ja 1990-luvulla alkoi tietokonei-

ta ilmestyä myös kouluihin, ja teolliset yritykset ja arkkitehdit alkoivat hyödyntää CAD-suunnittelua projekteissaan. [2; 4; 5.]

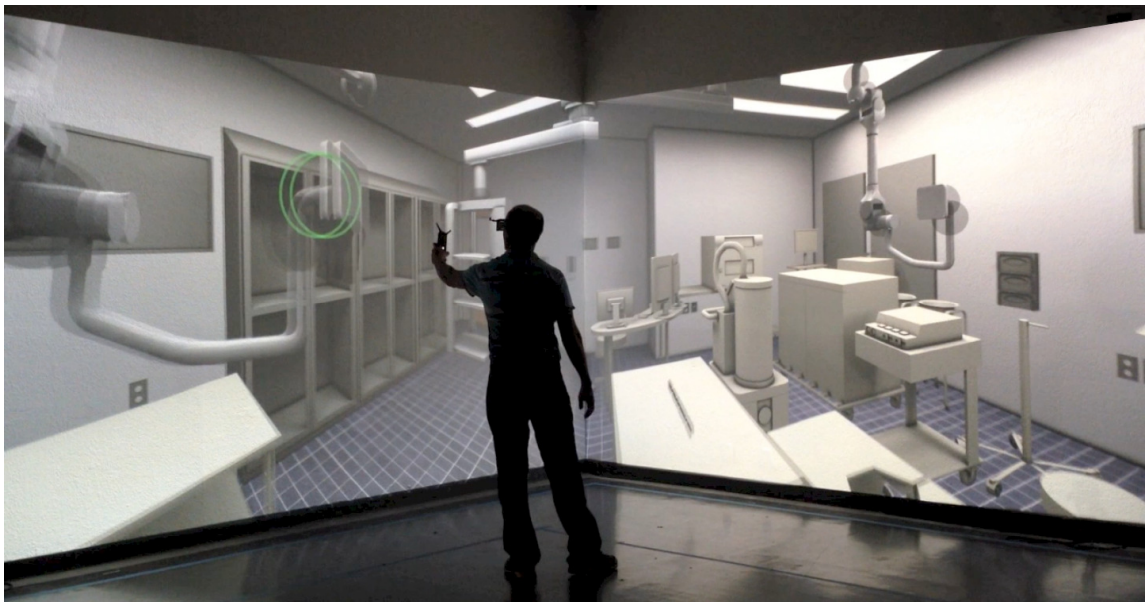
Nykyään käsin mallinnus ja piirtäminen ovat lähes kokonaan väistyneet tietokoneiden tieltä monenkin syyn takia. Pienoismallia rakentaessa joutuu varaamaan sille huomattavasti tilaa ja miettimään sen varastoimista jatkossa. Materiaalikustannuksetkin voivat nousta korkeiksi, samoin ajankäyttö, varsinkin hyvin monimutkaisissa ja realistisissa malleissa. CAD-ohjelmalla mallinnettaessa taas kaikki tapahtuu virtuaalisesti ja mallia pystytään muokkaamaan koska tahansa ilman kustannuksia. Jos mallin valmistuttua tahtoo muuttaa huoneen kokoa tai pinnan materiaalia, se onnistuu yleensä hyvinkin nopeasti verrattuna fyysiseen malliin. [2.]

Pienoismallilla on kuitenkin se etu, että se on paljon elävämpi, näyttävämpi ja havainnollisempi kuin virtuaalinen malli. Se voidaan kokea tavoilla, joita kaksiulotteisella näytöllä näkyvä malli ei pysty tarjoamaan. [2.] Juuri näiden syiden takia pienoismallien rakentamisesta ei ole kokonaan luovuttu. Etenkin harrastuksena pienoismallien rakentaminen on pysynyt vahvana.

Tietokoneella luotua mallia voi nykyisillä tekniikoilla tarkastella myös uusilla tavoilla, joissa malli pyritään tuomaan lähemmäksi todellisuutta. Mallin ympärille voidaan luoda todella realistinen tila ja ympäristö, johon voi olla reaaliaikaisesti vuorovaikutuksessa. Mallista voidaan myös luoda animaatioita, joissa se nähdään kuin osana reaalimaailmaa.

3D-tulostus antaa mahdollisuuden hypistellä tietokoneella tehtyä mallia fyysisesti. Mallin muokkaukseen voidaan palata, minkä jälkeen sitä voidaan tulostaa uudelleen niin monta kertaa kuin haluaa. 3D-tulostimien halventuessa pian kenellä tahansa on mahdollisuus tulostaa itse mallintamiaan esineitä. Jos itse ei osaa mallintaa, voi malleja ladata ilmaiseksi myös internetistä.

Virtuaalitodellisuudessa taas tietokonesimulaatiolla voi kolmiulotteiseen maailmaan hypätä sisään ja saada siitä lähes realistisen vaikutelman. Tämä tapahtuu esimerkiksi käyttäen laseja, joissa eri sensoreilla ja syvyysvaikutelmalla luodaan aistimuksia, jotka vastaavat todellisuutta. Esimerkiksi WorldVizin CornerCavern-järjestelmä (ks. kuva 2) on monipuolinen usean seinän kattava kokonaisuus, joka mahdollistaa interaktiivisessa 3D-mallissa liikkumisen. [6.]



Kuva 2. WorldVizin CornerCavern-virtuaalitodellisuusjärjestelmä [6].

Lisätyssä todellisuudessa kolmiulotteinen malli voidaan nähdä osana todellisuutta erityisten lasien tai vaikka puhelimen tai tabletin välityksellä. Erityisesti rakennusten ja muiden yksittäisten mallien katsominen lisätyssä todellisuudessa on kätevää, sillä se antaa pienoismallimaisen vaikutelman kohteesta. Mallin ympärillä voi kävellä ja katsoa sitä eri suunnista, ja joissain tilanteissa siihen voi jopa olla vuorovaikutuksessa.

2.3 3D-mallinnus ja visualisointi

Kolmiulotteinen malli on tietokoneella mallinnettu tai luotu objekti, jota pystytään käyttämään todella moniin eri käyttötarkoituksiin. 3D-malli voidaan luoda monilla eri tavoin, riippuen usein mallin lopullisesta käyttökohteesta ja siitä, millä tavoin työ saadaan hoidettua mahdollisimman laadukkaasti, nopeasti ja halvalla. Mallin yksittäisiä pisteitä voidaan esimerkiksi lisätä ja poistaa yksitellen käsin tai automaattisesti käyttäen siihen tarkoitettuja laitteita ja ohjelmistoja. Mallin geometria voidaan kopioida reaalimaailmasta erityisillä skannereilla, tai se voidaan generoida omanlaisekseen käyttäen sitä varten tehtyjä algoritmeja.

CAD- ja mallinnusohjelmistoja ja rakennusten mittaustapoja on paljon erilaisia, ja käyn niitä läpi luvussa 2.5. Projektissa mallinsin melko monimutkaisen 3D-mallin vesitornis-

ta, mikä ei onnistunut automaattisilla mallinnustavoilla vaan vaati paljon yksittäisten pisteiden siirtelyä.

Varsinkin rakennus- ja tuotesuunnittelussa 3D-mallit ovat isossa osassa. Nykyään kaikki teolliset tuotteet luodaan ensin kolmiulotteiseksi malliksi, jonka mukaan tuote sitten valmistetaan. Mallin avulla pystytään havainnoimaan tarkasti, miltä tuote näyttää, ja simuloimaan esimerkiksi siihen kohdistuvia voimia tai eri komponenttien liikkumista.

3D-mallien muita yleisimpiä käyttökohteita ovat seuraavat:

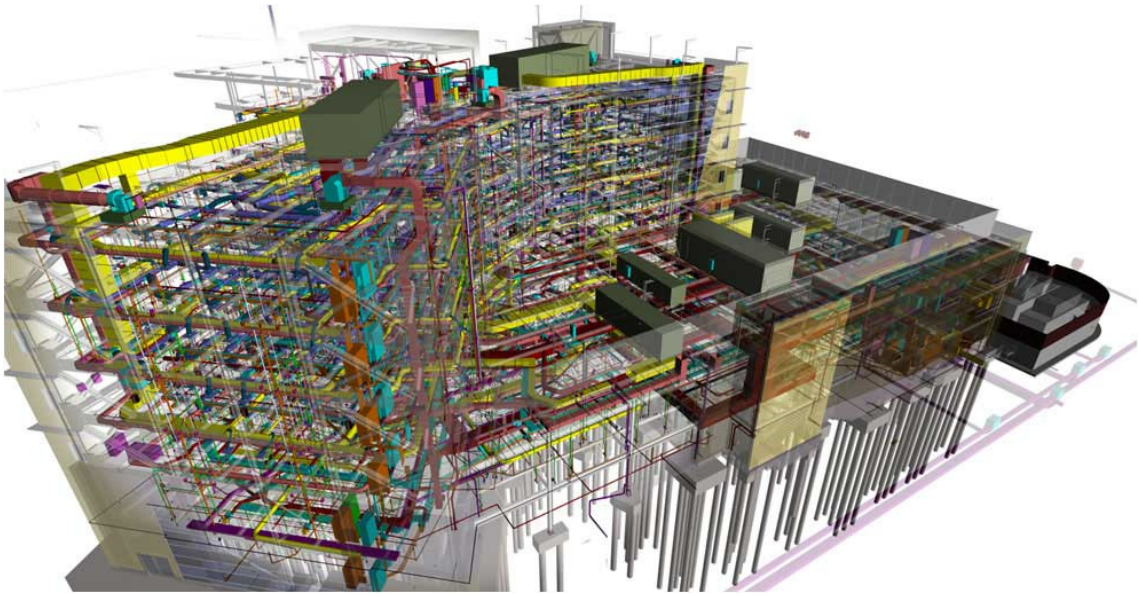
- Mainonnassa käytetään nykyään yhä useammin 3D-mallinnusta. Monesti mainoksissa näkyvät tuotteet luodaan jopa kokonaan tietokoneella. Tämä pätee niin televisio- kuin lehti- ja internetmainoksiin.
- Myös eri tiedotusvälineet käyttävät usein 3D-visualisaatioita viestin perille vientiin.
- Viihdeteollisuudessa 3D-visualisaatioiden käyttö on todella yleistä. Elokuvissa, televisiosarjoissa ja peleissä 3D:tä näkyy lähes kaikkialla. 3D mahdollistaa sellaisten maailmojen luonnin, joita ei todellisuuteen saisi rakennettua.
- Erilaiset virtuaaliset opetussimulaattorit hyödyntävät 3D-visualisointia, ja niillä pystytään harjoittelemaan vaativia tehtäviä turvallisesti. Tällaisia simulaattoreita on esimerkiksi armeijalla.
- Tieteellisissä tutkimuksissa ja lääketieteessä 3D-malleilla pystytään simuloimaan ja visualisoimaan tilanteita, joita ei muuten pystyisi näkemään. Esimerkiksi CERN-tutkimuskeskuksessa käytetään simulaatioita hiukkasten törmäyksistä. [7.]
- Lääketieteessäkin 3D-tulostaminen on yleistymässä. Monimutkaisia proteeseja pystytään tulostamaan halvalla, ja ne sopivat potilaalle täydellisesti. Uusimpina innovaatioina on jopa erilaisten sisäelimien tulostaminen kantasoluista. [8.]

2.4 CAD- ja BIM-ohjelmat

Tietokoneiden kehityttyä ja CAD-ohjelmien tullessa arkkitehtien ja rakennussuunnittelijoiden käyttöön alkoi rakennusten suunnitteluprosessi mullistua. Tietokoneiden myötä suunnitelmista tuli virheettömämpiä ja laadukkaampia ja kokonaisuuden pystyi näkemään paljon paremmin. Vaikka muutos tapahtui hitaasti luultavasti uuden oppimisen karttamisen takia, se oli kuitenkin selvä suunta tehokkaampaan suunnitteluun ja laa-

dukkaampiin lopputuloksiin. [9.] 2D- ja 3D-CAD-ohjelmat kehittyivät entisestään oliopohjaisiksi, mikä tarkoitti, että esimerkiksi ovet, seinät, ikkunat ja kokonaiset tilat pystyivät geometrisen mallin lisäksi pitämään sisällään niille ominaisia tietoja esimerkiksi materiaalista, massasta ja hinnasta. Tämän jälkeen huomattiin, että monia asioita, kuten rakentamisaikatauluja, pystyisi automatisoimaan näiden tietojen perusteella. CAD-ohjelmat edelleen suuntautuivat erityisesti mallien graafisen ulkoasun suunnitteluun eivätkä rakennustiedon moniulotteisempaan hallintaan, joten tietomallia oli kehitettävä uuteen suuntaan. [10, s. 1–2.]

Rakennuksen tietomallin (BIM) ajatus on ollut olemassa jo 1970-luvulta saakka [11], mutta käytännössä sen suosio kasvoi vasta vuonna 2003, kun Autodesk julkaisi aiheesta white paper -tiedotteen. Tietomallin perusajatuksena on, että CAD-ohjelmien tavoin se sisältää rakennuksesta kolmiulotteisen mallin, jonka lisäksi tietokantaan on tallennettuina kaikki rakennuksen ja sen elementtien tiedot, kuten materiaalit, mitat ja määrät. Ne luovat rakennushankkeen aikataulujen, rahoituksen ja muiden prosessien tietojen kanssa yhdessä kokonaisuuden, jota pystyy visualisoimaan todella monella eri tavalla. Pelkkien pohjapiirrosten lisäksi rakennuksesta voidaan esittää kuvia ja kaavioita minkä tahansa tietojen perusteella (ks. kuva 3). Rakennuksen tiedoista voi myös luoda simulaatioita, jotka auttavat osaltaan tilojen ja elementtien jatkosuunnittelua. [10, s. 2; 12.]



Kuva 3. Rakennuksen elementtejä korostettuina eri väreillä tietomallissa [13].

Tiedot ovat tietokannassa myös aina helposti muokattavissa siten, että ne haluttaessa vaikuttavat etenevästi myös muihin tietoihin. Toisin kuin CAD-mallin oliopohjaisissa tiedoissa, tietomallin relaatiotietokannan tietojen välille voidaan luoda suhteita. Rakennusprojektin kaikki osapuolet ja suunnittelijat voivat muokata yhteistyössä rakennuksen tietoja ja niiden suhteita, mikä tekee työstä sujuvaa ja tehokasta. Suunnittelua voidaan tehdä kunkin tekijän mieltymyksen mukaan käyttäen haluttua näkymää, esimerkiksi graafisista käyttöliittymää, taulukkoa, kaaviota tai isometristä kuvaa rakennuksen geometriasta. [10, s. 2–3.]

Kaiken kaikkiaan tietomallin käyttö on erityisen hyödyllistä silloin, kun rakennussuunnittelussa halutaan ottaa kaikki hankkeen vaiheet ja tiedot huomioon ja kun tekijöitä on paljon. Prosessin hallinta yhden tietokannan ja mallin ympärillä on tällöin huomattavasti helpompaa kuin esimerkiksi pelkän CAD-suunnitelman ja -mallin kanssa. Tämä kaikki vaikuttaa kuitenkin loppujen lopuksi hankkeen laatuun, nopeuteen ja hintaan, sillä kaikki aika ja työ, mitä käytetään dokumentointiin ja suunnitelmien hallintaan, kuluttaa siitä ajasta, joka käytetään itse rakennuksen suunnitteluun ja rakentamiseen. [10, s. 7.]

Kun kyseessä ei ole iso rakennusprojekti, vaan yksinkertaisempi kohde, kuten tuotteen mallin suunnittelu tuotantoa varten, on kätevää käyttää CAD-ohjelmaa. Tällöin tärkeintä on mallin geometria, mittasuhteet ja materiaalit ja niiden suunnittelu visuaalisesti. Toisin kuin tavallisilla 3D-mallinnusohjelmilla, CAD-ohjelmissa päätarkoituksena ei ole saada mallista visuaalisesti näyttävää tai realistista. CAD keskittyy lähes ainoastaan mallin tarkan geometrian luomiseen ja sen hyödyntämiseen rakennusten ja tuotteiden suunnittelussa ja kehityksessä. [14, s. 3–4.]

On kuitenkin hyvä, jos suunnitellusta kolmiulotteisesta CAD-mallista pystyy ohjelmassa luomaan edes hieman visuaalista näyttöä. Tämä tehdään yksinkertaisimmillaan siten, että pinnoille asetetaan halutut värit. Tarkempaan lopputulokseen tosin pääsee, kun pinnan materiaali luodaan itse tai valitaan valmiista vaihtoehdoista, esimerkiksi tiili-, puu- tai kivipinnoista. Mallin ympärille tai sisälle voi myös luoda esimerkiksi ihmisistä tai puista rekvisiittaa, joka luo kokonaisuuteen todentuntua. Rakennusta voidaan tämän jälkeen esitellä asiakkaalle kuvan muodossa tai animaatiolla, joka voi olla esimerkiksi kamera-ajo rakennuksen läpi. Interaktiivinen esityskin on mahdollinen; siinä voisi vaikka itse kulkea ihmisenä rakennuksen sisällä.

Tarkat pohjapiirrokset ja muut tekniset piirustukset ja kaaviot ovat välttämättömiä apuja kommunikoitaessa projektin tekijöiden kesken. Tällöin mallin näyttävyydellä ei ole väliä, vaan on tärkeää, että piirrokset ovat selkeitä ja niissä näkyy kaikki oleellinen tieto. CAD-ohjelmat tarjoavat työkaluja tätä varten, mikä nopeuttaa työn kulkua ja dokumentointia. Ohjelmat voivat tarjota myös analysointityökaluja ja projektinhallintaan liittyviä palveluja, ja monet tarjoavat myös ladattavia laajennuksia sekä lisäominaisuuksia esimerkiksi renderöinnin eli kuvantamisen puolelta. [14, s. 4.]

Kun rakennuksen mallia halutaan esimerkiksi hyödyntää markkinoinnissa tai hankkeen myymisessä, se halutaan visuaalisesti näyttäväksi ja mahdollisesti upotettuna realistiseen ympäristöön. Tällöin on otettava myös muita työkaluja käyttöön. CAD-ohjelmista saa vietyä mallin geometrian monessakin eri muodossa 3D-mallinnusohjelmiin, joissa mallin ympärille voi luoda ympäristön, asettaa materiaalit ja valonlähteet, ja lopuksi renderöidä lähes fotorealistisen kuvan, siitä miltä se todellisuudessa voisi näyttää.

2.5 Jo rakennetun rakennuksen mallinnus

Olemassa oleva yksittäinen rakennus voidaan haluta mallintaa kolmiulotteiseksi kappaleeksi monestakin syystä. Useimmiten rakennus on vanha ja arvokas, ja se halutaan säilyttää digitaalisesti myös jälkipolville. Vanhasta rakennuksesta ei ole rakennettaessa koskaan tehty tietokoneella mallia, ja mahdollinen pienoismallikaan ei ole säilynyt. Rakennus voi myös olla arkkitehtonisesti mielenkiintoinen ja tämän takia hyvä mallinnuksen kohde. Monesti ylläpitoa ja hallintaa varten voidaan mallintaa rakennus tai kiinteistö, josta ei ole vielä tehty CAD- tai tietomallia. Mallia hyödynnetään rakennuksen korjauksessa ja tulevien hankkeiden suunnittelussa sekä lopulta myös sen purkamisessa. [15.]

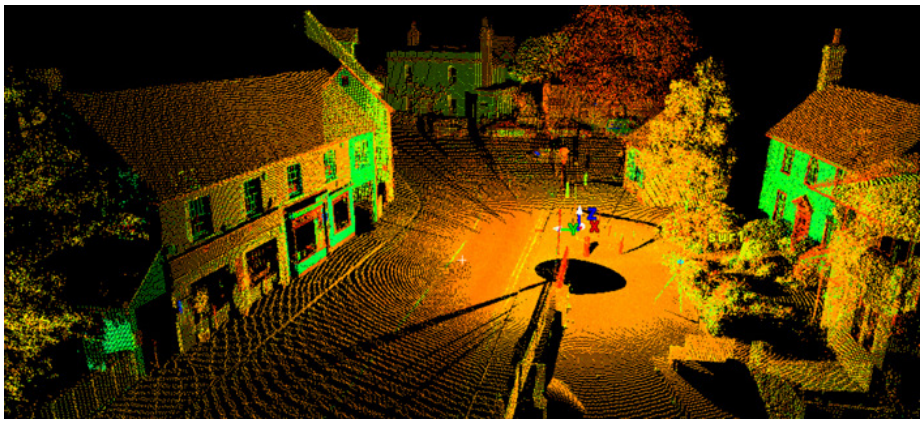
Yhden rakennuksen sijaan voidaan myös haluta mallintaa ja kartoittaa kokonainen naapurusto tai jopa kaupunki. Tämä yleensä tehdään eri tavalla kuin yksittäisen rakennuksen mallinnus, ja se vaatii myös kalliimpia laitteita ja menetelmiä. Isomman mittakaavan mallia voidaan käyttää muun muassa kaupunkisuunnitteluun, metsänhoitoon, navigaattoreihin ja karttapalveluihin. Mallista voi myös luoda kuvia, animaatioita ja interaktiivisia esityksiä, joita voi käyttää esimerkiksi kohdealueen mainontaan. [16.]

Mittaustavat

Jotta kolmiulotteinen malli voidaan tehdä kohteesta, siitä on oltava referenssimateriaalia ja mittaustuloksia. Mitä enemmän tätä materiaalia on ja mitä tarkempaa se on, sitä tarkemman mallin pystyy mallintamaan. Yksinkertaisimmillaan rakennuksesta tehtävä malli on kuutio tai muuten todella yksinkertainen malli, joka suurin piirtein noudattaa rakennuksen muotoja. Kokonaista kaupunkia mallinnettaessa pyritään huomioimaan vain rakennusten isoimmat piirteet. Tämä on tärkeää, ettei mallista tule turhan monimutkaista ja raskasta. Yksittäisestä rakennuksesta tehty malli taas voi olla hyvinkin yksityiskohtainen, riippuen käyttötarkoituksesta.

Alueen maastokartoitus ja rakennusten mallinnus tehdään yleensä ilmakuvilla ja laserkeilauksella. Ne tehdään joko ilmasta käsin lentokoneella, miehittämättömällä lennokilla tai helikopterilla tai maan pinnalla autolla, joka pystyy keräämään tarkkaa tietoa myös rakennusten seinien muodoista ja kadulla olevista muista yksityiskohdista. Laserkeilain on laite, joka lähettää laserimpulsseja ympärilleen tietyn välimatkan välein. Lasersäteet osuessaan kohteeseen heijastuvat takaisin ja osuvat keilaimeen, joka on mitannut säteen kulkuajan. Tämän perusteella pystytään laskemaan laitteen ja kohteen välinen etäisyys. Kun kaikki mittaustulokset yhdistetään, saadaan niin kutsuttu pistepilvi, jossa kolmiulotteisessa avaruudessa hahmottuvat kohteen muodot. [16.]

Kuvassa 4 nähdään, kuinka paljon yksityiskohtia voi yhdestä muutaman minuutin kestävästä laserkeilauksesta saada skannattua. Kuvan rakennukset erottuvat selkeästi, mutta koska keilauksia on tehty vain yksi, jää rakennusten ja puiden taakse isoja katvealueita. [17.]



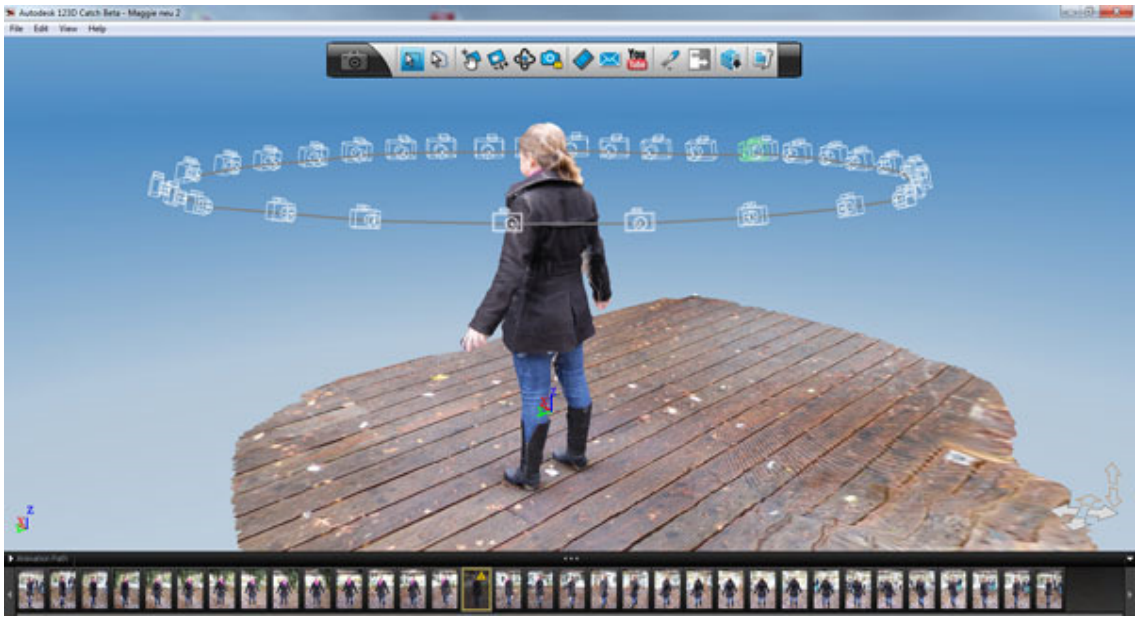
Kuva 4. Yhdestä laserkeilauksesta saatu pistepilvi [17].

Pistepilven jokainen piste vastaa yhtä etäisyyden mittausta, ja niiden tiheys voi laserkeilaimen tyypistä riippuen olla yli kymmenestä senttimetristä jopa alle millimetriin. Tämän jälkeen pistepilvi annetaan jollekin erityisohjelmalle, joka osaa pisteitä hyödyntäen luoda kohteesta halutunlaisen 3D-mallin. [18.]

Vastaavasti automaattinen muttei läheskään yhtä tarkka tapa mallintaa on mallin generoiminen kuvien perusteella. Google Earth -karttapalvelu tuki pitkään käyttäjien itse tekemiä 3D-malleja rakennuksista, mutta vuonna 2012 se ilmoitti siirtyvänsä automaattiseen mallinnustekniikkaan. Tekniikka perustuu kaupungeista otettaviin ilmakuviin, joita otetaan useasta eri kulmasta. Kuvista generoidaan kolmiulotteinen malli, johon tekstuuritkin pystytään asettamaan automaattisesti ilmakuvista sekä katunäkymistä (ks. kuva 5). Tämä on paljon nopeampaa kuin rakennusten yksittäinen mallinnus, ja mallien tarkkuus pysyy yhtenäisenä. [19.]

Kuva 5. Automaattisesti mallinnettuja rakennuksia Google Earthissä [19].

Tällaista tekniikkaa voi kuka tahansa hyödyntää syöttämällä kohteesta otettuja valokuvia tiettyihin ohjelmiin, jotka osaavat luoda niiden perusteella kolmiulotteisen mallin. Tällaisia ohjelmia ovat muun muassa Autodeskin *123D Catch (www.123dapp.com/catch)* ja yksinkertaisempi avoimeen lähdekoodiin perustuva *insight3d (insight3d.sourceforge.net/)*. Mallin luominen perustuu siihen, että kun kohteesta antaa riittävän montaa kuvaa eri kuvakulmista, ohjelma automaattisesti huomaa niissä olevat yhtäläisyydet ja pystyy laskemaan pisteiden sijainnin 3D-avaruudessa. Tästäkin saadaan tuotettua pistepilvi, jonka ohjelma osaa usein itse jäsennellä ja muuttaa käytettäväksi malliksi (ks. kuva 6). Kuvista saadaan pelkän geometrian lisäksi myös mallin tekstuurit, jotka sovitetaan automaattisesti. [20.]



Kuva 6. Ihmisen automaattinen mallinnus kuvien perusteella 123D Catchillä [21].

Monesti kalliita laitteita ei ole saatavilla yhden rakennuksen mallintamiseksi, eikä valokuvillakaan saa tarpeeksi laadukasta jälkeä. Rakennuksessa ei myöskään ole välttämättä mahdollisuutta päästä käymään, tai sitä ei ehkä ole enää olemassakaan. Tällöin parasta on tyytyä manuaaliseen työhön. Tämä oli tilanne myös aloittaessani omaa projektiani, jossa mallinnus tehtiin pelkästään pohjapiirrosten ja kuvien perusteella. Lisää projektin suunnittelusta on luvussa 3.2, ja rakennuksen mittaamisesta luvussa 4.1.

Kuvat ovat välttämättömiä oikeiden mittasuhteiden saamiseksi. Suunnittelu kannattaa aloittaa niiden ja pohjapiirrosten tutkimisella sekä rakennuksen tilojen hahmottamisella. Tämän jälkeen mallinnus aloitetaan pohjapiirrosten jäljittämisellä jossain CAD-ohjelmassa. Myös valokuvia kannattaa hyödyntää, koska aina pohjapiirrokset eivät pidä täysin paikkaansa, varsinkin jos rakennusta on remontoitu eikä uusia piirustuksia ole saatavilla.

Rakennusta mallinnettaessa on otettava huomioon mallinnuksen tarkkuus, mikä tarkoittaa sitä, kuinka monimutkainen mallista tehdään. Tämä riippuu pitkälti sen käyttötarkoituksesta, eikä yleispäteviä tarkkuusrajoja pysty sanomaan. Jos malli tulee käyttöön esimerkiksi interaktiivisissa visualisaatioissa, se ei saisi olla liian monimutkainen, jottei se hidastaisi tietokonetta liikaa. Jos mallista taas aiotaan renderöidä realistisia kuvia tai animaatioita, se voi olla yksityiskohtaisempi, mikä kuitenkin lopulta vaikuttaa myös renderöimisaikoihin.

Kuitenkin kaikissa tilanteissa pitäisi pitää huolta siitä, ettei mallissa olisi turhia pintoja ja pisteitä, koska monimutkaista geometriaa on aina hitaampi ja vaikeampi käsitellä. Monesti esimerkiksi pyöreisiin pintoihin saatetaan käyttää tarpeettoman paljon pisteitä. Pintojen materiaalilla voidaan myös erehdyttävästi esittää epätasaisuuksia tai läpinäkyviä kohtia, mikä säästää paljon mallin monimutkaisuudessa. Jos tarkkuutta ei tiedosteta, voi mallin yksityiskohtiin huomaamatta käyttää aivan liikaa aikaa, minkä jälkeen malli ei välttämättä ole edes käyttökelpoinen sen monimutkaisuuden takia.

Ohjelmistot

Ohjelmia on 3D- ja CAD-mallinnukseen todella paljon, eikä kaikkia ole tässä järkevää käydä läpi. Useimmat ohjelmat ovat erikoistuneet johonkin mallinnuksen osa-alueeseen tai menetelmään, mutta on myös monia ohjelmia, joissa on kaikki, mitä tarvitaan perusmallintamiseen, teksturoimiseen ja animoimiseen, ja joissa on jopa fysiikkamoottori pelinkehitystä varten.

CAD- ja BIM-ohjelmat eroavat toiminnoiltaan ja ulkoasultaan muista 3D-mallinnusohjelmista huomattavasti, sillä niillä on aivan eri käyttötarkoitukset. CAD- ja BIM-ohjelmat ovat mallinnuspuoleltaan ensisijaisesti suunnitteluohjelmia, joissa elementtien geometrinen malli, mittasuhteet ja ominaisuudet ovat tärkeimmässä osassa. Mallin näyttävyys ei tällöin ole olennaista. CAD-ohjelmissa mallinnus vastaa perinteistä teknisten piirrosten piirtämistä käsin, mutta tarkemmin ja virheettömämmin, ja kolmannen ulottuvuuden kanssa malli hahmottuu tekijälle aivan eri tavalla.

CAD-ohjelmat tarjoavat työkaluja moneen eri tarkoitukseen, ja ne helpottavat rakennusten mallintamista huomattavasti. Esimerkiksi seinät, ovet, ikkunat, portaat ja jopa huonekalut pystytään luomaan automaattisesti mittojen ja muiden määritysten perusteella. Tarkkojen mittojen määrittäminen on oikeaa kohdetta mallinnettaessa tärkeää, jotta kaikki elementit osuisivat kohdalleen ja malli vastaisi todellisuutta mahdollisimman tarkasti. Ohjelmissa on myös muita helpottavia ominaisuuksia ja työkaluja, joilla voi helposti luoda muun muassa suoria kulmia ja pyöreitä pintoja.

Yleensä CAD- ja BIM-ohjelmissa geometrian saa vietyä ulos muiden ohjelmien käyttöön monessakin eri tiedostomuodossa. Tiedostot sisältävät tiedot mallin geometriasta, materiaaleista ja muista mahdollisista asioista, kuten animaatioista ja fysiikasta. Yksi käytetyimmistä tiedostomuodoista on *COLLADA*, joka on Khronos Groupin kehittämä ja

perustuu XML (Extensible Markup Language) -rakenteeseen. Se on laajasti tuettu ja avautuu useimmilla mallinnusohjelmilla. Toinen laajasti käytetty tiedostomuoto on Wavefront *3D Object File* (OBJ). CAD-ohjelmista on mahdollista tuoda malli myös muodoilla, jotka ovat tyypillisiä CAD-malleille, sillä ne sisältävät myös muuta ei-geometrista tietoa. AutoCAD-ohjelman DWF (Design Web Format) -tiedostomuoto esimerkiksi toimii hyvin myös mallin viemiseksi muihin mallinnusohjelmiin. [22; 23.]

Otos käytetyimmistä CAD- ja BIM-ohjelmista ja ilmaisista vastineista:

- Autodeskin ensimmäinen tuote AutoCAD julkaistiin vuonna 1982, ja se on siitä asti ollut yksi tunnetuimmista ja suosituimmista CAD-ohjelmistoista. AutoCAD on maksullinen, mutta monipuolinen yleissuunnitteluohjelma 2D- ja 3D-vektoripohjaiselle grafiikalle. Siitä on olemassa useita eri versioita, jotka ovat erikoistuneet omiin suuntiinsa, sekä jopa mobiiliversio. Se on myös laajennettavissa moneen eri tarkoitukseen. [24; 25.] (www.autodesk.fi/products/autocad/)

- Progesoftin progeCAD on AutoCAD:lle tehty halvempi vastine, joka kuitenkin käyttää samaa tiedostoformaattia ja sisältää väitetysti myös samat ominaisuudet [26]. (www.progesoft.com/)

- ArchiCAD on Graphisoftin kehittämä arkkitehtisuunnitteluun keskittynyt maksullinen CAD- ja BIM-ohjelma, joka tarjoaa rakennussuunnittelun lisäksi muun muassa hankesuunnittelua ja kiinteistönhallintaa. ArchiCAD antaa monipuoliset mahdollisuudet myös visualisoinnille animaatioilla ja lähes realistisilla renderöinneillä. [27.] (www.graphisoft.com/archicad/)

- Autodesk Revit on rakennussuunnitteluun keskittynyt BIM-ohjelma, joka tarjoaa lukuisia työkaluja analysointiin, dokumentointiin ja visualisointiin. Revitin omalla renderöijällä pystyy rakennuksista luomaan myös näyttäviä kuvia. [28.] (www.autodesk.fi/products/revit-family/)

- Trimble SketchUp on helppokäyttöinen CAD-ohjelma, jonka ilmaislisenssillä CAD-suunnittelun opettelu onnistuu nopeasti. Työkaluja ja ominaisuuksia pystytään lisäämään sijoittamalla melko halpaan SketchUp Pro -lisenssiin, jolla rakennuksen luonnostelu, suunnittelu ja visualisointi onnistuvat jo kätevästi. Lisäosat ja mallien jakaminen 3D Warehousessa ovat myös iso osa ohjelmaa. [29.] (www.sketchup.com/)

- Ilmaisista ja avoimeen lähdekoodiin perustuvista CAD-ohjelmista on esimerkkinä FreeCAD, josta löytyy kaikki tarvittava perus 3D-CAD-suunnitteluun. Lisäosilla laajennettava ohjelma tukee monipuolisesti eri formaatteja ja sopii niin esineiden kuin rakennustenkin monipuoliseen suunnitteluun. Mutta esimerkiksi animointia tai näyttävää visualisointia sillä ei pysty tekemään. [30.] (www.freecadweb.org/)

Raja CAD-suunnitteluohjelmien ja 3D-mallinnusohjelmien välillä on melko häilyvä. Useimmat CAD-ohjelmat ovat selvästi suunnittelua varten ja useimmat mallinnusohjel-

mat visualisointia varten. Kuitenkin on myös ohjelmia, joilla pystyy molempiin, jos ei suoraan niin ainakin lisäosien avulla. Mallinnusohjelmissa tärkeintä kuitenkin on se, että malli saadaan materiaaleineen hyvän näköiseksi, sille voidaan luoda tila tai ympäristö, jossa voi olla lukuisia muita objekteja ja että siitä voidaan renderöidä kuvia tai animaatioita.

Mallintaminen voidaan tehdä todella monella eri tavalla mallinnusohjelmissa. Perinteisesti mallintaminen tehdään vertex-pisteiden ja pintojen luomisella, joita voi lisätä yksitellen tai työkaluja käyttämällä. Monesti malli aloitetaan luomalla jokin primitiivi muoto, kuten kuutio, pallo tai sylinteri, josta pisteitä siirtelemällä ja muun muassa *extrude*- ja *bevel*-työkaluilla saadaan luotua uusia pintoja ja pisteitä. Kaarevat pinnat ja käyrät vaativat monesti paljon pisteitä ja ovat täten raskaita, mutta erityisillä työkaluilla niitä pystyy luomaan kätevästi Bézier- tai B-splinikäyristä [31].

Henkilöiden, eläinten ja muiden hahmojen mallinnuksessa on suosittua käyttää niin kutsuttua digitaalista veistämistä, jossa mallia pystyy muovailemaan kuin savea. Tällainen ominaisuus on monessa mallinnusohjelmassa, mutta jos muovailusta haluaa saada kaiken hyödyn, kannattaa käyttää siihen erikoistuneita ohjelmia, kuten Zbrush tai Mudbox. Digitaalisella veistämisellä pystyy todella kätevästi muotoilemaan mallin ja luomaan siihen eloa yksityiskohdilla ja realistisilla materiaaleilla. Rakennusten tai tilojen suunnittelussa tästä ei kuitenkaan paljoa hyötyä ole, ellei realistisia hahmoja halua sijoittaa sinne rekvisiitaksi. [32.]

Automaattiset mallinnustavat säästävät huomattavan paljon aikaa ja pystyvät tuottamaan tuloksia, joita ei käsin välttämättä pystyisi mallintamaankaan. Monissa ohjelmissa on sisäänrakennettu skriptikieli, jolla mallinnusta, animointia, teksturointia ja renderöintiä pystytään tehostamaan ja kontrolloimaan halutulla tavalla. Skriptit myös mahdollistavat asioita, jotka olisivat käsin vaikeita, elleivät jopa mahdottomia tehdä. Esimerkiksi proseduraalisella generoimisella pystyy sisältöä, kuten objekteja, luomaan jonkin halutun algoritmin perusteella. [33.] Rakennusmallinnuksessa skriptejä voisi käyttää esimerkiksi rakennusten tai huoneiden luontiin automaattisesti tiettyjen sääntöjen mukaan. Todellisia esineitä mallinnettaessa taas 3D-skannaus on menetelmä, joka tehostaa prosessia tehden sen myös tarkemmin.

3D-mallinnusohjelmia on todella monipuolinen tarjonta, ja niitä on niin ilmaisia kuin todella kalliitakin. Usein ilmaisilla ohjelmilla pärjää jo pitkälle, eikä ole tarvetta käyttää

omaisuuksia maksullisiin ohjelmiin, ellei niissä ole jokin ominaisuus, joka on välttämätön omaa projektia varten. Ohjelmiin on myös mahdollista ostaa tai ladata ilmaiseksi lisäosia, jotka tarjoavat lisää ominaisuuksia. Usein maksullisilla ohjelmilla on tarjottavanaan hyvä tekninen tuki ja muita palveluita, kuten pilvitallennuspalvelu.

Käytetyimpiin 3D-mallinnusohjelmiin kuuluvat seuraavat:

- Autodesk 3ds Max on yksi laajimmin käytetyistä 3D-mallinnusohjelmista, ja sitä käytetään peliteollisuudessa, elokuvissa ja arkkitehtuurissa. Se sisältää kaikki tarvittavat työkalut kolmiulotteiseen mallinnukseen, animointiin, teksturointiin, renderöintiin ja moneen muuhun. 3ds Max on melko iso sijoitus harrastelijalle, mutta ammattimaiseen käyttöön se soveltuu hyvin monipuolisine ominaisuuksineen ja laajennuksineen. [34.] (www.autodesk.fi/products/3ds-max/)

- Yhtä lailla monipuolinen mallinnusohjelma on Autodesk Maya, joka sisältää paljon samoja laadukkaita työkaluja ja ominaisuuksia kuin 3ds Max. Mayalla on kuitenkin myös vahvuuksia muun muassa animoinnissa, skriptaamisessa, renderöimisessä ja riggaamisessa eli luurangon luomisesssa hahmolle. 3ds Maxissa vastaavasti on vahvuuksia mallinnuksessa ja arkkitehtuurisessa suunnittelussa. Toisin kuin 3ds Max, Maya on saatavilla Windowsin lisäksi myös Linuxille ja OSX:lle. [34.] (www.autodesk.fi/products/maya/)

- Ilmaisista 3D-mallinnusohjelmista Blender on suosituin ja ehkä laadukkain vaihtoehto. Se sisältää suurimman osan ominaisuuksista, joita muut kalliit ohjelmat sisältävät, mukaan lukien animoinnin ja pelinkehityksen, ja se on oiva tapa tutustua 3D-mallintaan. Sen ympärillä on laaja yhteisö, joka tarjoaa ohjeita ja tutoriaaleja, joten sisäänpääsy on helppoa. [35.] (www.blender.org/)

- Lightwave on NewTekin kehittämä vaihtoehto Autodeskin tuotteille, mutta sitä harvemmin käytetään isommissa tuotannoissa. Lightwavea on käytetty kuitenkin useissa elokuvissa, ja sen kattavat ominaisuudet, tehokkuus ja hyvä tuki tekevät siitä hyvän kilpailijan. [36.] (www.lightwave3d.com/)

- Cinema 4D on Maxonin kehittämä ammattikäyttöinen ohjelma, jota on laajasti käytetty etenkin elokuvien teossa. Se on pysynyt vahvana kilpailijana muille mallinnusohjelmille muun muassa sen vakauden ja nopeuden takia. Cinema 4D on myös tehty hyvin yhteensopivaksi muiden ohjelmistojen kanssa. [35.] (www.maxon.net/products/cinema-4d-studio/)

3D-mallien tuotannossa on osana myös useita muita ohjelmia, jotka eivät suoranaisesti liity kolmiulotteiseen mallintamiseen. Niitä voivat olla esimerkiksi renderöijät, kuvankäsittelyohjelmat, videoeditointiohjelmat, tekstieditorit, pelinkehitysalustat ja eri laitevalmistajien omat erityisohjelmat. 3D-tuotanto koostuu monesta eri vaiheesta riippuen siitä, mitä halutaan saavuttaa, ja usein se ei onnistu pelkästään yhdellä tai kahdella-

kaan ohjelmalla. Onneksi eri tekijöiden eri ohjelmat ovat yleensä kuitenkin hyvin yhteensopivia keskenään, koska niissä on standardoidut tiedostomuodot ja käytänteet.

Ohjelmistojen väliset tiedonsiirrot voivat olla monesti myös ongelmallisia. Kolmiulotteisille malleille ja niiden materiaaleille on useita eri tiedostomuotoja, joiden käytössä ei kaikki aina suju niin kuin pitää. Jotkin ohjelmat saattavat tulkita jotain tiedostomuotoja eri tavalla kuin toiset. Ne voivat tuottaa isoja ongelmia, jos tiedostojen rakennetta ei ymmärrä eikä virheitä osaa korjata. Ongelmia voivat tuottaa muun muassa seuraavat asiat: liian pitkät liitettävien tiedostojen nimet, koordinaattiakseleiden suunnat, valojen sijainnit, primitiivimuodot ja pintojen suunnat sekä niiden kaksipuolisuus. Toki tiedostomuotoja kehitetään koko ajan ja ongelmista pyritään pääsemään eroon, mutta se vie aikaa.

Omassa projektissani käytin monia eri ohjelmistoja ja työkaluja. Mallinnukseen käytin ensisijaisesti Trimble SketchUp -ohjelmaa, jolla kaikki vesitornin elementit sai tarkasti oikean mittaisiksi. SketchUpista toin mallin COLLADA-muodossa Blenderiin, jossa pystyin korjailemaan geometriaa ja sen yksittäisiä pisteitä paremmin sekä asettamaan materiaalit. Tekstuurien tekoon taas käytin Adobe Photoshopia, koska se on tuttu ja monipuolinen kuvankäsittelyohjelma. Blenderistä vein mallin .obj-tiedostomuodossa ulos ja käytin three.js:n omaa Pythonilla toimivaa konvertteria, joka muuntaa .obj-tiedoston ja .mtl-materiaalitiedoston .js-muotoon, jotta se toimii hyvin lopullisessa WebGL-sovelluksessa. Itse sivuston ja sovelluksen rakensin käyttäen Notepad++-tekstieditoria, joka taas toimii hyvin monilla eri kielillä ohjelmoimiseen. Lisää projektin tekemisestä ja vastaan tulleista ongelmista käyn läpi luvussa 4.

Materiaalit

Materiaalit ja tekstuurit ovat tärkeässä osassa kolmiulotteisen mallin tekoa, etenkin kun siitä tahdotaan visuaalisesti näyttävä. Oikeanlaisilla tekstuureilla mallista saadaan mielenkiintoinen ja uskottava. Jos tekstuureihin ei käytä tarpeeksi vaivaa, voi malli näyttää epärealistiselta. Nykyään realistisia tekstuureja saa luotua lähes mille materiaalille tahansa ja, internetin myötä valmiita tekstuureja voi helposti myös ostaa tai ladata ilmaiseksi laajoista tekstuurikirjastoista.

Tekstuuri voidaan luoda usealla eri tavalla. Se yleensä riippuu itse materiaalista ja siitä, kuinka isossa osassa se tulee olemaan lopullisessa tuotoksessa ja kuinka läheltä sitä

tullaan katsomaan. Sen tekotavat voidaan karkeasti jakaa kolmeen eri menetelmään, vaikka niitä kaikkia voi yhdistelläkin. Ensimmäinen menetelmä on digitaalinen kuvankäsittely tai maalaus, jolla taiteilijat pystyvät virtuaalisesti luomaan todella realistisiakin tekstuureja. Tähän voidaan myös yhdistää digitaalisia efektejä. Toinen menetelmä on tekstuurin proseduraalinen generoiminen annettujen algoritmien perusteella. Tällä tavoin pystyy luomaan satunnaisia ja sääntöihin perustuvia kuvia, joita voidaan käyttää pilviin, nahkaan ja moniin muihin materiaaleihin. Kolmas menetelmä on oikeiden valokuvien käyttö. Tämä on paras ja kätevin vaihtoehto esimerkiksi, kun yksinkertaiseen malliin oikeasta tai keksitystä rakennuksesta halutaan mahdollisimman realistiset tekstuurit. [37, s. 108.]

Kun tekstuurin teon aloittaa, se on ensin suunniteltava hyvin. Tämä säästää aikaa, kun tekstuuria ei joudu tekemään moneen kertaan uudelleen. Suunniteltaessa on pantava merkille ensinnäkin tekstuurin paikka ja koko mallissa. Useimmiten mallin kaikki tekstuurit tehdään samaan kuvatiedostoon, koska se on kätevämpää ja koko mallissa pysyy sama tarkkuus, mikä on myös otettava huomioon mallin UV-mappauksessa, jossa 3D-mallin pinnat projisoidaan tai avataan 2D-tasolle, johon tekstuuri liitetään. Toinen merkille pantava asia on mallin katseluetäisyys lopullisessa tarkoituksessaan, eli tarkastellaanko mallia läheltä vai kaukaa. Jos malli tekstuureineen nähdään vain kaukaa, voivat tekstuurit olla hyvinkin yksinkertaisia ja pienellä tarkkuudella.

Useimmat pelimoottorit ja grafiikkarajapinnat vaativat, että käytettävät tekstuurit noudattavat mitoiltaan kahden potenssin sääntöä. Tämä tarkoittaa, että tekstuureissa käytettävien bittikarttakuvien korkeuksien ja leveyksien tulisi olla kahden potensseja, esimerkiksi 2 x 2, 64 x 64, 256 x 128 tai 1024 x 2048. [37, s. 109.] Laajaan pintaan, jossa materiaali pysyy samanlaisena, ei kannata luoda yhtä isoa tekstuuria, vaan mieluummin käyttää pientä kuvaa, joka monistetaan pinnalle. Tällöin tekstuurin teossa on pidettävä huoli sen saumattomuudesta, sillä se helposti vie mallin realistisuutta. [37, s. 113.]

Usein ajatellaan 3D-mallien materiaalin tarkoittavan vain siinä nähtäviä värejä ja kuvioita. Kuten todellisillakin materiaaleilla, esimerkiksi puulla, kankaalla tai kivellä, niillä on myös muita ominaisuuksia. Kivipinnan karheuden pystyy tuntemaan, ja valo taittuu siitä aivan eri tavalla kuin sileästä pinnasta. Yhtä lailla kivipinnassa saattaa olla kiiltävämpiä kohtia, joita ilman se vaikuttaisi lattealta. Myös pinnan mahdollinen heijastuvuus, läpinäkyvyys ja taittokerroin on ominaisuuksia, jotka tuovat malliin realismia. Nämä kaikki ovat asioita, joita kannattaa miettiä, ennen kun aloittaa tekstuurien teon.

Mallin teksturointi kokonaisuudessaan tarkoittaa kaikkien eri ominaisuuksien mappausta pinnoille. Tällöin pintarakenne saadaan kiinnostavaksi ja lähemmäksi todellisuutta.

Tekstuuriin voidaan mapata seuraavat asiat:

- Diffusella tarkoitetaan tekstuuriin asetettavaa väriä tai esimerkiksi bittikarttakuvaa. Tämä on yksinkertaisin tapa muuttaa pinnan ulkonäkö halutunlaiseksi. Esimerkiksi tiiliseinää ei tarvitse mallintaa tiili kerrallaan, vaan koko seinälle voidaan asettaa tiiliseinää kuvaava bittikartta.

- Bump mapin käyttö luo vaikutelman kolmiulotteisista piirteistä pinnalla. Tällä tavoin pystyy pintaan luomaan esimerkiksi kohoumia, uria tai muita yksityiskohtia, ilman että niitä tarvitsee mallintaa. Tämä on todella hyödyllinen ja helposti lisättävä ominaisuus, varsinkin kun mallista ei haluta liian monimutkaista. Bump map perustuu harmaasävykuvaan, jossa pikselin kirkkausarvo korreloi suoraan kohouman korkeuteen. [38.]

- Normal map on uudempi ja tarkempi versio bump mapista. Toisin kuin bump map, se perustuu RGB-kuvaan, jossa värikomponentit vastaavat 3D-avaruuden X-, Y- ja Z-akseleita, eli suuntaa, johon pinnan kohta on kohdistunut. Normal mapia on tosin todella vaikea tehdä tai muokata kuvankäsittelyohjelmalla, sen sijaan se generoidaan usein kolmiulotteisesta kappaleesta. [38.]

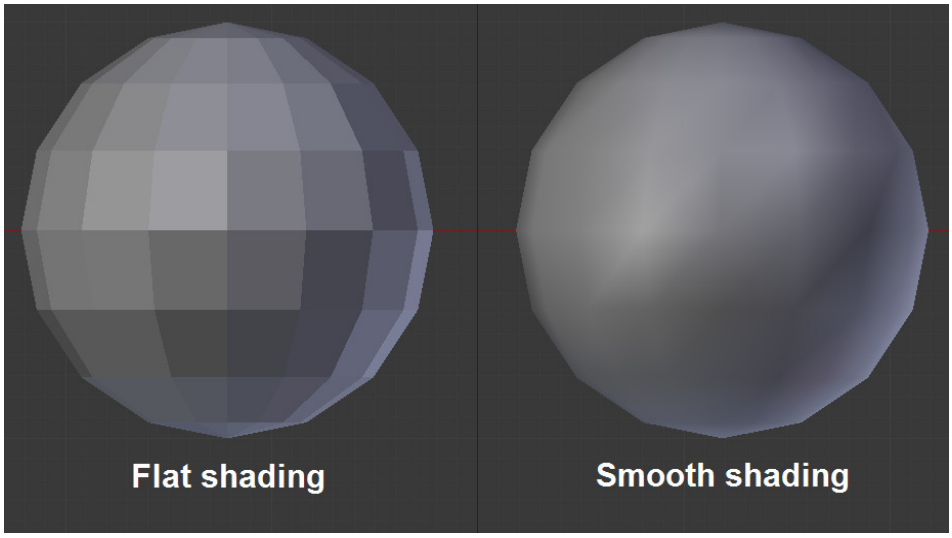
- Displacement map on kaikkein realistisin vaihtoehto, kun tahdotaan luoda pintaan muotoja ja yksityiskohtia. Mutta toisin kuin normal ja bump map, se muokkaa pinnan geometriaa, mikä voi tehdä mallista todella raskaan. Displacement mapia kannattaa käyttää harkiten ja vain, kun sitä oikeasti tarvitaan. Esimerkiksi maaston kolmiulotteiset muodot on kätevää luoda tällä tavoin. [38.]

- Specularity luo tekstuuriin vaikutelman kiillosta, ja specular mapilla voidaan määritellä, kuinka paljon kiiltoa näkyy missäkin pinnan kohdassa. Tähän käytetään harmaasävykuvaa, jossa mitä kirkkaampi pikseli on, sitä enemmän se luo kiiltoa.

- Reflection tarkoittaa yksinkertaisesti pinnan heijastuvuutta ympäristöstään. Sitäkin voidaan kontrolloida reflection mapilla. Siihen voidaan yhdistää myös refraction map, joka määrittää taittokertoimen avulla, miten tekstuurin takana oleva tausta taittuu läpinäkyvästä pinnasta. [39.]

- Alpha tai opacity map määrittää kunkin tekstuurin kohdan läpinäkyvyyden. Harmaasävykuvassa esimerkiksi mustilla pikseleillä voidaan luoda pintaan kokonaan läpinäkyviä kohtia. Alpha mapia hyödyntämällä pystytään mallin rakennetta helposti myös yksinkertaistamaan. Esimerkiksi puuaitaa ei tarvitse mallintaa kokonaan, kun laittaa kaksipuoliselle pinnalle aitakuvioisen tekstuurin, jossa alpha mapilla on merkitty aidan raot ja muut läpinäkyvät kohdat.

Tekstuureilla on myös muita ominaisuuksia, joihin kannattaa kiinnittää huomiota. Esimerkiksi pinnan suunnalla tarkoitetaan sitä, mikä puoli siitä on näkyvissä. Jos pinta on yksipuolinen, vain toinen puoli näkyy. Jos pinnan taas on oltava näkyvissä kummaltakin puolelta, se on asetettava kaksipuoliseksi. Toinen huomioitava asia on mallin pintojen shading eli värivarjostus. Kuvasta 7 näkee, miten varjostuksen tyypit vaikuttavat mallin ulkonäköön. Smooth shading eli pehmeä varjostus ilmenee yhtenäisenä sileänä pintana siten, etteivät yksittäisten tasojen rajat erotu. Flat shading taas jättää pinnat sellaisiksi, että jokaisella tasolla on oma värivarjostusarvo normaalin suuntaisesti, jolloin pinnat myös erottuvat toisistaan. [40.]



Kuva 7. Pintojen värivarjostustyypit.

Yksittäisistä pinnoista voidaan muodostaa malliin myös varjostusryhmiä, jolloin malli voi olla muuten pehmeästi varjostettu, mutta kuitenkin osittain terävillä reunoilla. Monet mallinnusohjelmat tarjoavat tähän avuksi työkalun, jolla voidaan asettaa pintojen välisille kulmille raja-arvo, jota loivemmat kulmat pehmennetään.

Projektin vesitorniin asetin paljon eri materiaaleja, jotta sain siitä mahdollisimman realistisen näköisen. Malliin ei ollut kuitenkaan tarvetta käyttää muita kuin diffuse- ja alpha-mappausta. Suurin osa mallista käyttää betonitekstuuria, joka luotiin pienestä kuvasta, joka monistettiin pinnoille. Joillekin pinnoille, kuten portaikkojen aidoille, tultiin diffusen lisäksi asettamaan myös alpha map, jolla aitojen läpinäkyvät kohdat saatiin luotua. Tällöin niitä ei tarvinnut mallintaa sen monimutkaisemmin. Smooth shading -varjostusta myös hyödynsin kaikissa kaarevissa pinnoissa, jolloin sain ne näyttämään sileiltä. Lisää projektissa käytetyistä materiaaleista käyn läpi luvussa 4.2.

Valaistus

Jotta materiaaleista saadaan kaikki hyöty, on valaistuksen suunnitteluun ja asetteluun käytettävä aikaa. Valolla pystytään vaikuttamaan todella paljon mallin tunnelmaan ja ajankohtaan, ja sillä voidaan korostaa haluttuja osia siitä. Tilanteesta riippuen tulee käyttää monipuolisesti erilaisia valonlähteitä, ja niiden voimakkuus, väri ja muut asetukset tulee määrittää sopiviksi.

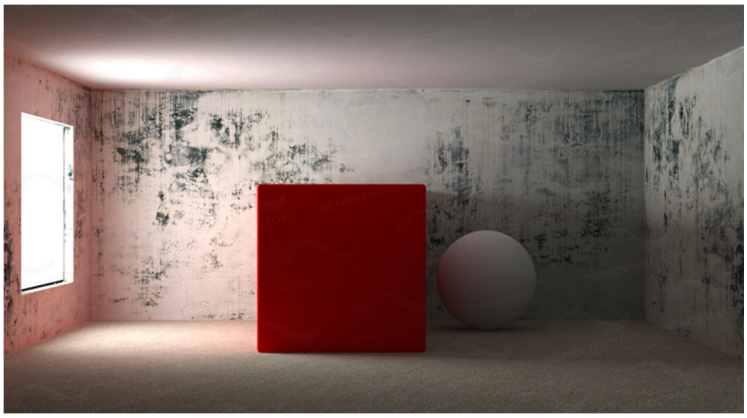
Todellisuudessa valoa on lähes kaikkialla, eikä valonlähteitä aina edes tiedosta. Kun kolmiulotteista mallia, tilaa tai ympäristöä halutaan valaista realistisesti, tulee olla hyvä käsitys valaistuksesta todellisuudessa, jossa valonsäteet yhtä lailla noudattavat samoja sääntöjä. Sijoittelussa kannattaa ottaa huomioon varjojen, kiillon ja heijastumien esiintyminen pinnoissa, ja esimerkiksi valokuvauksessa käytettävät valaistustekniikat pätevät hyvin myös virtuaalisesti. Tämä vaatii paljon harjoittelua ja kokeilua, jotta kaikki saa kohdalleen, eikä aina ole vain yhtä ratkaisua, vaan on myös taiteellisesti mietittävä, mikä sommitelma vie haluttua viestiä eteenpäin.

Yleensä valotyypit jaetaan mallinnusohjelmissa kahteen ryhmään: standardivaloihin ja fotometrisiin valoihin. Standardivalot ovat kevyitä ja yksinkertaisempia valoja, jotka eivät perustu todellisiin valon ominaisuuksiin. Oikein asetettuina niillä on kuitenkin mahdollista luoda realistisia valaistuksia. Standardivalojen vahvuus on se, että niillä on paljon enemmän ominaisuuksia ja parametreja, joita voidaan muokata ja saada haluttu tulos aikaiseksi. [37, s. 184–185.]

Fotometriset valot taas perustuvat todellisten valojen ominaisuuksiin. Ne pyrkivät mahdollisimman tarkasti simuloimaan valon kulkua realistisesti, mutta ovat raskaampia kuin tavalliset valot. Fotometristen valojen parametrit toimivat mallin mittayksikön mukaan, joten sen realistisesta koosta on myös huolehdittava. [37, s. 185.]

Kun valoihin yhdistää lisäksi monimutkaisempia tekniikoita, kuten global illuminationin, voidaan valaistuksen osalta saada jo hyvin fotorealistista jälkeä. Global illumination (GI) tarkoittaa tekniikkaa, jossa eri algoritmeilla voidaan parantaa valaistuksen realistisuutta muun muassa valon kimpoamisella ja heijastumisella epäsuorasti eri pinnoista. Tekniikalla voidaan simuloida tilanteita ja ympäristöjä, kuten auringonvalon kimpoilua huoneen sisällä, jotka muuten olisi todella vaikea saada tarpeeksi realistisiksi. Kuvasta

8 huomaa, kuinka auringonvalo levittyy pehmeästi koko huoneeseen ja kuution punainen väri heijastuu seinille ja palloon. [37, s. 185; 41.]



Kuva 8. Global illuminationin käyttö valaistuksessa [41].

Valaisintyyppejä on useita, ja niillä on kullakin omat hyötynsä. Tarkoituksesta ja tilanteesta riippuen tulee valita sopiva valonlähde. Esimerkiksi ulkotilanteissa valaistukseen käytettävät tekniikat poikkeavat sisätiloissa käytettävistä. Valaisintyypillä pystytään myös vaikuttamaan lankeaviin varjoihin ja niiden pehmeyteen. Yleisimpiin valaisintyyppeihin kuuluvat seuraavat:

- Piste- tai omni-valo on yksinkertaisesti yksi piste kolmiulotteisessa avaruudessa, josta valonsäteet lähtevät tasaisesti kaikkiin suuntiin. Esimerkiksi hehkulamppu tai kynttilä sopivat pistevaloiksi. [42.]

- Suuntavalo on valo, jolla ei ole sijaintia, vaan vain suunta. Sillä simuloidaan todella kaukana olevaa valonlähdettä, kuten aurinkoa, ja kaikki sen valonsäteet kulkevat rinnakkain samaan suuntaan. [42.]

- Spottivalo on yhdestä pisteestä lähtevä kartion muotoinen valokeila. Sitä voidaan käyttää esimerkiksi katulamppuihin, pöytälamppuihin tai valonheittimiin. [42.]

- Lineaarivalo on suora jana, joka lähettää säteitä ympärilleen tasaisesti. Tämä simuloi esimerkiksi loisteputkea. Aluevalo taas on kuin lineaarivalo, mutta sillä on myös leveyskomponentti. Se lähettää säteitä tasaisesti suorakaiteen muotoiselta alueelta, ja sillä voi simuloida esimerkiksi ikkunasta tulevaa valoa. Lineaari- ja aluevalo pehmentävät varjoja, ja niiden säteiden tiheys tulee määrittää sopiviksi, sillä ne voivat helposti tehdä renderöinnistä raskaan. [42.]

- Ambientvalo on tasainen koko mallin, tilan ja ympäristön kattava valo, joka ei jätä varjoja. Se usein asetetaan himmeäksi taustavaloksi, joka näkyy vain pinnoilla, joihin mikään muu valo ei osu. Tällöin vältytään täysin pikimustilta varjoilta, joita harvoin todellisuudessa on. [42.]

Valoilla on useita muutettavia parametreja, joilla niiden toimivuutta pystyy muokkaamaan. Niiden kaikkien läpikäyminen ei olisi tässä järkevää, mutta niistä muutama ainakin kannattaa huomioida. Sijainnin ja suunnan lisäksi tärkeimmät säädettävät asiat ovat valon intensiteetti ja väri. Valon heikkeneminen taas määritellään decay-parametreilla. Spottivalolla on lisäksi parametreja kartiokulman ja reunan pehmeyden määritykseen.

Jotta valoista saataisiin vieläkin realistisemmat, niihin voi käyttää IES (Illuminating Engineering Society) -valoprofiileja. Lähes kaikkiin valotyyppeihin pystyy asettamaan IES-valoprofiilin, jonka muodon mukaan valosäteet levittyvät. Useimmat valaisinvalmistajat käyttävät lamppujen fotometristen tietojen levitykseen IES-tiedostoja, joita pystyy ilmaiseksi lataamaan internetistä. Esimerkiksi rakennusmalliin pystyy asettamaan realistisia valonlähteitä, joiden valokeilat myös vastaavat todellisuutta. [43.]

Tässä projektissa valaistus on melko pienessä roolissa, mutta myös tärkeä osa kokonaisuutta. Koska kyseessä on WebGL-sovellus, asetin valot vasta sovellusta ohjelmoitaessa. Tärkein asetettava valo oli aurinkoa simuloiva suuntavalo, joka tuo hieman realistisuutta lankeavilla varjoillaan. Valo on hieman kellertävä ja suunnattu siten, että mallin kaikkein tärkeimmät piirteet on valaistu. Toinen tärkeä valo on tumman sininen ambientvalo, joka pitää varjoalueet näkyvissä.

2.6 HTML5-merkintäkieli, WebGL-rajapinta ja Three.js-grafiikkakirjasto

HTML (Hypertext Markup Language) -merkintäkielen uusin versio HTML5 julkaistiin vuonna 2008. Se kuitenkin käsittää uuden version lisäksi myös modernit web-tekniikat, kuten CSS (Cascading Style Sheets) -merkintäkielen, JavaScript-komentosarjakielen, Ajax (Asynchronous JavaScript And XML) -tekniikat sekä sovellusrajapinnat. HTML5:n tavoitteena on, että kaikki webissä toimisi saumattomasti ja tehokkaasti yhdessä. Ylimääräisten liitännäisten käyttäminen esimerkiksi multimediatoimintoihin ei ole käytännöllistä, ja ne rajoittavat selaimen käytettävyyttä. Adobe Flash ja Microsoft Silverlight toimivat selaimessa ladattavilla liitännäisillä ja näin ollen tekevät niiden käytöstä turhan hankalaa. Flashin käytön romahtaminen nosti HTML5:n suosiota, mikä johti erilaisten tekniikoiden kehityksen kiihtymiseen. HTML5 pysyykin elävänä standardina, mikä tarkoittaa sitä, että sitä kehitetään koko ajan eteenpäin. [44.]

Kun HTML5:ta alettiin laajasti tukea ja sen suosio nousi, tuli tarve uudistaa selaimien multimediaominaisuuksia. Flashin tilalle alettiin suositella HTML5:n tekniikoilla toteutettua sisältöä, joka pystyi vastaavaan ja jopa parempaan toiminnallisuuteen. CSS-kielen uusin versio CSS3 on kehittynyt monipuoliseksi tekniikaksi sulavien, näyttävien ja interaktiivisten sivustojen tekoon. Myös SVG (Scalable Vector Graphics) -kuvauskieli on kehittynyt ja sen tuki on parantunut. SVG:llä pystyy luomaan XML-pohjaisia vektorikuvia ja interaktiivisia animaatioita sivustoille. [45.]

Kun selaimelle tahtoo tehdä kokonaista sovellusta tai CSS:n käyttö ei muuten riitä, on hyvä tutustua muutamaan vaihtoehtoiseen tekniikkaan. Canvas-elementti on HTML5:een tuotu elementti, jonka päälle pystytään JavaScript-koodilla luomaan interaktiivista 2D- ja 3D-grafiikkaa, animaatioita, pelejä tai melkein mitä vain halutaan. Elementti on sivustolla suorakaiteen muotoinen alue, jolle määritellään korkeus ja leveys. 2D-sovelluksen rakentaminen canvas-alustalle on suhteellisen helppoa pienen opettelun jälkeen. Ohjeita ja dokumentaatiota eri toiminnoista voi myös etsiä internetistä, joten alkuun pääseminen on helppoa. [46.]

Grafiikkakirjastot ovat JavaScript-kirjastoja, joiden tarkoituksena on helpottaa 2D- ja 3D-grafiikan tuottamista sivustolle. Niitä on useita erilaisia, ja niillä on kullakin omat vahvuutensa. Niillä pystytään tuottamaan sisältöä monella eri tekniikalla, ja kirjastojen tarjoamat ominaisuudet ja valmiit työkalut tekevät sovellusten ohjelmoimisesta luontevaa ja ymmärrettävämpää. Jotkin grafiikkakirjastot voivat tuottaa sisällön käyttäen CSS3- tai SVG-renderöijiä, mutta yleensä sisältö renderöidään WebGL:n kautta tai suoraan canvas-elementtiin.

WebGL

WebGL on JavaScript-ohjelmointirajapinta, jonka päätarkoituksena on renderöidä 3D-grafiikkaa selaimessa. Se on laajasti tuettu useimmilla moderneilla selaimilla ja laitteilla, ja sitä voikin pitää jo standardina. WebGL tuottaa sisältönsä canvas-elementtiin, eikä ylimääräisiä liitännäisiä tarvita. Renderöinti sillä on nopeaa ja tehokasta, mutta sen käyttö on yksinään melko hankalaa sovellusta, peliä tai muuta vastaavaa tehtäessä. Ohjelmoiminen WebGL:llä vaatii paljon opettelua tai asian tuntemusta entuudestaan. [47, s. XXI.]

WebGL perustuu vuonna 2007 julkaistuun OpenGL Embedded Systems (ES) -ohjelmointirajapintaan, joka on OpenGL 2.0:sta kevyempi versio. OpenGL on ollut jo yli 20 vuotta yksi parhaista työkaluista kolmiulotteisen tietokonegrafiikan tuottamiseen, ja se tukee laitteistokiihdytystä ja mahdollistaa grafiikkaprosessorin (GPU) käytön laskennassa. OpenGL ES kehitettiin yksinkertaisemmille alustoille, kuten mobiililaitteille, joten se käyttää myös laitteen suorituskykyä mahdollisimman tehokkaasti hyödykseen. [47, s. XXI.]

Vladimir Vukićević alkoi vuonna 2007 kehittää Canvas 3D -prototyyppiä, joka hyödynsi OpenGL:ää. Tämä sai OpenGL:n kehittäjän, Khronos Groupin, aloittamaan WebGL Working Group -työryhmän, jonka tehtävänä oli kehittää WebGL:ää eteenpäin. Ensimmäinen versio siitä julkaistiin vuonna 2011, ja sen kehitys jatkuu edelleen. [47, s. XXI.]

WebGL:n myötä web-sivustoille on todella kätevää luoda kolmiulotteisia sovelluksia, esityksiä ja pelejä, ja ne voidaan siististi upottaa myös muun sisällön sekaan. Tämä mahdollistaa, että esimerkiksi jonkin tuotteen pystyy esittelemään pyöritettävänä 3D-kappaleena ja näin antamaan paljon paremman kuvan siitä asiakkaille. Myös kaupunkimallien tai -suunnitelmien esittäminen onnistuu interaktiivisessa sovelluksessa, jossa käyttäjä pääsee itse liikkumaan kaduilla. WebGL:n renderöinti on myös nopeaa, eikä sen käyttö vaadi käyttäjältä mitään muuta kuin modernin selaimen. Projektiin WebGL-alusta oli luonnollinen valinta sen selaintuen ja kätevyyden takia.

Three.js

WebGL-rajapinnan käyttö voi monelle olla vaikeaa, koska se vaatii paljon ymmärrystä siitä, miten se tai OpenGL sisäisesti toimii. Tätä helpottamaan on kuitenkin saatavilla paljon grafiikkakirjastoja, jotka toimivat helpottavina rajapintoina 3D-sovellusten ja pelien tekoon. Yksi suosituimmista ja käytetyimmistä grafiikkakirjastoista on Three.js, joka on todella monipuolinen ja nopeasti opittava ratkaisu, jolla pystyy luomaan lähes mitä tahansa sovelluksia ja pelejä. Three.js:n käyttäminen on suoraviivaista, eikä se vaadi selaimen lisäksi kuin jonkin tekstinkäsittelyohjelman. Renderöinti tehdään sillä yleensä WebGL:n päälle, mutta mahdollista on myös SVG-, CSS3D- tai canvas-renderöinti. Three.js on todella avoin kirjasto, ja sen tekijät kehottavat käyttäjiä myös osallistumaan sen kehitykseen. [48.]

Three.js sisältää paljon ominaisuuksia ja työkaluja monipuoliseen kehittämiseen. Kameroille, valoille, materiaaleille, geometrialle ja muille asioille on helpot komennot, ja primitiivimuotoja pystytään luomaan helposti omilla parametreilla. Mallien tuominen onnistuu Three.js-sovellukseen useilla eri importereilla, joilla malli voidaan tuoda esimerkiksi OBJ-, COLLADA- ja JSON (JavaScript Object Notation) -muodoissa. Three.js-sovellusten ohjelmointiin pääsee parhaiten sisään tutustumalla olemassa olevien esimerkkien koodiin ja muokkaamalla niitä. [48.] Liitteessä 1 on nähtävillä yksinkertaisen kuutioanimaation koodi, jota on helppo lähteä muokkaamaan omanlaisekseen.

Three.js on kätevä ratkaisu juuri jonkin kolmiulotteisen kappaleen visualisointiin, joten projektiani varten se sopi hyvin. Three.js tarjoaa myös valmiita kameraohjaimia, joista voi valita käyttöä varten sopivan tavan liikuttaa kameraa mallin ympärillä. Itselläni on myös aikaisempaa kokemusta Three.js:stä, joten sen opettelemiseen ei tarvinnut käyttää aikaa. Lisää Three.js-sovelluksen tekemisestä käyn läpi luvussa 4.3.

3 Lauttasaaren vesitorni

3.1 Vesitornin historia ja nykytilanne

Lauttasaaren vesitorni on ollut yli 50 vuotta yksi Lauttasaaren ja koko Helsingin merkittävimmistä maamerkeistä (ks. kuva 9). Se on Suomen ensimmäinen suuri sienen muotoinen vesitorni ja täten rakennushistoriallisesti tärkeä rakennus. Vesitornikäytöstä se poistettiin vuonna 1996, ja se on siitä asti ollut tyhjillään. Sille on sen jälkeen etsitty uutta käyttäjää ja tarkoitusta, kuitenkin ilman tulosta. Vuonna 2013 sille annettiin purkupäätös. [50, s.1; 51.]

Kuva 9. Lauttasaaren vesitorni [51].

Vesitornin historia

Lauttasaaren vesitornia alettiin ensimmäistä kertaa suunnitella 1950-luvulla, kun Länsi-Helsingissä sijaitsevasta Lauttasaaresta alkoi muotoutua oma kaupunginosansa eikä vedenpaine enää riittänyt kasvavalle asukasmäärälle. Vuonna 1955 laadittiin rahoitusohjelma vesijohtoverkon ja vesitornin rakennusta varten, ja tornin suunnittelutyö aloitettiin vuonna 1956. Vesitornin muodossa päädyttiin kartionmuotoiseen säiliöön muun muassa sen takia, että silloin veden pinnan ja paineen vaihtelu on pienimmillään. [50, s. 2.]

Lauttasaaren vesitornin rakennusurakka annettiin 64 000 000 markalla Oy Yleinen Insinööritoimisto Ab:lle (YIT). Pääpiirustukset hyväksyttiin maaliskuussa 1958, ja rakennustyöt aloitettiin huhtikuussa 1958. Torni otettiin käyttöön jo marraskuussa, vaikka sen viimeistely kesti vielä helmikuuhun 1959. Alun perin oli suunniteltu, että koko urakka kestäisi kesäkuuhun 1959 saakka, mutta rakennustyö suoritettiinkin paljon nopeammin Lauttasaaren huonon vedenpaineen ja veden ajoittaisen riittämättömyyden vuoksi. [50, s. 5–7.]

Vesitornista toivottiin myös näköalapaikkaa, jonka vuoksi sen huipulle rakennettiin kaiteilla varustettu tasanne, johon kulku onnistui portaita pitkin. Rakennusvirasto antoi syyskuussa 1958 kuitenkin lausunnon, jonka mukaan torni ei sopisi kaikille avoimeksi näkötorniksi, sillä se vaatisi erityisten suojarakennelmien rakentamista sekä valvontahenkilöstöä. Tasanne on ollut yleisön käytössä muutamina erityisinä päivinä, ja sinne on järjestetty yksittäisiä kutsutilaisuuksia. Sen muuttamista yleiseen käyttöön on toivottu monesti vuosien varrella, mutta toiveet on hylätty muun muassa portaiden parantamisen ja hissin rakentamisen suurten kustannusten vuoksi. [50, s. 11–12.]

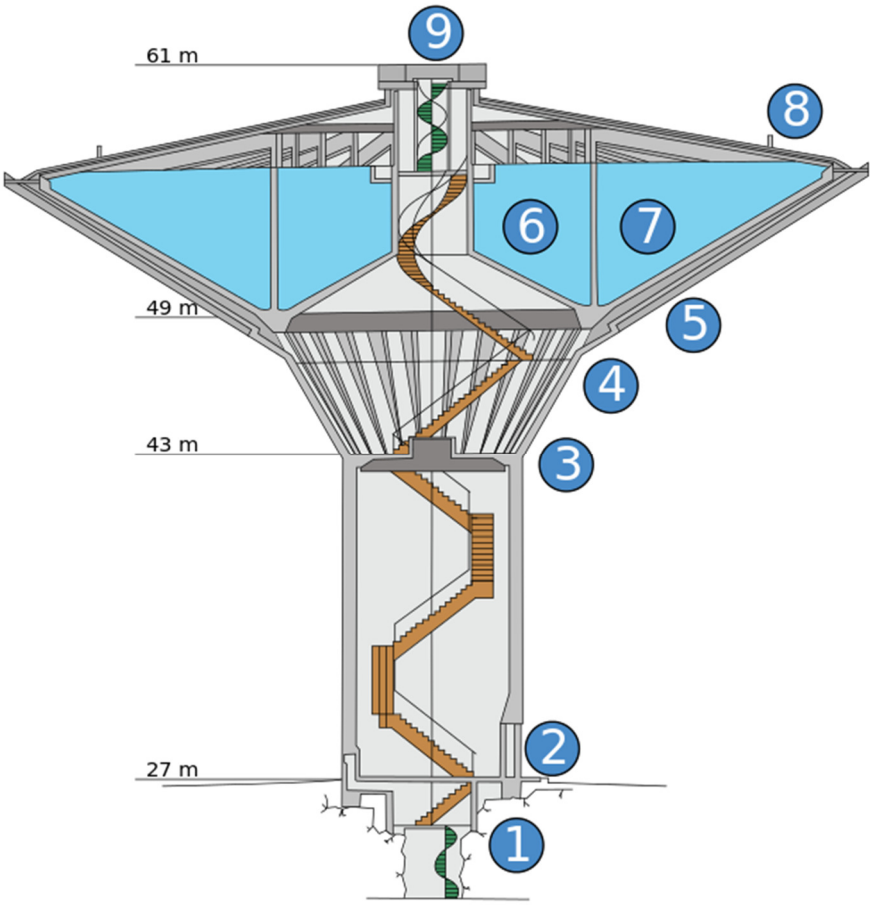
Nykytilanne

Vuonna 1982 vesitornin katolle rakennettiin kehikko, johon saatiin asennettua linkkiantennit puolustusvoimien ja merivartioston käyttöön. Siitä lähtien tasannetta on voitu vuokrata erilaisten antennien käyttöön, minkä vuoksi myös sisätiloihin on rakennettu niitä varten omia laitekaappeja. Tornille on tyhjennyksen jälkeen yritetty etsiä uutta käyttötarkoitusta, ja kiinnostuneita ostajia on myös ollut. [50, s. 14.]

Tornin kunto on kuitenkin viimeisen viidenkymmenen vuoden aikana huonontunut merkittävästi, ja vuonna 2008 tehdyn kuntoarvion mukaan muun muassa tornin jalan ja heltan betonielementit ovat selvästi vaurioituneet. Pelkästään mittava peruskorjaus tulisi maksamaan miljoonia euroja, eikä sopivaa ostajaa, jolla olisi tähän tarvittavat resurssit, ole löytynyt. Torniin on ollut kiinnostusta rakentaa muun muassa kahvila, asuntoja ja jopa uimahalli. Vesitornia myös suunniteltiin säilytettäväksi pelkästään maamerkkinä, mutta varoja ei siihenkään ollut saatavissa. [52; 53.]

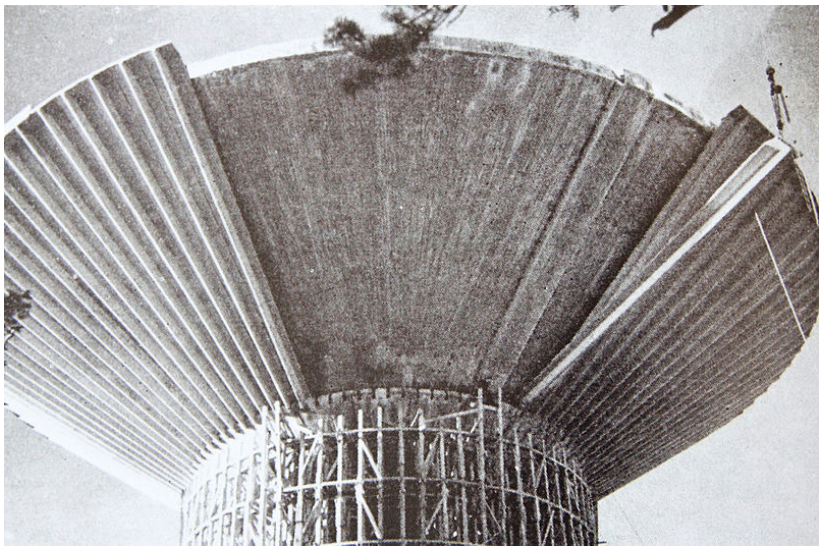
Rakenne

Lauttasaaren vesitorni rakennettiin Kotkavuoren huipulle 27 metriä merenpinnan yläpuolelle. Kuvassa 10 nähdään, kuinka vesitornin mielenkiintoinen rakenne jakaantuu. Sen ulkopinnat ovat suurimmalta osin teräsbetonia, ja sen korkeus on juuresta ylätasanteeseen saakka noin 34 metriä. Sen jalkaosan halkaisija on noin 9 metriä ja katon halkaisija noin 42 metriä. [50, s. 8.]



Kuva 10. Poikkileikkauskuva Lauttasaaren vesitornista: 1. kuilu ja portaat vesijohtotunneliin, 2. ulko-ovi, 3. aukko vesitornin keskellä tavaroiden nostamista varten, 4. keskihuone ja tukipalkisto, 5. tukirengas, 6. sisempi vesisäiliö, 7. ulompi vesisäiliö, 8. ilmanvaihtoputket, 9. kattotasanne. [54.]

Heltan alapinnalle asennetut 24 suojaelementtiä on asetettu 60 asteen kulmaan (ks. kuva 11), ja niistä muodostuvan suojakuoren tehtävänä on vähentää säiliön lämpötilavaihtelua ja tukea sen rakennetta. Tornin koko rakenne on myös lämpötilaeristetty ja räystäät on varustettu sähkölämmityksellä, jottei jääpuikkoja tippuisi alla olevaan puistoon. [50, s.11.]



Kuva 11. Suojaelementtien asennus vuonna 1958 [50, s. 8].

Vesitornin säiliö on jaettu kahteen ympyrän muotoiseen vesisäiliöön, tilavuudeltaan yhteensä 4 530 kuutiometriä. Sisempi säiliö on tilavuudeltaan 1 080 kuutiometriä ja ulompi 3 450 kuutiometriä. Vedenpinta voi korkeimmillaan nousta 57 metriin merenpinnasta, jolloin syvyys säiliössä on 7 metriä. Säiliö jaettiin kahteen osaan, jotta se oli mahdollista puhdistaa ilman, että veden saatavuus katkesi. [50, s. 8.]

3.2 Työn tarkoitus ja sen suunnittelu

Insinöörityön tarkoituksena oli paitsi tutkia rakennuksen digitaalista mallinnusta, myös mallintaa Lauttasaaren vesitorni kolmiulotteiseksi kappaleeksi. Projektiin kuului myös WebGL-sovelluksen rakentaminen ja sen upotus sivustoon, jossa on myös muuta vesitorniin liittyvää sisältöä. Torni tuli mallintaa sen käyttökohde huomioiden, joten se ei saanut olla liian monimutkainen. Myös mobiilialustat tuli ottaa huomioon. Mallin oli kuitenkin oltava sen verran yksityiskohtainen, että siitä voisi myöhemmin tutkia vesitornin rakennetta, sisätiloja ja ulkonäköä. Mallin ympäristö ja valaistus otettiin myös huomioon, ja sen pinnoille asetettiin realistiset materiaalit.

Sen jälkeen kun Lauttasaaren vesitorni on purettu, sovelluksen on toimittava hyvänä referenssinä tornista kiinnostuneille. Koko mallinnusprosessi tutki myös sitä, kuinka valmiin rakennuksen mallinnus onnistuu, kun käytössä on vain pohjapiirrokset ja kuvamateriaalia. Vastaan tulevat ongelmat tuli ottaa huomioon, etsiä ratkaisut tai vaihtoehtoiset tavat niihin ja dokumentoida ne. Tämä auttaa tulevaisuutta ajatellen, kun tiede-

tään, millä resursseilla ja aikataululla vastaavan rakennuksen mallinnus ja WebGL-sovelluksen teko onnistuu. Työn jälkeen tuloksiksi toivottiin selkeää käsitystä siitä, miten prosessi kokonaisuudessaan tapahtuu, mitä menetelmiä tulee välttää ja suosia ja mitkä ovat realistisesti saavutettavia ominaisuuksia lopputuotteessa. Lopuksi arvioin myös omaa onnistumistani ja sovelluksen käytettävyyttä jatkossa. Työn tuloksia käyn läpi luvussa 4.4.

Aloitin työn suunnittelun ensin tutustumalla saamaani materiaaliin Lauttasaaren vesitornista ja etsimällä itse lisää tietoa. Saamaani pohjapiirrokset vesitornista (liite 2) ovat kopioita alkuperäisistä käsin piirretyistä paperisista piirroksista 1950-luvulta, jolloin niitä ei vielä tietokoneella laadittu. Piirrokset ovat kuitenkin hyvälaatuiset, ja niistä pystyi saamaan hyvän käsityksen tornin rakenteesta. Ohessa sain myös monipuolisesti kuva-aineistoa, kuten kauko-ohjattavalla helikopterilla otettuja ilmakuvia ja sisäkuvia. Etsin kuvamateriaalia myös internetistä niin paljon kuin löysin. Itse en päässyt käymään vesitornissa sisällä, joten jouduin tyytymään siihen, mitä löytyi. Vesitornin sisäpuolelle ei ole kovin paljoa valokuvaajia muutenkaan päästetty, joten sen osalta kuvia oli suppeasti, mutta kuitenkin tarpeeksi tärkeimpien piirteiden ymmärtämiseksi. Kuvamateriaalia etsiessä otin huomioon myös malliin mapattavien tekstuurien teon.

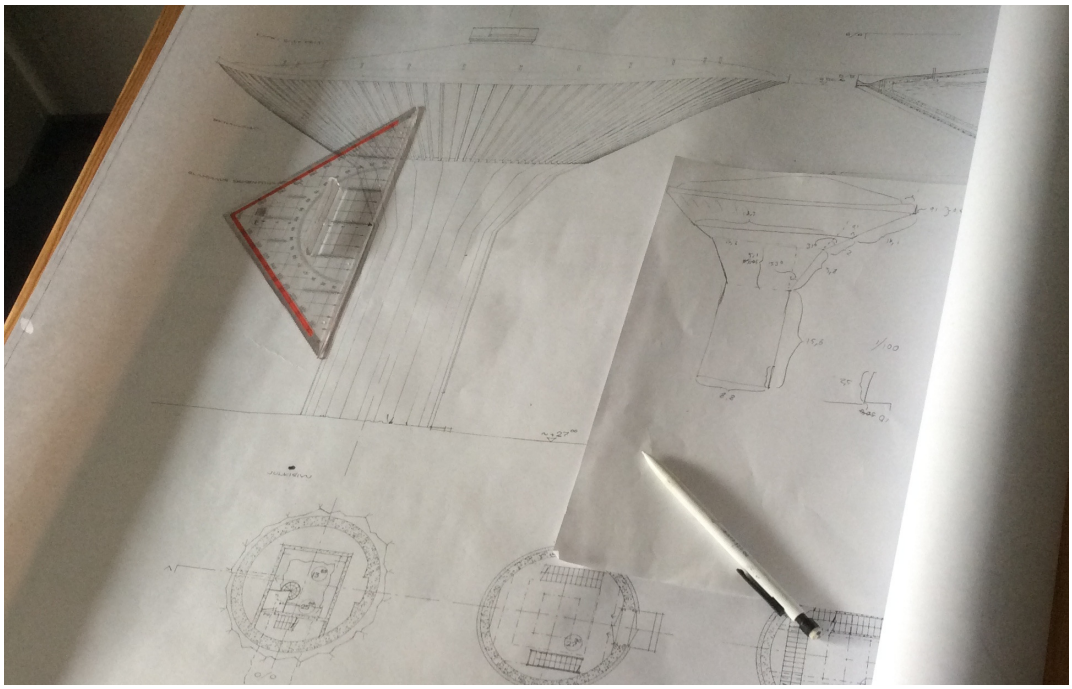
Vastaava työ oli aikaisemmin annettu toisellekin opiskelijalle, joka oli joutunut kuitenkin jättämään työn kesken pian aloituksen jälkeen. Häneltä oli jäänyt hiukan materiaalia tornin mallintamisesta Autodesk Maya -ohjelmalla. Tutkittuani materiaalia päätin aloittaa projektin puhtaalta pöydältä, sillä se oli nopeampaa.

4 Vesitornin mallinnus ja sovelluksen teko

4.1 Mittaaminen ja 3D-mallintaminen

Pohjapiirrosten mittaaminen

Kukin tornin kohta tuli mitata pohjapiirroksesta tarkkaan, jopa puolen millimetrin tarkkuudella, jotta mallin mitat saataisiin mahdollisimman lähelle oikeaa. Tätä tuli tehdä koko ajan mallinnuksen ohessa, aina kun jokin etäisyys on epäselvä. Ensin merkitsin mitat paperille (ks. kuva 12), mutta tämän jälkeen keksin ottaa tabletilla kuvan piirroksista ja kirjoittaa mitat kätevästi siihen päälle. Yksi huomioitava asia tulosten kirjaamisessa oli pohjapiirrosten mittakaava, joka tässä tapauksessa on 1/100. Tämä tarkoittaa, että piirroksesta mitattu tulos pitää kertoa 100:lla ennen sen syöttämistä CAD-ohjelmaan. Oikeiden mittayksiköiden käyttö on suositeltavaa, jotta malli näkyisi oikein kaikissa ohjelmistoissa ja koska sitä yleisestikin pidetään hyvänä käytänteenä.



Kuva 12. Paperisten pohjapiirrosten mittaaminen.

Mitatessa voi etäisyyksien lisäksi mitata myös kulmia, mutta niissä on oltava erityisen tarkka, varsinkin isommissa rakennuksen piirteissä. Jos mittaa pienellä kolmioviivaimella ja mitattu kulma poikkeaa asteenkin verran, se voi tarkoittaa useampienkin millimet-

rien virhe-etäisyyttä, joka taas skaalautuu moniin metreihin todellisessa koossa. Mittaamiseen kannattaa käyttää laadukkaita ja mahdollisimman isoja viivaimia, jotta tulokset ovat mahdollisimman tarkkoja.

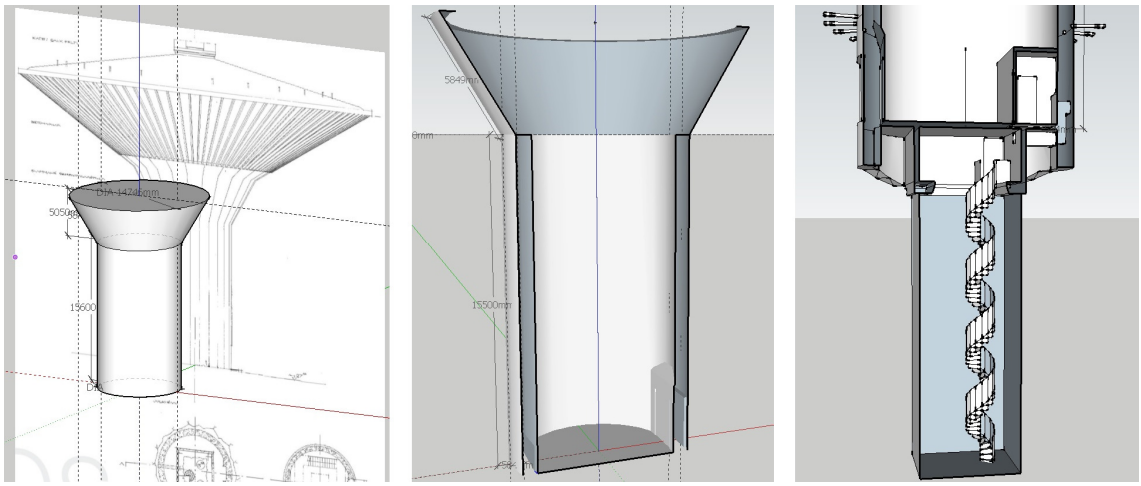
Mallintaminen

Valitessani mallinnusohjelmaa ensisijaista tornin mallinnusta varten, vaihtoehtoina olivat Autodesk 3ds Max, Maya, Blender tai SketchUp. Blender oli siitä hyvä vaihtoehto, että mallin vieminen siitä eteenpäin WebGL-sovellukseen on helppoa ja useimmiten onnistuu ilman ongelmia, kun taas muilla ohjelmilla tiedostojen siirto ei aina onnistu niin kuin pitää. Kokeilin mallin aloittamista kuitenkin ensin tutulla 3ds Maxilla, mutta totesin, ettei sillä pystynyt tarpeeksi tarkkaan lopputulokseen. Yhtä lailla Blenderillä ja Mayallakaan ei pystyisi CAD-tyyppiseen mallintamiseen tarkkojen mittojen perusteella, joten luonteva vaihtoehto oli SketchUp, joka on vaikkakin yksinkertainen, myös todella tehokas ja helppokäyttöinen ohjelma. SketchUp oli myös minulle tuttu aikaisemmista projekteista, joten ylimääräistä opettelua ei tarvittu.

Ennen mallinnuksen aloittamista piti miettiä, millä tarkkuudella se tulisi tehdä, kun lopullinen käyttötarkoitus on WebGL-sovellus. Tällöin se ei saisi olla liian monimutkainen. Myös materiaaleissa tuli ajatella, mikä olisi laadullisesti riittävä. Ensisijaisesti pyrittiin kuitenkin siihen, että vesitornin geometria olisi selkeä ja tutkittavissa, ja vasta tämän jälkeen keskityttäisiin hienoihin realistisiin tekstuureihin.

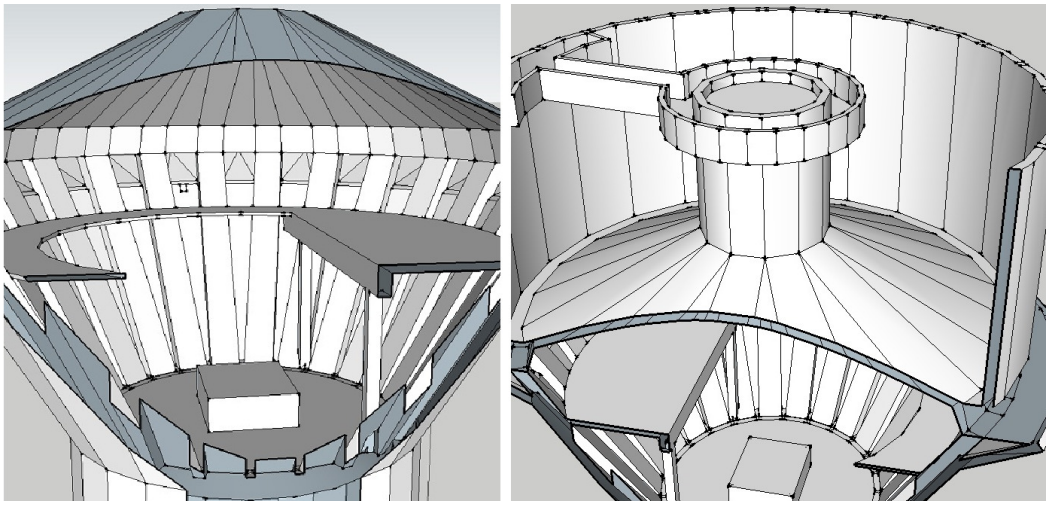
Aloitin vesitornin mallinnuksen luomalla SketchUpissa tornin mittoja vastaavan sylinterin ja sen viereen tason, jolle asetin pohjapiirroksesta otetun kuvan, mikä auttoi havainnoimaan tornin muotoa. Tämän jälkeen aloin pisteitä ja pintoja lisäten kasvattaa tornia ja muotoilla sitä enemmän vastaamaan piirroksia. SketchUpilla pystyy työkaluilla helposti lisäämään muotoja, kuten ympyröitä, ja kasvattamaan pintoja, joilla mallin saa kätevästi koottua.

Kuvasta 13 näkee hieman mallinnuksen alkuvaiheita. Oikeanpuolisessa kuvassa olevat pitkät kierreportaat vesijohtotunneliin on mallinnettu kokonaisuudessaan siten, että kun pieni osa portaista oli mallinnettu, kopioin sitä useaan kertaan alas asti. Tätä menetelmää käytin monessa muussakin tilanteessa, jossa geometria toistuu samanlaisena. Melkein kaikki etäisyydet, mitä vain on ollut mahdollista mitata pohjapiirroksista, on asetettu mallinnettaessa mahdollisimman lähelle oikeita pituuksia.



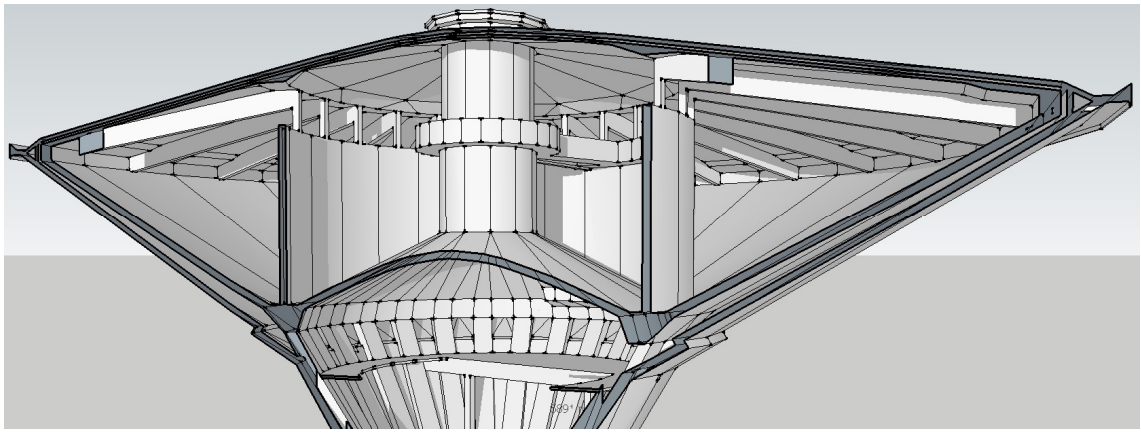
Kuva 13. Mallintamisen alkuvaiheet.

Kaikkea ei rakennuksesta pystynyt pelkillä pohjapiirroksilla hahmottamaan, joten keräsin myös paljon kuva-aineistoa tornin sisä- ja ulkopuolelta. Valokuvista pystyy päättelemään, miten pohjapiirrosten eri elementit, kuten portaat, sijoittuvat rakennuksen sisällä. Mallinnuksen yhteydessä huomasin, ettei käsin piirrettyihin pohjapiirroksiin pidä aina täysin luottaa, sillä niissä voi esiintyä ristiriitoja. Esimerkiksi vesitornin kellarikerroksen kierreportaiden sijainti näyttää yhdessä piirroksessa olevan eri reunalla tilaa kuin toisessa. Tämä on nähtävillä liitteestä 2, kellarikerroksen sivu- ja pystyleikkauksista. Kuva-aineistoa kellarikerroksesta ei löytynyt, joten minun piti miettiä, miten se järkevimmin istuisi rakennukseen, ja mallintaa se sen mukaan. Kuvat olivat erityisen hyödyllisiä, kun mallinsin vesitornin keskikerroksia (ks. kuva 14), joissa on paljon yksityiskohtia ja mielenkiintoisia muotoja.



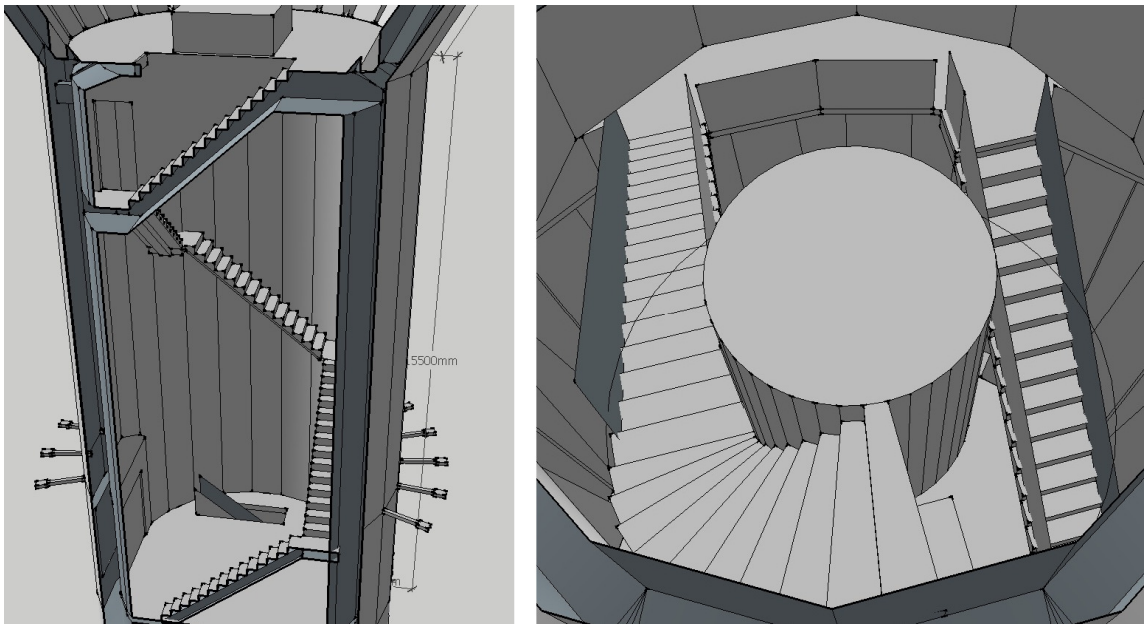
Kuva 14. Keskikerrosten mallintaminen.

Kompromisseja on myös tehtävä mittaustuloksien kanssa, sillä käsin piirretyissä pohjapiirroksissa voivat jotkin etäisyydet, esimerkiksi portaiden korkeudet, poiketa millimetrin kymmenyksiä, mitkä ovat jo pidempiä etäisyyksiä oikeassa mittakaavassa. Tällöin voi etäisyyksistä laskea niiden keskiarvon ja käyttää sitä, tai sitten asettaa korkeudeksi jonkin muun sopivan mitan. Mitattaessa kannattaa käyttää mahdollisimman tarkkaa viivainta, jolla pystyy erottamaan millimetriä lyhempiäkin etäisyyksiä. Vesitornissa on paljon kaltevia ja pyöreitä pintoja (ks. kuva 15), joiden kulmien tarkka mittaus ei käynyt aivan helposti pohjapiirroksista. Kulmien mittaamisen sijaan käsittelin kaltevia kohtia kuin suorakulmaisia kolmioita, joista laskin korkeudet ja leveydet. Niiden avulla sain luotua kaltevat pinnat tarkemmin, kuin jos olisin mitannut vain niiden kaltevuuskulmat.



Kuva 15. Vesitornin vesisäiliöt valmiina.

Viimeisenä mallinsin vesitornista heltan suojaelementit, sisätilojen putkistot ja kaikki loput portaikot. Portaikot oli hyvä tehdä kerralla, kun keksi, millä tavalla ne kannatti tehdä. Jotkin CAD-ohjelmat tarjoavat valmiita portaikkoja, joita pystyy luomaan helposti määrittämällä parametreja, kuten portaiden mitat ja lukumäärät. Myös SketchUpiin pystyy lataamaan tätä varten tehdyn lisäosan. Tämä ei kuitenkaan tuottanut haluttua tulosta omassa kokeilussani, joten päädyin mallintamaan kaikki portaikot itse alusta asti (ks. kuva 16).

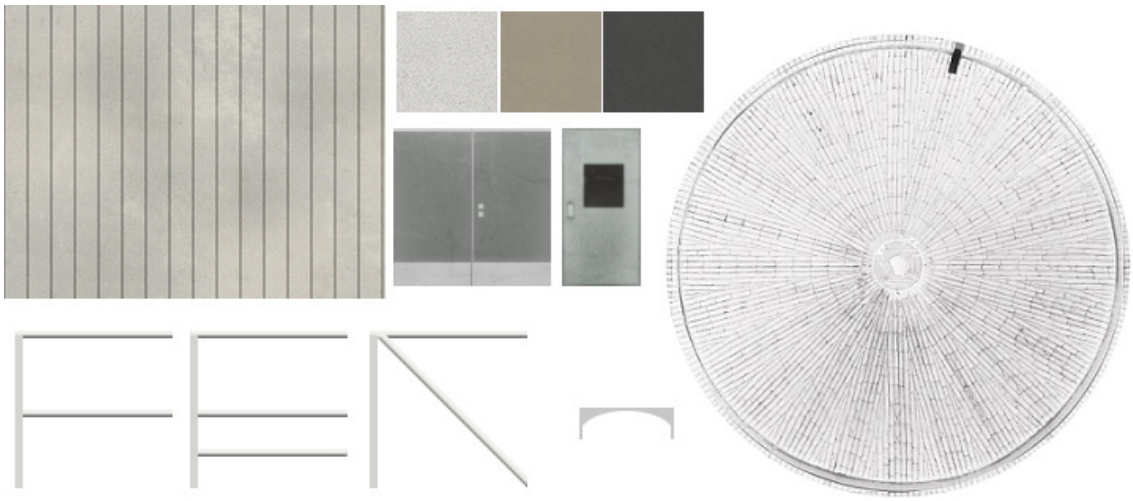


Kuva 16. Portaiden mallintaminen.

4.2 Materiaalit

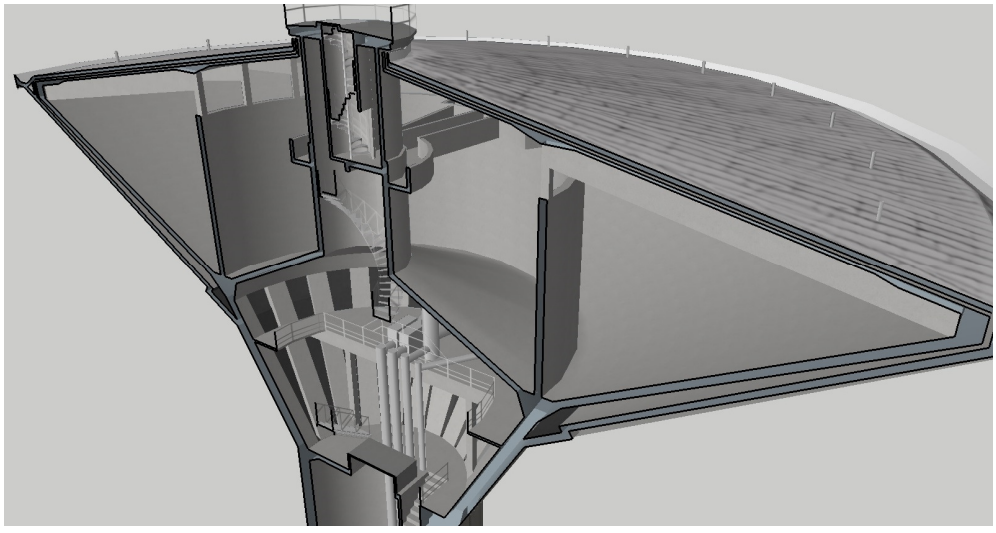
Vesitornin mallintamisen jälkeen aloin hahmotella myös käytettäviä materiaaleja. Alusta asti oli selvä, että ainakin jonkinlainen betonitekstuuri asetettaisiin suurimmalle osalle pinnoista. Tornin ulkopinnat ovat kuitenkin materiaaliltaan muutakin kuin pelkkää tasaista harmaata pintaa, ja varsinkin peltikaton mielenkiintoinen kuviointi oli sellainen, joka haluttiin saada myös lopullisessa mallissa näkyviin.

SketchUpissa tekstuurien mappaaminen pinnoille on todella helppoa siinä olevien työkalujen avulla. Ohjelmassa itsessään on mahdollista pelkästään diffuse- ja alpha-mappaus, mikä on kuitenkin usein riittävä, kun kyseessä on CAD-ohjelma. Jos kuitenkin haluaa renderöidä realistisempia otoksia ja käyttää esimerkiksi bump- ja reflection-mappeja, voi käyttää erillistä VRay-renderöijää, jonka avulla materiaaleista saa paljon realistisemmat [56]. Myös WebGL tukee materiaalien useita eri mappaustekniikoita, mutta vesitornin tapauksessa diffuse ja alpha olivat riittäviä. Kuvassa 17 ovat esiteltynä kaikki mallissa käytetyt tekstuurit.



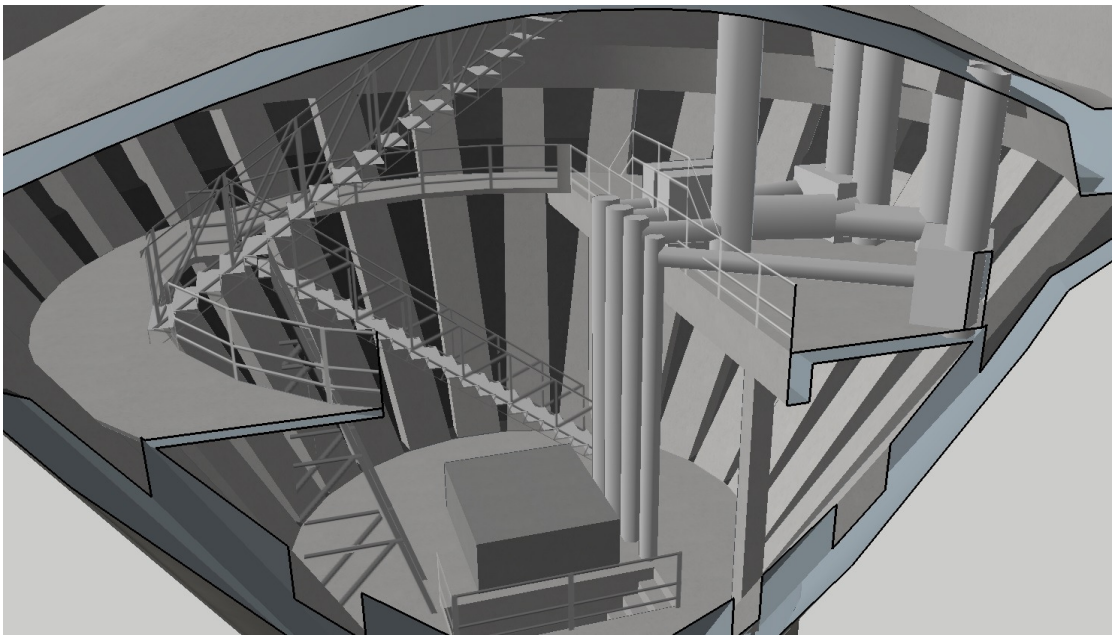
Kuva 17. Vesitornissa käytetyt tekstuurit.

Asetin SketchUpissa kaikille pinnoille tekstuurit, niin kuin ne tulisivat näkyviin lopullisessa tuotoksessa. Portaikoissa ja muualla olevat kaiteet tein kokonaan käyttäen tekstuureja, jolloin niitä ei tarvinnut mallintaa. Ainut geometria kaiteilla on siis tasainen litteä pinta, johon sama tekstuuri on asetettuna diffuse- ja alpha-mapiksi. Nämä tekstuurit ovat png-kuvia, joissa rgb-komponenttien lisäksi on myös läpinäkyvyys. Kun tekstuurit oli asetettu pinnoille (ks. kuva 18 ja 19), oli aika viedä kolmiulotteinen malli ulos SketchUpista, josta sen saa esimerkiksi käyttäen COLLADA-tiedostomuotoa. SketchUp Pro -lisenssillä 3D-mallin saa vietyä ulos myös muissa muodoissa, kuten .3ds, .obj ja .dwg. Ilmaisversiolla pystyy mallin viemään vain COLLADA, eli .dae-tiedostona, joka kuitenkin riittää moneen tarkoitukseen. [57.]



Kuva 18. Vesitorni teksturoituna valmiiksi.

Kun mallin vie ulos COLLADA-tiedostoksi, pystyy geometrian lisäksi valitsemaan myös muita liitettäviä tietoja, kuten pintojen kaksipuolisuuden ja tekstuurien asettuminen pinnoille. Nämä asetukset kannattaa huomioida, sillä näitä tietoja ei välttämättä liitetä tiedostoon automaattisesti.



Kuva 19. Keskikerrokset teksturoituina.

Mallin käyttäminen WebGL-sovelluksessa ei yleensä onnistu ensimmäisellä kerralla niin kuin haluaisi, ellei ole ennestään kokemusta vastaavasta tiedostojen ja tekstuurien käsittelystä. Vaikka Three.js-kirjasto sisältää työkalut monen eri formaatin, kuten COLLADAn, käsittelyyn, ei kaikki aina mene suunnitellusti. Itse yritin viedä vesitornin mallia monellakin eri tavalla sovellukseen, ennen kun löysin sopivan yhdistelmän ohjelmia ja työkaluja. Yritin ensin tapaa, jossa toin COLLADA-tiedoston 3ds Maxiin, josta pystyin viemään sen .obj-tiedostona ulos. Tämän jälkeen käytin Three.js:n Python-työkalua, jolla pystyi konvertoimaan .obj-tiedoston .js-tiedostoksi, jota taas Three.js osasi lukea ja käyttää sovelluksessa. Vaikka kuinka yritin, en kuitenkaan tällä tavoin onnistunut saamaan sekä tekstuureja että mallin geometriaa toimimaan oikein.

Seuraavaksi yritin käyttää Three.js:n omaa COLLADA-importeria, jolla mallin viemisen WebGL:ään pitäisi onnistua melko vaivattomasti. Vaikka mallin geometrian sain tällä tavalla näkyviin, ei tekstuureja tornissa näkynyt ollenkaan. Tämän jälkeen yritin pitkään säädellä eri asetuksia niin SketchUpissa kuin Three.js:ssäkin ja aloin jo luopua siitä toivosta, että tekstuurien käyttö onnistuisi lopullisessa sovelluksessa.

Lopulta kokeilin tapaa, jota minulle myös suositeltiin ja jolla tekstuurien käyttö varmimmin onnistuisi, eli Blender-ohjelman käyttö. Vein siis COLLADA-tiedoston Blenderiin ja yritin vastaavasti viedä mallin ulos .obj-tiedostona, jonka konvertoin .js-tiedostoksi. Tällä kertaa tekstuurit näkyivät, mutta aivan väärissä paikoissa ja väärän kokoisina, joten päätin, että asetan ne kokonaan uusiksi Blenderissä.

Kaikkien pintojen mappaaminen uudelleen ei onnistunut hetkessä, mutta ainakin sain tekstuurit kohdalleen. Blenderissä sain myös asetettua koko mallin värivarjostusalueet paikoilleen, sillä ne eivät olleet aivan oikein SketchUpin jäljiltä. Tämä onnistuu Blenderissä ja monessa muussakin ohjelmassa erityisellä *Edge Split* -työkalulla, jossa annetun asteluvun perusteella ohjelma muuttaa astetta pienemmät kulmat teräviksi ja muut pehmeiksi. Pintoja, niiden sivuja ja yksittäisiä pisteitä voi myös yksitellen asettaa teräviksi, jolloin ne eivät missään tapauksessa muutu pehmeiksi. Blenderillä muokkasin myös mallin geometriaa, leikkasin tornin kahtia ja poistin ylimääräisiä pintoja ja pisteitä.

Mallin .obj-tiedoston konvertoiminen .js-tiedostoksi siis onnistuu Three.js:n siihen tarkoitukseen tehdyllä Python-skriptillä. Skriptin saa ladattua Three.js:n github-sivuilta (*https://github.com/mrdoob/three.js/blob/master/utils/converters/obj/convert_obj_three.py*), ja se vaatii Pythonin version 2.7. Python-tiedosto ja mallin .obj- ja .mtl-tiedostot siirretään samaan hakemistoon python.exe-tiedoston kanssa, minkä jälkeen skripti ajetaan komentorivistä esimerkiksi komennolla ”*python convert_obj_three.py -i infile.obj -o outfile.js*”, jossa infile.obj on sisään menevä mallin .obj-tiedosto ja outfile.js on ulos tuleva malli JSON-muodossa. Tähän voidaan liittää oheen myös erilaisia komentoja, jotka vaikuttavat kaikki omalla tavallaan mallin konvertoimiseen.

Lopulta, kun sain tekstuurit mapattua ja muut mallin muokkaukset valmiiksi, huomasin, ettei Three.js osannut näyttää kaiteiden materiaaleissa olevaa alpha-mapia eli läpinäkyvyyttä. Myöskään niiden pintojen materiaalit, joiden tekstuurit monistetaan pinnoille useaan kertaan, eivät näkyneet oikein. Läpinäkyvyyden mappaukset vaativat hieman työtä taas Blenderissä, mutta suurimman avun toi se, kun huomasin, että lopullisessa .js-tiedostossa ei näitä asioita ollut määritelty oikein. Myös joidenkin tekstuurien tiedostonimet ja -sijainnit olivat väärin. Löysin onneksi ohjeet läpinäkyvyyden ja tekstuurin monistuksen asettamiseen, joten kaikki hoitui manuaalisella .js-tiedoston materiaalitietojen muokkauksella tekstieditorissa. Esimerkkikoodissa 1 ovat nähtävillä yhden vesitornissa käytetyn kaidemateriaalin määritykset .js-tiedostossa.

```
{
"DbgColor" : 238,
"DbgIndex" : 3,
"DbgName" : "aita3",
"colorAmbient" : [0.0, 0.0, 0.0],
"colorDiffuse" : [0.64, 0.64, 0.64],
"colorSpecular" : [0.5, 0.5, 0.5],
"illumination" : 2,
"mapAlpha" : "torni/aita3.png",
"mapDiffuse" : "torni/aita3.png",
"mapDiffuseWrap" : ["repeat", "repeat"],
"opticalDensity" : -1.0,
"specularCoef" : 96.078431,
"transparency" : 0.0
}
```

Esimerkkikoodi 1. Kaidemateriaalin määrittäminen JSON-tiedostossa.

Koodin ensimmäiset kolme riviä ovat materiaalin tunnistetietoja. Niiden alapuolella *colorAmbient* määrittää pinnan väriä pimeässä, *colorDiffuse* pinnan väriä valossa, jos sillä ei ole tekstuuria, ja *colorSpecular* pinnan kiillon väriä. Seuraavat tärkeät kohdat ovat *mapAlpha*, joka merkitsee alpha-mapin kuvatiedoston sijaintia, *mapDiffuse*, joka merkitsee diffuse-mapin kuvatiedoston sijaintia ja *mapDiffuseWrap*, joka tekee tekstuurista monistuvan. Viimeinen kohta, eli *transparency*, määrittää, kuinka läpinäkyviä tekstuurin läpinäkyvät kohdat ovat.

Vesitornin lisäksi tein tekstuurin myös sen ympärillä olevalle maa-alustalle, joka pyrkii havainnollistamaan sen oikeaa ympäristöä. Tämä ei suoraan onnistunut yhdestä kuvasta, vaan sitä varten piti yhdistellä ja muokata useita eri ilmakuvia. Tähänkin käytin pelkän kuvan lisäksi myös alpha-mapia, jolla sain siitä pyöreän muotoisen vähän läpikuultavilla reunoilla.

4.3 WebGL ja skenen luominen

Aloitin WebGL-sovelluksen teon käyttäen valmista Three.js-esimerkkiä pohjana. Tämä on kätevä tapa aloittaa, kun kaikki renderöintiin, kameran asettamiseen ja muuhun perusasiaan liittyvät koodit ovat valmiina. Toki nämäkin asiat kannattaa lukea ajatuksella

läpi ja jopa kokeilla muuttaa niitä, jotta oppii, mitä mikäkin rivi koodista tekee. Three.js:n sivuilta löytyy (*http://threejs.org/examples/*) esimerkkejä monesta erilaisesta sovelluksesta, joten helpoimmassa tapauksessa ei koodissa tarvitse tehdä paljoakaan omia muutoksia. Kuitenkin, jos sovelluksesta haluaa tehdä aivan omanlaisen, se vaatii ohjelmointitietämystä ja tutustumista WebGL:n ja Three.js:n metodeihin. Three.js tarjoaa myös verkossa toimivan editorin (*http://threejs.org/editor/*), jolla yksinkertaisen sovelluksen teko onnistuu kätevästi ilman ohjelmointia.

Skene on termi, jota käytetään muun muassa 3D-mallinnuksessa, animoinnissa ja pelien teossa. Se tarkoittaa kolmiulotteista tilaa, ympäristöä ja asetelmaa, jossa kaikki tapahtuu. Monesti yhdessä tuotoksessa on käytössä useitakin skenejä, joilla jokaisella on omat ominaisuutensa. Yksi voi tarkoittaa sisätilaa, toinen ulkotilaa ja kolmas jotain muuta paikkaa, tilannetta tai aikaa. Yhdessä skenessä on aina oma valaistus, kamerat ja objektit. WebGL-sovellukseen riittää useimmiten yksi skene, johon kaikki tarvittava liitetään, mutta useampienkin käyttäminen on mahdollista, vaikkakin raskaampaa.

Kun Three.js-sovelluksen teon aloittaa alusta asti, joutuu näkemään hieman vaivaa HTML-sivun tekoon, Three.js-kirjaston lisäykseen ja tarvittavien asetuksen ja metodien lisäämiseen. Liitteen 1 lyhyessä koodiesimerkissä on kaikki tarvittava yksinkertaisen sovelluksen tekoon. Script-tagin jälkeen määritetään sovellukselle skene, kamera ja renderöijä, joka on tyypiltään yleisimmin WebGL tai canvas. Tämän jälkeen asetetaan renderöintialueen koko, eli sovelluksen koko pikseleinä, ja liitetään se haluttuun HTML-sivun DOM (Document Object Model) -elementtiin.

Seuraavien rivien tehtävänä ennen render-funktiota on määrittää sovelluksessa nähtävästä kuutiosta sen koko ja materiaali ja lisätä se skeneen. Myös kameran sijaintia muutetaan. Render-funktion tehtävänä taas on määrittää kuution pyörimisanimaatio ja käskeä renderöimään skene sivulle. Silmukassa oleva requestAnimationFrame-metodi kutsuu render-funktiota ja näin ollen piirtää skenen näytölle maksimissaan 60 kertaa sekunnissa. Metodi on kätevä, koska se osaa myös pysäyttää silmukan silloin, kun käyttäjä navigoi jollekin toiselle välilehdelle, ja näin säästää tietokoneen prosessointitehoa ja mahdollisesti myös akkua. [49.]

Three.js tarjoaa monia eri tapoja kontrolloida sovelluksessa kameran sijaintia. Vaihtoehtoja löytyy muun muassa yksinkertaisesta mallin ympärillä pyörivästä kamerasta vapaasti liikutettavaan lentävään kameraan. Kameraohjain on helposti lisättävissä so-

vellukseen lataamalla sitä vastaava tiedosto ja liittämällä se koodiin. Omassa projektissani käytin Orbit Controls -nimistä ohjainta, joka sopi tarkoitukseen oikein hyvin.

Ulkoisen mallin tuominen Three.js-sovellukseen vaatii jonkin importerin käyttöä. Nämä ovat metodeita, jotka osaavat ladata tiedoston ja käsitellä sitä oikein. Itse olen kokenut parhaaksi importeriksi .js-tiedostoja lukevan JSONLoaderin. Tiedoston lataaminen ei kuitenkaan riitä, vaan malli on määriteltävä koodissa ensin, asetettava paikoilleen ja lisättävä skeneen. Esimerkkikoodissa 2 on esiteltynä, mitä kaikkea omassa projektissani jouduin vesitornin mallia varten tekemään, jotta sain sen paikoilleen ja näyttämään siltä kuin halusin. Materiaalin ja mallin asetuksia jouduin koodissa muokkaamaan, jotta mallin tekstuurit, värivarjostukset ja varjojen lankeamiset näkyivät oikein.

```
function addModelToScene( geometry, materials ) {
  geometry.computeVertexNormals();
  materials[0].shading = THREE.SmoothShading;
  var material = new THREE.MeshFaceMaterial( materials );
  for (i = 0, len = material.materials.length; i < len; i++) {
    material.materials[i].side = THREE.DoubleSide;
    material.materials[i].alphaTest = 0.7;
    material.materials[i].ambient = new THREE.Color(0xaaaaaa);
  }
  var model = new THREE.Mesh( geometry, material );
  model.scale.set(50,50,50);
  model.position.set(0,-70,0);
  model.castShadow = true;
  model.receiveShadow = true;
  scene.add( model );
}
```

Esimerkkikoodi 2. Mallin käsittely ja lisääminen Three.js-sovelluksen skeneen.

Lopullisessa tuotoksessa on siis kaksi eri vesitornin mallia, leikattu ja kokonainen, jotka kumpikin toin sovellukseen samalla tavalla. Yksinkertaisempaa geometriaa varten, kuten kuutiota, pintaa tai palloa, ei mallia kannata luoda toisaalla. Three.js tarjoaa tähän helppokäyttöiset metodit, joilla muutamilla parametreilla määritetään kappaleen koko ja muoto. Tämän jälkeen siihen voidaan lisätä materiaali käyttäen valmiita materiaaliluokkia, kuten MeshLambertMaterial tai MeshPhongMaterial, ja asettaa diffuse, alpha ja

muut mappaukset sekä halutut ominaisuudet. Tämän jälkeen sen sijainti voidaan määritellä ja lisätä skeneen.

Tällä tavalla loin maanpinnan, joka vaikkakin näyttää ympyrältä, on oikeasti neliskulmainen litteä taso. Ympyrän muoto on luotu käyttäen materiaalin alpha-mapia. Esimerkkikoodissa 3 on kaikki, mitä maanpinnan lisäämiseen tarvitaan koodin puolesta.

```
var floorTexture = new THREE.ImageUtils.loadTexture('maa.png',
    THREE.UVMapping(), function() {
  var floorMaterial = new THREE.MeshBasicMaterial( {
    map: floorTexture, side: THREE.DoubleSide,
    transparent: true, opacity: 0.8 } );
  var floorGeometry = new THREE.PlaneGeometry(500, 500, 10, 10);
  var floor = new THREE.Mesh(floorGeometry, floorMaterial);
  floor.position.y = -70.5;
  floor.rotation.x = Math.PI / 2;
  floor.rotation.z = (Math.PI / 2) + 0.5;
  floor.receiveShadow = true;
  scene.add(floor);
});
```

Esimerkkikoodi 3. Maanpinnan luominen ja lisääminen skeneen.

Koodista huomaa, että kaikki maan asettelu ja lisääminen tehdään funktiossa, joka suoritetaan vasta, kun maan tekstuuri on ladattu sovellukseen. Tällöin neliskulmainen pinta ei ilmesty sovellukseen ennen tekstuurin latautumista. Tämän jälkeen materiaali luodaan käyttäen Three.js:n valmista MeshBasicMaterial-luokkaa, johon pystyy tekstuurin lisäksi asettamaan myös sen läpinäkyvyyden, kaksipuolisuuden ja muita ominaisuuksia. Sen jälkeen koodissa luodaan maan geometria, joka yhdistetään materiaaliin. Lopuksi objekti asetetaan paikoilleen ja lisätään skeneen.

Valojen lisääminen Three.js:ssä on helppoa, kunhan vain tietää etukäteen, minkälaisen valaistuksen haluaa. Valotyyppejä on monia erilaisia, ja käsittelin niitä luvussa 2.5. Esimerkkikoodissa 4 nähdään auringonvalon lisääminen omaan sovellukseeni. Valona käytetään suuntavaloa, jolla siis ei ole sijaintia vaan vain suunta. Kuitenkin Three.js:ssä valon suuntavektori asetetaan sijaintia käyttäen, kuten koodista huomataan. Oletuksena suuntavalo osoittaa aina kohti koordinaatiston keskipistettä, jolloin valon säteet kulkevat sen sijainnista rinnakkain kohti pistettä (0,0,0). Suuntavalon kohteen voi myös

määrittää joksikin muuksi pisteeksi. Auringon sijainnin asettamisen jälkeen määrittelen sen varjojen ominaisuudet ja lisään sen lopulta skeneen, johon olen asettanut myös himmeän ambientvalon.

```
var sun = new THREE.DirectionalLight(0xffeecc);
sun.position.set(400,200,-300);
sun.shadowDarkness = 0.4;
sun.castShadow = true;
sun.shadowMapWidth = 1024;
sun.shadowMapHeight = 1024;
scene.add(sun);
```

Esimerkkikoodi 4. Auringonvalon luominen skeneen.

Kun sain sovelluksen valmiiksi, aloin kehittää sivuston ulkonäköä enemmän Arkkitehtuurimuseon tyyliin. Tutkin Arkkitehtuurimuseon oman sivuston rakennetta ja tyylitiedostoa, ja lisäsin sovelluksen ympärille samankaltaiset ylä- ja alaelementit sekä valikon. Sovimme myös, että sovelluksen oheen lisätään vesitorniin liittyvää muuta sisältöä, kuten tekstiä ja kuvia.

4.4 Onnistuminen ja käytettävyys

Lopullinen sovellus valmistui myöhemmin, kuin oli alun perin suunniteltu. Tämä johtui suurimmalta osin ongelmista mallin ja sen tekstuurien viennissä Three.js:ään sekä muista samaan aikaan sattuneista kiireistä. Sivusto saatiin kuitenkin valmiiksi ilman kummempia kiireitä. Kuvassa 20 näkyy valmis sivusto ja siinä olevat elementit, kuten alalaidassa olevat ohjeet mallin pyörittelyyn, siirtelyyn ja suurentamiseen. Sovellusikkunan vasemmassa alakulmassa on myös Kokokuva-painike, joka vaihtaa vesitornin mallia leikatun ja kokonaisen välillä.

Kuva 20. Lopullinen sivusto Lauttasaaren vesitornista valmiina.

Projekti onnistui mielestäni kokonaisuudessaan oikein hyvin. Se kesti odotettua kauemmin, mutta tein sen huolellisesti loppuun asti, ja kaikki ovat olleet tyytyväisiä lopputulokseen. Kuitenkaan työ ei edennyt aivan niin suoraviivaisesti, kuin työvaiheissa on esitelty, vaan välillä saatoin jumittua joihinkin kohtiin, joista ylipääsyyn tuli etsiä vaihtoehtoisia tapoja.

Yllätykset ja odottamattomat tilanteet kuuluvat osaksi jokaista sovelluskehitysprojektia, joten osasin ottaa ne huomioon myös työn edetessä. Ongelmienratkaisukyky on tässä erityisen tärkeää, ja siitä olen saanut aika paljon kokemusta aikaisemmissa projekteissa. Tällöin pitää osata etsiä ratkaisua oikeilla hakusanoilla ja oikeilta sivustoilta. Yleisimpiin ongelmiin löytyy vastaukset yleensä sivustolta nimeltä Stack Overflow (*http://stackoverflow.com/*), jossa käyttäjät kyselevät vastauksia vastaaviin ongelmiin. Välillä pitää myös osata luovuttaa joissakin vaikeissa asioissa eikä käyttää turhaan siihen liikaa aikaa. Sovellusta kehitettäessä pitää osata priorisoida sen ominaisuuksia eikä yrittää mahduttaa siihen kaikkea mahdollista, koska se on lopulta kuitenkin mahdotonta.

Vesitornin geometrian hahmottaminen oli alussa hieman hankalaa, mutta se helpottui, kun opin lukemaan pohjapiirroksia paremmin ja tutkimaan kuvamateriaalia. Tornin his-

torian lukeminen auttoi myös, kun tiesi, miksi mikin osa on rakennettu juuri niin kuin se on. Näiden asioiden oppiminen 1950-luvun arkkitehtuurista ja rakentamisesta on muutenkin ollut kiinnostavaa, joten mielenkiintoa tornin mallintamiseen on ollut.

Kaikkiin asioihin en täysin tyytyväinen kuitenkaan ole lopputuloksessa. Monia pieniä asioita olisi voinut tehdä paremmin mallinnusta tehdessä ja muissakin projektin vaiheissa. Kuitenkin eniten harmittaa näin jälkikäteen mallin monimutkaisuus. Tämä ei ilmene onneksi tietokoneella, jossa sovelluksen käyttö on sujuvaa. Mobiililaitteessa mallin kääntäminen on kuitenkin hidasta ja jopa epämiellyttävää. Kun mallin mallintaa SketchUpilla ja geometrian vie eteenpäin, voi malli olla hyvinkin monimutkainen. Kaikkien pintojen ja pisteiden lukumäärää on vaikea hahmottaa SketchUpissa, koska niitä ei suoraan näe. Tämän tajusin vasta jälkikäteen, kun uudelleenmallinnusta ei ollut enää järkevää tehdä, joten annoin sen olla. Jonkin verran turhia pintoja sain kuitenkin poistettua jälkikäteen Blenderillä, mutta tämä oli hidasta, joten koko mallin läpikäyminen olisi ollut todella työlästä.

Asia, jonka olisi voinut tehdä, jos olisi ollut siihen vielä aikaa, olisi ollut kolmannen vesitornimallin tuominen sovellukseen. Koska mallinsin vesitornin kokonaiseksi, sen toinen puolisko on jäänyt käyttämättömäksi. Leikkauskuvasta näkee toki paljon sen rakennetta ja siitä pystyy hahmottamaan, millainen torni kokonaisuudessaan on. Mutta sisätiloista, varsinkin keskikerrosten portaikoista ja muista tiloista (ks. kuva 19), nähdään vain puolet, mikä osaltaan harmittaa.

Yksi saaduista tuloksista on myös luonnollisesti työhön käytetty aika. Pidin koko ajan kirjaa ajankäytöstäni, ja arvioin, että yhteensä piirrosten mittaamiseen, vesitornin mallintamiseen ja sovelluksen tekoon kului noin 140 tuntia. Tämä sisältää myös torniin tutustumisen ja kaikki epäonnistumiset sekä virheiden korjaamiset, joten nopeamminkin sen olisi pystynyt tekemään. Kuitenkin, jos laskee täysiä työpäiviä, ei tekemiseen kuukautta enempää pitäisi kulua aikaa, mikä on mielestäni oikein nopeaa.

Käytettävyydeltään sovellus on hyvä, ja uskon, että siitä on monelle iloa ja apua. Sovellus tulee nähtäväksi Arkkitehtuurimuseon omalle sivustolle (*http://www.mfa.fi/*), ja siitä varmasti ollaan kiinnostuneita sitten, kun vesitorni puretaan. Mielestäni suoriuduin projektista hyvin, ja lopullinen sovellus sisältää ne ominaisuudet, joita työn alussa lähdettiin tavoittelemaan.

4.5 Tulevaisuus

Rakennusten mallintaminen

Rakennusten arkkitehtoninen mallinnus on kehittynyt paljon sitten ensimmäisten CAD- ja BIM-ohjelmistojen kehittämisen. Pienoismallien rakentaminen käsin on vaihtunut suurimmalta osin digitaaliseen 3D-mallintamiseen. Mallinnusohjelmat ovat itsessään kehittyneet paljon, eikä kehitys ole todellakaan hiljentynyt. Tietokoneiden suoritusteho kasvaa koko ajan, ja ohjelmistot pystyvät reaaliaikaisesti renderöimään jopa lähes fotorealistista kuvaa mallista. Myös suorituskykyä vaativat simulaatiot ja animointi kehittyvät entisestään ajan kuluessa.

Uskon, että käsin mallintaminen pysyy vielä elossa, mutta pääasiassa harrastuksena ja taidemuotona. Digitaalisilla malleilla on huomattavia etuja, ja tulevaisuudessa niitä käytetään vieläkin enemmän. Virtuaalitodellisuus ja lisätty todellisuus kehittyvät entisestään, jolloin rakennusmallit pystytään tuomaan osaksi todellisuutta. Mallien esitystavat lisääntyvät muutenkin ja monipuolistuvat, siten että ne ovat kaikille paremmin saatavilla.

WebGL ja muut modernit tekniikat

Interaktiiviset web-sovellukset ja pelit ovat olleet jo pitkän aikaa osana tavallista selainkäyttöä. Yleisesti ne ovat kuitenkin vaatineet ylimääräisten liitännäisten, kuten Flashin tai Silverlightin, asentamista, ja ne ovat olleet pääasiassa kaksiulotteisia. HTML5 ja WebGL ovat kuitenkin tarjonneet mahdollisuuksia riippumattomampiin ja monipuolisempiin sovelluksiin. Vaikka nämäkin tekniikat ovat olleet jo useamman vuoden käytössä, niitä kehitetään koko ajan eteenpäin.

IOS 8:n julkistaminen Applen mobiililaitteille toi vihdoin siihen myös WebGL-tuen, joka oli sen laitetukea ajatellen iso askel. Muutenkin WebGL:ää tuetaan nykyisin todella laajasti eri selaimissa, ja näin ollen sen käytettävyydenkään ei pitäisi laitetuesta riippua. Isot yritykset, kuten Sony ja Google, ovat myös alkaneet käyttää WebGL:ää, joten sen tulevaisuus näyttää valoisalta. Uskon, että WebGL ja muut modernit tekniikat, kuten Canvas, CSS3 ja SVG, syrjäyttävät Flashin ja muiden vanhojen tekniikoiden käytön selaimissa. Mahdollista voi myös olla muiden uusien tekniikoiden kehittyminen, jotka voivat ruveta kilpailemaan vastaavilla ominaisuuksilla.

Näitä tekniikoita kehitetään tulevaisuudessa sulavampaan ja riippumattomampaan suuntaan, mikä on myös vaatimuksena niillä yrityksillä, jotka haluavat laadukkaita ja moderneja web-sovelluksia. Myös sovellusten ja pelien tekoa pyritään tekemään helpommaksi, jotta kynnys ohjelmoinnin aloittamiseen olisi matalampi. Tässä olennaista on grafiikkakirjastojen, kuten Three.js:n, kehittyminen.

5 Yhteenveto

Insinöörityön tarkoituksena oli tutkia rakennuksen mallintamisen tekniikoita sekä 3D-mallintamista ja visualisointia yleisesti erilaisilla ohjelmistoilla ja menetelmillä. Rakennusmallinnus on kehittynyt paljon sitten ensimmäisten arkkitehtonisten pienoismallien, ja nykyään CAD- ja BIM-ohjelmilla luotuja malleja voidaan käyttää todella monipuolisiin tarkoituksiin. Mallinnustekniikoiden lisäksi työssä perehdyttiin myös moderneihin webtekniikoihin, kuten WebGL:ään, joilla tavallisiin selaimiin pystytään luomaan näyttäviä interaktiivisia 3D-sovelluksia.

Työssä oli tarkoitus luoda Lauttasaaren vesitornista kolmiulotteinen malli ja upottaa se WebGL-sovellukseen. Tavoitteena oli, että sovelluksessa pystyisi tutkimaan vesitornin ulkonäköä ja sen sisäistä ja ulkoista rakennetta. Sovelluksen tilaaja, Suomen Arkkitehtuurimuseo, toivoi sovelluksen toimivan tornin purkamisen jälkeen myös hyvänä visuaalisena esityksenä sen arkkitehtuurista.

Vaikka 3D-mallinnus ja WebGL olivat minulle ennestään tuttuja, sisälsi projekti silti paljon haasteita, joista oli päästävä yli. Rakennuksen geometriaa pystyy mittaamaan monilla eri tavoilla, mutta vesitornin mallinnukseen käytettävissä oli ainoastaan sen alkuperäisistä paperisista pohjapiirroksista tehdyt kopiot sekä valokuvamateriaalia sisä- ja ulkopuolelta. Insinöörityössä perehdyttiin hyväksi todettuihin menetelmiin kaikissa eri työvaiheissa.

Insinöörityö onnistui mielestäni oikein hyvin, ja se saavutti kaikki sille ennalta asetetut tavoitteet. Työ selvitti monipuolisesti, mitä jo rakennetun rakennuksen mallintaminen ja sovelluksen teko vaatii kokonaisuudessaan. Tuloksiksi saatiin työhön vaadittavan ajan lisäksi myös menetelmät ja tekniikat, joilla se kannattaa tehdä. Vaikka työtä tehdessä tuli vastaan myös vaikeita ongelmia, muun muassa mallin käyttämisessä sovelluksessa tekstuureineen, ne olivat yleensä kuitenkin ratkaistavissa. Ongelmiin löytyneet ratkaisut varmasti myös hyödyttävät muita, jotka tekevät vastaavanlaisia projekteja. Jos Arkkitehtuurimuseo luo muistakin vanhoista rakennuksista vastaavia sovelluksia, uskon tämän työn auttavan projekteissa huomattavasti. Tällöin pystytään paremmin ennalta arvioimaan niihin tarvittavia resursseja, samoin kuin sitä, mitä ominaisuuksia sovelluksiin on mahdollista lisätä.

Toivon, että insinöörityö toimii hyvänä oppaana kaikille asiasta kiinnostuneille. Työtä pystyisi jatkamaan pidemmällekin tutkimalla esimerkiksi vaihtoehtoisia mallinnusmenetelmiä eri ohjelmilla tai jopa kokeilemalla rakennuksen geometrian mittaamista laserkeilaimella, jolloin rakennuksen mallin käyttäminen WebGL-sovelluksessa vaatisi erilaisia toimenpiteitä, joita tässä työssä ei käyty läpi. Jatkotutkimuksiin olisi mielenkiintoista tutustua, koska uskon, että aihetta pystyisi viemään vielä paljon pidemmälle.

Lähteet

1 Arkkitehtuurimuseo. Verkkodokumentti. Arkkitehtuurimuseo. <http://www.mfa.fi/>. Luettu 30.3.2015.

2 A Brief History of Architectural Model Making. Verkkodokumentti. The Model Making Company. <http://www.modelmaking.co.uk/a-brief-history-of-architectural-model-making/>. Luettu 4.3.2015.

3 Astbury, Jon. 2014. Architects do it with models: the history of architecture in 16 models. Verkkodokumentti. The Architectural Review. <http://www.architectural-review.com/comment-and-opinion/architects-do-it-with-models-the-history-of-architecture-in-16-models/8658964.article>. 25 February 2014. Luettu 4.3.2015.

4 Madsen, David. 2011. Engineering Drawing and Design. New York: Delmar.

5 Bozdoc, Marian. The History of CAD. Verkkodokumentti. Marian Bozdoc's History of CAD. <http://mbinfo.mbdesign.net/CAD-History.htm>. Luettu 4.3.2015.

6 Yes, You Can Afford a CornerCavern. Verkkodokumentti. WorldViz. <http://www.worldviz.com/newsletter/yes-you-can-afford-a-cave>. Luettu 27.3.2015.

7 Wall, Mike. 2011. Scientists Abuzz Over Controversial Rumor that God Particle Has Been Detected. Verkkodokumentti. Live Science. <http://www.livescience.com/13853-higgs-boson-signal-lhc-cern.html>. April 22, 2011. Luettu 7.3.2015.

8 O'Toole, James. 2014. 3-D-printed organs are on the way. Verkkodokumentti. CNNMoney. <http://money.cnn.com/2014/11/04/technology/innovationnation/3d-printed-organs/>. November 4, 2014. Luettu 7.3.2015.

9 About. 2003. Verkkodokumentti. Design Computing. <http://bim.arch.gatech.edu/?id=402>. Last updated: October 21, 2003. Luettu 7.3.2015.

10 Building Information Modeling. 2002. White Paper. Autodesk, Inc.

11 Eastman, Charles, Fisher, David, Lafue, Gilles, Lividini, Joseph, Stoker, Douglas & Yessios, Christos. 1974. An Outline of the Building Description System. Research Report. Institute of Physical Planning.

12 BIM 101: An Introduction to Building Information Modeling. Video. Autodesk. <http://www.autodesk.com/solutions/building-information-modeling/overview>. Katsottu 9.3.2015.

13 Sutter Medical Center Castro Valley, USA. Verkkodokumentti. Tekla BIM Awards. <http://www.tekla.com/ae/bim-awards-2014/bimmodel1.html>. Luettu 9.3.2015.

14 Lalit Narayan, K., Mallikarjuna Rao, K., & Sarcar, M.M.M. 2008. Computer Aided Design and Manufacturing. New Delhi: Prentice-Hall of India.

15 Haapaniemi, Eija. 2013. Rakennusalan mittaustekniikka. Verkkodokumentti. Tampereen teknillinen yliopisto. <http://www.tut.fi/fi/tietoa-yliopistosta/laitokset/rakennustekniikka/tutkimus/rakennustuotanto-ja--talous/rakennusmittaukset/index.htm>. 21.05.2013. Luettu 10.3.2015.

16 Ympäristön 3D-mallinnus. Verkkodokumentti. MML Paikkatietokeskus FGI. <http://www.fgi.fi/fgi/fi/teemat/ymp%C3%A4rist%C3%B6n-3d-mallinnus>. Luettu 10.3.2015.

17 3D Laser Scanning. 2010. Verkkodokumentti. Terrain Surveys. <http://www.terrainsurveys.co.uk/3d-laser-scanning/11/>. 22nd December 2010. Luettu 11.3.2015.

18 Joala, Vahur. 2006. Laserkeilauksen perusteita ja mittauksen suunnittelu. Verkkodokumentti. Google. <https://drive.google.com/file/d/0B3MfAq-wXowlN2Q4MzJlYjktZTA5Ni00ZGMyLTlkOWUtNTQzMDIwZTI3NDVm/view> 30.11.2006. Luettu 11.3.2015.

19 Mellen, Mickey. 2013. Google continues to expand 3D imagery, but starting to phase out 3D Warehouse. Verkkodokumentti. Google Earth Blog. <http://www.gearthblog.com/blog/archives/2013/08/google-continues-to-expand-3d-imagery.html>. August 9, 2013. Luettu 28.3.2015.

20 Opensource image based 3d modeling software. Verkkodokumentti. Insight3d. <http://insight3d.sourceforge.net/>. Luettu 12.3.2015.

21 Workshop on rapid prototyping design practice. Verkkodokumentti. The Constitute. <http://theconstitute.org/form-inspires-function-workshop/>. Luettu 28.3.2015.

22 Collada - Digital Asset and FX Exchange Schema. Verkkodokumentti. Collada. <https://collada.org/>. Luettu 14.3.2015.

23 Suvo & Stonecypher, Lamar. 2010. Common File Types for 3D CAD Software. Verkkodokumentti. Bright Hub Engineering. <http://www.brighthubengineering.com/cad-autocad-reviews-tips/23469-common-file-types-for-3d-cad-software/>. Updated: 10/4/2010. Luettu 14.3.2015.

24 Autodesk, Inc. History. Verkkodokumentti. FundingUniverse. <http://www.fundinguniverse.com/company-histories/autodesk-inc-history/>. Luettu 15.3.2015.

25 Autodesk. Verkkodokumentti. Autodesk. <http://www.autodesk.fi/>. Luettu 15.3.2015.

26 ProgeCAD. Verkkodokumentti. ProgeSOFT. <http://www.progesoft.com/en>. Luettu 15.3.2015.

27 Archicad. Verkkodokumentti. Micro Aided Design. <http://www.mad.fi/mad/archicad.html>. Luettu 15.3.2015.

28 Revit. Verkkodokumentti. Autodesk. <http://www.autodesk.fi/products/revit-family/overview>. Luettu 15.3.2015.

29 SKETCHUP. Verkkodokumentti. Micro Aided Design. <http://www.mad.fi/mad/sketchup.html>. Luettu 15.3.2015.

30 FreeCAD: An Open Source parametric 3D CAD modeler. Verkkodokumentti. FreeCAD. <http://www.freecadweb.org/>. Luettu 15.3.2015.

31 Kurhila, Jaakko. 1997. Käyrät ja pinnat. Verkkodokumentti. Helsingin yliopiston graafinen oppiaineisto. <http://www.cs.helsinki.fi/group/goa/mallinnus/3dprim.html>. 17.2.1997. Luettu 14.3.2015.

32 5 Best Software Programs for Digital Sculpting. Verkkodokumentti. MakeItCG. <http://makeitcg.com/digital-sculpting-software/2447/>. Luettu 14.3.2015.

33 MAXScript Help. 2011. Verkkodokumentti. Autodesk. <http://docs.autodesk.com/3DSMAX/14/ENU/MAXScript%20Help%202012/>. Last updated: 2011/03/16. Luettu 15.3.2015.

34 Masters, Mark. 3ds Max vs. Maya: Is One Better than the Other?. Verkkodokumentti. Digital-Tutors. <http://blog.digitaltutors.com/3ds-max-vs-maya-is-one-better-than-the-other/>. Luettu 16.3.2015.

35 Fronczak, Tom. 2013. Top 20 Most Essential Software for Artists and Designers. Verkkodokumentti. Animation Career Review. <http://www.animationcareerreview.com/articles/top-20-most-essential-software-artists-and-designers>. September 4, 2013. Luettu 16.3.2015.

36 Choosing a 3D Animation Software Package. Verkkodokumentti. Animation Arena. <http://www.animationarena.com/choosing-a-3d-animation-software-package.html>. Luettu 16.3.2015.

37 Shiratuddin, Mohd Fairuz, Kitchens, Kevin & Fletcher, Desmond. 2008. Virtual Architecture: Modeling and Creation of Real-Time 3D Interactive Worlds. Lulu.com.

38 Russell, Eddie. Elliminate Texture Confusion: Bump, Normal and Displacement Maps. Verkkodokumentti. Digital-Tutors. <http://blog.digitaltutors.com/bump-normal-and-displacement-maps/>. Luettu 21.3.2015.

39 Reflect/Refract Map. 2014. Verkkodokumentti. Autodesk. <http://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/3DSMax/files/GUID-D3621A9C-931C-4D24-8D32-60C2494F5F9E-htm.html>. Dec 17, 2014. Luettu 21.3.2015.

40 Airaksinen, Harri. 2000. 3D-grafiikka ja animaatio. Luentomateriaali. EVTEK-ammattikorkeakoulu.

41 Anton & Max. Best V-Ray settings - Indirect illumination. Verkkodokumentti. RenderStuff. <http://renderstuff.com/vray-indirect-illumination-best-settings-cg-tutorial/>. Luettu 22.3.2015.

42 Slick, Justin. 3D Lighting Techniques - Lighting a 3D Scene. Verkkodokumentti. About Tech. <http://3d.about.com/od/Creating-3D-The-CG-Pipeline/a/3d-Lighting-Techniques-Standard-3d-Lighting.htm>. Luettu 22.3.2015.

43 Dravid, Atul. Understanding IES Lights. Verkkodokumentti. CGArena. <http://www.cgarena.com/freestuff/tutorials/max/ieslights/>. Luettu 22.3.2015.

44 Silverman, Matt. 2012. The History of HTML5. Verkkodokumentti. Mashable. <http://mashable.com/2012/07/17/history-html5/>. JUL 17, 2012. Luettu 23.3.2015.

45 HTML5 SVG. Verkkodokumentti. W3Schools. <http://www.w3schools.com/html/html5_svg.asp>. Luettu 23.3.2015.

46 HTML5 Canvas. Verkkodokumentti. W3Schools. <http://www.w3schools.com/html/html5_canvas.asp>. Luettu 23.4.2014.

47 Danchilla, Brian. 2012. Beginning WebGL for HTML5. Apress.

48 Features. 2012. Verkkodokumentti. GitHub. <https://github.com/mrdoob/three.js/wiki/Features>. 26 Nov 2012. Luettu 26.3.2015.

49 Creating a scene. Verkkodokumentti. Three.js. <http://threejs.org/docs/index.html#Manual/Introduction/Creating_a_scene>. Luettu 26.3.2015.

50 Elomaa, Ville. 2009. Lauttasaaren Vesitorni. Seinäjoki: Lauttasaari-seura.

51 Lust, Jari. Vesitornikortit. Verkkodokumentti. Vesitornit. <http://www.vesitornit.net/vesitornikortit.html>. Luettu 28.3.2015.

52 Mannila, Johanna. 2013. Lauttasaaren vesitorni sai purkutuomion. Verkkodokumentti. Helsingin Sanomat. <http://www.hs.fi/kaupunki/a1378697631032>. 9.9.2013. Luettu 28.3.2015.

53 Kuntoarvio: Lauttasaaren vesitorni. 2008. Helsingin kaupunki.

54 Sektori. 2013. Poikkileikkauskuva Lauttasaaren vesitornista. Verkkodokumentti. Wikimedia Commons. <http://commons.wikimedia.org/wiki/File:Lauttasaari_water_tower_cross-section_with_numbers.svg>. 25. elokuuta 2013. Luettu 28.3.2015.

55 Vesitornin pohjapiirrokset. 1958. Helsingin Rakennusvalvontavirasto.

56 Bump Maps. Verkkodokumentti. VRay. <http://www.vray.com/vray_for_sketchup/manual/bump_maps.shtml>. Artikkelin pvm. Luettu 4.4.2015.

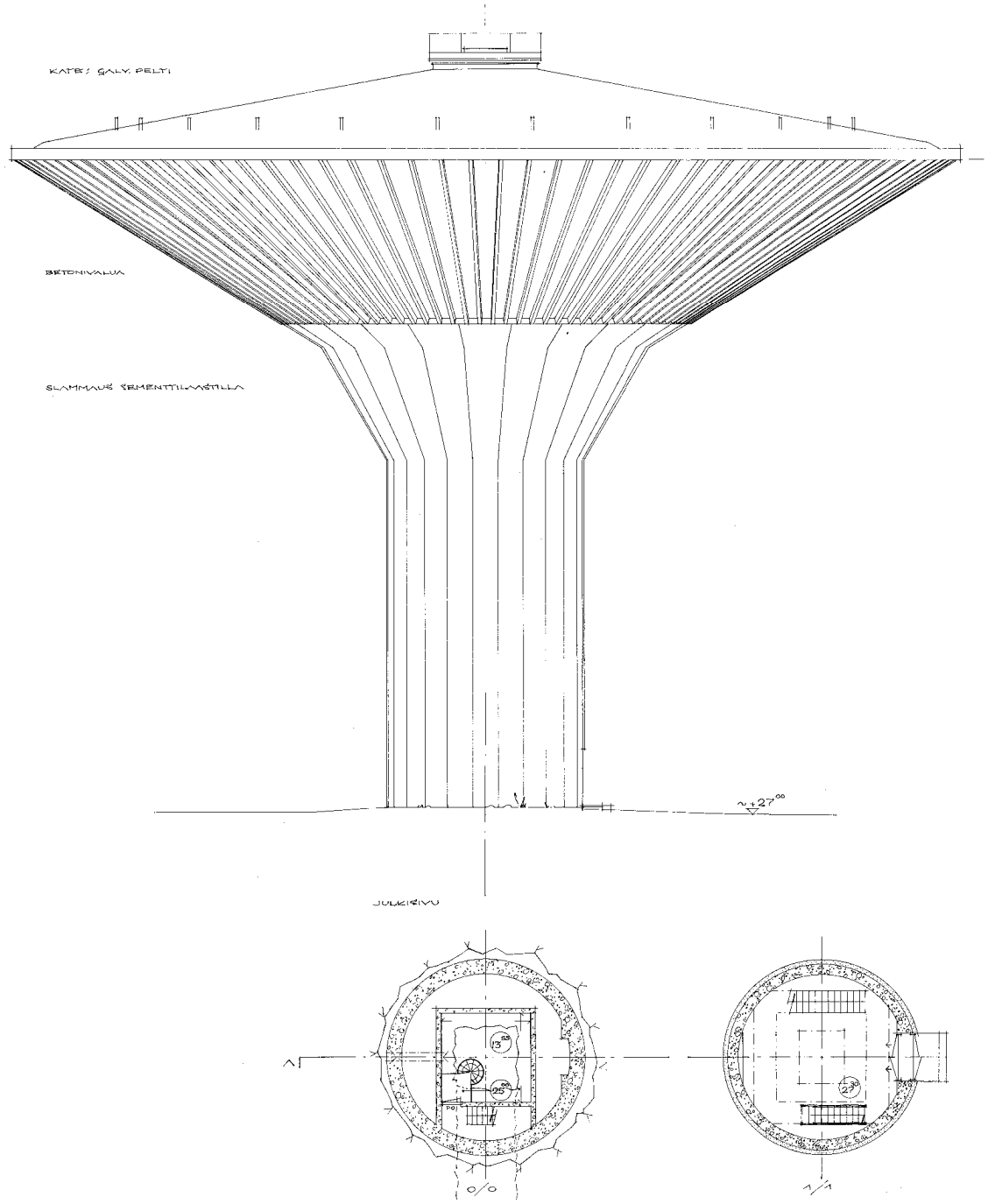
57 Can I export my SketchUp models to other programs or formats? Verkkodokumentti. SketchUp. <http://help.sketchup.com/en/article/36203>. Luettu 5.4.2015.

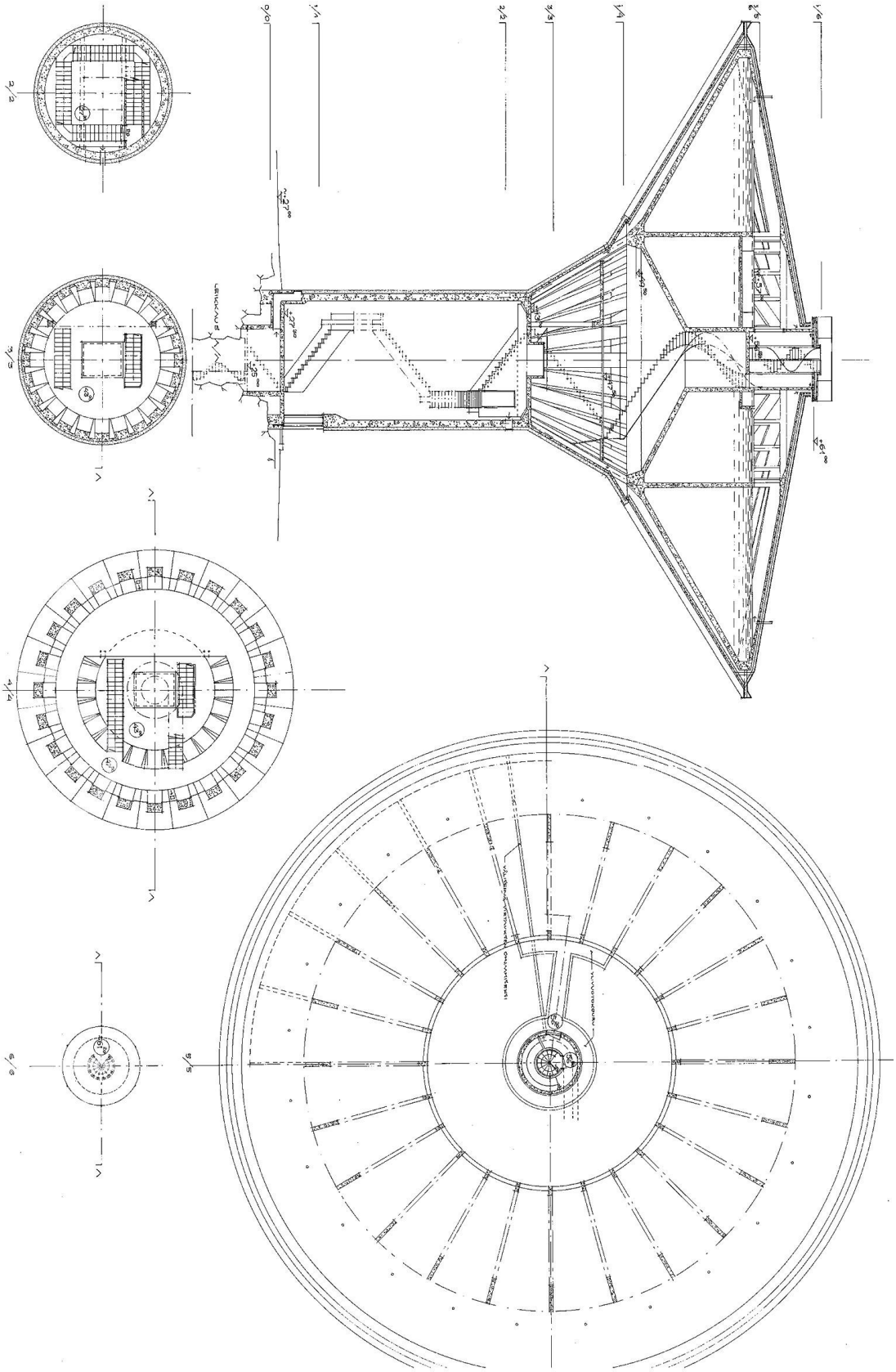
Esimerkkikoodi Three.js-animaatiosta

```
<html>
  <head>
    <title>My first Three.js app</title>
    <style>
      body { margin: 0; } canvas { width: 100%; height: 100% }
    </style>
  </head>
  <body>
    <script src="js/three.min.js"></script><script>
      var scene = new THREE.Scene();
      var camera = new THREE.PerspectiveCamera(
        75, window.innerWidth/window.innerHeight, 0.1, 1000 );
      var renderer = new THREE.WebGLRenderer();
      renderer.setSize( window.innerWidth, window.innerHeight );
      document.body.appendChild( renderer.domElement );

      var geometry = new THREE.BoxGeometry( 1, 1, 1 );
      var material = new THREE.MeshBasicMaterial(
        { color: 0x00ff00 } );
      var cube = new THREE.Mesh( geometry, material );
      scene.add( cube );
      camera.position.z = 5;

      var render = function () {
        requestAnimationFrame( render );
        cube.rotation.x += 0.1;
        cube.rotation.y += 0.1;
        renderer.render(scene, camera);
      };
      render();
    </script>
  </body>
</html>
```

[49.]

Lauttasaaren vesitornin pohjapiirrokset





[55.]