

CITYENGINE- JA SKETCHUP-MALLINNUSOHJELMIEN KÄYTTÖ PORIN KAUPUNGIN SUUNNITTELUSSA

Kirsi-Maria Viljanen

Opinnäytetyö
Maanmittaustekniikan koulutusohjelma
Maanmittausinsinööri (AMK)

2016

Tekniikka ja liikenne
Maanmittaustekniikan koulutusohjelma

Tekijä	Kirsi-Maria Viljanen	Vuosi	2016
Ohjaaja	Jaakko Lampinen		
Toimeksiantaja	Porin kaupunki, konsernihallinto		
Työn nimi	CityEngine- ja SketchUp-mallinnusohjelmien käyttö Porin kaupungin suunnittelussa		
Sivu- ja liitemäärä	82 + 14		

Tämän opinnäytetyön tavoitteena oli selvittää, miten CityEngine- ja SketchUp-ohjelmistot soveltuvat kolmiulotteiseen havainnollistamiseen Porin kaupungin suunnittelutyössä. Työssä tarkasteltiin ohjelmistojen tarjoamia mallinnusmetelmiä sekä niiden vahvuuksia ja heikkouksia. Lisäksi tarkasteltiin kolmiulotteisen mallin julkaisua verkkoselaimessa.

Opinnäytetyön teoreettinen pohja kerättiin kirjallisuustutkimuksen perusteella. Kirjallisuustutkimuksessa tarkasteltiin 3D-kaupunkimallinnusta, mallinnuksessa käytettäviä aineistoja, kaupunkimallinnuksen hyödyntämisestä Suomen kuntasektorilla sekä asemakaavoituksen prosessia. Käytännön osuudessa laadittiin tutkimusalueesta kolmiulotteiset kaupunkimallit CityEngine- ja SketchUp-ohjelmilla. Kaupunkimallien laatimisen pohja-aineistona hyödynnettiin ilmakuvaa, kaava-aineistoa, digitaalista pohjakarttaa sekä korkeusmallia.

Tutkimuskohteesta saatujen mallinnustulosten perusteella molemmille työssä käytetyille 3D-mallinnusohjelmille on tarvetta Porin kaupungin suunnittelussa. SketchUp-ohjelman helppokäyttöisyys ja yksinkertaisuus mahdollistavat ohjelman käyttöönoton suunnittelijoiden keskuudessa. Etenkin pieniin, suunnittelutyötä tukeviin luonnostelutarkoituksiin SketchUp on erinomainen työkalu. Heikkouksina ovat korkeusmallin hankala tuominen ohjelmaan, koordinaatistosidonaisuuden sekä verkkoselainpohjaisen julkaisumahdollisuuden puuttuminen. CityEnginen ylivoimaisena etuna on visuaalisesti tarkkojen ja laajojen alueiden mallintaminen hetkessä. Ohjelma soveltuu kaavojen, puistosuunnitelmien kuin tiesuunnitelmien mallintamiseen. Etuna on myös mallin helppo muunneltavuus ja virtuaalimallin yksinkertainen julkaisumahdollisuus verkkoselaimessa. Heikkoutena voidaan pitää monimutkaisuutta, jolloin ohjelmiston hyödyntäminen suunnittelutyössä tulee keskittää muutamille työntekijöille.

Asiasanat	3D-mallinnus, CityEngine, kaupunkimalli, SketchUp
Muita tietoja	CityEngine- (linkki liittessä 13) ja SketchUp-virtuaalimalliesitykset

Technology, Communication and Transport
Degree Programme in Surveying

Author	Kirsi-Maria Viljanen	Year	2016
Supervisor	Jaakko Lampinen		
Commissioned by	Town of Pori, Concern Management		
Subject of thesis	Using the CityEngine and SketchUp Modelling Programs in the Town of Pori planning		
Number of pages	82 + 14		

The aim of this study was to find out how the City Engine and the SketchUp programs are suitable for designing three-dimensional visualization of the town of Pori. This study was examined the software modeling methods as well as their strengths and weaknesses. The aim was also to investigate a three-dimensional model release in a web browser.

The theoretical part was based on a literature survey. Based on the literature study, the 3D city modeling, materials used in the modeling, city modeling utilization in Finnish local government sector and the city planning process were examined. In the practical part a three-dimensional city models with City Engine and SketchUp software of the research area were created. Aerial photo and a detailed plan, a digital base map, as well as digital elevation model were used in the city modeling.

According to this study, both the 3D modeling programs are needed in designing the city of Pori. Ease of use and simplicity of the SketchUp program can be used among a number of employees. In particular, SketchUp is a great tool for modeling small areas and illustrations of plans. The weaknesses are difficulties to import elevation model, the absence of the coordinate system and the lack of a Web browser-based publication. The main advantages of the CityEngine program are accurate visualization and instant modeling of extensive areas. The program is suitable for land use planning, park planning and road planning. The advantage is also the publishing possibility of a virtual model in the web browser and easy adaptability of the model. The weakness is complex usability.

Key words	3D modeling, city model, City Engine, SketchUp
Special remarks	The thesis includes CityEngine (weblink in Annex 13) and SketchUp multimedia presentations.

SISÄLLYS

1	JOHDANTO	8
2	3D-KAUPUNKIMALLINNUS	10
2.1	3D-kaupunkimalli	10
2.2	Kaupunkimallinnuksen muodot	10
2.3	Tarkkuustasot	12
2.4	3D-kaupunkimallinnuksessa käytettävät aineistot.....	14
2.5	Kaupunkimallinnuksen käyttömahdollisuudet	15
2.6	3D-kaupunkimallinnuksen käyttö kuntien kaupunkisuunnittelussa.....	16
2.7	Esrin CityEngine-mallinnusohjelma.....	19
2.8	Trimblen SketchUp-mallinnusohjelmisto	21
3	ASEMAKAAVOITUS.....	22
3.1	Asemakaava	22
3.2	Asemakaavaprosessi.....	23
3.3	Osallistumis- ja arviointisuunnitelma	25
3.4	Asemakaavamerkinnot ja -määräykset	26
4	AINEISTOT JA MENETELMÄT	27
4.1	Tutkimusalue.....	27
4.2	Tutkimuksen yleinen kulku	29
4.3	Alkuvalmistelut ja primaarianalyysit	30
4.3.1	Aineistot	30
4.3.2	Aineistojen esikäsittely ja muokkaus ArcGIS-ohjelmalla	31
4.4	Mallinnus CityEngine-ohjelmistolla.....	34
4.4.1	Sääntöihin perustuva mallinnus.....	37
4.4.2	Julkisivukuvaan perustuva mallinnus	51
4.4.3	Mallintaminen piirtämällä	55
4.5	Mallinnus SketchUp -ohjelmistolla	59
4.6	Ohjelmistojen välinen tiedonsiirto.....	64
4.7	3D-mallien julkaisu.....	66
5	3D-MALLINNUKSIEN TARKASTELU.....	69
5.1	Ohjelmistojen vahvuudet kaupunkimallinnuksessa.....	69
5.2	Ohjelmistojen heikkoudet kaupunkimallinnuksessa	71

5.3	Ohjelmistojen sopivuus Porin kaupungin suunnittelutarpeisiin.....	74
6	JOHTOPÄÄTÖKSET	76
	LÄHTEET.....	78
	LIITTEET	82

ALKUSANAT

Haluan kiittää Porin kaupungin konsernihallinnon tietohallintoyksikköä opinnäytetyöni teknisten puitteiden mahdollistamisesta. Työni ohjaajana toimi Osmo Leppäniemi, tietohallinnon pääsuunnittelija, jota haluan kiittää tuesta työni aikana. Kiitos kuuluu myös tietopalveluasiantuntija Timo Widbomille. Lisäksi haluan kiittää työni valvojaa Jaakko Lampista Lapin ammattikorkeakoulusta.

Pori 29.1.2016

Kirsi-Maria Viljanen

KÄYTETYT MERKIT JA LYHENTEET

CAD	Tietokoneavusteinen suunnittelu, (Computer Aided Design)
CGA	CityEngine-ohjelman ohjelmointikieli, (Computer Generated Architecture)
COLLADA	3D-formaatti, jolla esitetään 3D- geometriaa, (Collaborative Design Activity)
DEM	Digitaalinen korkeusmalli, (Digital Elevation Model)
DDF	Korkeustietoa sisältävä aineistomuoto
DXF	CAD-tiedostomuoto
DSM	Digitaalinen pintamalli, (Digital Surface Model)
DTM	Digitaalinen maastomalli, (Digital Terrain Model)
DWG	CAD-tiedostomuoto
FileGDB	Spatiaalista dataa käsittävä Esrin tiedosto, File Geodatabase)
GIS	Paikkatietojärjestelmä, (Geographical Information System)
LAS	Laserkeilausaineiston kolmiulotteista pistepilviaineistoa sisältävä formaatti
LoD	3D-kaupunkimallin tarkkuustaso, (Level of Detail)
MP4	Videokuvaa ja ääntä sisältävä säiliömuoto, (MPEG-4)
OBJ	3D-formaatti, jolla esitetään 3D-geometriaa
Paikkatieto	Tietoa kohteista, joiden paikka Maan suhteen tunnetaan. Paikkatiedossa tiedolla on tietosisältönsä lisäksi sijainti.
Tarkkuustaso	Kuvaa 3D-kaupunkimallin tarkkuustasoa, (Level of Detail, LoD)
TIN	Epäsäännölliseen kolmioverkkoon perustuva korkeusmallipinta, (Triangulated Irregular Network)

1 JOHDANTO

Kolmiulotteinen kaupunkimallinnus on yhä laajemmin tullut osaksi kuntien kaupunkisuunnittelua ja kaavojen havainnollistamista. 3D-kaupunkimallinnus on kolmiulotteista digitaalista mallinnusta, joka esittää maastoa, rakennuksia, kasvillisuutta, infrastruktuuria ja muita kaupunkikohteita. Mallinnettavat kohteet voivat olla pieniä tai suuria, ja malli voi kuvata joko todellista tai kuvitteellista kohdetta. Nykyisten mallinnusohjelmien avulla voidaan kuvata laajoja kaupunkialueita kohtuullisella työpanoksella.

Kolmiulotteisen mallinnuksen avulla suunnittelukohteiden havainnollistamien ja kuvaus onnistuvat kaksiulotteista kuvaa paremmin. Myös suunnittelijoiden, päättäjien ja asukkaiden välinen vuorovaikutus helpottuu, kun kaavoitettu alue pystytään hahmottamaan rakennettuna mallina olemassa olevassa ympäristössä. Mallinnuksen perusteella havaitut ongelmatilanteet pystytään korjaamaan jo suunnitteluvaiheessa, mikä säästää aikaa ja rahaa.

Opinnäytetyöni aihe syntyi intohimostani hyödyntää 3D-mallinnusta työssäni Porin kaupungin kaupunkisuunnittelussa. Myös Porin kaupungilla oli tavoitteena selvittää 3D-mallinnusohjelmien soveltuvuutta kaupungin suunnitteluun. Porin kaupungilla on käytössä Esrin CityEngine- ja Trimblen SketchUp-mallinnusohjelmat, joista SketchUp-ohjelmasta on kaupungissa jonkin verran käyttökokemuksia. Yhteisten päämäärien saavuttamiseksi opinnäytetyöni on tehty toimeksiantona Porin kaupungin konsernihallinnolle.

Työn tavoitteena on selvittää kuinka CityEngine ja SketchUp-ohjelmistot soveltuvat Porin kaupungin suunnittelutyön havainnollistamiseen etenkin kaupunkisuunnittelun kannalta. Opinnäytetyössäni selvitetään minkälaisia menetelmiä kyseiset ohjelmistot tarjoavat rakennetun- sekä luonnonympäristön 3D-mallintamiseen. Lisäksi työssä tarkastellaan kolmiulotteisen mallin esittämistä interaktiivisessa selainsovelluksessa. Tarkoituksena ei ole luoda fotorealistisesti parasta mahdollista kuvaa mallinnettavasta alueesta, vaan tarkastella ja kokeilla erilaisia ohjelmien tarjoamia mallinnusmenetelmiä. Koska CityEngine on varsin uusi 3D-mallinnusohjelmisto ja sen käyttö kuntien suunnittelussa on vielä

vähäistä, tarkoitukseni on myös tuottaa mallinnussääntöjä, joita voidaan käyttää apuna myös muiden kuntien suunnittelussa.

Työn ensisijaisena mallinnusohjelmistona käytetään CityEngine-ohjelmistoa, sillä ohjelma kattaa SketchUp-ohjelmistoon verrattuna monipuolisemmat kaupunkimallinnusmahdollisuudet. Kumpikin ohjelmisto on uusi Porin kaupungin suunnittelussa, joista halutaan vertailevia käyttökokemuksia. Molemmat ohjelmistot ovat myös minulle tuntemattomia, joten mallinnus perustuu täysin ohjelmistojen sivuilta löytyviin käyttöoppaisiin, internetistä löytyviin ohjeisiin sekä ohjelmistojen opiskeluun.

Tutkimuskysymykset ovat seuraavat:

Miten CityEngine-ohjelmistossa 3D-mallit muodostetaan?

Miten SketchUp-ohjelmistossa 3D-mallit muodostetaan?

Voidaanko laadittuja 3D-malleja siirtää joustavasti ohjelmistojen välillä?

Mitä vahvuuksia ohjelmistoilla on?

Mitä heikkouksia ohjelmistoilla on?

Miten ohjelmistot palvelevat Porin kaupungin suunnittelutarpeita?

Työn toisessa ja kolmannessa luvussa käsitellään opinnäytetyön aiheeseen liittyvää teoriaa. Toinen luku käsittelee 3D-kaupunkimallinnusta. Luvussa selvitetään mitä on 3D-mallinnus ja miten kaupunkimallinnusta hyödynnetään kuntien kaupunkisuunnittelussa. Lisäksi käsitellään 3D-mallintamiseen tarkoitettuja ohjelmistoja. Ohjelmistoista esitellään työssä käytetyt CityEngine- ja SketchUp-mallinnusohjelmistot. Kolmannessa luvussa käsitellään asemakaavaprosessia. Neljännessä luvussa kuvataan opinnäytetyön tutkimusosuus. Luvussa käsitellään opinnäytetyössä käytetyt menetelmät ja aineistot sekä kuvataan 3D-mallintamisen prosessia. Lisäksi perehdytään 3D-mallin julkaisemiseen selainsovelluksessa. Viidennessä luvussa käsitellään 3D-mallinnusprosessin tuloksia ja arvioidaan ohjelmistojen vahvuuksia ja heikkouksia. Kuudennessa luvussa arvioidaan testattujen 3D-ohjelmistojen soveltuvuutta Porin kaupungin suunnitteluun sekä kaava-alueiden mallintamiseen.

2 3D-KAUPUNKIMALLINNUS

2.1 3D-kaupunkimalli

3D-kaupunkimalli on digitaalinen kolmiulotteinen malli kaupungista. Sitä käytetään yhä useammin kaupunkialueiden esittämiseen, tutkimiseen ja arviointiin (Döllner, Bauman & Buchholz 2006a, 107). 3D-kaupunkimalli rakentuu korkeusmallista, rakennusmalleista, kasvillisuusmalleista ja katualuemalleista. Kolmiulotteisen kaupunkimallin perusteella arkkitehdit, kaupunkisuunnittelijat, viranomaiset ja kansalaiset voivat analysoida kaupunkirakennetta todentuntuisesti virtuaalisessa tilassa. Reaaliaikainen 3D-mallien visualisointi ja tarkastelu helpottaa suunnitteluprosessia auttamalla tunnistamaan suunnitelmien konflikteja ja epäkohtia sekä helpottamaan suunnitelmien ymmärtämistä. Voidaan myös olettaa, että kolmiulotteinen kaupunkimallinnus auttaa suunnittelun vuorovaikutusprosessissa ja johtaa parempaan suunnitteluun. (Döllner ym. 2006b, 107; Döllner, Kolbe, Liecke, Sgouros & Teichmann 2006a, 2 & Buhur, Ross, Büyüksalih, Baz 2009, 1).

Kolmiulotteinen kaupunkimallinnus voi perustua manuaaliseen tai automatisoituun mallintamiseen. Laaja-alaisiin suunnittelukohteisiin tulee käyttää automatisoitua mallinnusta. Manuaalinen mallinnus on hidasta ja tehotonta. Manuaalista mallinnusta voidaan suorittaa vain pienialaisissa suunnittelukohteissa. (Döllner ym. 2006a, 2).

2.2 Kaupunkimallinnuksen muodot

Kolmiulotteiset kaupunkimallit voidaan karkeasti jakaa graafisiin, visuaalisuuteen perustuviin malleihin tai semanttisiin, tietosisältöön perustuviin malleihin. Graafiset mallit on tarkoitettu visuaalista esitystä varten, kun tietomalleissa kohteet sisältävät geometrinen ja graafisten tietojen lisäksi ominaisuustietoa kohteista (Kolbe 2009, 15; Gröger & Plümer 2012, 12; Gröger, Kolbe, Nagel & Hägele 2012, 14). Mallinnuksen käyttötarkoitus määrää lopulta käytetäänkö graafista vai semanttista mallia. Graafista visuaalista mallinnusta käytetään, kun

halutaan saada aikaan mahdollisimman realistinen vaikutelma suunnitellusta ympäristöstä (kuvio 1). Graafinen 3D-malli sisältää kohteen geometriamuodon, mutta ei sen tietopohjaa. Täten graafisen 3D-mallin sisältämä tieto on vain visuaalista.



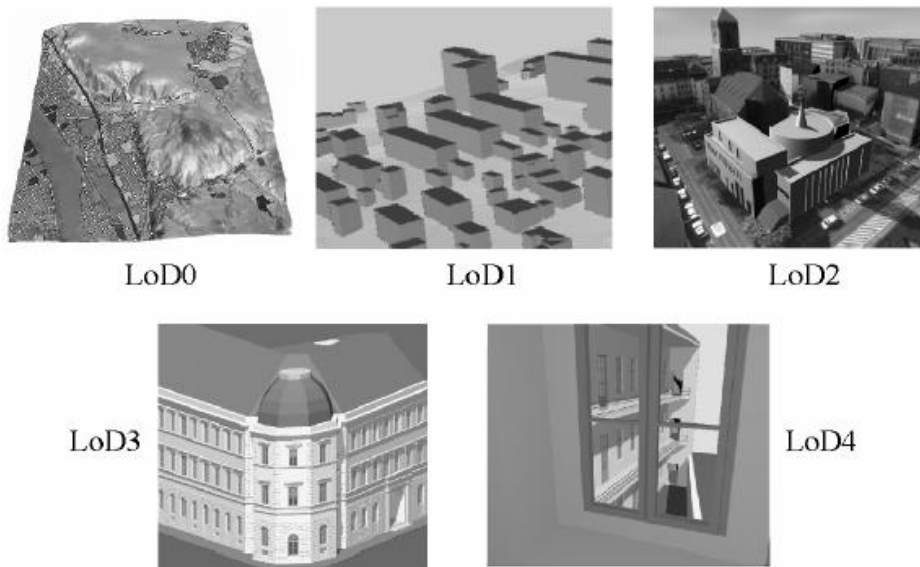
Kuvio 1. Kuviossa vasemmalla on visuaalinen 3D-malli. Kuviossa oikealla on tietosisällöllinen 3D-malli (Döllner ym. 2006b, 107).

Tietosisältöön perustuvia kaupunkimalleja voidaan käyttää geometrisiä kaupunkimalleja monipuolisemmin esimerkiksi analyysihin (kuvio 1), simulointeihin ja suunnitteluun (Döllner ym. 2006b, 107). Tietomalli sisältää tietoa kohteen rakenteesta, sisällöstä ja suhteista muihin kohteisiin. Tietomallin ominaisuustietoja voidaan myös monipuolistaa lisäämällä analyysin tulokset kohteen ominaisuustiedoksi tietokantaan (Suomisto 2014, 10). Tietomallin avulla voidaan esimerkiksi vertailla suunnitteluehdotuksia ja laskea rakennustehokkuuksia, kerrosaloja sekä ikkunoiden ja rakennuksien lukumääriä.

Kolmiulotteinen kaupunkisuunnittelu on kehittymässä kohti tietomallipohjaista 3D-mallinnusta. Kolmiulotteisen tiedon esittämistä varten on kehitetty avoin ja maksuton kansainvälinen CityGML -standardi (City Geography Markup Language). CityGML on yleinen tietomalli, joka mahdollistaa tärkeimpien kaupunkiympäristön topografisten kohteiden geometrysten, topologisten, semanttisten ja visuaalisten ominaisuuksien määrittelyn viidellä eri tarkkuustasolla. CityGML-kehityksen painopisteenä on ollut rakentaa 3D-kaupunkimallin kohteiden ominaisuuksille, rakenteille ja luokille yhteiset määritelmät. (Kolbe 2009, 17; Gröger & Plümer 2012, 12). Standardimuotonsa avulla CityGML mahdollistaa useiden erilaisten aineistojen ja ohjelmistojen yhteiskäytön, jolloin tiedon jakaminen, käsittely, tallennus ja esittäminen helpottuvat huomattavasti (Erving 2006, 4).

2.3 Tarkkuustasot

Tarkkuustaso Level of Detail (LoD) on tärkeä osa kaupunkimallinnusta. Tarkkuustasoa käytetään määrittämään, miten yksityiskohtaisesti kohteet mallinnetaan (kuvio 2). Tarkkuustasot eroavat toisistaan aineiston, yksityiskohtien, visuaalisten ominaisuuksien, semanttisuuden ja objektien monimutkaisuuden perusteella (Biljecki, Ledoux, Stoter & Zhao 2014, 3). Tarkkuustason kasvaessa lisääntyvät kohteiden geometriset ja semanttiset yksityiskohdat. Tarkkuustason tavoitteena on asettaa optimaalinen määrä muotoja ja tekstuureja katseluetaisuudelle. Mitä kauempaa kohdetta katsotaan, sitä vähemmän yksityiskohtia kohteella voi olla.



Kuvio 2. 3D-kaupunkimallinnuksen tarkkuustasot CityGML:n mukaan (Kolbe 2009, 18).

Kaupunkimallinnuksessa hyödynnetään yleisesti CityGML-standardiin perustuvaa tarkkuustasojärjestelmää. Tarkkuustasoja CityGML:ssä on viisi; LoD0–LoD4 ja ne perustuvat lähinnä geometrian ja semanttisuuden monimutkaisuuteen (kuvio 2, taulukko 1). LoD0 on tarkkuustasoista yleistetyin ja se muodostuu maastomallista (Digital Elevation Model, DEM) sekä sen päälle sijoitetusta ilmakuvasta ja polygoneista. (Kolbe 2009, 18). LoD1 -tarkkuustasossa talot esitetään laatikkomaisina ja suorakattoisina suorakulmioina. LoD1 -tasoa käytetään usein esimerkiksi korttelien talojen esittämiseen. LoD2 -tasolla rakennuksilla on erilaisia tekstuureja ja kattomuotoja. LoD3 -tasoa kutsutaan arkkitehtuurimalliksi,

jossa rakennuksien seinät ja katto ovat teksturoituja. LoD3 -taso sisältää muun muassa mahdolliset ikkunat, ovet ja parvekkeet. Myös yksityiskohtainen kasvillisuus, kävelijät, autot ja liikenteelliset yksityiskohdat voidaan esittää tällä tarkkuustasolla. Tarkimmassa LoD4 -tasossa esitetään LoD3 -tason ominaisuuksien lisäksi talojen sisäpuoliset ominaisuudet, kuten huoneet, sisäovet, portaat, huonekalut ja sisustus. (Kolbe 2009, 18; Yalcin & Selcuk 2015, 425).

Taulukko 1. LoD -tarkkuustasojen ominaisuudet (Gröger ym. 2012 mukaillen).

	LoD0	LoD1	LoD2	LoD3	LoD4
Kuvaus	Aluemalli	Kaupunkimalli	Kaupunkiosa-malli	Kaupunginosa-malli, arkkitehtimalli	Sisäinen arkkitehtimalli
Yleistys	Teksturoitu maastomalli	Laatikkomalli	Kattomuodot ja tekstuurit	Yksityiskohtaiset rakennukset	Yksityiskohtaiset rakennukset
Tarkkuus (sijaint/korkeus)	< LoD1	5/5 m	2/2 m	0,5/0,5 m	0,2/0,2 m
Rakennuksen rakenteet	Ei	Ei	Kyllä	Yksityiskohtainen julkisivu	Todelliset muodot
Rakennuksen kattomuoto	Ei	Tasainen	Erilaisia kattomuotoja	Todelliset muodot	Todelliset muodot
Katon ulkonevat osat	Ei	Ei	Kyllä, jos tiedossa	Kyllä	Kyllä
Kaupunkiympäristön kalusteet	Ei	Tärkeitä kohteet	Yleistetyt	Todelliset muodot	Todelliset muodot
Yksittäiset puut/pensaat	Ei	Tärkeitä kohteet	> 6 m korkeat	> 2 m korkeat	Todelliset muodot
Kasvipeite		> 50*50 m	> 5*5 m	< LoD2	< LoD3

Aineistovaatimukset eri tarkkuustasoilla vaihtelevat yleistysasteen mukaan. LoD0–LoD2 tasoilla mallinnuksessa voidaan käyttää ilmakuvaa, mutta LoD4 -tasolla tarvitaan yksityiskohtaisia dokumentteja tai selvityksiä mallinnuksen laadittamiseksi. (Yalcin & Selcuk 2015, 425). Kolmiulotteinen kaupunkimalli voi sisältää samanaikaisesti myös useita tarkkuustasoja (Kolbe 2009, 17–18). Esimerkiksi kaupungin keskusta voidaan esittää ympäröivää aluetta tarkemmin.

2.4 3D-kaupunkimallinnuksessa käytettävät aineistot

Kolmiulotteinen kaupunkimalli rakentuu yleensä muun muassa paikkatieto-, ilmakeku- sekä korkeusmalliaineistosta (Yalcin & Selcuk 2015, 425). Kaupunkimallinnuksessa käytettävä paikkatietoaineisto käsittää usein kaupungin pohjakartan, joka sisältää tiedot rakennuksista ja muista kaupungin kohteista. Näitä tietoja ovat esimerkiksi kohteiden sijainnit, rajat ja muodot. Esimerkiksi rakennusaineiston esittämiseen kolmiulotteisessa muodossa voidaan tarvita tietoja rakennuksien muodon ja sijainnin lisäksi korkeudesta, kerroslukumäärästä sekä kattorakenteista. Fotorealistisen kaupunkimalliin tarvitaan myös tekstuureita ja kuvia julkisivuista, katoista, maanpinnasta, puista, pensaista, jne (Yalcin & Selcuk 2015, 425).

Ortoilmakuva on ilmakeku, joka kuvaa maanpintaa kohtisuoraan ilmasta katsottuna. Ortokuvat soveltuvat erinomaisesti kaupunkimallinnuksen pohjaksi. Kaupunkimallinnuksessa voidaan käyttää myös viistokuvia. Esimerkiksi SketchUp-ohjelmassa laaditut 3D-mallit voidaan upottaa viistokuvaan. Viistokuvien käytönä on kohteiden pintatekstuurien hyödyntäminen mallinnuksessa. Viistokuvista voidaan laatia myös fotorealistinen kaupunkimalli, jolloin kolmiulotteinen geometria laaditaan viistokuvien perusteella (BLOM 2013). Tällöin kaupunkimalli ei sisällä ominaisuustietoa, vaan perustuu pelkkään valokuvainformaatioon.

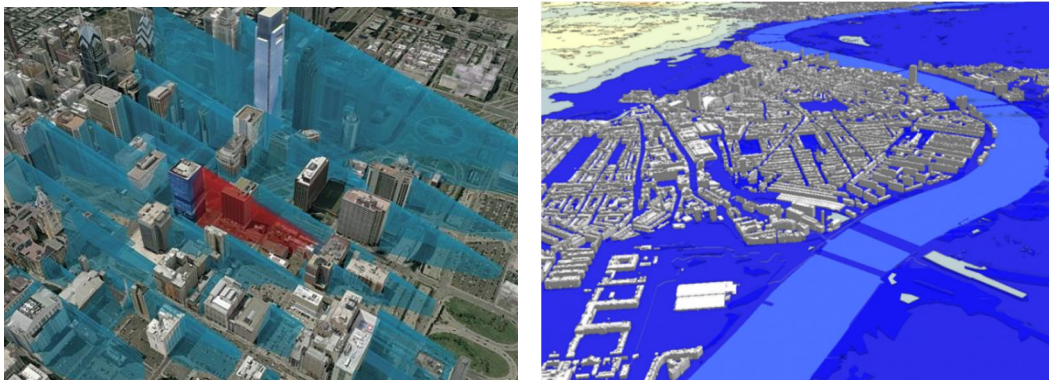
Kolmiulotteiset kaupunkimallit suunnitellaan yleensä korkeusmallin päälle (DEM, Digital Elevation Model). Korkeusmalli kuvaa maanpinnan korkeuseroja x-, y- ja z-koordinaattien muodossa. Korkeusmallit laaditaan yleensä laserkeilausaineistosta, joka perustuu lasersäteillä mitattuun mittatarkkaan korkeustietoon. Korkeusmalli voidaan esittää rasteripintaisena tai epäsäännölliseen kolmioverkkoon perustuvana mallina (TIN, Triangulated Irregular Network). Korkeusmallin yhteydessä käytetään usein myös termejä digitaalinen maastomalli (DTM, Digital Terrain Model) tai digitaalinen pintamalli (DSM, Digital Surface Model). Maastomallilla tarkoitetaan laserkeilausaineistoon perustuvaa mallia, joka kuvaa maastoa sellaisena kuin se maanpinnalla esiintyy. Maastomalli voi sisältää tietoa myös esimerkiksi rinteiden kaltevuuksista ja viettosuunnista. Pintamalli kuvaa maanpinnan ylimmälle korkeustasolle asetettua pintaa. Pintamalli

kuvaa maanpinnan korkeutta ainoastaan avoimella maalla. Muilla alueilla pinta noudattaa esimerkiksi talojen kattoja tai puiden latvuksia. Pintamalli perustuu korkeusmallin ja maastomallin tapaan laserkeilausaineistoon. (Finnish Geospatial Research Institute 2015; Maanmittauslaitos 2015).

2.5 Kaupunkimallinnuksen käyttömahdollisuudet

Kolmiulotteisen kaupunkimallinnuksella on paljon suunnittelua, havainnointia ja päätöksentekoa edistäviä käyttömahdollisuuksia. Analyysityökalujen hyödyntäminen 3D-kaupunkimallinnuksessa auttaa suunnittelun tavoitteiden optimoinnissa. Kaupunkimallia voidaan käyttää apuna esimerkiksi energia-, varjostus-, näkyvyys-, melu-, tulvakartoitus- ja onnettomuusanalyysissä (Kolbe 2009, 15; Moser, Albrecht & Kosar 2010, 144–145 ; Alam, Coors, Zlatanova & Oostrerom 2011, 1; Gröger & Plümer 2012, 12; Gröger ym. 2012, 14; Krüger & Kolbe 2012, 145, Suomisto, J. 2013, 11–21).

Näkyvyys- ja varjostusanalyysit ovat hyödyllisiä etenkin tarkastellessa kaupunkikuvaa ja korkeita rakenteita (kuvio 3). Kolmiulotteinen kaupunkimalli on ensiarvoisen tärkeä myös kaupunkien tulvien torjuntasuunnitteluun ja ennakkointiin. Tulvatilanteiden mallintaminen auttaa havainnollistamaan mahdollisen tulvan aiheuttamia seurauksia (kuvio 3). (Suomisto, J. 2013, 11–21).



Kuvio 3. Kuviossa vasemmalla on CityEnginellä laadittu varjostusanalyysi (Esri 2010). Oikealla on tulvamallinnus Lontoosta (GeoPlanIT 2014).

Kaupunkimallia voidaan hyödyntää myös kaupungin energiatarpeen suunnittelussa. Berliinissä rakennusten energiatarvetta on analysoitu yhdistämällä kaupunkimalli aurinkoenergian tai maalämmön hyödyntämismahdollisuuksiin (Energy Atlas Berlin 2015).



Kuvio 4. 3D-kaupunkimallin hyödyntäminen Berliinin rakennusten lämmöntarpeen laskennassa sekä johtoverkoston lisätarpeen laskennassa (Energy Atlas Berlin 2015).

2.6 3D-kaupunkimallinnuksen käyttö kuntien kaupunkisuunnittelussa

Kolmiulotteinen kaupunkimallinnus on tulossa kiinteäksi osaksi kuntien kaupunkisuunnittelua. Useassa kunnassa, kuten Helsingissä, Espoossa, Tampereella, Oulussa, Vantaalla ja Mikkelissä on tuotettu 3D-kaupunkimalleja. Kaupunkimallinnus on kuitenkin vielä suhteellisen uutta, eikä kansallisia pelisääntöjä kaupunkimallinnuksen laadintaan ole. Siksi kunnat ovat kokeilleet kaupunkimallinnusta parhaiksi katsomillaan menetelmillä ja laatineet omia ohjeistuksiaan kaupunkimallinnuksen tietosisällön hallintaan. Kuntaliiton kunnille teettämän kaupunkimallinnusta koskeva sähköpostikyselyn mukaan 3D-kaupunkimallinnusta tehdään kuntasuunnittelussa yleisimmin SketchUp-, MicroStationin Bentley, Autodesk 3DS MAX, Vianova Virtual Map, Autodesk Infracore -ohjelmilla (Suomen Kuntaliitto 2014, 11). Näistä ohjelmista käytetyimmät olivat SketchUp- ja MicroStationin Bentley-ohjelmistot. CityEngineä käytettiin 3D-kaupunkimallinnukseen vain yhdessä kunnassa.

Kuviossa viisi on esitetty ote Oulun yleiskaavan suunnittelua varten laadistusta 3D-mallista. Mallin on laatinut WSP Finland Oy CityEngine-ohjelmalla. Mallinnuksessa on tarkasteltu Kemintien maankäyttömahdollisuuksia.



Kuvio 5. Oulun Kemintien maankäyttömahdollisuuksia havainnollistava 3D-malli (WSP Finland 2015).

Kuviossa kuusi on esitetty ote Mikkelin kaupungin keskustasta SketchUp-ohjelmalla laaditusta 3D-kaupunkimallista. Kaupunkimalli on nähtävissä myös verkkoselaimella Mikkelin kaupungin internet-sivujen kautta.



Kuvio 6. Mikkelin kaupungin SketchUp-ohjelmalla laatima 3D-kaupunkimalli (Mikkelin kaupunki 2015).

Porin kaupungissa kolmiulotteiset kaupunkimallinnukset ovat perustuneet pääasiassa konsulttien tekemiin 3D-malleihin. 3D-malleja on laadittu lähinnä suurimmista ja tärkeimmistä kaavoituskohteista, kuten asuntomessualueesta (kuvio 7). Porin kaupungilla on ollut jo useamman vuoden ajan Esrin CityEngine-ohjelma sekä vuoden 2015 alusta lähtien Trimblen SketchUp-ohjelma. Vuoden 2016 aikana SketchUp-ohjelman käyttömahdollisuudet tulevat laajenemaan merkittävästi, kun Porin kaupungin suunnittelutyössä käytetyn Trimble Locukseen saatiin tiedonsiirtomahdollisuus SketchUp-ohjelmiston skp-formaattiin (Leppäniemi 2015). Tietokannasta voidaan tuoda SketchUp-ohjelmaan korkeusmalli sekä rekisteritiedot (rakennukset, aidat, puut, tiet, vesistö ym). Trimblen Locuksessa voidaan määrittää millaista SketchUp 3D Warehousen valmista 3D-mallia kukin kohde käyttää tai määräytyykö rakennuksien korkeus kerrosten määrällä. (Leppäniemi 2015).



Kuvio 7. 3Dee Oy:n laatima Porin kaupungin asuntomessualueen asemakaavan 3D-mallinnus (Porin kaupunki 2015c).

Kuntaliiton paikkatiedon yhteistyöryhmä perusti 3D-kaupunkimallinnushankkeen (KM3D) yhteistyössä BuildingSmartin kanssa tutkimaan 3D-mallinnuksen tilaa, kehittämistä ja standardointia Suomessa. Hankkeen tavoitteena on tietomallinnuksen sisältövaatimusten lisäksi valmistella avoin ohjeistus kuntien, suunnittelijoiden ja ohjelmistotalojen käyttöön (Suomen Kuntaliitto 2015). Osana KM3D-hanketta, Kuntaliitto toteutti kunnille kyselyn kesäkuussa 2014 3D-

mallinnuksesta ja sen vaatimuksista. Kyselyyn vastasi 44 henkilöä 40 kaupungista ja kunnasta, joista 21 maankäytön tai kaavoituksen asiantuntijaa. (Savisaalo 2015, 8–10).

Kuntaliiton teettämän kyselyn mukaan kaupunkimallinnusta käytetään kunnissa eniten konkreettisissa yksittäisissä hankkeissa sekä kaupunkien keskustoja käsittelevien mallien laadinnassa. Kyselyn mukaan 3D-kaupunkimallit koetaan suunnittelun, päätöksenteon, visioinnin ja vuorovaikutuksen apukeinoiksi. Tulevaisuuden kaupunkimallinnus haluttuun kytkeä kunnan perusaineistoihin ja rekistereihin, jotta vältetään aineiston ylimääräiseltä ylläpidolta. Kaupunkimallista toivotaan myös tulevaisuuden kantakarttaa. Kaupunkimallinnuksen toivotaan myös olevan tulevaisuudessa osana kuntien suunnittelua, vuorovaikutusta ja palvelua. Ennen kaikkea kaupunkimallinnuksen toivotaan olevan tulevaisuuden työväline. (Suomen Kuntaliitto 2014 2–21; Savisalo 2015, 10–15).

Kaupunkimallinnus on vielä uutta useissa kaupungissa ja haasteita sen toteuttamiseksi on paljon. Kuntaliiton kyselyn mukaan haasteita luovat etenkin kustannus- ja koulutuskysymykset. Lisäksi mallinnuksen tarpeet ovat hyvin erilaisia, jolloin yleispätevien toimintatapojen löytäminen on vaikeaa. Eniten yhteisiä toimintatapoja 3D-kaupunkimallinnuksessa kaivattiin tiedonsiirtoon, mallintamiseen, käyttöviin tarkkuustasoihin ja kaupunkimallistrategiaan. (Suomen Kuntaliitto 2014, 19–20 ; Savisalo 2015, 12–14).

2.7 Esri CityEngine-mallinnusohjelma

Esri on paikkatietoratkaisujen toimittaja, jonka tuoteperheeseen kuuluu useita eri paikkatietosovelluksia. Esri CityEngine-ohjelmisto on 3D-mallinnusohjelma, jolla voidaan sääntöjen perusteella luoda 3D-kaupunkimalleja olemassa olevasta 2D-paikkatietoaineistosta. 3D-malleja voidaan luoda vaihtoehtoisesti myös manuaalisesti suoraan ohjelmassa. Ohjelmisto tarjoaa työkaluja muun muassa paikkatiedon keräämiseen, katuverkon suunnitteluun ja muokkaukseen, kiinteistöjen jakamiseen, 3D-rakennusten luomiseen ja muokkaukseen, kaupunkirakenteen mallinnukseen sekä kaupunkisuunnitteluhankkeiden analysoimiseen

(Esri 2015a, 5–7). Ohjelmiston avulla voi laatia analyysejä esimerkiksi valojen ja varjojen käyttäytymisestä kaupunkimallissa.

CityEngine perustuu dynaamiseen sääntöpohjaiseen CGA-mallinnukseen (Computer Generated Architecture). Sääntö kertoo ohjelmistolle, miten 2D-paikkatietoaineistosta muodostetaan 3D-kohteita. Säännöillä määrätään esimerkiksi minkälaisista geometrioista, mittasuhteista ja tekstuureista mallinnettavat kohteet muodostuvat. Ohjelma tarjoaa valmiita sääntöjä, joita voidaan hyödyntää katuverkon, rakennuksien ja kasvillisuuden mallintamiseen. Valmiita sääntöjä voidaan muokata tai niitä voidaan rakentaa itse CityEnginen sääntö-editorissa. Valmiin mallin yksityiskohtia voidaan muokata julkisivueditorin avulla valokuvantarkoiksi kopioiksi alkuperäisestä. Julkisivueditorilla muodostetut säännöt ovat koosta riippumattomia ja niitä voidaan laajentaa monipuolisemmiksi säännöiksi. CityEngine tarjoaa myös laajan tyylivalikoiman, jonka avulla suunnitelmiin voidaan tehdä helposti muutoksia. (Esri 2012, 6).

Olemassa oleva paikkatietoaineisto voidaan tuoda CityEngineen sijainti- ja ominaistietoineen. Ohjelmisto tukee lukuisia tiedostoformaatteja, jonka vuoksi ohjelmistoon voidaan tuoda valmiita 3D-kohteita useista eri ohjelmistoista tai hakea aineistoa OpenStreetMap-palvelusta. CityEngine on myös ArcGIS-paikkatieto-ohjelmiston kanssa yhteensopiva, jolloin aineistojen käsittely ja siirtely ohjelmien välillä sujuu vaivattomasti. (Esri 2012, 6).

CityEngine mahdollistaa sääntöpohjaisten raporttien tuottamisen kaupunkisuunnitelman analysoimiseksi. Esri tarjoaa yhdenmukaistettuja tietomalleja 3D-paikkatietoaineistoista, joilla kaupunki- ja kaavasuunnittelun tuotokset voidaan esitellä kuntalaisille. Nämä tietoaineistot voidaan edelleen visualisoida ja muokata CityEnginellä. Valmiit mallinnukset voidaan julkaista verkkoselaimelle kansalaisten nähtäville. CityEngine tarjoaa myös useita opetusvideoita ja koulutusmateriaalia itseopiskelua varten. (Esri 2012, 6–8). Ohjelmisto on maksullinen, mutta ilmaisversion voi ladata 30 päivän ajaksi käyttöönsä Esrin verkkosivuilta (Esri 2015c).

2.8 Trimblen SketchUp-mallinnusohjelmisto

Graafiseen kaupunkimallinnukseen soveltuvia mallinnusohjelmia on kaupunkisuunnittelussa käytössä lukuisia erilaisia. Käytettävyydeltään yksinkertaisena 3D-suunnitteluohjelmiana voidaan pitää Trimblen Sketchup-ohjelmistoa. SketchUp on käytössä monien kuntien kaupunkisuunnittelussa, esimerkiksi luonnostelussa, koska sillä saa aikaiseksi nopeasti informatiivisen ja työn määrään nähden suhteellisen laadukkaan mallinnuksen.

Kolmiulotteinen mallinnus SketchUp-ohjelmistolla perustuu ääriviivojen piirtämiseen, joiden perusteella ohjelma luo automaattisesti pintoja. Pintoja edelleen vetämällä ja työntämällä voidaan muodostaa kolmiulotteisia kohteita. SketchUp tarjoaa myös lukuisia valmiita 3D-malleja Warehouse-mallikirjastosta sekä mahdollisuuden mallintaa kohteita valokuvan perusteella (SketchUp 2015a).

SketchUp-ohjelmistolla on mahdollisuus luoda korkeusmalli olemassa olevasta korkeuskäyräaineistosta. Korkeusmallin sekä ilmakuvan voi myös ladata Google Earth -palvelusta kaikkialta maailmasta. Ohjelman avulla voidaan tarkastella myös valojen ja varjojen käyttäytymistä. SketchUp tukee useita tiedostomuotoja, jonka ansiosta aineiston tuonti ja vienti ohjelmistojen välillä on helppoa. (SketchUp 2015b).

SketchUp-ohjelmisto tarjoaa 3D-mallinnukseen ilmaisen Make-version sekä maksullisen Pro-version. Make-versio on työkaluiltaan ja ominaisuuksiltaan tyypistetympi Pro-versioon verrattuna. Pro-version voi ladata 30 päiväksi ilmaiseksi käyttöönsä (SketchUp 2016).

3 ASEMAKAAVOITUS

3.1 Asemakaava

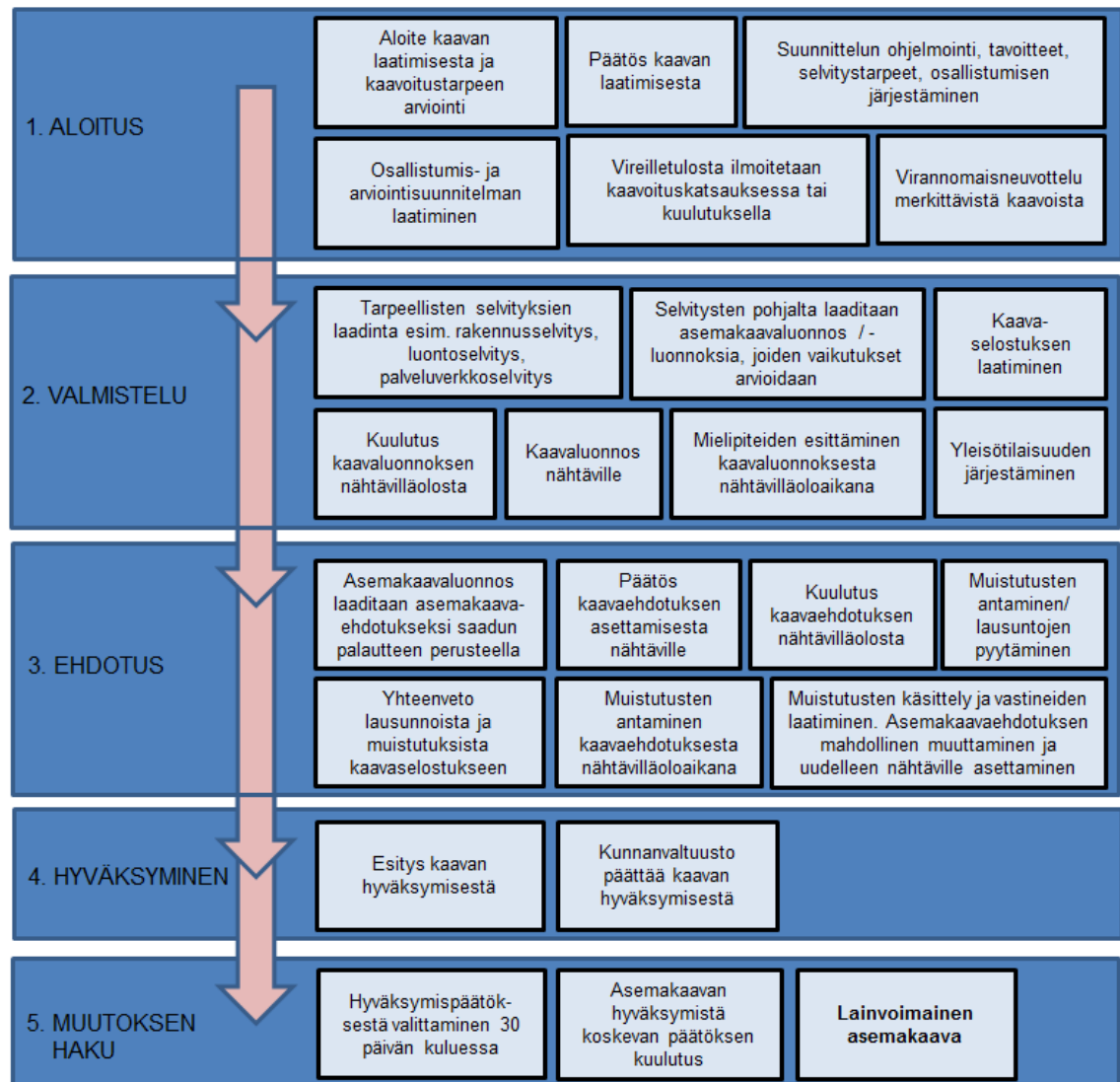
Kaavoitus jakautuu kaavatasoihin: maakuntakaavaan, yleiskaavaan ja asemakaavaan, joista asemakaava on yksityiskohtaisin. Asemakaavan suunnittelua ohjaavat valtakunnallisten alueidenkäyttötavoitteiden lisäksi maakuntakaava ja oikeusvaikutteinen yleiskaava (Ympäristöministeriö 2003, 14). Asemakaavalla tarkoitetaan kunnan laatimaa yksityiskohtaista suunnitelmaa, jossa osoitetaan tarpeelliset alueet eri tarkoituksia varten ja ohjataan alueiden järjestämistä, rakentamista, kehittämistä paikalliset olosuhteet huomioon ottaen (Maankäyttö- ja rakennuslaki 132/1999 7:50 §).

Asemakaava esitetään kartalla, jossa osoitetaan kaavaan sisältyvien eri alueiden rajat, käyttötarkoitus, rakentamisen määrä sekä sijoitusta ja tarvittaessa rakentamistapaa koskevat periaatteet. Asemakaava voi käsittää kokonaisen asuinalueen tai vain yhden tontin. Asemakaava koostuu asemakaavakartasta, kaavamerkinnöistä ja -määräyksistä sekä asemakaavaselostuksesta. (Maankäyttö- ja rakennuslaki 132/1999 7:55 §). Asemakaavaselostuksessa kerrotaan kaavan laatimisesta ja keskeisistä ominaisuuksista. Lisäksi kaavakarttaa voidaan selventää havainnepiirroksella.

Kunnan on Maankäyttö- ja rakennuslain 132/1999 7:51§:n mukaan velvollisuus laatia ja pitää ajan tasalla asemakaavaa sitä mukaa kuin kunnan kehitys sitä edellyttää. Asemakaava on laadittava siten, että luodaan edellytykset terveelle, turvalliselle ja viihtyisälle elinympäristölle, palvelujen alueelliselle saatavuudelle ja liikenteen järjestämiselle. Rakennettua ympäristöä ja luonnonympäristöä on myös vaalittava, eikä niihin liittyviä erityisiä arvoja saa hävittää. Kaavoitettavalla alueella tai sen lähiympäristössä on oltava riittävästi puistoja tai muita lähivirkistykseen soveltuvia alueita. Jos asemakaava laaditaan alueelle, jolla ei ole oikeusvaikutteista yleiskaavaa, on asemakaavaa laadittaessa soveltuvin osin otettava huomioon myös yleiskaavan sisältövaatimukset. (Maankäyttö- ja rakennuslaki 132/1999 7:54 §).

3.2 Asemakaavaprosessi

Asemakaavan laatiminen koostuu monivaiheisesta prosessista, joka alkaa päätöksestä kaavan laatimiseen, päättyen kaavan hyväksymiseen ja lainvoiman saavuttamiseen (kuvio 8). Usein kaavan laatimispäätöstä edeltää kaavoitustarpeen arviointi. Asemakaavatyön käynnistyessä suunnittelutyö ohjelmoidaan ja laaditaan osallistumis- ja arviointisuunnitelma (OAS), johon kirjataan kaavan tavoitteet, määritellään selvitystarpeet sekä suunnitellaan osallistumisen järjestäminen. Kaavan vireilletulosta sekä osallistumis- ja arviointisuunnitelmasta ilmoitetaan osallisille. Merkittävistä asemakaavoista järjestetään viranomaisneuvottelu. (Ympäristöministeriö 2007, 17).



Kuvio 8. Asemakaavaprosessin eteneminen.

Asemakaavan valmisteluvaiheessa laaditaan kaavan sisältöä koskevat ratkaisut. Valmisteluvaiheessa kaavan tavoitteita tarkennetaan, laaditaan tarvittavat selvitykset, pohditaan erilaisia kaavaratkaisuja sekä selvitetään kaavan vaikutuksia. Laadittavien selvityksien on annettava riittävät tiedot, jotta voidaan arvioida suunnitelman toteuttamisen merkittävät vaikutukset (Maankäyttö- ja rakennuslaki 132/1999 1:9 §). Selvitysten sekä valitun kaavaratkaisun perusteella laaditaan asemakaavaluonnos. Lisäksi laaditaan asemakaavaselostus, joka toimii asemakaavan keskeisenä asiakirjana koko asemakaavaprosessin ajan. Asemakaavaselostuksessa esitetään kaavan tavoitteet, sisältö sekä vaikutusten arvioimiseksi tarvittavat tiedot. Selostuksen tehtävänä on auttaa ymmärtämään ja tulkitsemaan kaavaratkaisua sekä turvaamaan toteutus kaavan tarkoittamalla tavalla. Asemakaavaluonnos kuulutetaan nähtäville osallisten mielipiteen esittämistä varten. Ennakkolausuntoja pyydetään tarpeen mukaan esimerkiksi erityisen tärkeistä asemakaavakohteista. Osana vuorovaikutusprosessia, asemakaavaluonnoksesta ja sen aineistosta voidaan pitää yleisötilaisuus asukkaille ja tiedotusvälineille. (Maankäyttö- ja rakennuslaki 2000, 11–17; Ympäristöministeriö 2007, 17).

Kaavaluonnoksesta saadut mielipiteet sekä suunnitelmaan tulleet muutokset kirjataan asemakaavaselostukseen. Asemakaavaluonnoksesta laaditaan saatujen mielipiteiden pohjalta asemakaavaehdotus. Asemakaavaehdotuksen julkisesti nähtävilleasettamisesta päättää yleensä kunnanhallitus tai lautakunta (Ympäristöministeriö 2007, 19). Kaavaehdotus kuulutetaan nähtäville vähintään 30 päivän ajaksi. Vaikutukseltaan vähäinen asemakaavamuutos voidaan pitää nähtävillä vain 14 päivän ajan. Nähtävilläoloaikana osalliset voivat tehdä kaavaehdotuksesta kirjallisen muistutuksen. Kaavaehdotuksesta pyydetään tarpeelliset lausunnot maakunnanliitolta, elinkeino-, liikenne- ja ympäristökeskukselta, kunnalta sekä viranomaisilta ja yhteisöiltä. (Maankäyttö- ja rakennusasetus 895/1999 5:27 §). Saatuihin muistutuksiin ja lausuntoihin laaditaan vastineet ja tarvittaessa muutetaan asemakaavaehdotusta. Jos asemakaavaehdotusta joudutaan muuttamaan merkittävästi, kaava on asetettava uudelleen julkisesti nähtäville (Maankäyttö- ja rakennusasetus 895/1999 6:32 §).

Asemakaavan hyväksyy kunnanvaltuusto (Ympäristöministeriö 2007, 19). Kaavan hyväksymispäivämäärä ja kaavan hyväksynyt toimitus merkitään kaavakarttaan. Kunnan on lähetettävä asemakaavan hyväksymistä koskeva päätös sekä kaavakartta että -selostus ja rakennusjärjestys tiedoksi elinkeino-, liikenne- ja ympäristökeskukselle. Kaavan hyväksymisestä koskevasta päätöksestä on myös ilmoitettava niille viranomaisille, kunnan jäsenille ja muistutuksen tehneille, jotka ovat sitä pyytäneet kaavan ollessa nähtävillä. (Maankäyttö- ja rakennusasetus 895/1999 16:94 §). Asemakaavan hyväksymisestä voi valittaa hallinto-oikeuteen valitusajan kuluessa. Valituksen voi tehdä, jos asemakaavapäätös on syntynyt virheellisessä järjestyksessä, se on lainvastainen tai päätöksen tehnyt viranomainen on ylittänyt toimivaltansa (Kuntalaki 365/1995 11:90 §). Valituksen saa tehdä kunnan jäsen tai henkilö, jonka oikeuteen, velvollisuuteen tai etuun päätös välittömästi vaikuttaa. Hallinto-oikeuden antamaan päätöksestä voi edelleen valittaa korkeimpaan hallinto-oikeuteen. (Kuntalaki 365/1995 11:92 §). Asemakaava saa lainvoiman kaavan hyväksymisen tai mahdollisten valitusten ratkaisemisen jälkeen.

3.3 Osallistumis- ja arviointisuunnitelma

Osallistumis- ja arviointisuunnitelma (OAS) on maankäyttö- ja rakennuslain 132/1999 8:63 §:n mukainen suunnitelma, joka välittää tietoa kaavan etenemisestä, vaikutusten arvioinnista ja mahdollisuuksista osallistua valmisteluun. Osallistumis- ja arviointisuunnitelmasta tiedotetaan yleensä samalla, kun ilmoitetaan kaavoituksen vireilletulosta ja sitä päivitetään tarvittaessa kaavaprosessin aikana.

Osallisilla on keskeinen rooli kaavaprosessissa, sillä he tuovat kaavoittajien tietoon, mitkä asiat ovat asukkaille ja alueella toimijoille tärkeitä. Osallistumis- ja arviointisuunnitelma tehdäänkin pääasiassa asukkaita, järjestöjä, kunnan ja valtion viranomaisia sekä muita osallisia varten. Liittämällä osallistumis- ja arviointisuunnitelmaan havainnollisia kuvia, palvelee suunnitelma sekä asukkaiden että viranomaisten tiedontarpeita ja innostaa osallistumaan kaavatyöhön. (Ympäristöministeriö 2007, 36).

3.4 Asemakaavamerkinnot ja -määräykset

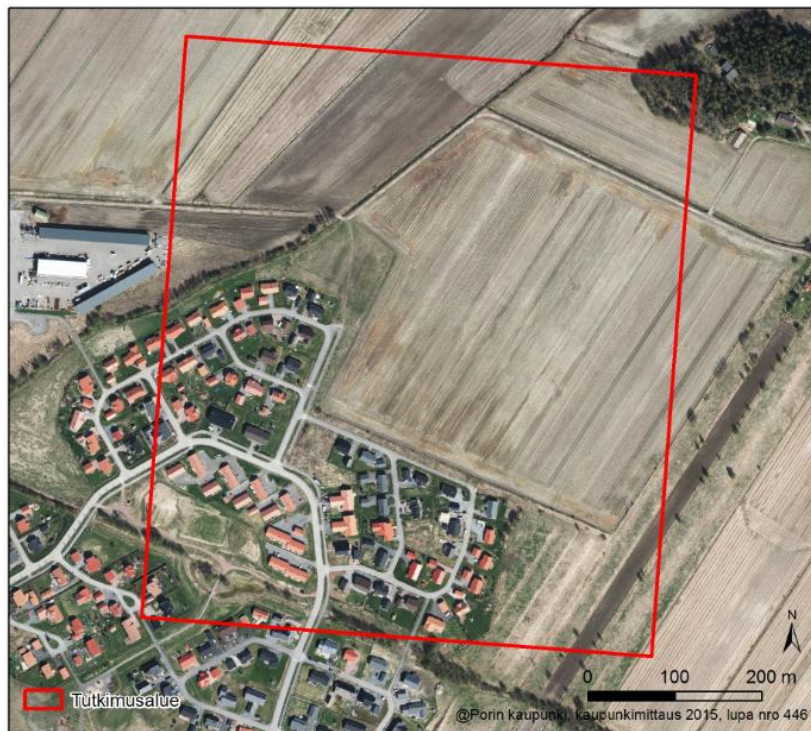
Kaavamääräykset ovat keskeinen osa asemakaavaa. Ne täsmentävät kaavan oikeudellisia vaikutuksia ja niiden avulla voidaan myös merkittävästi vaikuttaa ympäristön laatuun. Kaavamääräysten tulee liittyä rakentamiseen tai rakennusten tai alueen käyttöön, eivätkä ne saa olla ristiriidassa lain, asetuksen tai muun ylemmän asteen säännöksen kanssa. (Ympäristöministeriö 2003, 20)

Asemakaavamääräykset toimivat alueiden käytön ja rakentamisen ohjaajana. Kaavamääräysten esitystavat voidaan jakaa yleismääräyksiin ja indeksimääräyksiin. Yleismääräykset koskevat koko kaavan aluetta tai sen osaa, esimerkiksi tiettyjä kortteleita tai tiettyntyyppisiä alueita. Indeksimääräykset sen sijaan liittyvät kortteleiden tai muiden alueiden käyttötarkoituksimerkintöihin. Asemakaavamääräykset voidaan esittää kaavamerkintöjen selitysten yhteydessä tai koko kaava-alueella tai sen osaa koskevin yleismääräyksinä. (Ympäristöministeriö 2003, 23)

4 AINEISTOT JA MENETELMÄT

4.1 Tutkimusalue

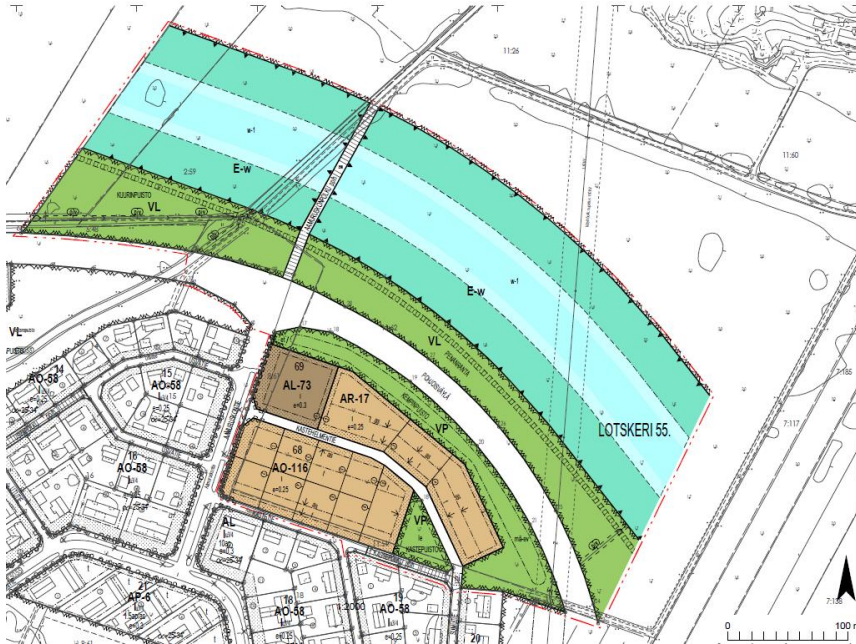
Kolmiulotteisen kaupunkimallinnuksen tutkimusalueeksi valittiin Porin kaupungin Lotskerin kaupupunginosassa sijaitseva vireillä oleva asemakaavakohde ympäristöineen (kuviot 9 ja 10). Tutkimusalue sijoittuu pientaloalueelle noin kolme kilometriä Porin keskustasta pohjoiseen. Tutkimusalue on pinnanmuodoiltaan hyvin alavaa ja on pinta-alaltaan noin 39,5 hehtaaria.



Kuvio 9. 3D-kaupunkimallinnuksen tutkimusalue.

Tutkimusalueella sijaitseva vireillä oleva asemakaavakohde on pinta-alaltaan 11,5 hehtaarin kokoinen (kuvio 10, liitteet 1 ja 2). Asemakaavakohde on osa vaiheittain toteutettavaa asemakaavahanketta, jolla on tarkoitus liittää Lotskeri ja Ruosniemi kaupunkirakenteellisesti yhteen. Vireillä olevalla asemakaavalla täydennetään Lotskerin pientaloalueen koillisosa ja jatketaan Pohjoisväylän linjasta sekä laaditaan aluevaraus jokiuomalle patorakenteineen (Porin kaupunki 2013). Uuden jokiuoman rakentaminen kantakaupungin pohjoispuolitse ja suunnitellun kaava-alueen itäpuolelle vähentää merkittävästi Porin keskustan, suunnittelualueen ja sitä ympäröivien alueiden tulvariskiä. Hanke halutaan to-

teuttaa vaiheittain. Ensin rakennetaan pieniä sisäjärviä, jotka lopulta yhdistetään yhtenäiseksi uomaksi.



Kuvio 10. Ote Lotskerin pientaloalueen asemakaavaehdotusluonnoksesta (Porin kaupunki 2015a ja d).

Asemakaava on tullut vireille kaupungin aloitteesta ja vireilletulosta on kuulutettu 17.5.2013 sanomalehdissä ja kaupungin verkkosivuilla sekä ilmoitettu kirjeellä osallisille. Asemakaavakohteen osallistumis- ja arviointisuunnitelma on ollut vireilletulon yhteydessä nähtävänä 18.5–17.6.2013. (Porin kaupunki 2013, 3). Ympäristöministeriön 30.11.2011 vahvistamassa ja korkeimmalta hallinto-oikeudelta lainvoiman 13.3.2013 saaneessa Satakunnan maakuntakaavassa asemakaavoitettava alue on osoitettu taajamatoimintojen alueeksi (A) (kuvio 11). Maakuntakaavaan on merkitty suunnittelualueen läpi kulkeva Pohjoisväylän suunniteltu linjaus uutena pääkatuna (st) sekä suunnittelualueen itärajalla kulkeva voimalinja (z). Kantakaupungin yleiskaavassa 2025 (hyväksytty 10.12.2007) suunnittelualue on osoitettu pientalovaltaiseksi asuntoalueeksi (AP). Yleiskaavaan on merkitty suunnittelualueen kohdalla kaakkoon taittuva Pohjoisväylän jatkoon linjaus sekä sen meluhaitan torjuntatarve pientaloalueeseen rajautuvalla osuudella. Lännessä suunnittelualue rajautuu pääkatuverkkoa täydentävään kevyen liikenteen reittiin. (Porin kaupunki 2013, 2).



Kuvio 11. Vasemmalla ote Satakunnan maakuntakaavasta ja oikealla ote Porin kantakaupungin yleiskaavasta 2025 (Satakuntaliitto 2013, Porin kaupunki 2007).

Tutkimusalueeksi pientaloalueen laajennus on erinomainen, sillä kohde ei ole liian suuri kaupunkimallinnuksen laatimista varten. Lisäksi alue on yksi tyypillisimmistä kaavoituskohteista Porissa, jolloin mallinnuksen käytettävyyttä asemakaavoituksen sekä liikenne- ja puistosuunnittelun havainnollistamisessa pystytään hyvin arvioimaan. Lisäksi alue mahdollistaa erilaisten mallinnusmenetelmien hyödyntämisen, sillä alue on osin rakennettu, osin rakentamaton ja käsittää monia erilaisia maankäyttömuotoja. Valmista kaupunkimallia hyödynnetään asemakaavatyön ehdotusvaiheessa, jolloin valmis malli on katsottavissa verkkoselaimen kautta.

4.2 Tutkimuksen yleinen kulku

Tutkimus jaettiin menetelmällisesti kolmeen kokonaisuuteen (liite 3): **(1) Alkuvalmistelut ja primaarianalyysit**, **(2) 3D-mallinnukset** sekä **(3) tulosten analysointi**. **Alkuvalmisteluihin ja primaarianalyysihin** kuuluivat aineiston hankinta, esikäsittely sekä maankäyttöaineistojen luominen ilmakuvaa ja kaavaluonnosta hyödyntäen. **3D-mallinnuksessa** laadittiin kaupunkimalli Esrin CityEngine- ja Trimblen SketchUp-ohjelmien avulla. CityEnginellä laadittiin 3D-malli hyödyntäen sääntöpohjaista-, manuaalista- sekä kuvamallinnusta. Lisäksi 3D-mallien tiedonsiirto suoritettiin molemmissa ohjelmissa. Valmis 3D-malli siirrettiin verkkoselaimelle, jossa 3D-mallia voitiin tarkastella interaktiivisesti. SketchUp-ohjelmassa laadittu malli tallennettiin MP4-muotoon. **Tulosten ana-**

lysoinnissa vertailtiin 3D-mallinnusmenetelmien soveltuvuutta kaupunkimallintamiseen sekä Porin kaupungin suunnitteluntarpeisiin.

Tutkimuksen yleinen kulku on esitetty tarkemmin liitteessä kolme. Työn kuvauksessa pyöristetyt suorakulmiot kuvaavat aineistoja. Pistekatkoviivalla kuvatut suorakulmiot kuvaavat erilaisia analyysejä tai työvaiheita.

4.3 Alkuvalmistelut ja primaarianalyysit

4.3.1 Aineistot

Mallinnustyön aineistoina käytettiin Porin kaupungin digitaalista pohjakarttaa, ortoilmakuvia, Maanmittauslaitoksen laserkeilausaineistoa sekä talojen julkisivukuvia (taulukko 2). Aineistoja laadittiin Esrin ArcGIS-, CityEngine- sekä SketchUp-ohjelmissa. Maanmittauslaitoksen laserkeilausaineistoa käytettiin korkeusmallin luomiseen. Ortokuvia käytettiin sekä korkeusmallin että rakennusten kattojen teksturoinnissa. Porin kaupungin pohjakarttaa käytettiin 3D-mallin talojen ja tonttien pohja-aineistona. Talojen julkisivukuvia käytettiin kolmiulotteisten talojen julkisivujen mallintamiseen.

Rakennuksien julkisivujen kuvaaminen suoritettiin maastossa kuvaamalla. Kuva otettiin mahdollisimman kohtisuoraan rakennuksen sivua kohden. Kuvausajankohta keskittyi heinäkuuhun sekä loppuvuoteen 2015. Julkisivukuvat muokattiin CityEngine-ohjelmassa ennen niiden hyödyntämistä mallinnuksessa.

Taulukko 2. Kaupunkimallinnuksessa käytetyt aineistot.

Aineisto	Tuotantovuosi/kuvausajankohta	Lähde
Kantakartta	01/2015	Porin kaupunki, kaupunkimitaus 2015, lupa nro 446
Ortokuvamosaiikki	2013	Porin kaupunki, kaupunkimitaus 2015, lupa nro 446
Rakennusten julkisivukuvat	2015	Kirsi-Maria Viljanen
Laserkeilausaineisto (karttalehdet M3412F1 ja M3412F2)	Automaattisesti maanpinta-luokiteltu pistepilviaineisto. Tarkkuus 0.5 pistettä/m ²	Maanmittauslaitos

4.3.2 Aineistojen esikäsittely ja muokkaus ArcGIS-ohjelmalla

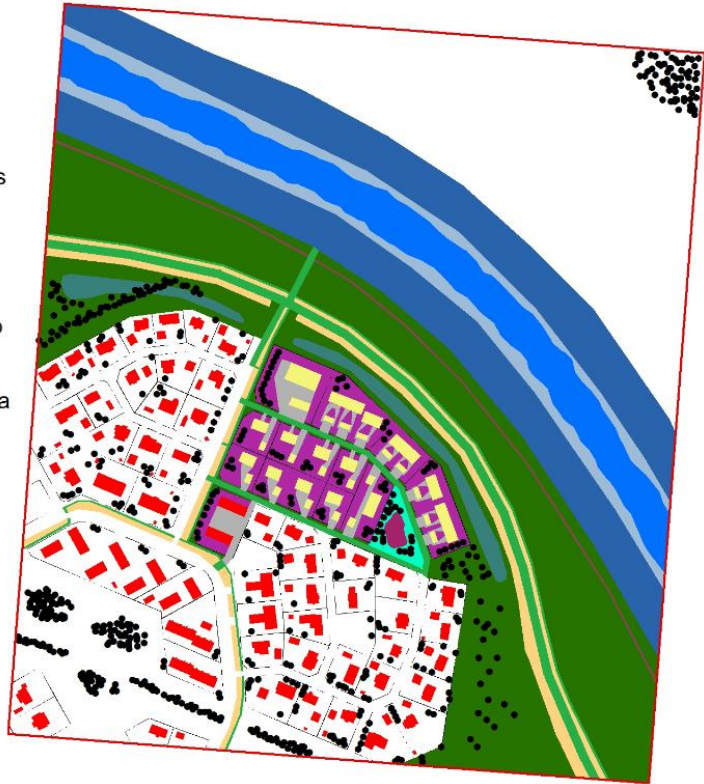
Aineistojen esikäsittely ja muokkaus on lähes välttämätöntä Esrin ArcGIS-ohjelmalla ennen niiden siirtämistä CityEngine-ohjelmaan. Ilman esikäsittelyä aineistot voivat olla laajuudeltaan erikokoisia, sijaita eri koordinaatistoissa sekä olla väärässä tiedostoformaattissa. Lisäksi aineistojen attribuuttitiedot kannattaa lisätä esikäsittelyvaiheessa, jotta niitä voidaan hyödyntää sujuvasti CityEnginen sääntöpohjaisessa mallinnuksessa. Mallinnusta CityEnginellä voi tehdä ilman aineistojen esikäsittelyäkin, mutta tällöin mallinnuksen mahdollisuudet sekä ulkonäkö eivät välttämättä tue toivottua tulosta. Aineistoon voidaan lisätä attribuuttitietoja myös CityEngine-ohjelmassa.

Aineistojen esikäsittelyssä mallinnuksessa käytetyt ilmakuvat sekä pohjakartta muutettiin laserkeilausaineiston kanssa samaan etrs tm 35-fin -koordinaatistoon. Tämän jälkeen aineistot leikattiin tutkimusaluearajalla. Maankäyttöä kuvaavat aineistot luotiin hyödyntäen ilmakuvaa, pohjakarttaa sekä kaavaluonnosta. Maankäyttöä kuvaavaan aineistoon digitoitiin ne alueet, jotka haluttiin mallintaa 3D-malissa. Alueita, jotka haluttiin esittää mallinnuksessa ilmakuvalla teksturoidulla korkeusmallilla, jätettiin digitoimatta. Tiestö jätettiin asemakaava osuutta lukuunottamatta digitoimatta, sillä se muodostettiin suoraan CityEngine-ohjelmassa fotorealistisemmän kuvan saamiseksi. Asemakaavan havainnekuvan (liite 2) rakennukset digitoitiin omaksi rakennusaineistoksi. Osa mallinnuksessa käytetyistä puista digitoitiin pisteaineistoksi. Omaksi aineistoksi muodostettiin myös tonttirajat. Tonttirajoille muodostettiin aluemainen bufferi. Bufferointia käytettiin, jotta voitiin mallintaa CityEnginessä tontteja ympäröivät pensasaidat.

Kuviossa 12 on kuvattu mallinnuksen perustana käytetty pohja-aineisto. Pohja-aineisto muodostui digitoiduista maankäyttöalueista, puustopisteistä sekä Porin kaupungin pohjakartan rakennuksista.

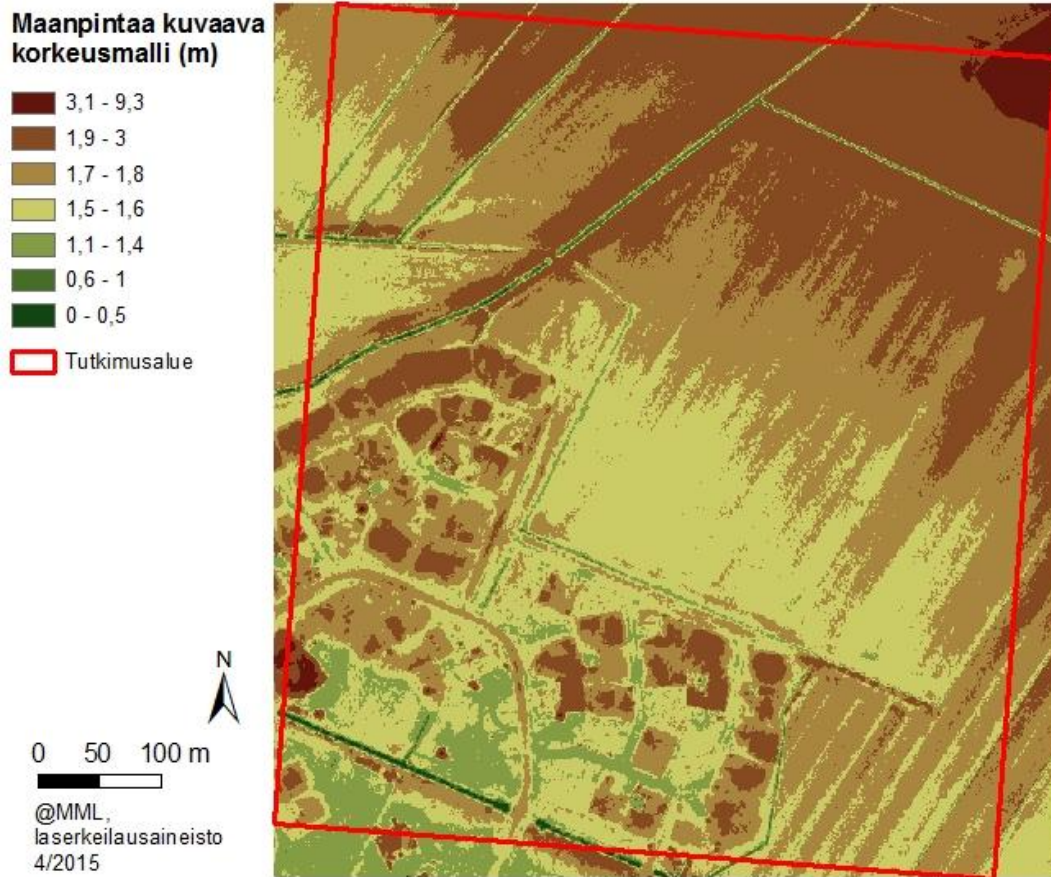
Pohja-aineisto

	aluerajaus	Maankäyttö
	rakennus	 asfalttatie
	tonttiaita	 soratie
	uusi rakennus	 pihakivetyks
		 tontti
		 meluvalli
		 nurmi
		 uusi puisto
		 viheralue
		 leikkipaikka
		 joenreuna
		 jokivalli
		 vesialue
		• Puu



Kuvio 12. Mallinnuksessa käytetty pohja-aineisto.

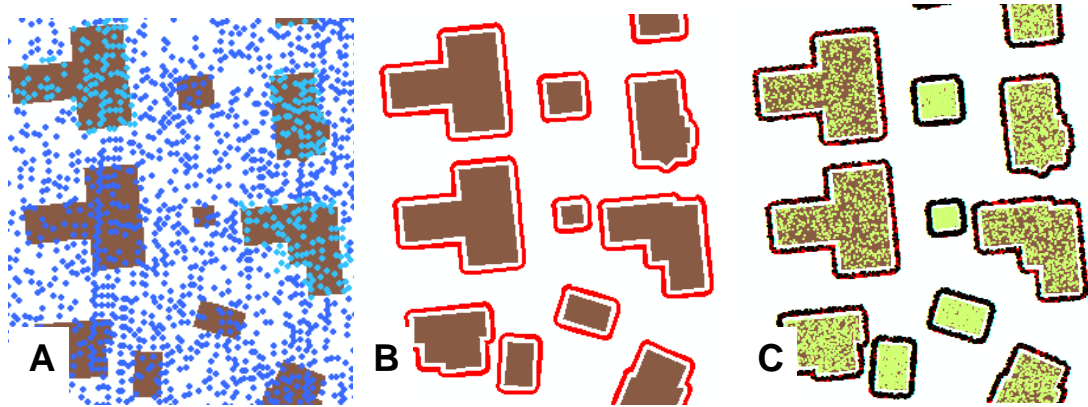
Mallinnuksessa käytetty korkeusmalli laadittiin Maanmittauslaitoksen laserkeilausaineistosta. Laserkeilausaineisto ladattiin Maanmittauslaitoksen avoimien aineistojen tiedostopalvelusta, jonka jälkeen se muutettiin LAS-tiedostomuotoon. LAS-muotoista aineistoa ei voida suoraan hyödyntää korkeusmallipintana, joten se muutettiin ArgGis-ohjelmassa MultiPoint-pisteaineistoksi ja tämän jälkeen vektorimuotoiseksi Terrain-korkeusmallipinnaksi (kuvio 13). Multipoint-aineiston laatimisessa erotettiin laserkeilausaineiston luokkakoodien perusteella ensimmäinen havaittu pinta (esimerkiksi puut ja rakennukset) ja maanpinta. Maanpinnan luokkakoodia käyttämällä saatiin muodostettua maanpinnan korkeussuhteita kuvaava Terrain-korkeusmallipinta. Koska CityEngine ei lue Terrain-aineistoa, muutettiin aineisto rasteripinnaksi.



Kuvio 13. Maanpinnan korkeussuhteita kuvaava vektorimuotoinen Terrain-korkeustasopinta.

Laserkeilausaineistoa hyödynnettiin myös rakennusten korkeuksien määrittämisessä (kuvio 14, A–C). Jotta laserkeilausaineiston korkeustieto voitiin siirtää rakennusaineistoon, luotiin rakennusaineistojen perusteella näytepisteitä (sample points), joihin siirrettiin korkeustieto laserkeilausaineistosta. Jokaisen talon näytepisteistä luotiin taulukko, joka kertoi kunkin rakennuksen minimi-, maksimi- ja keskikorkeuden merenpinnan tasosta. Jotta jokaisen rakennuksen todellinen rakennuskorkeus voitiin määrittää, tarvittiin tieto myös rakennuksien maanpinnan korkeudesta. Rakennusten maanpinnan korkeuden laskemista varten luotiin jokaiselle rakennuksen ympärille vyöhyke (kuvio 14, B), jonka perusteella luotiin näytepisteitä (kuvio 14, C). Näytepisteisiin siirrettiin korkeustieto laserkeilausaineistosta. Tämän jälkeen näytepisteaineistojen minimiarvot yhdistettiin rakennusaineistoon. Minimiarvot otettiin näytepisteaineistosta siitä syystä, ettei arvoihin sekaannu korkeuspistetietoja esimerkiksi kasvillisuudesta, autoista tai teistä. Todellinen rakennuskorkeus saatiin vähentämällä maanpinnan korkeus rakennusten merenpintatason korkeudesta. Koska osa alueen rakennuksista

ei ollut valmistunut laserkeilausaineiston kuvausajankohtana, asetettiin korkeustiedoiltaan puuttuviin rakennuksiin arviokorkeudet. Rakennusaineistoon lisättiin korkeustiedon lisäksi mallinnuksessa tarvittavia tietoja, kuten talotyyppi.



Kuvio 14. Maanmittauslaitoksen laserkeilausaineiston hyödyntäminen rakennuksien korkeuksien laskemisessa.

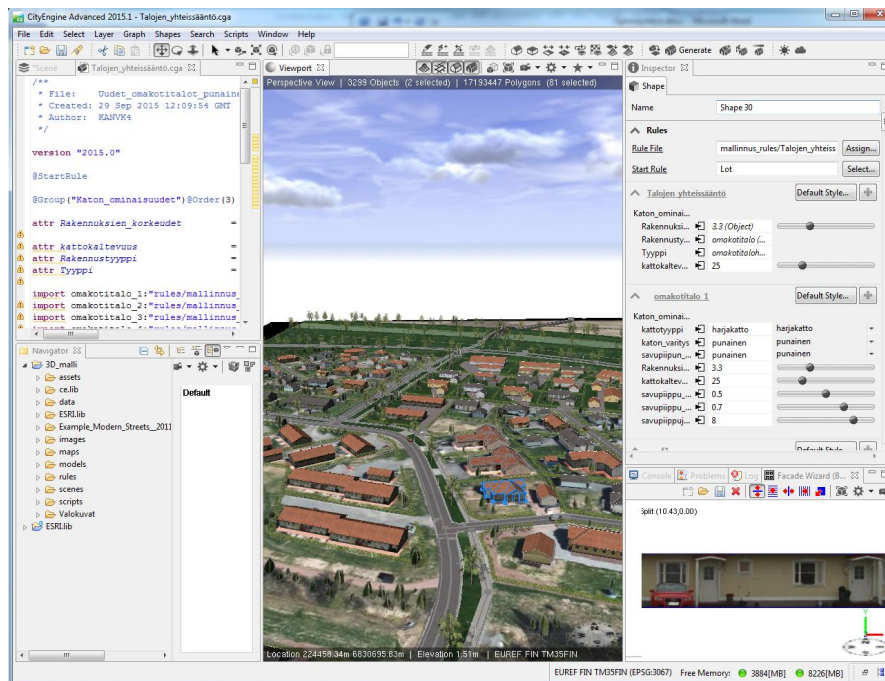
4.4 Mallinnus CityEngine-ohjelmistolla

Tässä työssä yhtenä tarkoituksena oli tutkia CityEngine-ohjelmiston mallinnusmenetelmiä ja perehtyä etenkin sääntöpohjaiseen mallinnukseen. Sääntömallinnuksella laadittiin asemakaavaluonnoksen havainnekuvan mukaisille rakennuksille (liite 2) neljä erilaista sääntöä; omakotitaloille, rivitaloille, paritaloille ja autotalleille (liitteet 4–8). Maankäyttöaineisto sekä tonttiraja-aidat mallinnettiin myös sääntömallinnusta hyödyntäen (liitteet 9 ja 10). Tutkimusalueen muille rakennuksille sovellettiin erityyppisiä mallinnusmenetelmiä. Osalle rakennuksista laadittiin säännöt, osa mallinnettiin julkisivukuvan perusteella (liite 11), osa mallinnettiin manuaalisesti ja osa tuotiin tiedonsiirtona SketchUp-ohjelmasta. Julkisivukuvan ja osa manuaalisesti mallinnettujen talojen säännöistä yhdistettiin yhteen sääntötiedostoon (liite 12), jotta välttyttiin useiden erilaisien sääntöjen käyttämiseltä. Sääntöjen yhdistäminen myös nopeutti mallinnustyötä. Kuvamallinnukseen perustuvaa yhtenäistä sääntöä voitiin käyttää mihin tahansa pientaloalueen rakennukseen leveydestä tai pituudesta riippumatta. Rakennuksen tyyppi valikoitui aineiston attribuuttitiedon perusteella. Säännöt sovitettiin 1–1,5 kerroksisille taloille. Sääntöjä hieman muokkaamalla voidaan mallintaa korke-

ampienkin taloja. Puiden ja teiden mallintamisessa hyödynnettiin CityEnginen harjoituksista löytyviä valmiita sääntöjä.

CityEngine-ohjelmassa mallinnus tapahtuu työtilassa, joka perustetaan mallinnustyön alussa. Työtilaan muodostettiin projekti (project) ja sille halutussa koordinaatistossa sijaitseva näkymä (scene). Työtilan ESRI.lib-kansiorakenne sisältää valmiita kuvia ja sääntöjä mallinnuksen laatimiseen. Luotu projekti sisälsi myös kansiorakenteen, johon mallinnuksessa käytettävät aineistot tallennettiin. Myös mallinnuksessa käytetyt valmiit säännöt tallennettiin kuvineen ja objekteineen projektin kansiorakenteeseen. Projekti voi sisältää useita eri näkymiä, mutta vain yksi näkymä voi olla kerrallaan auki.

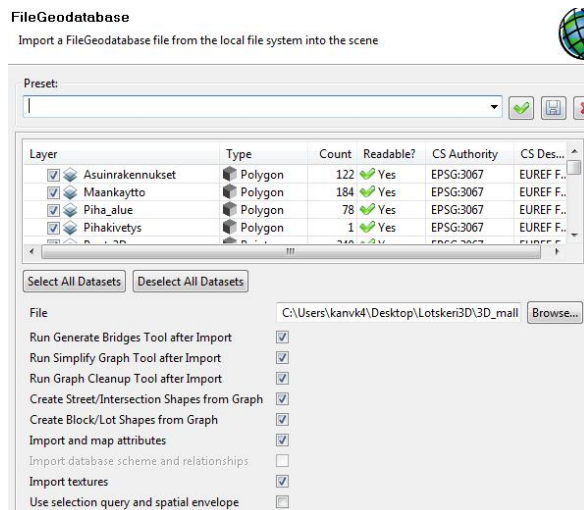
Kuviossa 15 on esitetty CityEnginen työtila projektin kansiorakenteineen. Vasemmassa ylälaudassa on säännön editointi-ikkuna ja oikealla alhaalla julkisivumallinnukseen tarkoitettu Facade wizard. Inspector-ikkunassa oikealla ylhäällä näkyy valitun rakennuksen tietoja.



Kuvio 15. CityEnginen työtila.

Vektorimuotoiset paikkatietoaineistot tallennettiin data-kansioon FileGDB -muotoiseen tiedostokansioon feature class -muodossa. Aineistot tuotiin projek-

tin näkymään import-komennolla. Komentoasetuksista voitiin esimerkiksi valita, yleistettiinkö tuotavaa dataa, muodostettiin tieverkko tai muodostettiin tie-aineistolle korttelirakenne (kuvio 16). Korkeusmallin tuominen näkymään tapahtui import-komennolla, kuten muidenkin aineistojen osalla. Komentoasetuksissa voitiin valita korkeusmallille haluttu tekstuuripinta. Tässä työssä tekstuuripintana käytettiin ilmakuvaa.



Kuvio 16. Aineistojen tuominen projektin näkymään Import-toiminnolla.

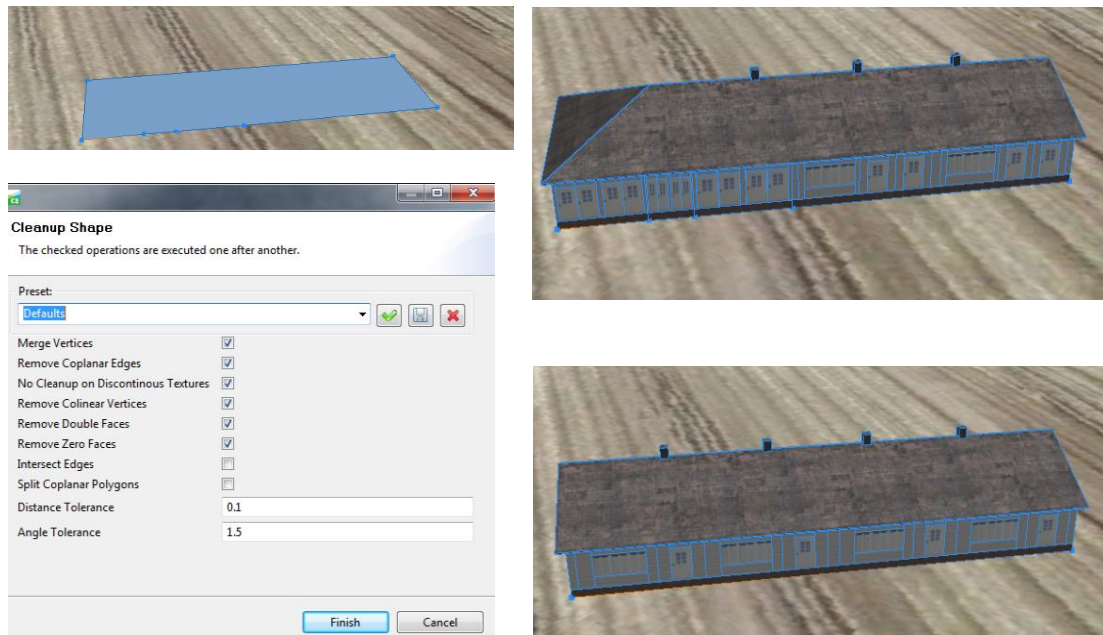
Kuviossa 17 vasemmalla on esitetty CityEngineen tuotu rakennusaineisto, buferoidut tonttirajat sekä puut. Kuviossa oikealla on esitetty mallinnuksen pohjaksi luotu maankäyttöaineisto kokonaisuudessaan.



Kuvio 17. ArcGIS-ohjelmassa luotu ja CityEngineen tuotu pohja-aineisto.

Ennen mallinnuksen aloittamista rakennusaineistoille oli tehtävä aineiston geometrian puhdistus Cleanup Shapes -toiminnolla vääristyneiden katto- ja seinä-

muotojen välttämiseksi. Kuviossa 18 on esitetty aineiston geometrian puhdistamisen merkitys mallinnuksen lopputulokseen. Kuviossa ylävasemmalla on kuvattu rakennuksen pohja ennen geometrian puhdistamista. Ylhäällä oikealla on kuvattu säännöllä mallinnettu rakennus ilman geometrian puhdistusta. Alhaalla vasemmalla on esitetty geometrian puhdistamiseen tarkoitettu työkalu Cleanup Shape. Alhaalla oikealla on säännöllä mallinnettu ja geometrialtaan puhdistettu rakennus.



Kuvio 18. Kuviossassa on esitetty aineiston geometrian puhdistamisen merkitys mallinnuksen lopputulokseen.

4.4.1 Sääntöihin perustuva mallinnus

CityEngine-ohjelmiston pääasiallinen 3D-mallintaminen perustuu sääntöpohjaiseen mallintamiseen. Sääntöpohjainen mallinnus perustuu ohjelman omaan CGA -ohjelmointikielen (CGA, Computer Generated Architecture). Sääntöjä voi ladata CityEngine-ohjelmiston harjoituksista tai niitä voi laatia itse ohjelmointikielen periaatteiden mukaisesti. Sääntöjä rakennetaan muodoille (esimerkiksi rakennusaineistolle), ja ne määrittävät miten muodon geometria ja muut yksityiskohdat rakennetaan. Sääntöjen peruseriaatteena on korvata muoto aina yksityiskohtaisemmalla muodolla erilaisten operaattoreiden avulla.

Sääntömallinnuksen tarkastelu on tässä opinnäytetyössä jaettu viiteen osioon seuraavasti: **asemakaavan uusien rakennuksien mallintaminen, maankäyttöaineiston mallintaminen, puiden mallintaminen, pensasaitojen mallintaminen sekä teiden mallintaminen**. Jokaisessa osiossa tarkastellaan sääntöjen muodostamisen rakennetta pääpiirteissään. Kerran selitettyä sääntörakennetta ei toisteta uudelleen sääntöjen rakenteen esittelyssä. Sääntömallinnuksella luodut säännöt ovat kokonaisuudessaan esitetty liitteissä 4–10. Tieaineiston ja pistemäisen puuaineiston sääntöjä ei tässä työssä ole esitetty tarkemmin, sillä teiden ja pistemäisten puiden mallintamiseen käytettiin Esrin omia sääntötiedostoja.

Asemakaavakohteen uusien rakennuksien mallintaminen aloitettiin luomalla uusi sääntötiedosto, joka avattiin sääntöeditoriin. Ohjelma muodosti automaattisesti sääntötiedostoon nimen, luontipäivän sekä luojan nimen. Säännön luettavuuden vuoksi kaikki säännössä käytetyt attribuutit (attr), tekstuurit sekä muut määritteet koottiin yhtenäiseksi kokonaisuudeksi säännön alkuun seuraavasti:

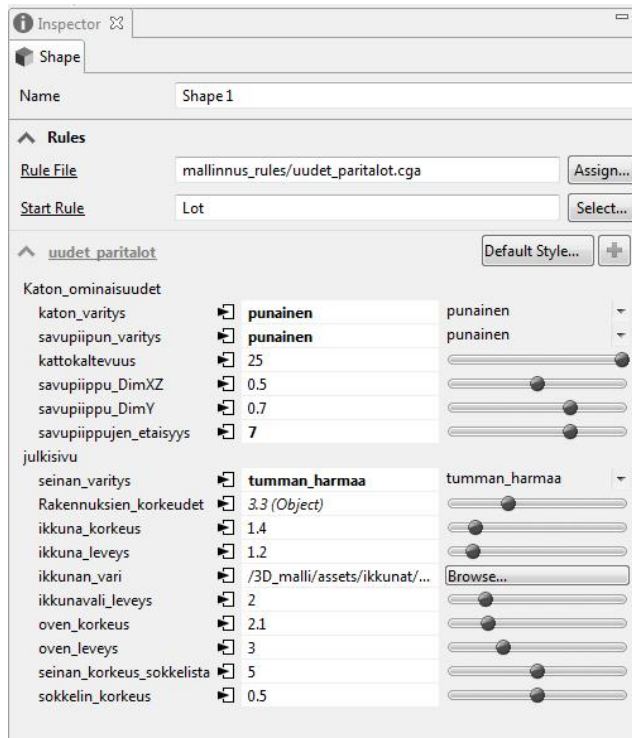
```
////////ATRIBUUTIT//////////////////////////////////////
attr attribuutin nimi                      = arvo
//////////////////////////////////////
```

Atribuutit ovat säännössä määritellyjä ominaisuuksia, joita voitiin hallinnoida joko sääntöikkunassa tai inspector-ikkunassa (kuviot 15 ja 19). Atribuutit sijoitettiin inspector-ikkunaan halutun pääotsikon alle käyttämällä attribuuttien edellä seuraavaa selitettä:

```
@Group ("nimi")
```

Mallinnettavan kohteen sisältäessä vaihtoehtoisia ominaisuuksia (esimerkiksi seinätekstuurit), määritettiin attribuutin edellä Range-selitteessä kaikki vaihtoehtoiset ominaisuudet. Ominaisuudet voivat olla joko tekstityyppisiä tai numeroita. Kohteen ominaisuuden vaihtoehdot olivat valikoitavissa inspector-ikkunan nuolinäppäintä painamalla. Käyttämällä Range-selitteen edellä selitettä Order(), saatiin attribuutit sijoitettua inspector-ikkunassa haluttuun kohtaan (kuvio 19):

```
@Order(1) @Range("punainen","harmaa")
attr katon_varitys = "punainen"
```



Kuvio 19. Inspector-ikkunassa esitetään näkymäikkunassa valitun kohteen tiedot.

Säännön muodostaminen aloitettiin aina StartRule-komennolla. Rakennusaineiston kohteet (Lot) saatiin nostettua extrude-operaattorilla rakennusaineiston attribuuttitiedoissa määritellyn korkeuteen. Jotta sääntö osasi hyödyntää rakennusaineiston attribuuttitietoja, oli korkeustiedon sisältävän sarakkeen nimi kerrottava säännössä ja annettava attribuuttitiedon arvoksi 0 seuraavasti:

```
attr Rakennuksien_korkeudet      = 0
```

```
Lot -->
```

```
    extrude (Rakennuksien_korkeudet) rakennukset
```

Jos rakennus haluttiin nostaa johonkin tiettyyn korkeuteen, voitiin arvon 0 tilalle antaa haluttu luku. Tämä tapahtui seuraavasti:

```
Lot -->
```

```
    extrude (4) rakennukset
```

Rakennuksien seinät jaettiin osiin käyttäen comp() -operaattoria (component split). Operaattorin (f)-parametri (faces) kertoi säännölle, että rakennuksen pinnat jaetaan osiin, jolloin rakennus sisälsi tässä tapauksessa neljä osaa:

```
rakennukset-->
```

```
comp(f){ street.front : etuseina | street.back: takaseina
          | street.side : sivut | top: katto }
```

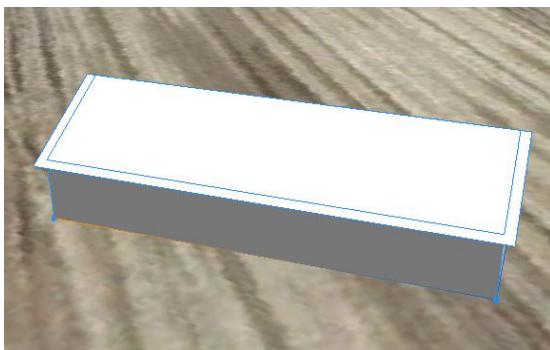
Luoduille uusille osille voitiin edelleen määritellä uusia operaatioita. Katon muodostamista varten luotiin kolme erilaista kattosääntöä, joiden perusteella muodostui harjakatto, aumakatto tai pulpettikatto. Katto-operaattorille annettiin kattokaltevuus sekä rakennuksen reunan ylittävän kattoalueen pituus. Harjakatolle ja aumakatolle reunan ylittäväksi osuudeksi annettiin 0,5 metriä. Pulpettikatolle tätä määritettä ei pystynyt antamaan. Katto jaettiin edelleen osiin katon yksityiskohtaisempaa mallintamista varten sekä halkaistiin 3D-volyymina y-akselin suuntaisesti. Halkaisu tehtiin, jotta voitiin muodostaa alue, johon savupiippu mallinnetaan. NIL-operaatio poistaa annetun mittaisen osuuden muodosta:

```
katto -->
case kattotyyppi == "harjakatto":
    roofGable(kattokaltevuus, 0.5, 0.5)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä }
    split(y) { ~5 : NIL | ~0.1 : savupiipun_alue }

case kattotyyppi == "aumakatto":
    roofHip(kattokaltevuus, 0.5)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä }
    split(y) { ~5 : NIL | ~0.1 : savupiipun_alue }

case kattotyyppi == "pulpettikatto":
    roofShed(kattokaltevuus, 2)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä }
    split(y) { ~5 : NIL | 0.7 : savupiipun_alue }
else: X.
```

Rakennuksen noston ja katonmuodostamisen jälkeen rakennus saatiin mallinnettua kuvion 20 mukaiseen pelkistettyyn muotoon. Pelkistetyn rakennuksen sääntöä jatkettiin tarkemman mallinnuksen aikaansaamiseksi.



Kuvio 20. Esimerkki yksinkertaisella säännöllä luodusta 3D-rakennuksesta.

Rakennusten katoille annettiin kaksi vaihtoehtoista kattotekstuuria, jotka olivat valittavissa inspector-ikkunassa. Vaihtoehtoiset kattotekstuurit määritettiin säännössä case-määritteellä. Vaihtoehtojen erittely lopetettiin else:X.-määritteellä, joka palautti tekstuurittoman pinnan. Rakennusten katot tehtiin al-

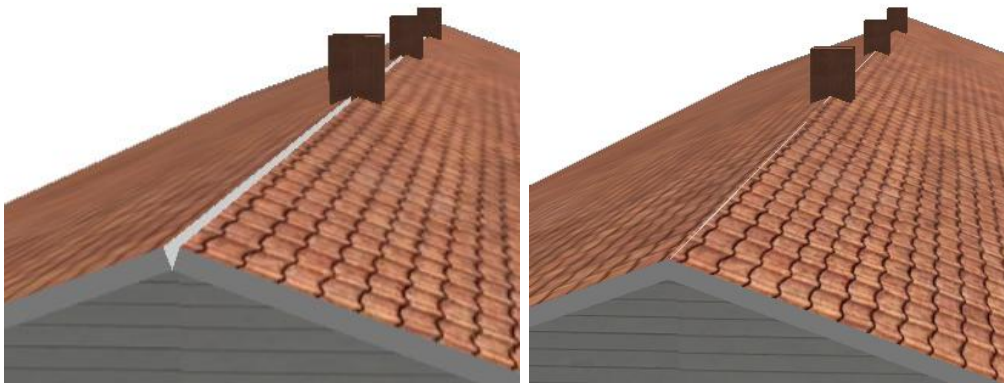
kuperäistä paksummiksi todellisemman ulkoasun saamiseksi extrude-operaattorilla. Tämän jälkeen kattoa siirrettiin t-operaattorilla vektorin z-suuntaisesti, jotta katon harjoille muodostuneet kolot saatiin umpeen (kuvio 21). Katoille määritettiin projektiot ja katon pinnalle projisoitiin tekstuuri `setupProjection` ja `projectUV`-operaattoreiden avulla. Katon tekstuuri sovitettiin objektin pinnalle `tileUV`-operaattorin avulla. Operaattoriin määriteltiin tekstuurin haluttu pituus ja leveys. Pituus- ja leveysasetukset olivat olennaisia, jotta tekstuuri kuvautui tarkoituksenmukaisesti. Tämän jälkeen kattorakenne jaettiin vielä osiin, jotta voitiin teksturoida myös kattorakenteen sivut ja alapuoli. Yksityiskohtaiset kattorakenteet muodostettiin seuraavasti:

```
kattotekstuuri-->
case katon_varitys == "punainen":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("katto/katto_1.jpg")
    tileUV(0,7, 10)
    comp(f) {side: katonreuna|bottom:katonreuna|top:X.}

case katon_varitys == "harmaa":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("katto/katto_2.jpg")
    tileUV(0,10, 10)
    comp(f) {side: katonreuna|bottom:katonreuna|top:X.}

else: X.

katonreuna-->
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_6.jpg")
```



Kuvio 21. Katon harjan korjaaminen säännön t-operaattorilla.

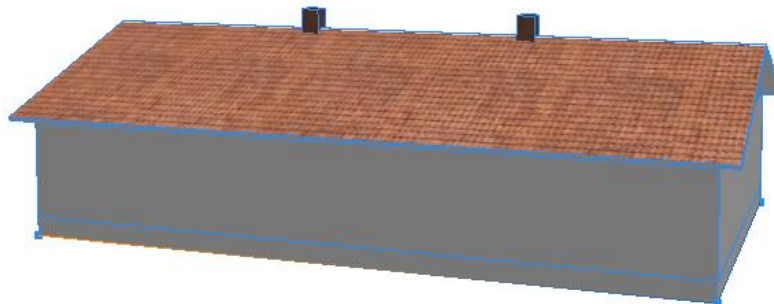
Katolle muodostettu savupiipun alue halkaistiin x-akselin suuntaisesti, jolloin saatiin määritettyä savupiippujen jakautuminen katolle. Halkaisun jälkeen muodostettiin savupiippu. Savupiipulle määritettiin samat väri vaihtoehdot kuin katolle case-määritteen avulla. Savupiipun ulottuvuudet määriteltiin s-operaattorilla. Ulottuvuuksien arvot määriteltiin attribuuttiedoissa. Savupiippu sijoitettiin keskelle kattoa center-operaattorilla ja muodostettiin kuution muotoiseksi seuraavasti:

```
savupiipun_alue -->
    split(x) {{~savupiippujen_etaisyys : NIL |
    0.1 : savupiippu }*
    | ~savupiippujen_etaisyys : NIL}

savupiippu -->
case savupiipun_varitys== "punainen":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    texture("katto/katto_1.jpg")
    tileUV(0,7, 10)
    s(savupiippu_DimXZ, savupiippu_DimY, savupiippu_DimXZ)
    center(xz)
    i("builtin:cube")
case savupiipun_varitys== "harmaa":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    texture("katto/katto_2.jpg")
    tileUV(0,~10, 20)
    s(savupiippu_DimXZ, savupiippu_DimY, savupiippu_DimXZ)
    center(xz)
    i("builtin:cube")
else: X.
```

Rakennuksen jokainen seinä jaettiin y-suuntaisiin osiin split-operaattorilla (kuvio 22). Y-suuntaisiin osiin jakamisella saatiin erotettua sokkeli talon muusta seinästä seuraavasti:

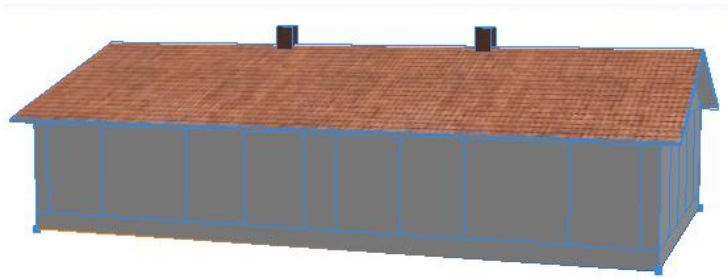
```
etuseina-->
    split (y) {sokkelin_korkeus:sokkeli |
    seinan_korkeus_sokkelista: etuosan_seina}
```



Kuvio 22. Talon sokkeli erotettuna muusta seinästä.

Sokkelin yläpuolinen seinäosuus jaettiin edelleen x-suuntaisiin osiin (kuvio 23). Pystysuuntaisilla seinäjoilla määriteltiin alueet ikkunoiden ja ovien sijoittumiselle seuraavasti:

```
etuosan_seina-->
    split(x){~0.5: seinä |
    {~ikkunavali_leveys : vali} |
    ~oven_leveys: ovialue|
    {~ikkunavali_leveys : vali} *|
    ~1: seinä|{~ikkunavali_leveys : vali} *|
    ~oven_leveys: ovialue|
    {~ikkunavali_leveys : vali}* |
    ~0.5: seinä }
```



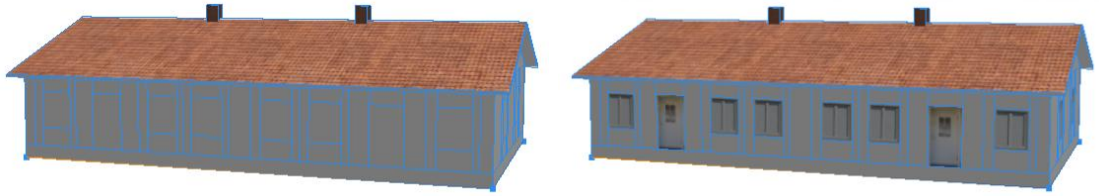
Kuvio 23. Talon sokkelin yläpuolinen seinäosuus jaettuna x-suuntaisiin osiin.

Käyttämällä leveysarvojen edessä merkkiä ~, muodostuivat arvot suhteelliseksi. Jos kyseistä merkkiä ei ole arvon edessä, oli annettu mitta absoluuttinen. Tähtimerkillä voitiin monistaa kohdetta. Edellä mainittujen merkkien käyttäminen oli olennaista rakennuksen seinien sovittamiseksi erimittaisiin taloihin. Jaetut seinien välit jaettiin edelleen pienempiin osiin ikkuna-aukkojen ja oviaukkojen muodostamiseksi (kuvio 24). Nämä muodostetut ikkuna-aukot ja oviaukot muodostettiin ikkunoiksi ja oviksi asettamalla niihin ikkuna- ja ovitekstuurit (kuvio 24). Ikkunatekstuurit valikoituvat satunnaisotannan perusteella fileRandom-funktiolla neljästä erilaisesta vaihtoehdosta:

```
attr ikkunan_vari = fileRandom ("assets/ikkunat/ikkuna_*.jpg")

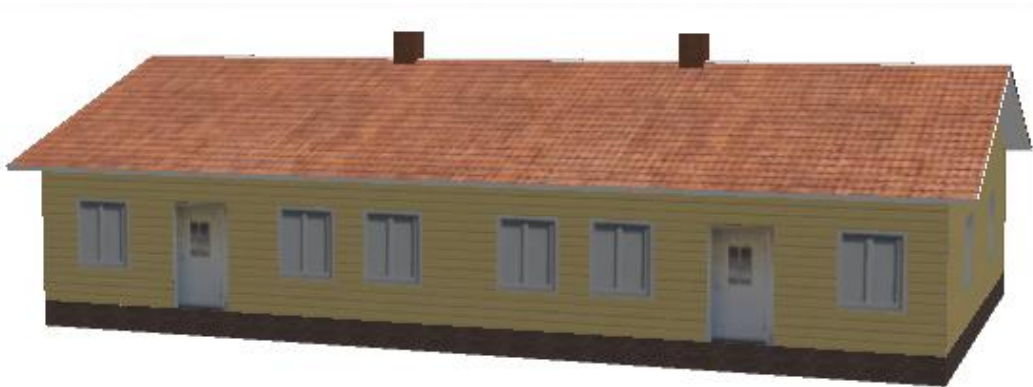
vali-->
    split (x) {~1.5: seinä| ikkuna_leveys:ikkuna_aukko |
    ~1:seinä}*
ikkuna_aukko-->
    split (y){0.7:seinä |ikkuna_korkeus:ikkuna |~1:seinä }*
ikkuna-->
    setupProjection(0,scope.xy,scope.sx,scope.sy) projectUV(0)
    texture(ikkunan_vari)
ovialue-->
    split (x) {~0.2:seinä |1.2:split (y) {oven_korkeus :ovi
    |~0.2:seinä }*|~0.2: seinä}
ovi-->
    setupProjection(0,scope.xy,scope.sx,scope.sy) projectUV(0)
```

```
texture("ovet/valk_ovi.jpg")
```



Kuvio 24. Vasemmalla talon seinärakenteeseen muodostetut ikkuna- ja oviauheet. Oikealla valmiiksi mallinnettu talo tekstuureineen.

Ikkunatekstuuria vaihdettiin painamalla update seed and generate models -painiketta. Talon seinien teksturointia varten valokuvattujen talojen seinäpinnoista leikattiin tekstuurinäytteet, jotka tallennettiin projektin kansiorakenteeseen. Tekstuurinäytteitä käytettiin seinäosuuden ja sokkelin teksturoimiseen (kuvio 25). Seinän tekstuureina käytettiin viittä erilaista seinätekstuuria, joiden ominaisuudet määriteltiin säännössä case-määritteen avulla. Lopuksi talon sokkelille määritettiin tekstuuri.



Kuvio 25. Valmiiksi mallinnettu paritalo tekstuureineen.

Maankäyttöaineisto mallinnettiin pääasiassa vireillä olevalle asemakaava-alueelle sekä juuri rakennetuille tonteille. Näin tehtiin siitä syystä, että ilmakuvalla teksturoitu korkeusmalli ei soveltunut kuvaamaan näiden alueiden maapohjaa. Maankäyttöaineisto käsitti myös nurmialueita kaava-alueen ulkopuolelta, joihin haluttiin mallintaa puustoa. Tutkimusalueen muu alue maankäyttöaineiston ulkopuolella esitettiin ilmakuvalla teksturoidun korkeusmallin avulla. Maankäyttöaineiston eri maankäyttötyyppien mallinnuksessa hyödynnettiin paikkatietoaineiston attribuuttitietoja, joiden avulla saatiin mallinnettua jokaiselle maankäyttötyypille omanlaisensa ulkoasu. Maankäyttöaineiston lisäksi pistemäisille

kohteille käytettiin Esrin internetsivujen harjoituksista löytyvää valmista puustosäätöä (Esri 2015b), joka muodosti pisteistä 3D-puita. Puustosäätö käytti puiden mallintamiseen valmiita 3D-puumalleja, jotka olivat obj-formaatissa assets-kansiossa.

Maankäyttösäännön tekstuureina käytettiin sekä Esrin omia että itse kuvattuja tai ilmakuvasta leikattuja tekstuuripintoja. Käytetyt tekstuurit kuvasivat hiekkaa, soraa, asfalttia, ruohoa ja pihakivetystä (kuvio 26). Tekstuurit tallennettiin projektin kansiorakenteen assets-kansioon.



Kuvio 26. Maankäyttöaineiston mallintamisessa käytetyt tekstuurit vasemmalta oikealle: sora, nurmi, viheralue, hiekka, joenreuna.

Ennen maankäyttöaineiston säännön muodostamista, maankäyttöaineisto asetettiin lähelle korkeusmallin pintaa align shapes to terrain-työkalulla ja käyttämällä translate to maximum -funktiota. Funktio asetettiin maankäyttöaineiston kokonaisuudessaan korkeusmallin maksimitasoon, jolloin maankäyttöaineisto sijoittui korkeusmallin päälle. Maankäyttöaineiston niitä kohtia, jotka olivat merkittävästi ilmakuvaan nähden eri korkeustasolla, muutettiin manuaalisesti lähelle korkeusmallin pintaa.

Maankäyttöaineiston mallintaminen toteutettiin siten, että jokaiselle maankäyttöluokalle laadittiin omat sääntönsä, jotka on esitetty kokonaisuudessaan liitteessä 9. Vesialueen mallintamiseen käytettiin Esrin vesisäätöä, jonka mukaisesti interaktiivinen 3D-malli (web scene) osasi muuntaa mallinnetun veden interaktiiviseen muotoon. Vesisäätö käsitti viisi erilaista vesistövaihtoehtoa (joki, puro, lampi, järvi ja meri), jotka voitiin vaihtaa inspector-ikkunassa. Jokaiselle vesistötyypille määritettiin oma RGB -väriarvot, mittakaava ja aallonnopeus seuraavasti:

```

Vesi -->
    case Water_Type == "River" :
        set(material.name, "river__water__waterparams_5_10")
        color(.44,.55,.44)
    case Water_Type == "Stream" :
        set(material.name, "river__water__waterparams_5_30")
        color(.44,.55,.44)
    case Water_Type == "Pond" :
        set(material.name, "river__water__waterparams_1_3")
        color(.2,.4,.45)
    case Water_Type == "Lake" :
        set(material.name, "river__water__waterparams_4_8")
        color(.2,.4,.45)
    case Water_Type == "Ocean" :
        set(material.name, "river__water__waterparams_15_20")
        color(.2,.4,.45)
    else: x.

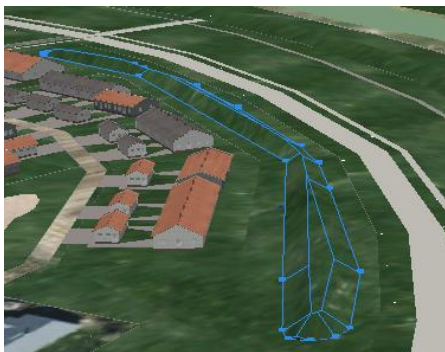
```

Asemakaava-alueen meluvalli mallinnettiin harjakattosääntöä (roofGable) käyttäen ja leikkaamalla harjan huippu tasaiseksi split-operaattorilla (kuvio 27). Lopuksi meluvalli teksturoitiin vieheraluetekstuurilla ja sovitettiin kohteen pinnalle tileUV-operaattorin avulla seuraavasti:

```

Meluvalli-->
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    roofGable(30,0,0,false,0)
    split(y){3:Huippu}
Huippu-->
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture(viheralue_tekstuuri)
    tileUV(0,~100,80)

```



Kuvio 27. Säännöllä muodostettu maankäyttöaineiston meluvalli.

Tienvierustojen nurmiluokka muodostettiin jakamalla alue puuosaan ja viheralueosaan. Puuosalle muodostettiin puut ja viheralueosa laadittiin kuvaamaan nurmipeitteistä maanpintaa. Puuosuus muodostettiin leikkaamalla split-operaattorilla ja käyttämällä parametriä `unitSpace`. Sääntö leikkasi pinnan pisimmän sivun mukaisesti ja sijoitti tyhjän alueen puiden etäisyystribuutin arvon mittaisille välimatkoille. Tämän jälkeen sääntö sijoitti seuraavassa säännössä määritellyn valmiin 3D-puun tyhjän alueen viereiselle nurmialueelle puiden etäisyystribuutin mukaisesti (kuvio 28):

```
Nurmi-->
  alignScopeToGeometry(yUp, 0, longest)
  setupProjection(0,scope.xz,scope.sx,scope.sz)
  projectUV(0)
  comp(f) {top : Viheralue Nurmen_puut}

Nurmen_puut-->
  alignScopeToGeometry(yUp, 0, longest)
  t(0,0,0)
  split(u,unitSpace,0){ ~ puiden_etaisyydet*0.5 : NIL
    | {5: PUU_nurmi | ~ puiden_etaisyydet:NIL}*}

PUU_nurmi-->
  alignScopeToAxes(y)
  s(0,Nurmen_puiden_korkeus,0) center(xz)
  r(0,rand(0,360),0)
  i("/ESRI.lib/assets/Plants/Paper_Birch/Paper_Birch_Model_0.obj")
  set(material.opacity,1.0)
```



Kuvio 28. Puiden asettuminen tienviereiselle nurmialueelle.

Jokivallin vallimainen muoto muodostettiin `roofGable`-operaattorilla meluvallin tapaan ja jaettiin osiin samalla tekniikalla, kuten nurmialue. Jako tehtiin säännöl-

lä kolmeen osaan, viheralueeseen ja kahdenlaiseen puualueeseen. Puut sijoitettiin jokivallille tasaisesti scatter-operaattorin avulla. Operaattorille määritettiin kuinka paljon puita halutaan sijoittaa. Tämän jälkeen sääntö sijoitti valmiin 3D-puun annetuin ominaisuuksin i-funktion perusteella (kuvio 29). Puun korkeudet ja pyöriminen y-akselin suhteen määriteltiin vaihtelevaan satunnaisesti rand-funktiolla:

```
Puut-->
    scatter(surface, 2, uniform) {PUU_X }

Havupuut-->
    scatter(surface, 2, uniform) {PUU_Y }

attr PUU_X_korkeus    = rand (7,15)
attr PUU_Y_korkeus    = rand (8,20)

PUU_X -->
    alignScopeToAxes(y)
    s(0,PUU_X_korkeus,0) center(xz)
    r(0,rand(0,360),0)
    i("/ESRI.lib/assets/Plants/Norway_Spruce/
      Norway_Spruce_Model_0.obj")

PUU_Y -->
    alignScopeToAxes(y)
    s(0,PUU_Y_korkeus,0) center(xz)
    r(0,rand(0,360),0)
    i("/ESRI.lib/assets/Plants/Paper_Birch/Paper_Birch_Model_0.obj")
```



Kuvio 29. Säännöllä muodostettu joki, jokivalli puineen sekä joenreuna.

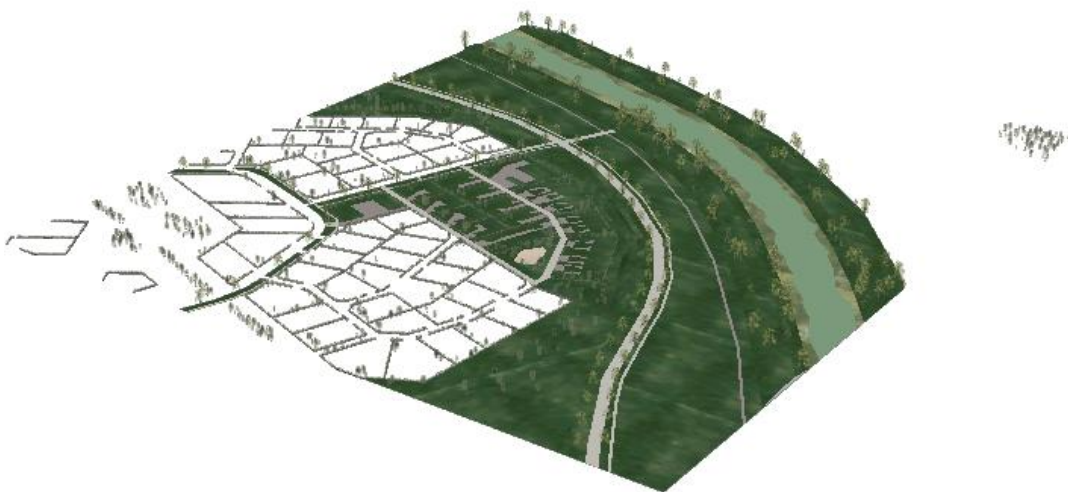
Maankäyttöaineiston muut maankäyttöluokat teksturoitiin annettujen parametrien mukaisesti (kuvio 30, liite 9). Maankäyttöaineiston asemakaava-alueen tiealueet muutettiin myöhemmin viheralueeksi, jotta harmaat alueet eivät näy erillisen tieaineiston alta.

Pistemäistemäisten puiden mallintaminen tapahtui Esrin verkkosivuilta löytyvän puusäännön avulla. Sääntö loi puut satunnaisvalinnalla hyödyntäen assets-kansiossa sijaitsevia valmiita 3D-puita. Ennen puuaineiston mallintamista puuaineisto asetettiin korkeusmallin pinnalle käyttämällä project all-funktiota, jolloin kaikki pisteet asettuivat täsmälleen korkeusmallin pinnalle.

Pensasaitojen mallintaminen tontteja ympäröiville bufferoiduille tonttirajoille tapahtui sääntömallinnuksella. Sääntö hyödynsi Esrin assets-kansiosta löytyvää valmista obj-formaatissa olevaa 3D-pensasta. Pensasaitauksen muodostamisessa käytettiin hyvin samankaltaista sääntörakennetta kuin maankäyttöaineiston nurmialueen puiden muodostamisessakin (liite 10). Pensasaidan alue jaettiin pensasosuuteen ja pohjatekstuuriin. Pensasosuuteen muodostettiin pensaat ja pohjatekstuuriksi asetettiin ilmakeku. Ilmakeku asetettiin pohjatekstuuriksi määrittämällä ilmakekuvan ulottuvuudet projektioasetuksiin seuraavasti:

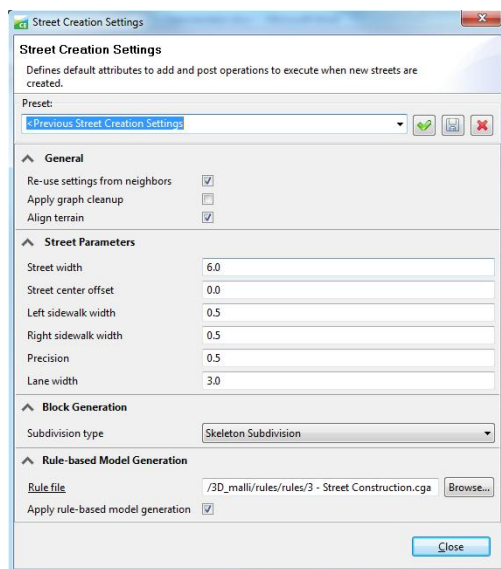
```
attr katto_Ilmakeku = ("images/ilmakeku_ETRSTM35fin3.tif")
attr ilmakekuvan_koko_X = 639.200
attr ilmakekuvan_koko_Z = 715.400
attr ilmakekuvan_offset_X = 224111.957
attr ilmakekuvan_offset_Z = -6831115.704

pohjatekstuuri-->
    setupProjection (0,world.xz,ilmakekuvan_koko_X,-
        ilmakekuvan_koko_Z,ilmakekuvan_offset_X,ilmakekuvan_offset_Z)
    projectUV(0)
    texture(katto_Ilmakeku)
```



Kuvio 30. Valmis säännöllä mallinnettu maankäyttöaineisto, puuaineisto sekä tonttiraja-aita-aineisto.

Teiden mallintamiseen käytettiin Esrin verkkosivuilta ladattavaa Modern Street Template -sääntöä (Esri 2013), jonka asetuksia muokattiin soveltumaan tutkimuskohteeseen. Tieaineistona ei käytetty Porin kaupungin pohjakartan tieaineistoa, vaan tieaineisto luotiin testimielessä suoraan CityEngine-ohjelmassa. Ennen tieaineiston luomista asetettiin tienmuodostusasetukset Street Creation Settings -valikosta (kuvio 31). Valikosta voitiin muun muassa määrittää käytetäänkö tien muodostamisessa sääntöä.



Kuvio 31. Asetusvalikko teiden muodostamista varten.

Tieasetusten jälkeen tieaineisto luotiin Polygonal Street Creation -työkalulla. Tiet ja kävelytiet muodostettiin maankäyttöaineistossa määriteltujen teiden sekä ilmakuvassa näkyvien teiden päälle. Myöhemmässä vaiheessa sillaksi muodostettava tiealue piirrettiin oikealle kohdalleen asemakaavan perusteella. Kävelyteiden päätepisteitä ei liitetty kiinni varsinaiseen tieaineistoon, vaan ne jätettiin irrallisiksi tieosuuksiksi tieaineiston välittömään läheisyyteen (kuvio 32). Kävelyteiden katkaisu tehtiin, koska CityEngine ei hyväksynyt teiden liittämistä liian lähelle toista risteysaluetta. Teiden mutkat pyöristettiin curves auto smooth -työkalulla visuaalisesti näyttävämmän mallinnuksen saamiseksi. Tieaineistossa käytetyn säännön asetuksia (kaistaleveydet, ihmisten ja autojen määrä, katuvalaistus, kävelytiet) muokattiin soveltumaan tutkimusalueen tieaineistoon. Lopuksi tieaineistolle muodostettiin silta edit streets/curves-työkalun avulla, jolla kävelytie nostettiin siltaa muistuttavaan kaarevaan muotoon (kuvio 32).

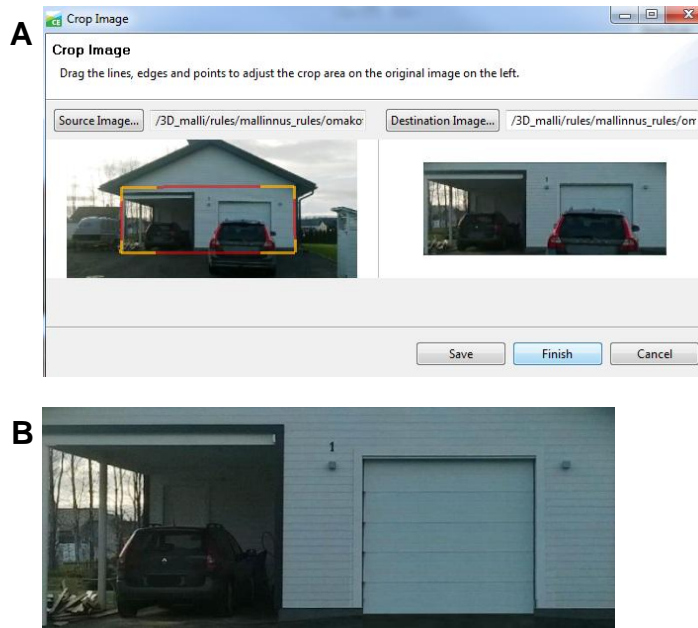


Kuvio 32. Tieaineiston, kävelytieaineiston sekä sillan muodostaminen.

4.4.2 Julkisivukuvaan perustuva mallinnus

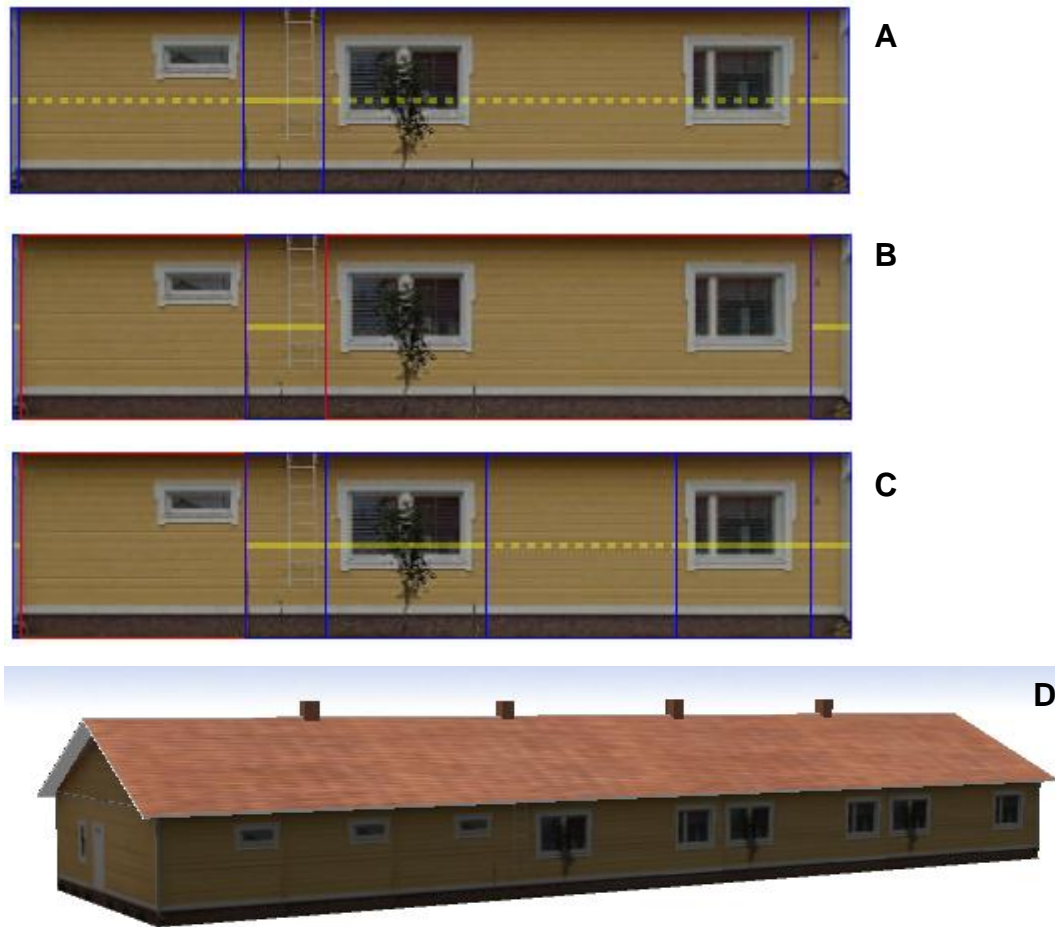
CityEngine-ohjelmalla on mahdollista mallintaa esimerkiksi rakennuksia myös julkisivukuvan perusteella. Kuvan perusteella mallintaminen tapahtuu FacadWizard-ikkunassa (kuvio 15). FacadeWizardilla voidaan luoda GCA -sääntö kustakin rakennuksen julkisivusta kuvan perusteella (liite 11). Kuvamallinnuksessa syntyneet seinien säännöt yhdistettiin talosääntöön, jolloin talo saatiin mallinnettua kokonaisuudessaan.

Kuvamallinnuksen perusteella mallinnettiin erilaisia rakennuksia, joista viisi oli omakotitaloa, kolme autotallia, kolme varastoa sekä yksi rivitalo. Kaikkia kuvamallinnuksella mallinnettujen rakennuksien sääntöjä ei ole esitetty liitteissä. Liitteessä kahdeksan on esitetty kuvamallinnuksella syntyneen keltaisen omakotitalon sääntötiedosto. Ennen kuvamallinnusta kuvat leikattiin CityEnginen Crop Image -työkalulla (kuvio 33, A) ja käsiteltiin tarvittaessa paint-kuvankäsittelyohjelmassa. Kuvankäsittelyssä parannettiin kuvan värejä ja poistettiin tarvittaessa mallinnusta häiritseviä elementtejä (kuvio 33, B).



Kuvio 33. Kuvassa A julkisivukuvan leikkaaminen Crop Image -työkalulla. Kuvassa B on esitetty Paint-kuvankäsittelyohjelmalla editoitu kuva.

Käsittelyt kuvat avattiin FacadeWizardiin säännön muodostamista varten. FacadeWizardissa kuvalle määritettiin viivojen avulla toistettavat, venyvät ja muuttumattomaksi jätettävät pinnat. Näin saatiin luotua sääntö, joka soveltui erikokoisten rakennuksien mallintamiseen. Käyttämällä punaisia viivoja, voitiin pintoja monistaa. Sinisiä viivoja sekä keltaisia viivoja käyttämällä määritettiin pysyvä pinta muuttumattomana, vai saako pinta venyä. FacadeWizardin käyttö oli yksinkertaista, mutta viivojen asettamisen järjestyksestä tuli olla tarkka, jotta sääntö toimi halutunlaisesti. Sinisillä viivoilla jaettiin julkisivu alueisiin, jonka jälkeen luotujen alueiden venyvyys tai muuttumattomuus määritettiin keltaisten viivojen avulla (kuvio 34, A). Keltainen katkoviiva mahdollistaa alueen venymisen ja keltainen yhtenäinen viiva säilyttää alueen muuttumattomana. Punaisella viivalla määritettiin monistetaanko aluetta vaakasuuntaan tai pystysuuntaan (kuvio 34, B). Monistuksen jälkeen määriteltiin sinisten sekä keltaisten viivojen avulla, mitkä monistuvan alueen kohteista halutaan säilyttää muuttumattomina (kuvio 34, C). Luoduille osille määritettiin tarvittaessa niiden todelliset korkeudet tai leveydet set region width/length -työkalun avulla. Kuviossa 34, D on esitetty valmis julkisivukuvaan perustuvalla mallinnuksella laadittu rakennus.



Kuvio 34. Julkisivukuvaan perustuva kuvamallinnuksen eteneminen FacadeWizardilla.

Talon kaikkien julkisivujen mallintamisen jälkeen luotiin uusi sääntötiedosto. Luotuun sääntötiedostoon yhdistettiin kuvamallinnuksessa muodostetut yksittäisten seinien sääntötiedostot import-operaattorilla seuraavasti:

```
Import f1: "rules/mallinnus_rules/omakotitalo1_kuvamallinnus/
omakotitalo_etusivu.cga"
```

Uudessa sääntötiedostossa hyödynnettiin osittain aiemmin luodun talosäännön rakennetta (liite 4). Kuvan perusteella mallinnettujen rakennuksien säännöissä talon seinät jaettiin comp-operaattorilla useampiin osiin (liite 11). Useaan osaan jaetut seinät mahdollistivat erilaisen julkisivusäännön sijoittamisen jokaiselle erilliselle seinälle. Tämä oli erityisen tärkeää mallintaessa taloja, joissa on useita seinäelementtejä. Kuviossa 35 on havainnollistettu seinien jakamisen vaikutus talon julkisivukuvaan. Kuviossa vasemmalla puolella talon seinät on jaettu

comp-operaattorilla kolmeen osaan (etuseinä, takaseinä ja sivut). Kuviossa oikealla puolella jokainen talon seinä on jaettu omaksi seinäkseen, jolloin jokaiselle erilliselle seinälle muodostuu säännössä määriteltä kuva. Sääntötiedostoon tuodut julkisivusääntötiedostot osoitettiin kullekin julkisivulle seuraavasti:

```
sivu2--> f4.Facade
sivu3--> f2.Facade
sivu4--> f1.Facade
sivu5--> f3.Facade
```



Kuvio 35. Talon säännön seinäjakojen määrän vaikutus mallinnetun talon julkisivukuvaan.

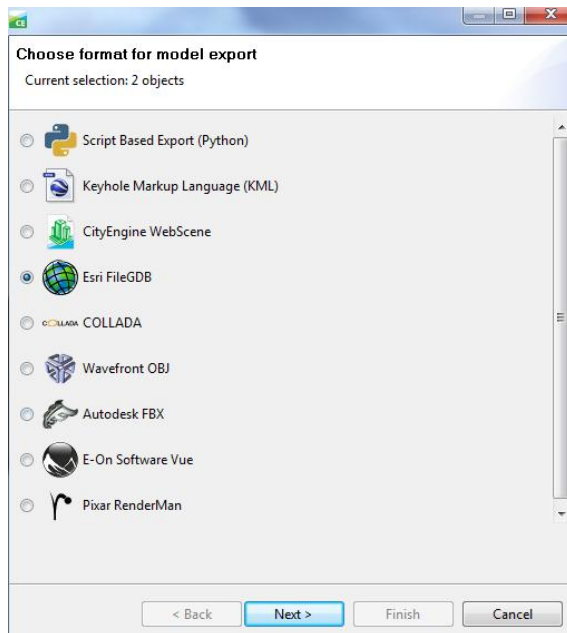
Mallintamisen yksinkertaistamiseksi yhdistettiin rakennuksien säännöt asema-kaava-alueen rakennuksien sääntöjä lukuun ottamatta yhteen sääntötiedostoon (liite 12). Talojen yhteissäännön hyödyntäminen perustui rakennusaineiston attribuuttitaulun ominaisuustietoihin.

Muutamia sääntömallinnuksella laadittuja 3D-malleja muodostettiin kiinteiksi malleiksi manuaalista editointia varten. Kiinteäksi malliksi muuttaminen mahdollistaa sekä mallin käytön muissa mallinnuksissa että sen manuaalisen editoinnin. Kuviossa 36 on esitetty kuvamallinnuksen jälkeen kiinteäksi malliksi muutettu rakennus, jonka katon muotoa on editoitu.



Kuvio 36. Kuvamallinnuksen jälkeen kiinteäksi malliksi muutettu ja katon muodoilta editoitu kattorakenne.

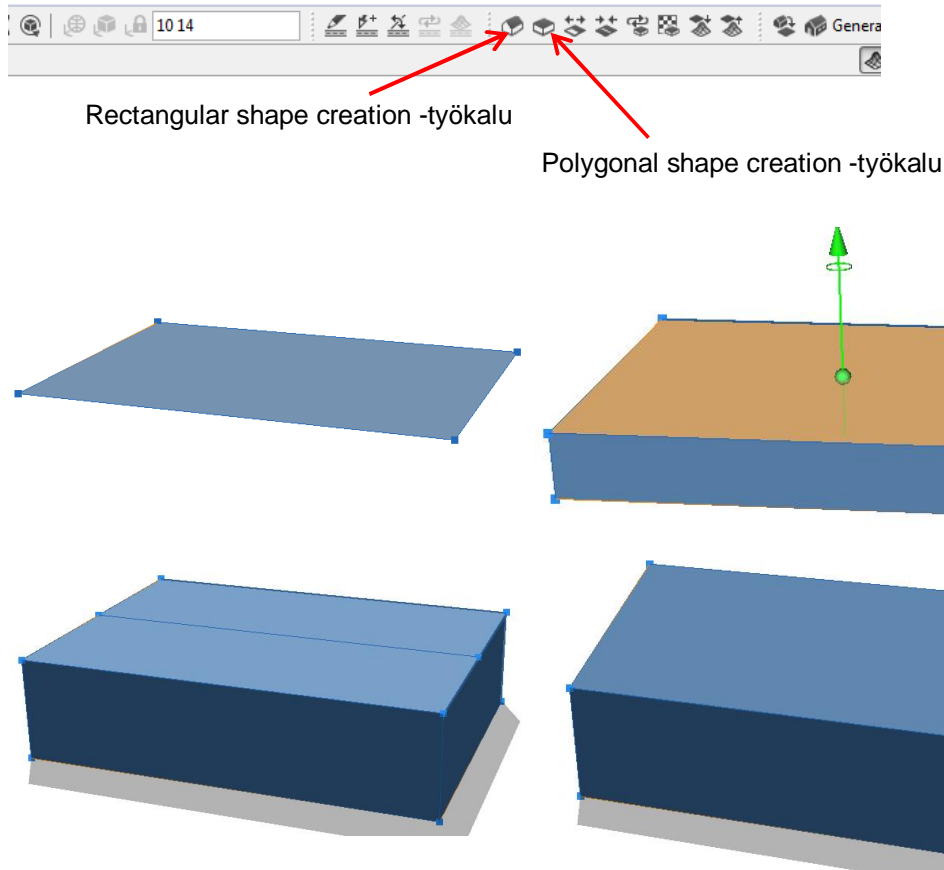
Mallit tallennettiin kiinteiksi malleiksi Esrin FileGDB -muotoon, sekä COLLADA-muotoon. FileGDB -muotoon tallennettaessa malleja voitiin editoida jälkikäteen CityEnginessä. Mallit tallennettiin COLLADA-muotoon myöhemmin tapahtuvaa SketchUp-tiedonsiirtoa varten. CityEngine tarjoaa myös useita muita tiedonsiirtoformaatteja (kuvio 37).



Kuvio 37. City Enginen tarjoamat tiedonsiirtoformaatit.

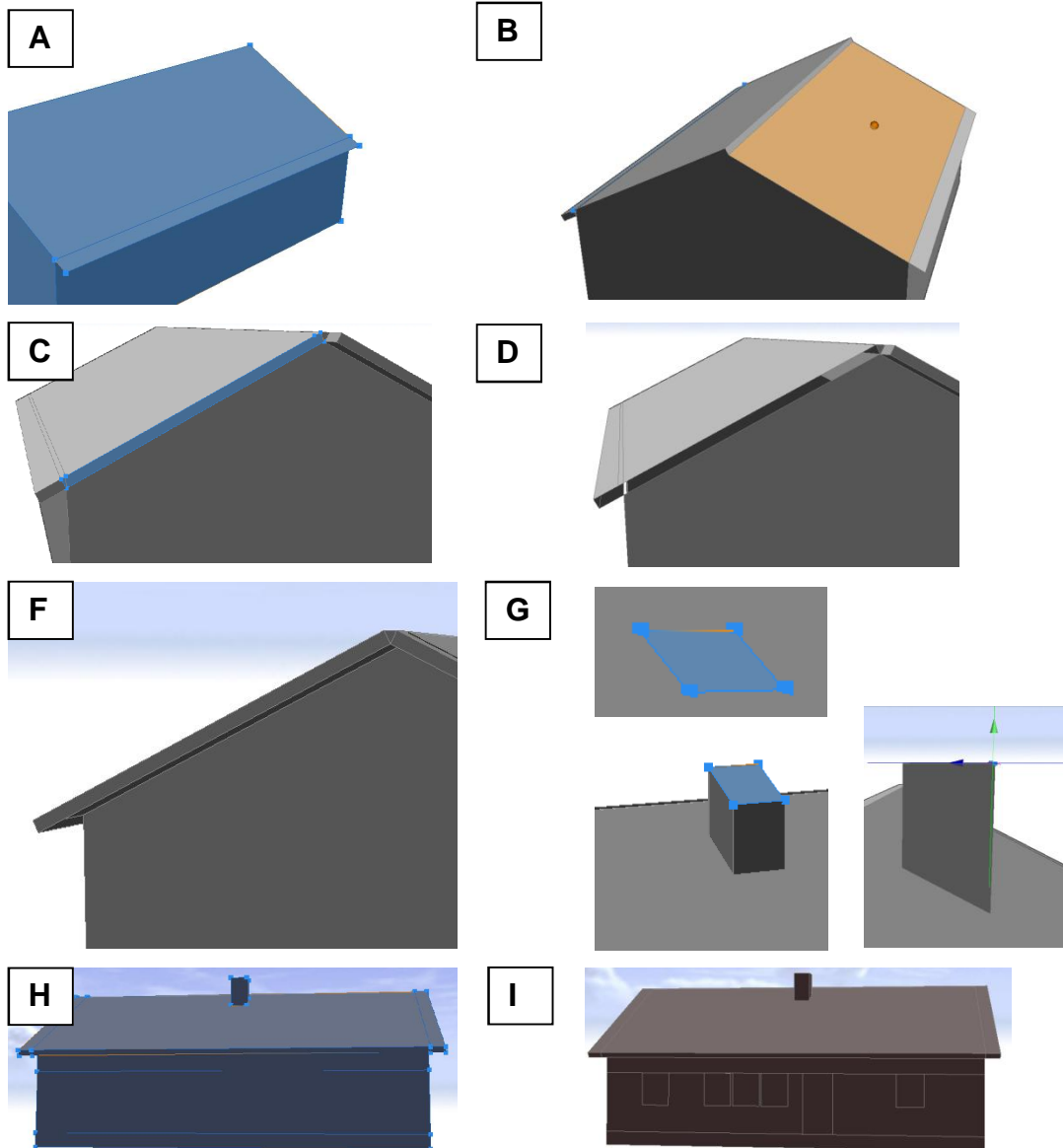
4.4.3 Mallintaminen piirtämällä

Piirtämiseen perustuva manuaalinen mallinnus tapahtuu CityEngine-ohjelmistolla polygonal shape creation ja rectangular shape creation -työkalujen avulla (kuvio 38). Polygonal shape creation -työkalulla laadittiin malli yksittäisiä viivoja piirtämällä. Rectangular shape creation -työkalulla malli laadittiin suorakulmaisesti. Polygonal shape creation -työkalulla voitiin jakaa pintoja piirtämällä viiva katkaistavalle kohdalle. Osoittamalla mallia halutuista kohdista, voitiin vetämällä ja työntämällä luoda 3D-malli (kuvio 38). Halutut mitat annettiin mittakenttään, jolloin ohjelma asetti viivan tai suorakulmion halutun pituiseksi.



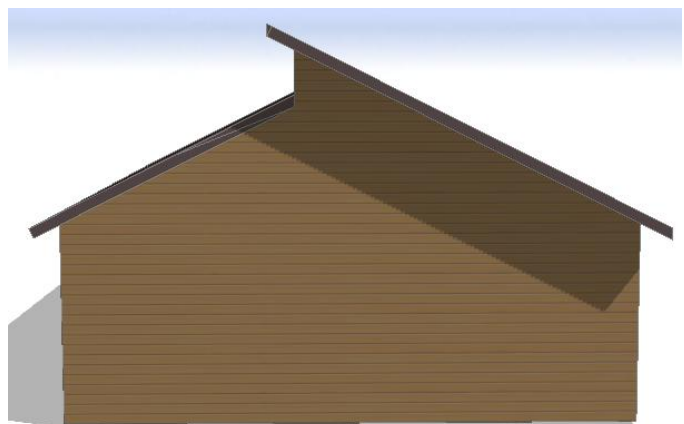
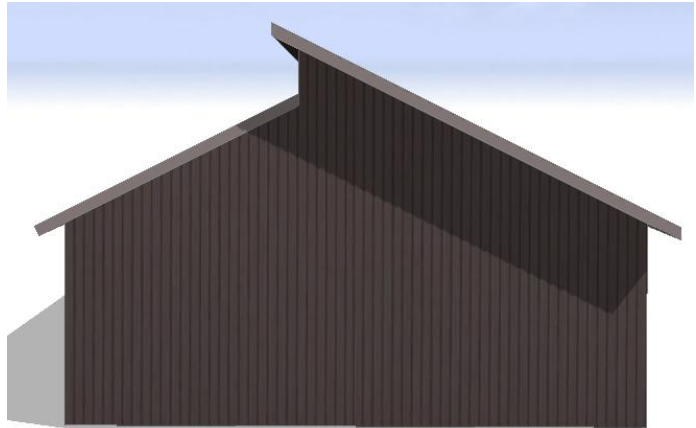
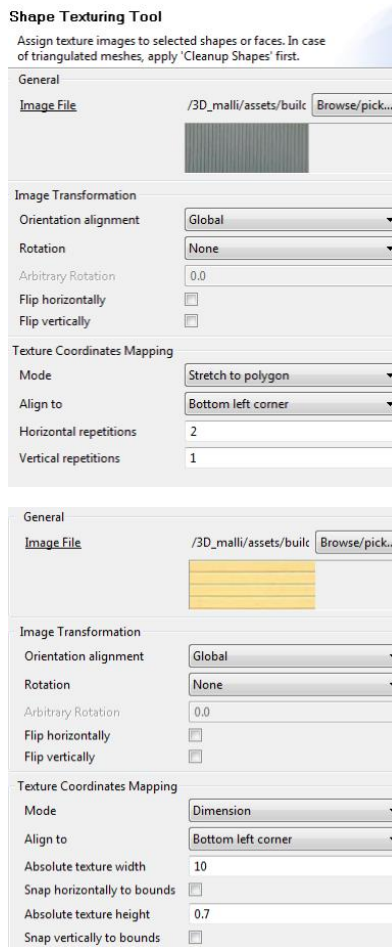
Kuvio 38. CityEnginen manuaalisesti tapahtuva mallinnus perustuu rectangular ja polygonal shape creation -työkalujen käyttöön.

Tässä työssä mallinnettiin piirtämällä muutama rakennus sekä autotalli. Piirtäminen aloitettiin talon pohjan piirtämisellä kuvion 38 osoittamalla tavalla. Tämän jälkeen piirrettiin katon levikkeet ja nostettiin kattoa ylöspäin (kuvio 39, A ja B). Katon sivuosien pinnoilta poistettiin ylimääräiset viivat ja muodostettiin tilalle yhtenäinen pinta (kuvio 39, C ja D). Yhtenäisen pinnan muodostamisen jälkeen pintaa pystytettiin vetämään, jolloin muodostettiin myös katon sivuille levikkeet (kuvio 39, F). Katolle muodostettiin savupiippu suorakulmaista polygonal shape creation -työkalua käyttämällä. Tämän jälkeen muodostettu pinta vedettiin haluttuun pituuteen move-työkalun avulla (kuvio 39, G). Savupiipun muodostamisen jälkeen talo oli valmis yksinkertainen malli, johon pystyi asettamaan julkisivukuvan tai muun halutun tekstuurin. Autotalli mallinnettiin käyttämällä samoja tekniikoita kuin rakennuksen mallinnuksessa.



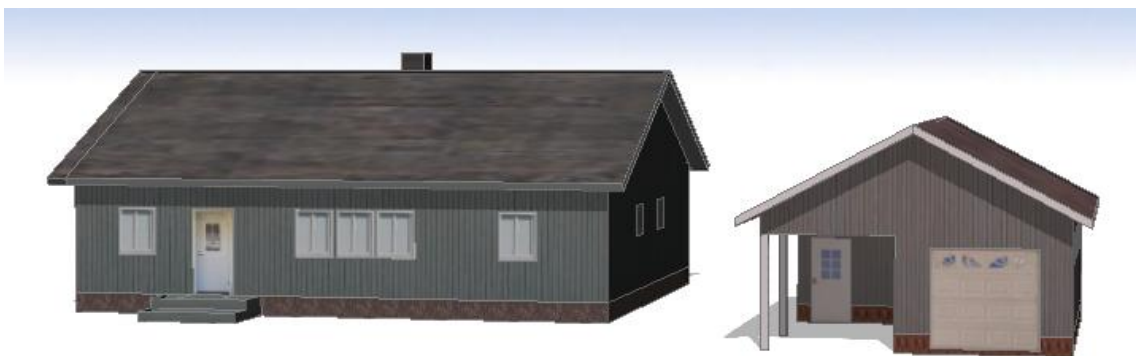
Kuvio 39. Piirtämällä tapahtuvan rakennuksen mallintamisen vaiheet.

Yksityiskohtaisempaa mallinnusta varten taloon piirrettiin ikkunat ja ovet polygonal shape creation -työkalua käyttämällä (kuvio 39, H ja I). Lisäksi taloon piirrettiin sokkeli, terassi sekä rappuset rectangular shape creation -työkalua käyttämällä. Ovet ja ikkunat teksturoitiin ikkunan ja oven kuvilla. Talon tekstuurit valittiin shape texturing tool -työkalulla. Työkalun asetuksissa määriteltiin miten kuva tuodaan kohteen pinnalle (kuvio 40). Tekstuurin tuontiasetuksissa voitiin säätää pysty- ja vaakasuuntaisten toistojen määrä tai pysty- ja vaakasuuntainen pituus.



Kuvio 40. Tekstuurien asettaminen mallin pinnalle stretch to polygon ja dimension-menetelmällä.

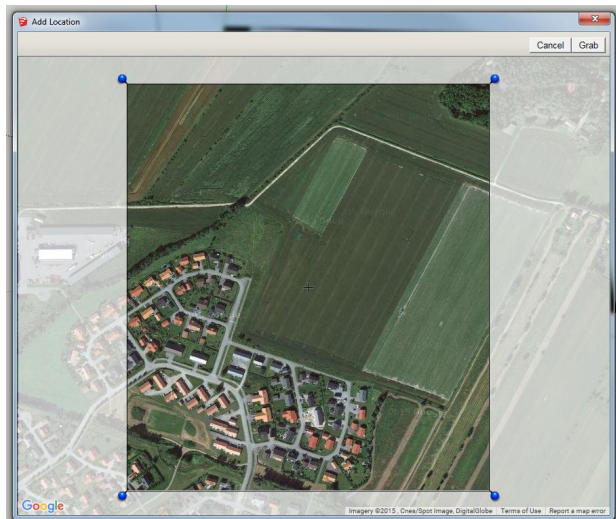
Valmiit mallinnukset muutettiin lopuksi kiinteiksi malleiksi export models -komennolla. Kiinteät mallit tallennettiin FileGDB -muotoon. Kuviossa 41 on esitetty valmis manuaalisesti mallinnettu rakennus autotalleineen.



Kuvio 41. Piirtämällä mallinnetut rakennukset tekstuureineen.

4.5 Mallinnus SketchUp -ohjelmistolla

SketchUp-ohjelmalla oli tarkoitus laatia samantapainen 3D-mallinnus tutkimusalueesta kuin CityEngine-ohjelmalla, mutta huomattavasti lyhyemmässä ajassa. Mallinnuksessa käytettiin apuna ilmakuvaa, rakennusaineistoa sekä ArcGIS-ohjelmassa tuotettua maankäyttöaineistoa. Pohja-aineistot muutettiin ArcGIS-ohjelmassa AutoCAD-muotoon, jolloin aineistot oli siirrettävissä SketchUp-ohjelmaan. Korkeusmallia ei voitu siirtää SketchUp-ohjelmaan, sillä ohjelman tiedostoformaatit tukevat vain dem- ja ddf-tiedostoformaatteja. SketchUp-ohjelmistoon voitiin ladata geo-location-työkalun avulla Google Earth -palvelun korkeusmalli (kuvio 42).

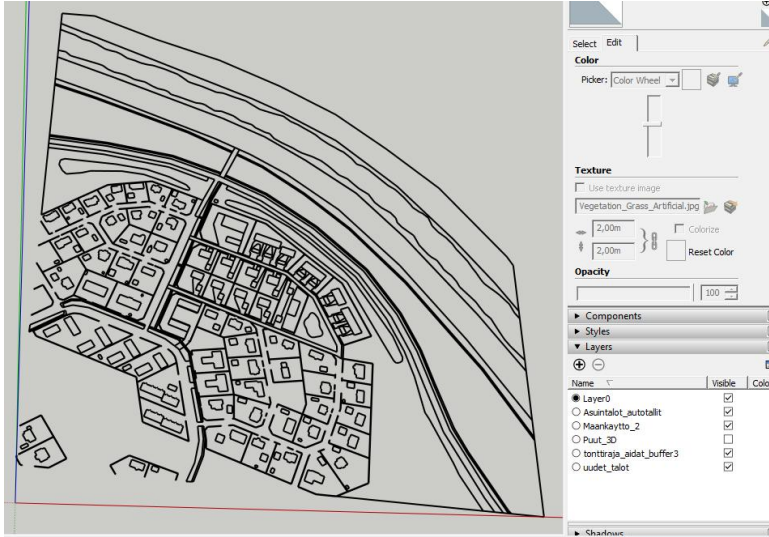


Kuvio 42. Korkeusmallinpinnan tuominen geo-location -työkalun avulla Google Earth -palvelusta.

Google Earth -palvelun korkeusmallia ei käytetty mallinnuksen korkeusmallipintana sen heikon korkeustarkkuuden vuoksi. Korkeusmallia ei myöskään saatu kohdistettua tarpeeksi hyvin olemassa olevaan pohja-aineistoon. Tästä syystä SketchUp-mallinnuksen pohjana käytettiin Porin kaupungin ilmakuvaa.

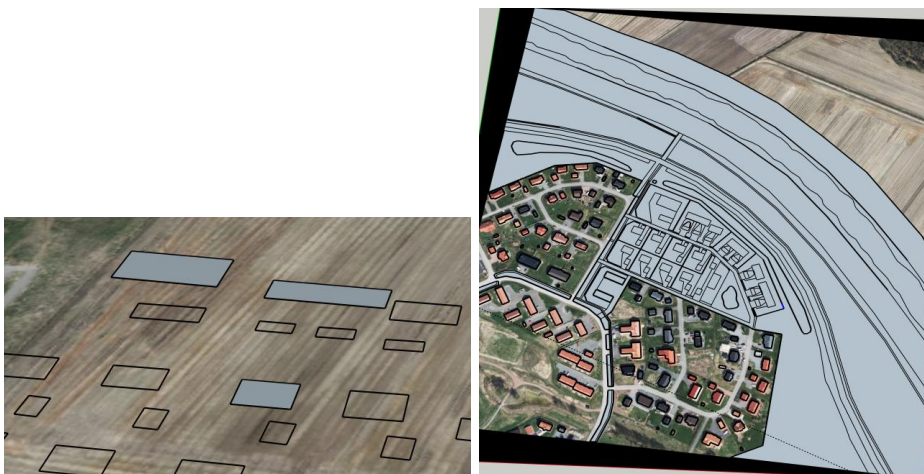
Ennen varsinaista mallinnusta ilmakehä sekä pohja-aineistot tuotiin työtilaan omiksi tasoikseen (kuvio 43). Mallinnuksen pohjatekstuurina käytettävä ilmakehä skaalattiin oikealle kohdalle tutkimusaluetta, sillä SketchUp-ohjelmisto ei tunnistanut koordinaattitietoja. Maankäyttöaineisto ja rakennusaineisto kuvau-

tuivat ohjelmaan tuodessa viivamaisina kohteina. Aineiston aluemaiseksi muuttaminen tapahtui hajottamalla aineisto explode-työkalulla osiin ja tämän jälkeen muodostamalla alueiksi piirtotyökalua hyödyntäen. Piirtotyökalulla näpätettiin alueen muutamaa päätepistettä, jolloin kohde muuttui alueeksi (kuvio 44).



Kuvio 43. SketchUp-ohjelmistoon tuotu pohja-aineisto.

Kuviossa 43 on esitetty AutoCAD-muotoisen aineiston aktivoiminen aluemuotoiseksi. Kuviossa vasemmalla on esitetty muutamia piirtotyökalulla aluemuotoiseksi aktivoituja kohteita. Kuviossa oikealla on esitetty aktivoituina kaikki mallinnuksessa käytettävät kohteet.

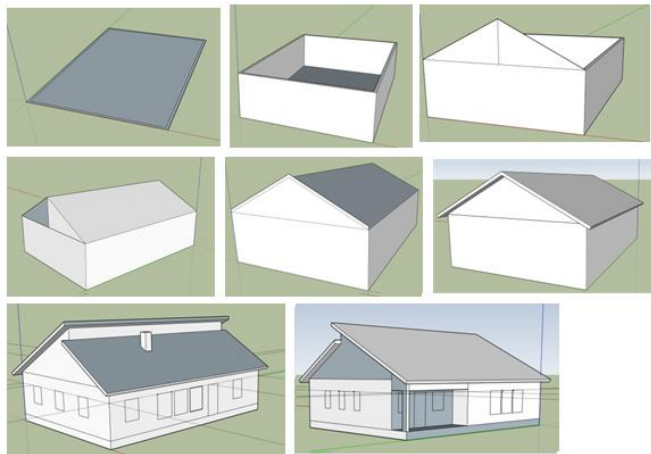


Kuvio 44. AutoCAD-muotoisen aineiston aktivoiminen aluemuotoiseksi

Mallinnus SketchUp-ohjelmistolla tapahtui manuaalisesti piirtämällä. Mallintamiseen käytettiin SketchUp-ohjelman tarjoamia perustyökaluja. Mallintaminen ta-

pahtui hyvin pitkälti samalla tavalla kuin CityEngine-ohjelmassa tapahtuva ma-
nuaalinen mallinnus. Koska tutkimusalue sisälsi lukuisia rakennuksia, kopioitiin
mallinnetut rakennukset tutkimusalueen rakennusten paikoille. Ajallisten rajoit-
teiden takia koko rakennuskantaa ei mallinnettu SketchUp-ohjelmassa, vaan
osa rakennuksista tuotiin tiedonsiirtona CityEnginestä.

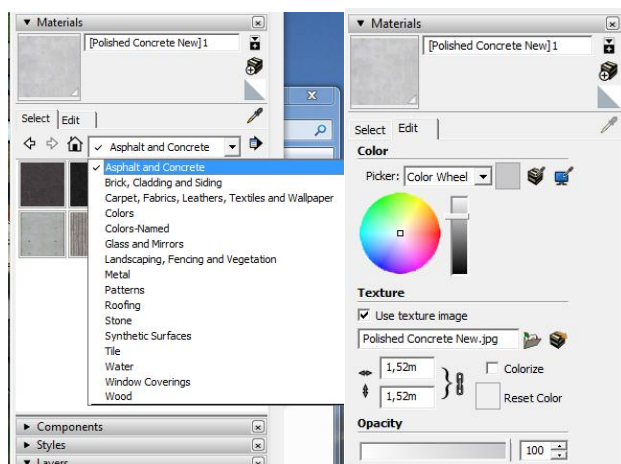
Rakennuksien mallintaminen aloitettiin käyttämällä apuna muutamaa rakennus-
aineiston rakennuksen pohjaa (kuvio 45). Rakennuksen pohja nostettiin
push/pull-työkalulla haluttuun korkeuteen. Korkeus syötettiin measurements-
ikkunaan, jolloin ohjelma asetti rakennuksen täsmälleen halutulle korkeudelle.
Rakennuksen katon muodostaminen tapahtui CityEngine-ohjelmaan verrattuna
eri tavalla. Katon muodostamiseksi piirrettiin rakennuksen reunalle kolmiot, joita
vetämällä muodostettiin katto. Katon paksuuden kasvattamiseksi tehtiin katon
reunoille halutun paksuiset kaistaleet, joita vetämällä muodostui kattopinta. Piir-
tämällä viivoja haluttuihin kohtiin, voitiin pintoja erottaa toisistaan. Vastaavasti
poistamalla viivoja pinnoilta, voitiin muodostaa yhtenäinen pinta (esimerkiksi
talon päätykolmio). Savupiippu piirrettiin muodostamalla savupiipun sivuprofiili
halutulle kohtaa kattoa. Tämän jälkeen savupiipun sivuprofiilia vedettiin halutun
levyiseksi. Rakennuksen ikkunoiden ja ovien piirtämisessä hyödynnettiin tape
measure tool -työkalua helpottamaan ikkunoiden ja ovien sijoittamista. Ikkunat
ja ovet piirrettiin piirtotyökaluilla apuviivojen osoittamiin kohtiin (kuvio 45). Sket-
chUp-ohjelma osasi myös ennakoida piirtämistä, jolloin se osoitti esimerkiksi
viivan suorakulmat.



Kuvio 45. SketchUp-ohjelmistolla mallintamisen eteneminen.

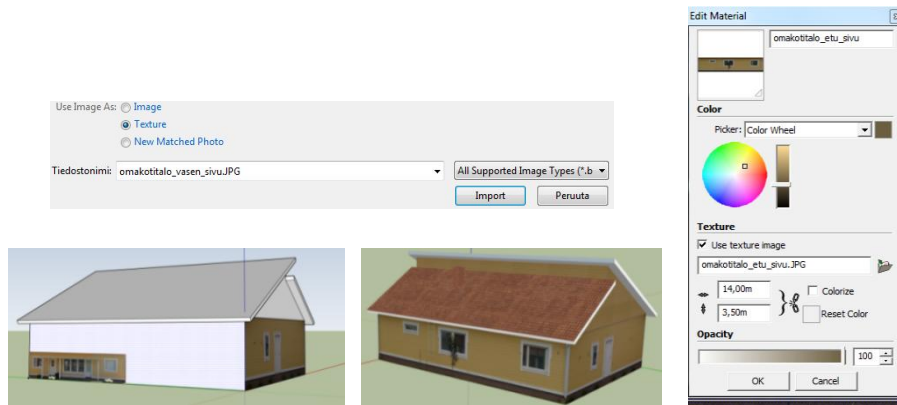
Mallien teksturoinnissa käytettiin SketchUp-ohjelman omia tekstuureita sekä kuvia. SketchUp-ohjelma ei sisällä kuvien muokkaustyökalua, jolla rajataan ja leikataan talon julkisivukuva. Tästä syystä rakennuksien julkisivukuvien esikäsittely tehtiin CityEngine-ohjelmassa. Tässä työssä käytettiin sekä SketchUp-ohjelmiston tarjoamia tekstuuripintoja sekä CityEngine-ohjelmistolla leikattuja julkisivukuvia, ovia, ikkunoita ja kattotekstuureita.

Valmis malli teksturoitiin paint bucket -työkalulla ja valitsemalla haluttu tekstuuri materials-valikosta. Tekstuurin ominaisuuksia muokattiin halutuksi materials-välilehden asetuksissa (kuvio 46). Teksturoidessa useaa pintaa samalla tekstuurilla, voitiin tekstuuri kopioida helposti jo teksturoidulta pinnalta. Tämä tehtiin valitsemalla paint bucket-työkalu ja painamalla alt-näppäintä tekstuurin pinnalla. Tämän jälkeen painettiin teksturoitavaa pintaa, jolloin kopioitu tekstuuri siirtyi kohteen pinnalle.



Kuvio 46. Tekstuurin valitseminen SketcUp-ohjelman materials-tekstuurivalikosta ja tekstuurin ominaisuuksien muuttaminen.

Julkisivukuvien, ikkunoiden ja ovien lisääminen tekstuuripinnaksi tapahtui import-komennolla. Julkisivukuvan tuontiasetuksista valittiin texture-vaihtoehto. Tuotu kuva venytettiin kohteen pinnalle, jonka jälkeen edit material-valikosta muutettiin kuvan mittasuhteet kohteeseen sopivaksi (kuvio 47). Tuotu julkisivukuva kopioitiin tarvittaessa haluttuihin pintoihin edellisessä kappaleessa kuvatulla tavalla.



Kuvio 47. Julkisivukuvan asettaminen rakennuksen julkisivun tekstuuriksi.

Valmiiksi mallinnetut rakennukset (kuvio 48) muutettiin komponenteiksi. Komponenttina olevaa kappaletta voitiin muokata ja liikutella näkymän pinnoilla ilman, että muut aineistot muuttuivat. Jos mallia ei ole muutettu komponentiksi, liikkuvat myös muut näytöllä olevat aineistot mallia liikutettaessa. Valmiit komponentit monistettiin ja liitettiin ilmakuvalla näkyvillä talojen paikoille. Asemakaava-alueen talot sijoitettiin rakennusaineiston mukaisille paikoille. Kuviossa 48 on esitetty SketchUp-ohjelmassa laaditun mallinnuksen 3D-rakennukset. Kuvion yläladassa on esitetty rakennukset, joiden julkisivu on laadittu piirtämällä ja tekstuureja lisäämällä. Kuvion alalaidan rakennuksissa on hyödynnetty CityEngine-ohjelmassa muokattuja julkisivukuvia.



Kuvio 48 SketchUp-ohjelmistolla mallinnetut rakennukset.

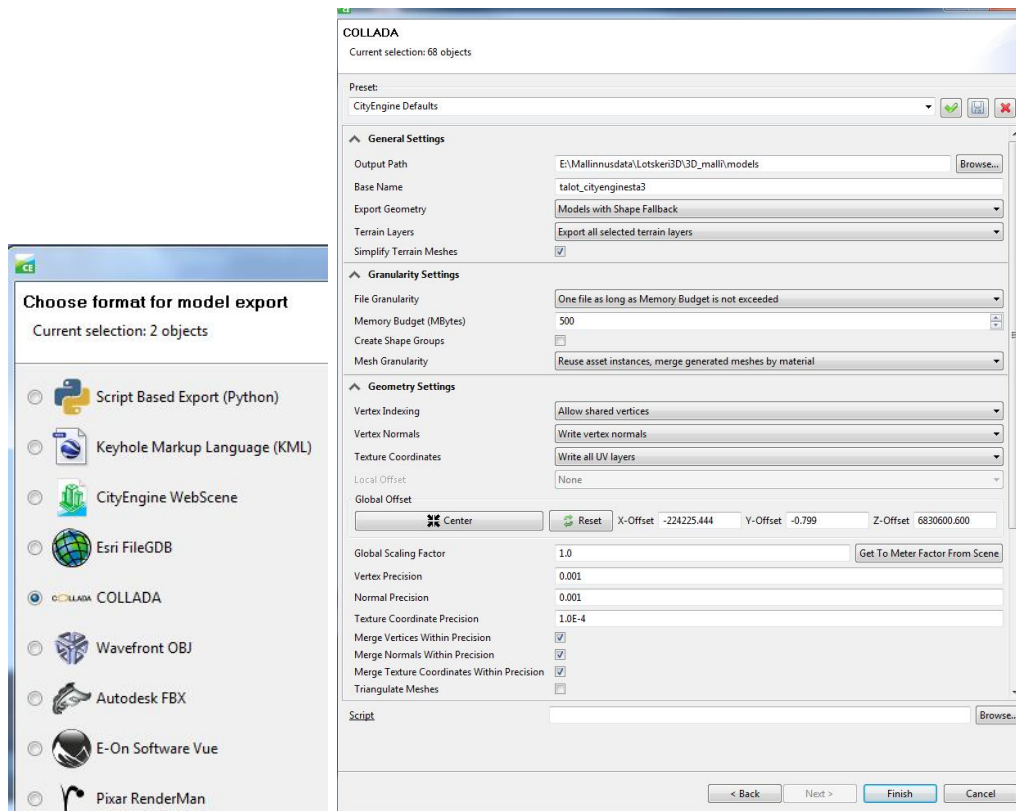
Maankäyttöaineisto teksturoitiin materiaalivalikosta löytyvillä tekstuureilla (kuvio 49). Muutamia tonttien pihakivetyksen alueita jouduttiin muuttamaan, jotta mallinnettujen talojen sisäänkäynnit asettuisivat tarkoituksenmukaisesti maankäyttöaineistolle. Jokivallin ja meluvallien mallintamiseen käytettiin kaarityökalua, jolla saatiin muodostettua vallimainen ulkoasu. Valmiiksi mallinnettu valli muutettiin komponentiksi ja monistettiin. Suora valli haluttiin muuttaa kaarevaksi, jotta se olisi sopinut hyvin mallinnukseen. SketchUp extension warehouse -lisäosapalvelusta löytyi työkalu, jolla komponentti voitiin sijoittaa kaarevaan muotoon. Työkalu ei kuitenkaan toiminut ohjelmistoversiossani toivotulla tavalla, jolloin vallit laadittiin useita suoria vallinpalasia yhdistelemällä. Tonttien raja-aidat muodostettiin piirtämällä suorakaiteen muotoinen alue, joka nostettiin 1,5 metrin korkeuteen. Tämän jälkeen muodostettu aita teksturoitiin, muodostettiin komponentiksi ja monistettiin. Aidat siirrettiin asemakaava-alueen tonttien ympärille. Kaikkille tutkimusalueen tonteille ei muodostettu aitoja ajallisten puitteiden takia. Tutkimusalueen puut, katulamput, silta, ihmiset, autot ja leikkipaikka tuotiin valmiina 3D-malleina SketchUp-ohjelman 3D-warehouse-palvelusta. Tuodut mallit monistettiin ja siirrettiin yksitellen omille paikoilleen maankäyttöaineiston päälle.

Valmiille mallille luotiin useita eri kuvakulmista otettuja näkymiä animaation luomiseksi. Näkymät luotiin animation valikon add scene -toiminnon kautta. Animaatio-toiminnolla pystyttiin katsomaan kuvakulmista muodostettu animaatio. Animaatio siirrettiin MP4-muotoon export-toiminnolla, jolloin se voitiin katsoa ilman SketchUp-ohjelmaa MP4-tiedostomuotoa lukevan soittimen avulla. MP4-muotoinen animaatio tallennettiin Googlen drive-palveluun, josta se voitiin jakaa halutuille henkilöille.

4.6 Ohjelmistojen välinen tiedonsiirto

CityEngine ja SketchUp tarjoavat aineistojen siirtoon useita aineistoformaatteja. Aineistoa siirrettäessä CityEngine-ohjelmasta SketchUp-ohjelmaan, käytettiin COLLADA-aineistoformaattia. Siirrettäessä aineistoa SketchUp-ohjelmasta Ci-

tyEngine-ohjelmaan käytettiin OBJ-aineistoformaattia (kuvio 49). Tässä työssä tehtiin mallinnetuille rakennukselle tiedonsiirto molempiin ohjelmistoihin.



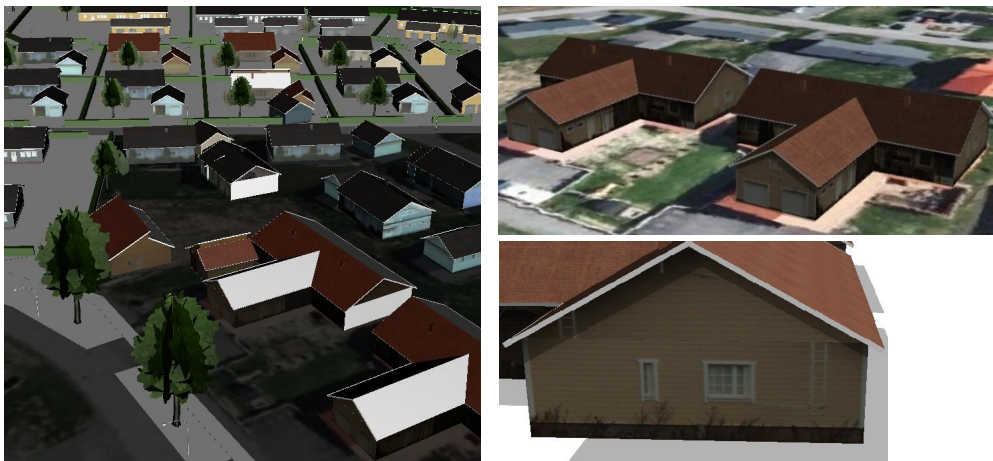
Kuvio 49. Tiedonsiirtoasetukset siirrettäessä aineistoa CityEngine-ohjelmasta SketchUp-ohjelmaan.

CityEnginestä siirretyt 3D-mallit tallennettiin SketchUp-malliin (kuvio 50). CityEngine-ohjelmassa tiedonsiirto tapahtui export models -toiminnon avulla ja SketchUp-ohjelmassa export-toiminnon avulla. 3D-mallit muutettiin CityEngine-ohjelmassa COLLADA-muotoon. COLLADA-tiedoston avaaminen tapahtui SketchUp-ohjelmassa import-toiminnon kautta. Jos tuotava malli piti sisällään tekstuureja, joiden nimissä oli käytetty ä- tai ö-kirjainta, jätti SketchUp-ohjelma tuomatta kyseiset tekstuurit. Tämä oli hyvä muistaa, jotta tekstuurien nimiä ei tarvinnut jälkikäteen vaihtaa. 3D-mallit siirtyivät täysin alkuperäisen näköisenä tiedostoformaattista toiseen ja näkyvät molemmissa ohjelmissa yhtenäisinä komponentteina.



Kuvio 50. Kuvassa on punaisella viivalla rajattu rakennukset, jotka on laadittu CityEngine-ohjelmassa ja siirretty SketchUp-ohjelmaan.

SketchUp-ohjelmasta tedyissä tiedonsiirroissa ilmeni useita ongelmia. Siirrettäessä 3D-malleja SketchUp-ohjelmasta CityEngine-ohjelmaan, puuttui osa rakennuksen tekstuureista. Ongelma saatiin korjattua asettamalla SketchUp-ohjelmassa tekstuurit uudelleen niille pinnoille, joista ne CityEngine-ohjelman puolella puuttuivat (kuvio 51). Tämänkin jälkeen osa tekstuureista oli kääntynyt peilikuvaksi. Tiedonsiirron jälkeen SketchUp-mallit tallennettiin CityEnginen mallinnukseen.



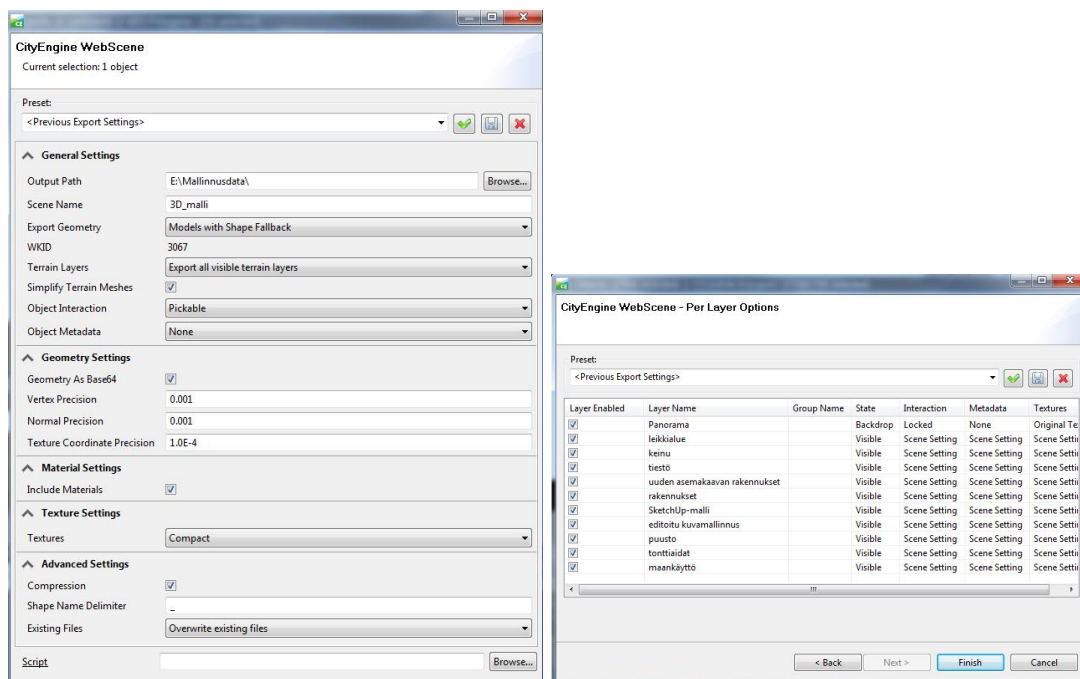
Kuvio 51. SketchUp-ohjelmasta obj-formaatissa CityEnginen siirretyt 3D-mallit.

4.7 3D-mallien julkaisu

CityEngine-ohjelmassa laadittu 3D-malli voidaan julkaista Esrin webscene-selainsovelluksessa (liite 13). Selainsovellus avautuu internet-selaimeen, jolloin

ihmiset voivat katsella ja kommentoida mallia omalta tietokoneeltaan internetin välityksellä. Aineisto voidaan jakaa halutuille henkilöille linkin välityksellä tai selainsovellus voidaan istuttaa internet-sivuille linkkinä. Ennen mallin siirtämistä selainsovellukseen lisätään eri kuvakulmista otettuja näkymiä bookmarks-toiminnolla. Tallennetut näkymät näkyvät webscnessä animaationa.

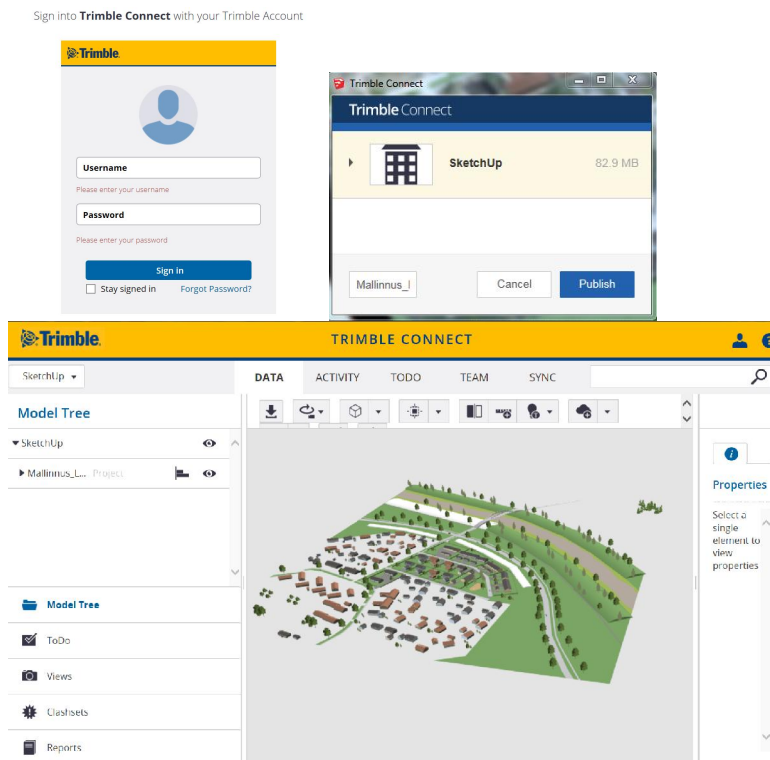
CityEnginessä valmis 3D-malli viedään selainsovellukseen export models -toiminnon avulla. Toiminnon asetuksista valitaan, mitä aineistoja halutaan viedä selainsovellukseen, millä nimellä ne halutaan esittää ja mitä attribuuttitietoja halutaan jakaa (kuvio 52). Tämän jälkeen webscene-julkaistaan share as -toiminnon avulla. Webscene tallentuu Esrin online-palveluun, jossa määritellään webscenen julkaiseminen. Valmis mallinnus (liite 13) on katsottavissa liitteessä 13 mainitussa verkko-osoitteessa. Mallinnus ei toimi Internet Explorer -selaimella. Google Chrome ja Mozilla Firefox -selaimilla mallinnust toimii hyvin.



Kuvio 52. 3D-mallin siirtäminen webscene-selainsovellukseen.

SketchUp-ohjelmassa valmis 3D-malli (liite 14) voidaan julkaista Trimble connect-palvelualustalle (kuvio 53). Palvelualusta vaatii kirjautumista, joten jaettu malli on jaettavissa vain connect-palveluun kirjautuneiden tiimihenkilöiden välille. Ohjelmisto ei tue suoraa CityEnginen kaltaista nettiselaimen kautta ta-

pahtuvaa julkaisua. Valmis 3D-malli animaatioineen voidaan julkaista linkkinä verkkosivuilla eri sovellustuottajien laatimien lisäohjelmien välityksellä. Animaation siirtäminen MP4-muotoon mahdollistaa kuitenkin esimerkiksi sähköpostin välityksellä tapahtuvan mallin jakamisen tai sen sijoittamisen linkkinä esimerkiksi kaupungin verkkosivuille.



Kuvio 53. Valmiin 3D-mallin julkaiseminen TrimbleConnect-palvelussa.

5 3D-MALLINNUKSIEN TARKASTELU

5.1 Ohjelmistojen vahvuudet kaupunkimallinnuksessa

CityEngine-ohjelmisto osoittautui erittäin monipuoliseksi mallinnusohjelmaksi mallinnusmahdollisuuksiensa sekä mallinnustyötä helpottavien näkymäikkunoiden (Viewport, inspector, scene, navigator) vuoksi. Ohjelmisto tarjosi myös hyviä neuvoja help-palvelun kautta ja lukuisia opetusvideoita. Ohjelmistolla saatiin luotua realistinen sekä visuaalisesti hyvä mallinnus, ja tietokone pyöritti tutkimusalueen aineistoa suhteellisen hyvin. Ohjelmistolla voitiin mallintaa ja editoida useita eri aineistotasoja helposti. Editoidessa yhtä tasoa (esimerkiksi siirtäessä talomalleja) eivät muut aineistot liikkuneet samanaikaisesti. Aineistotasot voitiin myös lukita käytöstä yksinkertaisesti. CityEnginestä voitiin siirtää aineistoja toiseen ohjelmistoon ja tuoda aineistoa useissa eri formaateissa. Ohjelmassa GIS-datan käsittely oli helppoa ja tietokantapohjainen mallintaminen oli mahdollista. Ohjelmiston ehdottomana etuna oli sen koordinaatistosidonnaisuus sekä korkeusmallipinnan ja aineistojen yhteensovittaminen. Koordinaatist ominaisuuksien ansiosta aineistot sijoittuivat aina oikeille paikoilleen ja näkyivät aina oikeassa suhteessaan. Korkeusmallipinta ilmakuvatekstuureineen oli helppo tuoda ohjelman näkymään. Myös muut aineistot oli helppo asettaa korkeusmallin pinnalle.

CityEnginen vahvuutena on myös sääntöpohjainen mallinnus, joka mahdollistaa laajojen alueiden, jopa koko kaupungin mallintamisen hetkessä. Sääntömallinnus myös mahdollistaa mallin muuttamisen helposti ja joustavasti. Kerran laadittuja sääntöjä voidaan hyödyntää myöhemmissä mallinuksissa. Ohjelmaan tuotu viivamainen verkosto muodostuu helposti tieverkostoksi ohjelman tarjoaman valmiin tiesäännön avulla.

CityEnginen kuvapohjainen mallinnus todettiin myös erittäin tehokkaaksi tavaksi mallintaa rakennuksia fotorealistisiksi. Julkisivukuvien käsittelymahdollisuus suoraan ohjelmassa helpottaa ja nopeuttaa kuvamallinnuksen laadintaa. Ohjelmisto tarjoaa myös kiinteiden mallien editoinnin, jolloin saatiin luotua rakennuk-

siin halutessa uniikkeja kattomuotoja. Ohjelmisto tarjoaa lukuisia valmiita tekstuureita ja 3D-malleja mallinnuskäyttöön. Valmiit 3D-mallit ja tekstuurit ovat osittain tarkan resoluution 3D-malleja. Ohjelman sisältäessä paljon tarkan resoluution 3D-malleja, ohjelman toiminta hidastui selvästi.

Ohjelman erittäin tärkeänä vahvuutena voidaan pitää mallinnuksen julkaisu-mahdollisuutta virtuaalisena 3D-mallina webscene-selainikkunassa. Verkkoselaimen kautta katsottava 3D-malli mahdollistaa mallinnuksen esittämisen kansalaisille, päättäjille ja suunnittelijoille. Etenkin kaavatyön havainnollistamisessa mallinnuksen verkkoselainpohjainen jakaminen on erittäin tärkeää. Webscene mahdollistaa myös mallin kommentoinnin. Mallinnus CityEnginellä on kaiken kaikkiaan erittäin kustannustehokasta, jos omaa kokemusta ja taitoa ohjelman käyttöön.

SketchUp-ohjelmistolla mallintamiseen käytettiin murto-osa CityEnginellä mallintamiseen käytetystä ajasta. Tämä johtui siitä, että työssä haluttiin keskittyä enemmän CityEnginen tarjoamien mallinnusmenetelmien opettelemiseen. SketchUp-ohjelmiston tärkeimpänä etuna on sen yksinkertaisuus, jonka ansiosta SketchUp-ohjelmisto sopii hyvin käyttöönotettavaksi kaupungin suunnitteluelimissä. Mallintaminen oli helppoa yksinkertaisten mallinnustyökalujen ansiosta. Mallin rakentamisessa auttoi erinomainen snapping-ominaisuus, jolloin ohjelma tarttui kiinni helposti pisteeseen. Erinomainen tarttumisominaisuus takasi myös hyvän mallinnustarkkuuden. Ohjelma ennakoi viivan piirtoa, jolloin se muisti edellisien piirrosten perusteella, mihin seuraava piste tulee (esimerkiksi vierekkäisten ikkunoiden laatiminen seinälle). Mallin pinnat muodostuivat usein myös ilman, että pinnan jokaista kulmaa tarvitsi piirtää. Tämä nopeutti huomattavasti mallinnusta. Ohjelman apuviivat olivat myös ehdoton etu muodostettaessa mitasuhteiltaan tarkkaa mallia. Julkisivukuvien käyttäminen mallinnuksessa teki malleista nopeasti fotorealistisia. Julkisivukuvaa ja muita tekstuureita voitiin vai-vatta monistaa muihin pintoihin pikanäppäimen (painamalla paintbuket-työkalua alt-näppäin pohjassa kopioitavaa pintaa). SketchUpilla saatiin ohjelman ohjautuvuuden ja helppokäyttöisyyden vuoksi rakennettua nopeasti yksinkertainen visualisointi. Ohjelmaan on lisäksi ladattavissa ilmakehä korkeusmallilla Google Earth-palvelusta.

SketchUp-ohjelmiston etuna on animaation katsominen suoraan sovelluksessa. Mallinnuksen animaatio voidaan myös tallentaa MP4-muotoon, jolloin se voidaan lähettää eteenpäin tai katsoa MP4-soittimella. Trimplen tarjoama Trimble connect -palvelu mahdollistaa mallinnuksen jakamisen palveluun kirjautuneiden välillä. Kokonaisuudessaan SketchUp on käyttäjäystävällinen ja soveltuu parhaiten nopeasti tarvittavien ja pienien mallien yksinkertaiseen visualisointiin.

5.2 Ohjelmistojen heikkoudet kaupunkimallinnuksessa

CityEngine-ohjelmiston heikkoutena on monimutkaisuus ja sääntökielen vaikeus. Ohjelman käyttö vaatii paljon taitoa, kokemusta ja harjoitusta. Sääntökielen oppiminen on erittäin työlästä vaikka ohjelma tarjoaakin valmiita sääntöjä ja ohjeita mallintajan käyttöön. Laadittujen sääntöjen hyödyntäminen myöhemmissä mallinnuksissa ilman sääntöihin tehtäviä muutoksia, vaatii standardimuotoisen tietomallin luomisen. Standardimuotoista tietomallia tai CityEnginen oman informaatiomallia käyttämällä, vältetään aineiston attribuuttitaulun tai säännön turhalta muokkaamiselta. Tässä työssä ei käytetty CityEnginen tarjoamaa tietomallipohjaa, sillä sen laatiminen vaati ArcGIS-ohjelman datainteroperability-työkalulisenssin.

Kuvamallinnuksessa käytettävien viivojen asettaminen vaati harjoittelua, jotta malli muodostui halutulla tavalla. Lisäksi kuvamallinnusta varten tuli luoda sääntötiedosto, johon kuvamallinnuksessa laaditut säännöt yhdistettiin. Sääntöön perustuva rakennusten katonmuodostus ei tapahtunut aina tarkoituksenmukaisesti. Rakennuksiin saattoi muodostua virheellinen kattomuoto huolimatta rakennukselle tehdystä cleanup shapes -puhdistustoimenpiteestä.

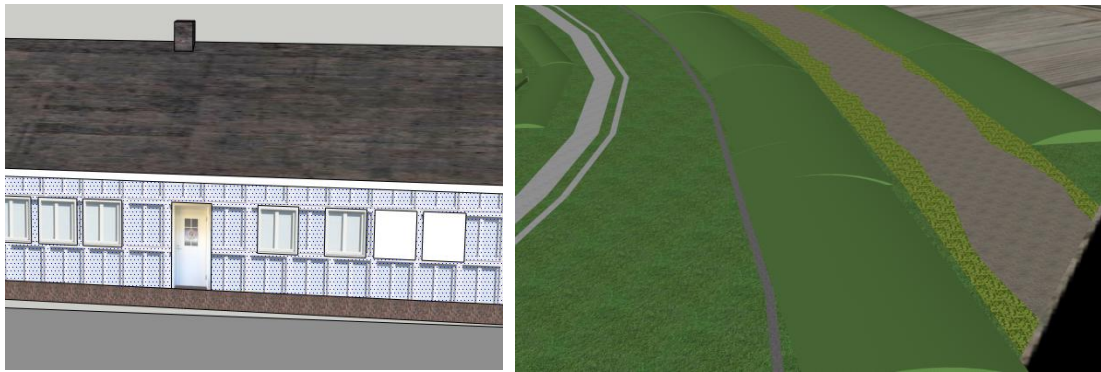
CityEnginen heikoimmat puolet liittyivät olennaisesti manuaaliseen mallinukseen. Piirtämällä tapahtuva mallinnus perustui käytännössä kahden työkalun varaan. Näillä työkaluilla piirrettiin, vedettiin, työnnettiin ja jaettiin pintoja. Snapping-toiminto oli CityEnginessä selvästi SketchUp-ohjelmaa heikompi. Ohjelma ei myöskään ennakoinut piirtoa eikä siinä ollut hyödynnettävissä apuviivoja, kuten SketchUpissa. Toisinaan pintojen muodostaminen oli monimutkaista ja työ-

lästä. Erityisesti pieniin pintoihin tarttuminen oli vaikeaa. Mallin pintojen ylimääräisiä viivoja oli myös vaikea poistaa ilman pinnan häviämistä. Toisinaan monesta osasta koostuva pinta tuli kokonaisuudessaan poistaa ja piirtää tilalle yhtenäinen pinta, jotta ylimääräiset viivat saatiin poistettua.

CityEnginen tiemallinnuksessa oli puutteita, jotka korostuvat etenkin risteysalueilla. Ohjelma ei hyväksynyt lähellä toisiaan risteäviä teitä. Tästä syystä kävelytiet jouduttiin jättämään irralleen virallisesta tieaineistosta. Tiemallinnuksen perusteella voitiin määrittää rakennuksien tienpuoleinen sijainti. Tämä oli olennaista, jotta rakennuksille luodut säännöt muodostivat rakennukset tarkoituksenmukaisesti ja oikein. Jos tienpuoleista sivua ei rakennukselle määritetty, saattoi rakennuksen julkisivu muodostua puutteellisesti (esimerkiksi osa julkisivua saattoi puuttua). Tienpuoleisen sijainnin määrittäminen toimii automaattisesti koko mallinnetulle rakennusaineistolle vain, jos rakennukset on mallinnettu tieaineiston luomisen yhteydessä muodostuneista tonteista. Tässä työssä rakennukset mallinnettiin erillisestä rakennusaineistosta, jolloin tienpuoleisen seinän määrittäminen jouduttiin tekemään manuaalisesti. Tämä tapahtui valitsemalla rakennukset ja rakennuksien edessä oleva tienpätkä sekä käyttämällä compute street edges -työkalua. Tämä menetelmä vei aikaa ja on mahdoton toteuttaa esimerkiksi koko kaupungin käsittävässä mallinnuksessa. Jos seinien tienpuolista määrittelyä ei tehty, muodostuivat rakennuksen seinät virheellisesti. Valmista mallinnusta ei voitu CityEnginessä suoraan katsoa animaationa, vaan mallinnus näkymineen tuli siirtää websceneen. Ohjelmisto myös vaatii kapasiteettia ohjelmaa pyörittävältä tietokoneelta. Käyttämällä paljon valmiita 3D-malleja, hidastuu ohjelman käsittely merkittävästi suhteellisen tehokkaallakin tietokoneella käsiteltynä. Tarkkojen 3D-mallien käyttäminen tutkimusalueen kokoisella alueella hidasti merkittävästi myös webscene-animaatiota.

SketchUp-ohjelmiston heikkoutena on sen soveltumattomuus laajempien alueiden mallintamiseen. Mallinnus tehdään täysin manuaalisesti, joten laajojen alueiden mallinnus on erittäin työlästä ja kustannustehotonta. Laajoja malleja käsitellessä ohjelma myös toimii hitaasti. Mallinnuksen muokkauksessa oli myös merkittäviä ongelmia. Laadittujen mallien siirtämien komponenttittomassa muodossa johti muiden aineistojen muuttumiseen siitäkin huolimatta, että muut ai-

neistotasot olivat kytkettyinä pois näkyvistä. Malleja saattoi helposti laatia myös väärälle aineistotasolle, jos ei huomionnut tarkasti aktiivisena olevaa tasoa. Vaikka julkisivukuvien käyttäminen oli SketchUpissa yksinkertaista, ei ohjelmisto sisällä kuvien editointityökalua. Tästä syystä kuvien esikäsittely tuli tehdä muussa ohjelmassa, joka hidasti mallinnustyötä. Kuvien siirtämisessä teksturoitavalle pinnalle ilmeni myös ongelmia. Tekstuuria skaalatessa halutulle ikkunapinnalle, siirtyi tekstuuri usein seinäpinnalle (kuvio 54). SketchUp-ohjelmassa on haasteellista luoda pitkiä kaarevia kolmiulotteisia muotoja. Kaarevat muodot jouduttiin muodostamaan suorista malleista, jolloin mallien jatkokset jäivät helposti näkyviin (kuvio 54).



Kuvio 54. SketchUp-mallinnuksessa havaittuja ongelmia.

SketchUp-ohjelma ei ole koordinaatistosidonnainen, joka on yksi ohjelman heikkouksista. Ohjelmaan tuodut aineistot jouduttiin skaalaamaan oikeille paikoilleen. Koordinaatissa sijaitsevia aineistoja ei aina pysty skaalaamaan Google Earth -palvelusta tuodun korkeusmallin pinnalle. Yksittäisten talojen sijoittaminen korkeusmallille onnistuu, mutta suurempien georeferoitujen aineistojen sijoittaminen on mahdotonta. Google Earth -palvelusta ladattava korkeusmalli oli tarkkuudeltaan myös erittäin huono. Tutkimusalue kuvautui korkeusmallissa täysin tasaiseksi alueeksi. Edes tutkimusalueen koillislaidassa sijainnut metsäinen kohouma ei kuvautunut korkeusmallissa. Työssä käytettyä maanmittauslaitoksen korkeusmallipintaa ei voitu tuoda ohjelmistoon, sillä ohjelmisto tuki vain muutamaa korkeusmalliformaattia. SketchUp-ohjelman heikkoutena voidaan pitää myös sen huonoa julkaisumahdollisuutta. Trimble connect -palvelun kautta ei voida jakaa mallinnusta julkisesti nähtäville ja kommentoitavaksi. Animaation

tallentaminen MP4-muotoon ja upottaminen esimerkiksi kaupungin verkkosivuille onnistuu, mutta vaatii lisätyötä ja osaamista.

5.3 Ohjelmistojen sopivuus Porin kaupungin suunnittelutarpeisiin

Tässä työssä laadittujen mallinnuksien ja ohjelmistokokemusten perusteella molemmille ohjelmille on tarvetta ja omat käyttökohteensa Porin kaupungin suunnittelussa. SketchUp-ohjelmiston käyttäjäystävällisyys ja yksinkertaisuus mahdollistavat 3D-mallintamisen käyttöönoton yleisessä suunnittelutyössä ilman laajempaa koulutusta. CityEngine sen sijaan on ohjelmana erittäin monimutkainen ja vaatii kokemusta ja käyttötaitoa sekä CityEnginen että ArcGIS-ohjelmiston käytöstä. Vaikka CityEngine mahdollistaa laajojen alueiden mallintamisen nopeasti sekä visuaalisesti laadukkaasti, sen vaikeakäyttöisyys on esteenä ohjelman laajempaan käyttöönottoon suunnittelussa.

CityEnginellä laadittuja mallinnuksia voidaan hyödyntää kaupunkisuunnittelun isompien kaavahankkeiden suunnittelussa, jossa visuaalisella ilmeellä, mallin muunneltavuudella ja selaimessa näkyvällä virtuaalimallilla on merkitystä. Opin näytetyössä laadittu mallinnus tullaan hyödyntämään vireillä olevan kaavatyön ehdotusvaiheessa. Myös tiesuunnitteluun CityEngine soveltuu visuaalisuutensa ja tierakenteiden helpon mallinnettavuuden vuoksi. Tiesääntö tuo kaikki tarvittavat kohteet teille: katuvalaisimet, liikennevalot, suojatiet, liikenneympyrät, tie-merkinnät, autot, kävelijät, puut, sillat. Tiesuunnittelussa ainoana ongelmana on lähekkäisten risteyksien muodostamisen vaikeus. Puistosuunnitteluun CityEngine soveltuu myös hyvin, sillä CityEnginessä on saatavana valmiina lukuisia tarkkoja 3D-malleja puista ja pensaista. Laajempien puistoalueiden mallintamiseen voidaan käyttää sääntöä, jossa säännön perusteella muodostuu tietylle alueelle satunnaisesti tietyt puut. Yksityiskohtaisessa puistosuunnittelussa manuaalinen mallinnus tulee kysymykseen. Tarkkuutta ja yksityiskohtaisuutta vaativaan rakennussuunnitteluun CityEngine ei sovellu.

Kaupunkisuunnittelussa SketchUp voi toimia kaavoittajien suunnittelun tukena. SketchUp-mallit ovat hyödyllisiä pienien kaavojen havainnollistamiseen, joita ei

tarvitse julkaista virtuaalisena verkkoselaimessa. Myös muiden pienempien suunnittelukohteiden suunnitteluun SketchUp soveltuu hyvin. SketchUp-ohjelman käyttömahdollisuudet tulevat laajenemaan merkittävästi, kun Porin kaupungissa suunnittelukäytössä olevasta Trimplan Locus-palvelusta voidaan kirjoittaa aineistoa suoraan SketchUp-malliin.

6 JOHTOPÄÄTÖKSET

Opinnäytetyön tavoitteena oli tutkia 3D-mallinnuksien avulla, miten CityEngine- ja SketchUp-ohjelmat soveltuvat kolmiulotteiseen mallintamiseen Porin kaupungin suunnittelutyössä. Tavoitteena oli myös tutkia ohjelmien tarjoamia mallinnusmenetelmiä, ohjelmistojen välistä tiedonsiirtoa sekä ohjelmien vahvuuksia ja heikkouksia. Lisäksi selvitettiin, miten mallinnuksien julkaisu verkkoselaimessa onnistuu.

Opinnäytetyön mallinnusprosessi oli erittäin työläs ja vaati lukuisten ongelmien ratkaisua lopulliseen mallinnustulokseen pääsemiseksi. Molemmilla ohjelmilla suoritettu mallinnusprosessi antoi arvokasta tietoa CityEngine- ja SketchUp-ohjelmien ominaisuuksista, mahdollisuuksista ja käytön vaatimista resursseista. Työn aikana ilmeni, että CityEngine vaatii erittäin paljon resursseja mallinnusprosessien oppimiseen, mutta johtaa visuaalisesti hyvään lopputulokseen. SketchUp eroaa täysin CityEnginestä vaativuudeltaan ja sen käyttöönoton kynystä suunnitelmien mallintaisessa voidaan pitää erittäin matalana.

CityEngine-ohjelman tärkeimpinä vahvuuksina voidaan pitää seuraavia asioita: isoja alueita voidaan mallintaa sääntömallinnuksella hetkessä, mallien muuttaminen on helppoa, mallinnustulos on visuaalisesti tarkka, ohjelmassa on hyvä GIS-datan yhteiskäytön mahdollisuus sekä mallinnuksen julkaisumahdollisuus verkkoselaimessa. Ohjelman merkittävimpiä heikkouksina voidaan pitää seuraavia asioita: ohjelman monimutkaisuus vaatii kokemusta ja käyttötaitoa, manuaalinen mallinnustoiminto ei toimi sujuvasti, tiemallinnus ei hyväksy hyvin lähellä toisiaan sijaitsevia risteyksiä sekä talojen kadunpuoleinen sivu täytyy osoittaa manuaalisesti jokaiselle erillisestä pohja-aineistosta muodostetulle talolle.

SketchUp-ohjelman tärkeimpinä vahvuuksina voidaan pitää seuraavia asioita: ohjelma on erittäin helppo ja nopea oppia, ohjelmassa on hyvä snapping-ominaisuus ja ennakoiva piirto sekä animaation tallennusmahdollisuus MP4-muotoon. Ohjelman merkittävimpiä heikkouksina voidaan pitää seuraavia asioita: ohjelma ei sovellu laajojen alueiden mallintamiseen, mallien muuttaminen

erittäin työlästä, ohjelma ei ole koordinaatistosidonnainen, Maanmittauslaitoksen korkeusmallia ei voida tuoda ohjelmaan, Google Earth -palvelusta ladattava korkeusmalli on erittäin epätarkka sekä verkkoselainpohjaisen julkaisumahdollisuus puuttuu.

Opinnäytetyössä laadittujen mallinnusten perustella voidaan antaa muutamia suosituksia Porin kaupungin suunnittelussa hyödynnettävän 3D-mallinnustyön organisoinniseksi. Porin kaupungissa on käytössä kaksi 3D-mallinnusohjelmaa (SketchUp ja CityEngine), joiden potentiaali suunnittelutyön visuaalisessa esittämisessä ja vuorovaikutusprosessissa tulee hyödyntää. SketchUp-ohjelmisto tulee ottaa käyttöön suunnittelijoiden keskuudessa nykyistä laajemmin, sillä ohjelmalla voidaan helposti luoda yksinkertaisia visualisointeja suunnittelun tueksi. SketchUp-ohjelma on hyvä suunnitteluprosessin hahmottelutyöhön tai pienialaisten mallien laatimiseen. CityEngine-ohjelmistolla voidaan luoda nopeasti näyttäviä ja tarkkoja visualisointeja laajoista alueista. Sääntöpohjainen mallinnus ja valmiit säännöt mahdollistavat esimerkiksi laajempien liikennesuunnitelmien, puistosuunnitelmien sekä kaavasunnitelmien suhteellisen nopean laatimisen. Laaditut mallinnukset animaatioineen voidaan myös helposti siirtää verkkoselaimelle kaikkien nähtäväksi ja kommentoitavaksi. Tämä on ensiarvoisen tärkeää etenkin kaupunkisuunnittelun vuorovaikutusprosessin kannalta. Verkkoselaimesta katsottava kolmiulotteinen mallinnus hyödyttää sekä suunnittelijoita, päättäjiä että osallisia. Verkkoselaimessa pyörivästä animaatiosta voidaan tarkastella esimerkiksi suunnitelmien dynaamisempia osa-alueita osana suunnitelman kokonaiskuvaa. CityEngine-ohjelman monimutkaisuus rajaa kuitenkin ohjelmiston laajempaa käyttöönottoa suunnittelijoiden keskuudessa.

LÄHTEET

Alam, N., Coors, V., Zlatanova, S. & Oosterom, P. V. 2011. Shadow Effect on Photovoltaic Potentiality Analysis using 3D City Models. In Proceedings of the Joint ISPRS Workshop on 3D City Modelling & Applications and the 6th 3D GeoInfo Conference, 26–28.

Biljecki, F., Ledoux, H., Stoter, J. & Zhao, J. 2014. Formalisation of Level of Detail in 3D City modelling. *Computers, Environment and Urban Systems* 48, 1–15.

BLOM 2013. Fotorealistic kaupunkimalleja Blomilta. Newsletter. Viitattu 4.9.2015

http://newsletter.blomasa.com/newsletter/2013/december/fi/december_fi_1.htm

Buhur, S., Ross, L., Büyüksalih, G. & Baz, I. 2009. 3D City modelling for planning activities, Case Study: Haydarpasa Train Station, Haydarpasa Port and Surrounding Backside Zones, Istanbul. *ISPRS proceedings XXXVIII-1-4-7 W5*.

Döllner, J., Kolbe, T. H., Liecke, F., Sgouros, T. & Teichmann, K. 2006a. The Virtual 3D City Model of Berlin – Managing, integrating and communicating complex Urban Information. In *Proceedings of the 25th Urban Data Management Symposium UDMS, Vol. 2006*.

Döllner, J., Baumann, K., & Buchholz, H. (2006b). Virtual 3D City Models as Foundation of complex Urban Information Spaces. 11 th International Conference on Urban Planning and Spatial Development for the Information Society. 107–112.

Erving, A. 2006. Paikkatiedoista kaupunkimalleihin CityGML selvitystyö. *Fotogrammetrian ja kaukokartoituksen laboratorio, Teknillinen korkeakoulu* 41.

Energy Atlas Berlin 2015. Viitattu 4.9.2015

<http://energyatlas.energie.tu-berlin.de/energy-atlas/?lang=en>.

Esri 2010. Virtual city template enables 3D city modeling. *ArcNews Online*. Viitattu 8.9.2015

<http://www.esri.com/news/arcnews/summer10articles/virtual-city.html>.

Esri 2012. CityEngine kaupunkien 3D-mallinnukseen. Viitattu 1.9.2015

http://www.esri.fi/midcom-serveattachmentguid-1e15e0777cef5b25e0711e197841f5dd425307a307a/cityengineflier_fin_lowres.pdf.

Esri 2013. Example modern streets. Viitattu 1.6.2015

<https://www.arcgis.com/home/item.html?id=6e45334d75f5423f9e44436cd0f53183>.

Esri 2015a. CityEngine. Viitattu 1.9.2015

<https://www.esri.com/~media/Files/Pdfs/library/brochures/pdfs/esri-cityengine.pdf>.

Esri 2015b. Esri 3D City: Creation part 2. Viitattu 1.10.2015

<http://www.arcgis.com/home/item.html?id=b7142081948b4cd9ac1c852d8790c9c9>.

Esri 2015c. Transform 2D GIS data into smart 3D City Models. Viitattu 27.8.2015

<http://www.esri.com/software/cityengine>.

Finnish Geospatial Research Institute 2015. Korkeusmallit. Viitattu 4.9.2015.

<http://www.fgi.fi/fgi/fi/teemat/korkeusmallit>

GeoPlanIT 2014. 3D Flood mapping of London using CityEngine. Viitattu 8.9.2015.

<http://www.geoplanit.co.uk/?p=1433>.

Gröger, G. & Plümer, L. 2012. CityGML-interoperable semantic 3D City Models. ISPRS Journal of Photogrammetry and Remote Sensing 71, 12–33 .

Gröger, Kolbe, Nagel & Hägele 2012. OGC City Geography Markup Language (CityGML) encoding Standard. Open Geospatial Consortium.

Kolbe, T. H. 2009. Representing and exchanging 3D City Models with CityGML. in 3D Geo-Information Sciences. Springer Berlin Heidelberg, 15–39.

Krüger, T. & Kolbe, T. H. 2012. Building Analysis for Urban Energy planning using Key Indicators on Virtual 3D City Models - the Energy Atlas of Berlin. In Proceedings of XXII ISPRS Congress, Melbourne, Australia, Vol. 25.

Kuntalaki 17.3.1995/365.

Leppäniemi, O. 2015. SketchUp. Email kirsi-maria.viljanen@pori.fi 8.12.2015. Tulostettu 8.12.2015. 2015.

Maanmittauslaitos 2015. Tuotekuvaukset. Viitattu 4.9.2015.

<http://www.maanmittauslaitos.fi/digituotteet>.

Maankäyttö- ja rakennusasetus 10.9.1999/895.

Maankäyttö- ja rakennuslaki 5.2.1999/132.

Maankäyttö- ja rakennuslaki 2000. Opas 3. Asemakaavan selostus.

Mikkelin kaupunki 2015. 3D-malli Mikkelin keskustasta. Viitattu 20.10.2015

<http://www.mikkeli.fi/palvelut/3d-malli-mikkelin-keskustasta>.

Moser, J., Albrecht, F. & Kosar, B. 2010. Beyond Visualisation – 3D Gis analyses for Virtual City Models. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XXXVIII-4/W15.

Porin kaupunki 2007. Kantakaupungin yleiskaava 2025.

Porin kaupunki 2013. Lotskerin pientaloalueen laajennus. Osallistumis- ja arviointisuunnitelma.

Porin kaupunki 2015a. Asemakaavaehdotusluonnos.

Porin kaupunki 2015b. Asemakaavaehdotusluonnoksen havainnekuva.

Porin kaupunki 2015c. Havainnekuvia 3Dee. Viitattu 22.1.2016
<https://www.pori.fi/kaupunkisuunnittelu/asebakaavat/1632.html>.

Porin kaupunki 2015d. Kaupunkimittaus, lupa nro 446.

Savisalo, A. 2015. KM3D-hanke:Kohti 3D-kaupunkimallia. Paikkatietoseminaari 10.2.2015.

Satakuntaliitto 2013. Satakunnan maakuntakaava. Viitattu 1.10.2015
http://www.satakuntaliitto.fi/sites/satakuntaliitto.fi/files/tiedostot/Alueidenkaytto/MK_13_3_2013_Lehti_B.pdf.

SketchUp 2015a. Architecture. Viitattu. 27.8.2015
<http://www.sketchup.com/3Dfor/architecture>.

SketchUp 2015b. Landscape Architecture. Pro. Viitattu. 27.8.2015
<http://www.sketchup.com/3Dfor/landscape-architecture>.

SketchUp 2016. SketchUp Pro. Viitattu. 27.8.2015
<http://www.sketchup.com/products/sketchup-pro>.

Suomen Kuntaliitto 2015. Kolmiulotteinen kaupunkimalli (KM3D) -hanke.Viitattu 15.9.2015
<http://www.kunnat.net/fi/asiantuntijapalvelut/mal/verkko-opaat/paikkatiedon-opas/hankkeet/kaupunkimalli-3D/Sivut/default.aspx>.

Suomen Kuntaliitto 2014. Kaupunkimallikysely. Viitattu 2.9.2015
https://asiakas.kotisivukone.com/files/buildingsmart.kotisivukone.com/tiedostot/bSF_kaupsu/kolmiulotteisetkaupunkimallitkuntienvastaukset.pdf

Suomisto, J. 2013. Kaupunkien tietomallit Euroopassa – standardit – teknologia – sovellukset. Fiksu Kalasatama -seminaari, Helsinki 12.12.2013. Viitattu 10.9.2015
http://www.forumvirium.fi/sites/default/files/fiksu_kalasatama_12_12_2013.pdf.

Suomisto, J. 2014. 3D-tietomallit yleistyvät kaupungeissa. Positio 2/14,10–12. Viitattu 10.0.2015

https://www.paikkatietoikkuna.fi/c/document_library/get_file?uuid=2a55c72f-a033-4409-83ee-fc8c4ae7437d&groupId=108478.

WSP Finland Oy 2015. Kemintien bulevardisointi. City Engine Web Scene 28.8.2015

<https://www.arcgis.com/home/item.html?id=af5f1e610cc34277b85bf93716a0c505>.

Yalcin, G & O. Selcuk 2015. 3D City modelling with Oblique Photogrammetry Method. Procedia Technology 19, 424–431.

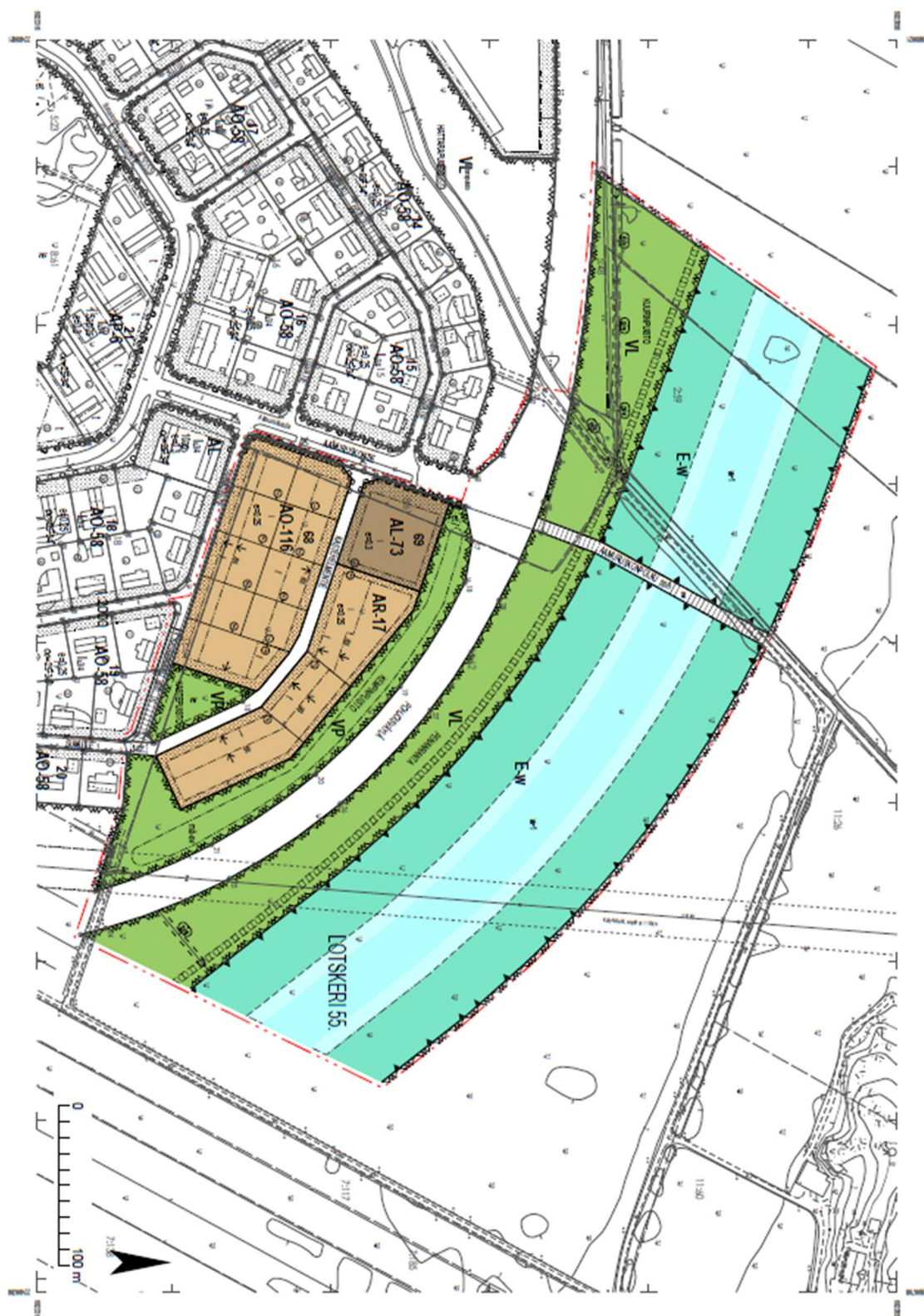
Ympäristöministeriö 2003. Asemakaavamerkinnot ja -määräykset.

Ympäristöministeriö 2007. Osallistuminen yleis- ja asemakaavoituksessa. Ympäristöhallinnon ohjeita I/2007. Edita Prima Oy, Helsinki 2007.

LIITTEET

- Liite 1. Asemakaavaehdotusluonnos
- Liite 2. Asemakaavanselostusluonnoksen havainnekuva
- Liite 3. Työnkulku
- Liite 4. Uusien omakotitalojen mallinnusta varten laadittu sääntö
- Liite 5. Uusien paritalojen mallinnusta varten laadittu sääntö
- Liite 6. Uusien rivitalojen mallinnusta varten laadittu sääntö
- Liite 7. Uusien omakotitalojen autotallien mallinnusta varten laadittu sääntö
- Liite 8. Uusien rivi- ja paritalojen autotallien mallinnusta varten muodostettu sääntö
- Liite 9. Maankäytön mallintamista varten laadittu sääntö
- Liite 10. Tonttien pensasaitojen mallintamista varten laadittu sääntö
- Liite 11. Esimerkki yhdestä kuvamallinnuksella laaditun rakennuksen säännöstä
- Liite 12. Rakennuksien säännöt yhdistävä sääntötiedosto
- Liite 13. CityEngine-ohjelmalla laadittu mallinnus
- Liite 14. SketchUp-ohjelmalla laadittu mallinnus

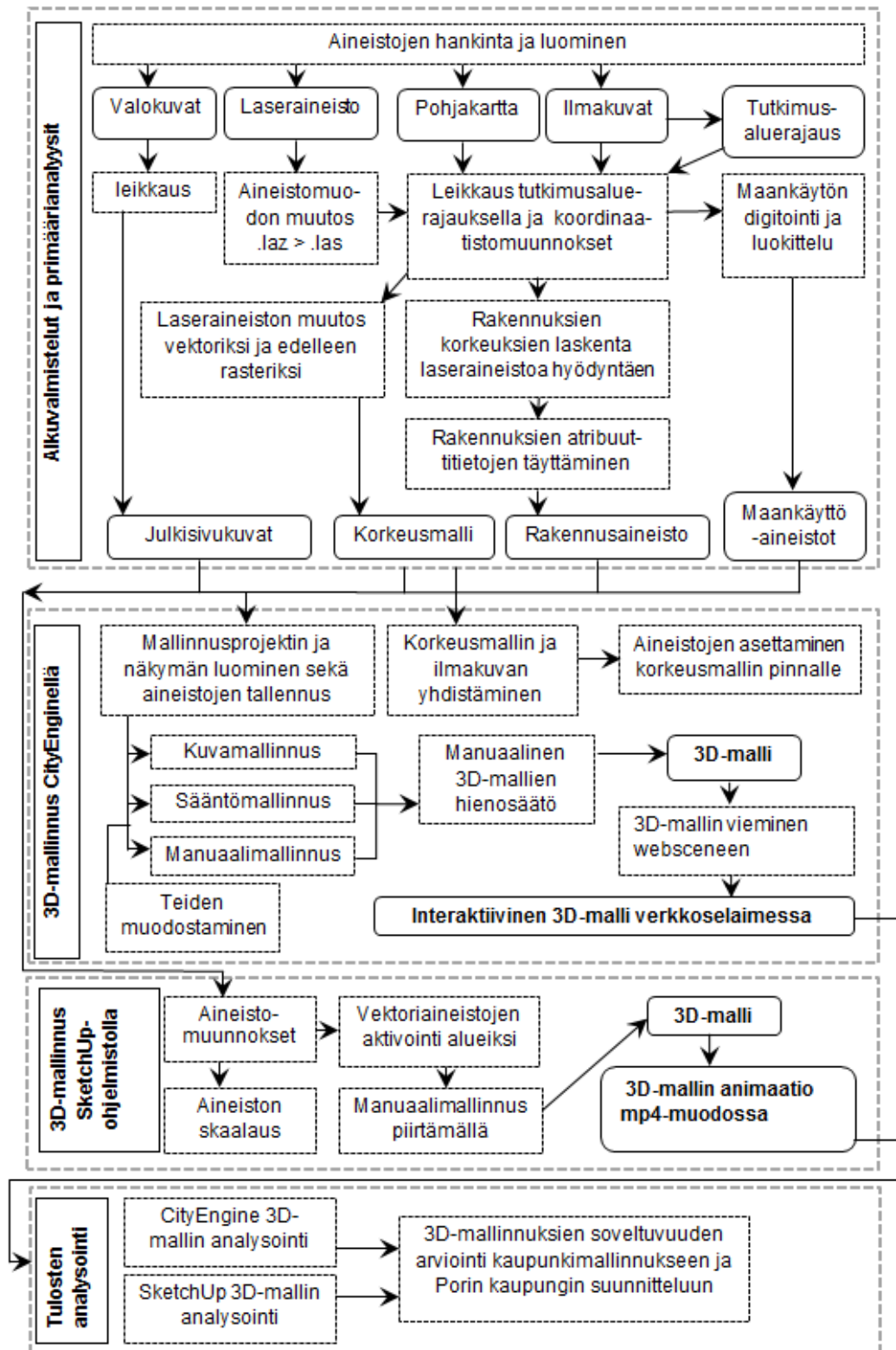
Liite 1 1(2) Asemakaavaehdotusluonnos



[illegible]

Lähde: Porin kaupunki 2015b

Liite 3 Työnkulku



Liite 4 1(4) Uusien omakotitalojen mallinnusta varten laadittu sääntö

```
/**
 * File:      uudet omakotitalot.cga
 * Created:   18 May 2015 11:40:56 GMT
 * Author:    KANVK4
 */
version "2015.0"

@Group ("julkisivu",1)
attr Rakennuksien_korkeudet = 0
attr ikkunavali_leveys      = 2
attr oven_leveys            = 5
attr sokkelin_korkeus       = 0.5
attr seinan_korkeus_sokkelista = 5
attr ikkuna_leveys          = 1.2
attr ikkuna_korkeus         = 1.4
attr oven_korkeus           = 2.1
attr ikkunan_vari = fileRandom ("assets/ikkunat/ikkuna_*.jpg")
@Order(1) @Range("tumman_harmaa","vaalean_harmaa","valkoinen"
,"tumman_keltainen","kirkkaan_keltainen")
attr seinan_varitys = "tumman_harmaa"

@Group("Katon ominaisuuudet",2)
attr savupiippu_DimXZ = 0.5
attr savupiippu_DimY  = 0.7
attr savupiippujen_etaisyys = 6.5
@Order(3) @Range(20,25)
attr kattokaltevuus = 20
@Order(2) @Range("punainen","harmaa")
attr katon_varitys = "punainen"
@Order(4) @Range("punainen","harmaa")
attr savupiipun_varitys = "punainen"
@Order(1) @Range("harjakatto","aumakatto","pulpettikatto")
attr kattotyyppi = "harjakatto"

@StartRule
//Nostetaan rakennukset keksikorkeuteen
Lot -->
    extrude (Rakennuksien_korkeudet) rakennukset

// nimetään rakennuksen seinät
rakennukset-->
comp(f){ front : etuseina | back: takaseina | side : sivut | top: kat-
to}

//tehdään harjakatto ja lisätään kattoväri ja savupiippu
katto -->
case kattotyyppi == "harjakatto":
    roofGable(kattokaltevuus,0.5, 0.5)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}
    split(y) { ~5 : NIL | ~0.1 : savupiipun_alue }

case kattotyyppi == "aumakatto":
    roofHip(kattokaltevuus, 0.5)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}
```

Liite 4 2(4)

```
split(y) { ~5 : NIL | ~0.1 : savupiipun_alue }

case kattotyyppi == "pulpettikatto":
    roofShed(kattokaltevuus,2)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}
    split(y) { ~5 : NIL | 0.7 : savupiipun_alue }
else: X.

kattotekstuuri-->
case katon_varitys == "punainen":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("katto/katto_1.jpg")
    tileUV(0,7, 10)
    comp(f) {side: katonreuna|bottom:katonreuna|top:X.}

case katon_varitys == "harmaa":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("katto/katto_2.jpg")
    tileUV(0,~20, ~10)
    comp(f) {side: katonreuna|bottom:katonreuna|top:X.}
else: X.

katonreuna-->
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_6.jpg")

savupiipun_alue -->

split(x) {{~savupiippujen_etaisyys : NIL | 0.1 : savupiip-
pu}*
| ~savupiippujen_etaisyys : NIL}

savupiippu -->
case savupiipun_varitys== "punainen":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    texture("katto/katto_1.jpg")
    tileUV(0,7, 10)
    s(savupiippu_DimXZ, savupiippu_DimY, savupiippu_DimXZ)
    center(xz)
    i("builtin:cube")
case savupiipun_varitys== "harmaa":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    texture("katto/katto_2.jpg")
    tileUV(0,~10, 20)
    s(savupiippu_DimXZ, savupiippu_DimY, savupiippu_DimXZ)
    center(xz)
    i("builtin:cube")
else: X.
```


Liite 4 3(4)

```
//jaetaan seinät x ja y-suuntaisiin osiin
etuseina-->
    split (y) {sokkelin_korkeus:sokkeli | sei-
        nan_korkeus_sokkelista: etuosan_seina}
etuosan_seina-->
    split(x){~0.5: seinä |
        {~ikkunavali_leveys : vali}* |
            oven_leveys: ovialue |
        {~ikkunavali_leveys : vali}* |
        ~0.5: seinä }
takaseina-->
    split (y) {sokkelin_korkeus:sokkeli | sei-
        nan_korkeus_sokkelista: takaosan_seina}
takaosan_seina-->
    split(x){~1: seinä |
        {~ikkunavali_leveys : vali}* |
            oven_leveys: ovialue |
        {~ikkunavali_leveys : vali}* |
        ~1: seinä }
sivut-->
    split (y) {sokkelin_korkeus:sokkeli | sei-
        nan_korkeus_sokkelista: sivuosan_seina }
sivuosan_seina-->
    split (x) {~1: seinä |
        {~ikkunavali_leveys : vali}* |
        ~1.: seinä}

//tehdään ikkuna-aukot
vali-->
    split (x) {~1.5: seinä | ikkuna_leveys:ikkuna_aukko
        |~1:seinä}*
ikkuna_aukko-->
    split (y){0.7:seinä | ikkuna_korkeus:ikkuna | ~1:seinä }*
ikkuna-->
    setupProjection(0,scope.xy,scope.sx,scope.sy) projectUV(0)
    texture(ikkunan_vari)
//tehdään ovi ja tekstuuri oveen
ovialue-->
    split (x) {~0.2:seinä
        |1.2:split (y) {oven_korkeus :ovi |~0.2:seinä }*|
        ~0.2: seinä}
ovi-->
    setupProjection(0,scope.xy,scope.sx,scope.sy) projectUV(0)
    texture("ovet/valk_ovi.jpg")
seinä-->
    case seinan_varitys == "tumman_harmaa":
        setupProjection(0,scope.xz,scope.sx,scope.sz)
        projectUV(0)
        texture("buildings/seinat/vari_8.jpg")
        tileUV(0,4,3)

    case seinan_varitys== "vaalean_harmaa":
        setupProjection(0,scope.xz,scope.sx,scope.sz)
        projectUV(0)
        texture("buildings/seinat/vari_3.jpg")
        tileUV(0,5,0.5)
    case seinan_varitys== "tumman_keltainen":
```

Liite 4 4(4)

```
        setupProjection(0,scope.xz,scope.sx,scope.sz)
        projectUV(0)
        texture("buildings/seinat/vari_5.jpg")
        tileUV(0,2,2.26)
case seinan_varitys == "kirkkaan_keltainen":
        setupProjection(0,scope.xz,scope.sx,scope.sz)
        projectUV(0)
        texture("buildings/seinat/vari_4.jpg")
        tileUV(0,5,0.7)
case seinan_varitys == "valkoinen":
        setupProjection(0,scope.xz,scope.sx,scope.sz)
        projectUV(0)
        texture("buildings/seinat/vari_7.jpg")
        tileUV(0,1.4,1.405)
else: x.

sokkeli-->
        setupProjection(0,scope.xy,~0.8,~0.3)
        projectUV(0)
        texture("sokkeli/vari_4.jpg")
```

Liite 5 1(4) Uusien paritalojen mallinnusta varten laadittu sääntö

```
/**
 * File:      uudet_paritalot.cga
 * Created:   29 Sep 2015 16:34:55 GMT
 * Author:    KANVK4
 */

version "2015.0"
//////////ATRIBUUTIT//////////
@Group ("julkisivu",1)
attr Rakennuksien_korkeudet = 0
attr ikkunavali_leveys      = 2
attr oven_leveys            = 3
attr sokkelin_korkeus       = 0.5
attr seinan_korkeus_sokkelista = 5
attr ikkuna_leveys          = 1.2
attr ikkuna_korkeus         = 1.4
attr oven_korkeus           = 2.1
attr ikkunan_vari = fileRandom ("assets/ikkunat/ikkuna_*.jpg")
@Order(1) @Range("tumman_harmaa","vaalean_harmaa","valkoinen"
,"tumman_keltainen","kirkkaan_keltainen")
attr seinan_varitys = "tumman_harmaa"

@Group("Katon ominaisuudet",2)
attr savupiippu_DimXZ      = 0.5
attr savupiippu_DimY       = 0.7
attr savupiippujen_etaisyys = 6.5
@Order(3) @Range(20,25)
attr kattokaltevuus        = 20
@Order(2) @Range("punainen","harmaa")
attr katon_varitys          = "punainen"
@Order(4) @Range("punainen","harmaa")
attr savupiipun_varitys     = "punainen"
@Order(1) @Range("harjakatto","aumakatto","pulpettikatto")
attr kattotyyppi            = "harjakatto"

//////////
@StartRule
//Nostetaan rakennukset rakennusaineistossa kerrottuun korkeuteen
Lot -->
    extrude (Rakennuksien_korkeudet) rakennukset
// jaetaan seinät osiin ja nimetään ne
rakennukset-->
comp(f){ street.front : etuseina | street.back: takaseina
        | street.side : sivut | top: katto }
//tehdään katto ja lisätään kattoväri ja savupiippu
katto -->
case kattotyyppi == "harjakatto":
    roofGable(kattokaltevuus,0.5, 0.5)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}
    split(y) { ~5 : NIL |~0.1 : savupiipun_alue }
case kattotyyppi == "aumakatto":
    roofHip(kattokaltevuus, 0.5)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}
    split(y) { ~5 : NIL |~0.1 : savupiipun_alue }
```


Liite 5 2(4)

```
case kattotyyppi == "pulpettikatto":
    roofShed(kattokaltevuus,2)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}
    split(y) { ~5 : NIL | 0.7 : savupiipun_alue }
else: X.

kattotekstuuri-->
case katon_varityys == "punainen":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("katto/katto_1.jpg")
    tileUV(0,7, 10)
    comp(f) {side: katonreuna|bottom:katonreuna|top:X.}
case katon_varityys == "harmaa":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("katto/katto_2.jpg")
    tileUV(0,~20, ~10)
    comp(f) {side: katonreuna|bottom:katonreuna|top:X.}
else: X.

katonreuna-->
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_6.jpg")

savupiipun_alue -->

    split(x) {{~savupiippujen_etaisyys : NIL | 0.1 : savupiip-
    pu}*
    | ~savupiippujen_etaisyys : NIL}

savupiippu -->
case savupiipun_varityys== "punainen":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    texture("katto/katto_1.jpg")
    tileUV(0,7, 10)
    s(savupiippu_DimXZ, savupiippu_DimY, savupiippu_DimXZ)
    center(xz)
    i("builtin:cube")
case savupiipun_varityys== "harmaa":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    texture("katto/katto_2.jpg")
    tileUV(0,~10, 20)
    s(savupiippu_DimXZ, savupiippu_DimY, savupiippu_DimXZ)
    center(xz)
    i("builtin:cube")
else: X.

//jaetaan seinät x ja y-suuntaisiin osiin
etuseina-->
    split (y) {sokkelin_korkeus:sokkeli |
    seinan_korkeus_sokkelista: etuosan_seina}
```

Liite 5 3(4)

```
etusan_seina-->
    split(x){~0.5: seinä |
        {~ikkunavali_leveys : vali} |
        ~oven_leveys: ovalue |
        {~ikkunavali_leveys : vali} * |
        ~1: seinä|{~ikkunavali_leveys : vali} * |
        ~oven_leveys: ovalue |
        {~ikkunavali_leveys : vali}* |
        ~0.5: seinä }

takaseina-->
    split (y) {sokkelin_korkeus:sokkeli |
        seinan_korkeus_sokkelista: takaosan_seina}
takaosan_seina-->
    split(x){~0.5: seinä |
        {~ikkunavali_leveys : vali} |
        ~oven_leveys: ovalue |
        {~ikkunavali_leveys : vali} * |
        ~1: seinä|{~ikkunavali_leveys : vali} * |
        ~oven_leveys: ovalue |
        {~ikkunavali_leveys : vali}* |
        ~0.5: seinä }

sivut-->
    split (y) {sokkelin_korkeus:sokkeli |
        seinan_korkeus_sokkelista: sivuosan_seina }
sivuosan_seina-->
    split(x){~2: seinä |
        {~ikkunavali_leveys : vali}* |~2: seinä |
        {~ikkunavali_leveys : vali}* |
        ~2: seinä }

//tehdään ikkuna-aukot
vali-->
    split (x)
    {~1.5: seinä| ikkuna_leveys:ikkuna_aukko | ~1:seinä}*

ikkuna_aukko-->
    split (y){0.7:seinä |ikkuna_korkeus:ikkuna |~1:seinä }*

ikkuna-->
    setupProjection(0,scope.xy,scope.sx,scope.sy) projectUV(0)
    texture(ikkunan_vari)

//tehdään ovi ja sille tekstuuri
ovialue-->
    split (x) {~0.2:seinä
    |1.2:split (y) {oven_korkeus :ovi |~0.2:seinä }*|
        ~0.2: seinä}

ovi-->
    setupProjection(0,scope.xy,scope.sx,scope.sy) projectUV(0)
    texture("ovet/valk_ovi.jpg")

//laaditaan seinän ja sokkelin tekstuuri
seinä-->
case seinan_varitys == "tumman_harmaa":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
```

Liite 5 4(4)

```
        texture("buildings/seinat/vari_8.jpg")
        tileUV(0,4,3)
    case seinan_varitys== "vaalean_harmaa":
        setupProjection(0,scope.xz,scope.sx,scope.sz)
        projectUV(0)
        texture("buildings/seinat/vari_3.jpg")
        tileUV(0,5,0.5)

    case seinan_varitys== "tumman_keltainen":
        setupProjection(0,scope.xz,scope.sx,scope.sz)
        projectUV(0)
        texture("buildings/seinat/vari_5.jpg")
        tileUV(0,2,2.26)

    case seinan_varitys == "kirkkaan_keltainen":
        setupProjection(0,scope.xz,scope.sx,scope.sz)
        projectUV(0)
        texture("buildings/seinat/vari_4.jpg")
        tileUV(0,5,0.7)

    case seinan_varitys == "valkoinen":
        setupProjection(0,scope.xz,scope.sx,scope.sz)
        projectUV(0)
        texture("buildings/seinat/vari_7.jpg")
        tileUV(0,1.4,1.405)

    else: X.

sokkeli-->
    setupProjection(0,scope.xy,~0.8,~0.3)
    projectUV(0)
    texture("sokkeli/vari_4.jpg")
```

Liite 6 1(4), Uusien rivitalojen mallinnusta varten laadittu sääntö

```
/**
 * File:      uudet_rivitalot.cga
 * Created:   29 Sep 2015 12:23:05 GMT
 * Author:    KANVK4
 */
version "2015.0"

@StartRule
@Group ("julkisivu")
attr ikkunavali_leveys = 5
attr oven_leveys = 3
attr sokkelin_korkeus = 0.5
attr seinan_korkeus_sokkelista = 5
attr oven_korkeus = 2.1
attr ikkuna_leveys = 1.2
attr ikkuna_korkeus = 1.4
@Order(1) @Range("tumman_harmaa", "vaalean_harmaa", "valkoinen",
, "tumman_keltainen", "kirkkaan_keltainen")
attr seinan_varitys = "tumman_harmaa"
@Group("Katon ominaisuudet")
attr Rakennuksien_korkeudet = 0
attr savupiippu_DimXZ = 0.5
attr savupiippu_DimY = 0.7
attr savupiippujen_etaisyys = 8
attr kattokaltevuus = 25
@hidden
attr katto_tekstuuri = ("assets/katto/katto_1.jpg")@Order(2)
@Range("punainen", "harmaa")
@Order(2) @Range("punainen", "harmaa")
attr katon_varitys = "punainen"
@Order(3) @Range("punainen", "harmaa")
attr savupiipun_varitys = "punainen"
@Order(1) @Range("harjakatto", "aumakatto", "pulpettikatto")
attr kattotyyppi = "harjakatto"
////////////////////////////////////

//Nostetaan rakennukset keksikorkeuteen
Lot -->
    extrude (Rakennuksien_korkeudet) rakennukset
// nimetään rakennuksen seinät
rakennukset-->
comp(f){ street.front : etuseina | street.back: takaseina |
street.side : sivut | top: katto }

//tehdään harjakatto ja lisätään kattoväri ja savupiippu
katto -->
case kattotyyppi == "harjakatto":
    roofGable(kattokaltevuus, 0.5, 0.5)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}
    split(y) { ~5 : NIL | ~0.1 : savupiipun_alue }

case kattotyyppi == "aumakatto":
    roofHip(kattokaltevuus, 0.5)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}
    split(y) { ~5 : NIL | ~0.1 : savupiipun_alue }
```

Liite 6 2(4)

```
case kattotyyppi == "pulpettikatto":
    roofShed(kattokaltevuus,2)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}
    split(y) { ~5 : NIL | 0.7 : savupiipun_alue }
else: X.

kattotekstuuri-->
case katon_varitys == "punainen":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("katto/katto_1.jpg")
    tileUV(0,~20, ~10)
    comp(f) {side: katonreuna|bottom:katonreuna|top:X.}
case katon_varitys == "harmaa":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("katto/katto_2.jpg")
    tileUV(0,~20, ~10)
    comp(f) {side: katonreuna|bottom:katonreuna|top:X.}
else: X.

katonreuna-->
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_6.jpg")

savupiipun_alue -->

    split(x) {{~savupiippujen_etaisyys : NIL | 0.1 : savupiip-
    pu}*
    | ~savupiippujen_etaisyys : NIL}

savupiippu -->
case savupiipun_varitys== "punainen":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    texture("katto/katto_1.jpg")
    tileUV(0,~10, 20)
    s(savupiippu_DimXZ, savupiippu_DimY, savupiippu_DimXZ)
    center(xz)
    i("builtin:cube")
case savupiipun_varitys== "harmaa":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    texture("katto/katto_2.jpg")
    tileUV(0,~10, 20)
    s(savupiippu_DimXZ, savupiippu_DimY, savupiippu_DimXZ)
    center(xz)
    i("builtin:cube")
else: X.
```

Liite 6 3(4)

```
//jaetaan seinät x ja y-suuntaisiin osiin
etuseina-->
    split (y) {sokkelin_korkeus:sokkeli | sei-
nan_korkeus_sokkelista: etuosan_seina}

etuosan_seina-->
    split(x){~0.5: seinä |{~ikkunavali_leveys : vali2}* |
~oven_leveys: ovalue | {~ikkunavali_leveys : va-
li2} *|~oven_leveys: ovalue|~0.8: seinä|
{~ikkunavali_leveys : vali2} |
~oven_leveys: ovalue|
{~ikkunavali_leveys : vali2}* |
~oven_leveys: ovalue|
{~ikkunavali_leveys : vali}*| ~0.5: seinä }

takaseina-->
    split (y) {sokkelin_korkeus:sokkeli | sei-
nan_korkeus_sokkelista: takaosan_seina}

takaosan_seina-->
    split(x){~0.5: seinä |{~ikkunavali_leveys : vali}* |
~oven_leveys: ovalue | {~ikkunavali_leveys : va-
li} *|~2: seinä|~oven_leveys: ovi-
value|{~ikkunavali_leveys : vali} |~oven_leveys:
ovalue|
{~ikkunavali_leveys : vali}* |
~oven_leveys: ovalue|{~ikkunavali_leveys : va-
li}*|~0.5: seinä }

sivut-->
    split (y) {sokkelin_korkeus:sokkeli | sei-
nan_korkeus_sokkelista: sivuosan_seina }

sivuosan_seina-->
    split(x){~0.5: seinä |
{~ikkunavali_leveys : vali} |
oven_leveys: ovalue|
{~ikkunavali_leveys : vali} *|
~0.5: seinä }

//tehdään ikkuna-aukot
vali2-->
    split (x)
        {~1.5: seinä| 3:ikkuna_aukko2 | ~1:seinä}*

ikkuna_aukko2-->
    split (y){0.7:seinä | ikkuna_korkeus:ikkuna2 |~1:seinä }*

ikkuna2-->
    setupProjection(0,scope.xy,scope.sx,scope.sy) projectUV(0)
    texture("ikkunat/leveä_ikkuna.jpeg")

vali-->
    split (x) {~1.5: seinä| ikkuna_leveys:ikkuna_aukko
|~1:seinä}*

ikkuna_aukko-->
    split (y){0.7:seinä | ikkuna_korkeus:ikkuna |~1:seinä }*
```

Liite 6 4(4)

```
ikkuna--> setupProjection(0,scope.xy,scope.sx,scope.sy) projectUV(0)
          texture("ikkunat/ikkuna_4.jpg")

//tehdään ovi ja tekstuuri oveen
ovialue-->
          split (x) {~0.2:seina
                    |1.2:split (y) {oven_korkeus :ovi |~0.2:seina }*|
                    ~0.2:seina}

ovi-->
          setupProjection(0,scope.xy,scope.sx,scope.sy) projectUV(0)
          texture("ovet/valk_ovi.jpg")

//seinän tekstuuri
seina-->
case seinan_varitys == "tumman_harmaa":
          setupProjection(0,scope.xz,scope.sx,scope.sz)
          projectUV(0)
          texture("buildings/seinat/vari_8.jpg")
          tileUV(0,4,3)
case seinan_varitys== "vaalean_harmaa":
          setupProjection(0,scope.xz,scope.sx,scope.sz)
          projectUV(0)
          texture("buildings/seinat/vari_3.jpg")
          tileUV(0,5,0.5)
case seinan_varitys== "tumman_keltainen":
          setupProjection(0,scope.xz,scope.sx,scope.sz)
          projectUV(0)
          texture("buildings/seinat/vari_5.jpg")
          tileUV(0,2,2.26)
case seinan_varitys == "kirkkaan_keltainen":
          setupProjection(0,scope.xz,scope.sx,scope.sz)
          projectUV(0)
          texture("buildings/seinat/vari_4.jpg")
          tileUV(0,5,0.7)
case seinan_varitys == "valkoinen":
          setupProjection(0,scope.xz,scope.sx,scope.sz)
          projectUV(0)
          texture("buildings/seinat/vari_7.jpg")
          tileUV(0,1.4,1.405)
else: x.

sokkeli-->
          setupProjection(0,scope.xy,~0.8,~0.3)
          projectUV(0)
          texture("sokkeli/vari_4.jpg")
```


Liite 7 1(4), Uusien omakotitalojen autotallien mallinnusta varten laadittu sääntö

```
/**
 * File:      omakotitalon_autotalli.cga
 * Created:   29 Sep 2015 10:45:24 GMT
 * Author:    KANVK4
 */
version "2015.0"
@Group ("julkisivu",1)
attr Rakennuskorkeus = 3
@Order(1) @Range("tumman_harmaa","vaalean_harmaa","valkoinen"
,"tumman_keltainen", "kirkkaan_keltainen")
attr seinan_varitys = "tumman_harmaa"
attr ikkunavali_leveys = 2
attr oven_leveys = 2.2
attr sokkelin_korkeus = 0.5
attr seinan_korkeus_sokkelista = 2.5
attr ikkuna_leveys = 1.5
attr ikkuna_korkeus = 0.6
attr ikkunan_vari = fileRandom ("assets/ikkunat/ikkuna_*.jpg")
attr oven_korkeus = 2.1

@Group("Katon ominaisuudet",2)
@Order(3)
attr kattokaltevuus = 30
@Order(2) @Range("punainen","harmaa")
attr katon_varitys = "punainen"
@Order(1) @Range("harjakatto","aumakatto","pulpettikatto")
attr kattotyyppi = "harjakatto"

@StartRule
////////////////////////////////////
//Nostetaan rakennukset keksikorkeuteen
Lot -->
    extrude (Rakennuskorkeus) rakennukset

// nimetään rakennuksen seinät
rakennukset-->
    comp(f){ street.front : etuseina | street.back: takaseina |
    street.left : sivu_vasen |street.right:sivu_oikea| top:
    katto }

katto -->
case kattotyyppi == "harjakatto":
    roofGable(kattokaltevuus,0.5, 0.5)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}
case kattotyyppi == "aumakatto":
    roofHip(kattokaltevuus, 0.5)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}
case kattotyyppi == "pulpettikatto":
    roofShed(kattokaltevuus,2)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}
else: X.
```

Liite 7 2(4)

```
kattotekstuuri-->
case katon_varitys == "punainen":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)

    projectUV(0)
    texture("katto/katto_1.jpg")
    tileUV(0,7, 10)
    comp(f) {side: katonreuna|bottom:katonreuna|top:X.}
case katon_varitys == "harmaa":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("katto/katto_2.jpg")
    tileUV(0,~20, ~10)
    comp(f) {side: katonreuna|bottom:katonreuna|top:X.}
else: X.

katonreuna-->
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_6.jpg")

//jaetaan seinät x ja y-suuntaisiin osiin
sivu_oikea-->
    split(x){~1: takaosan_seina |
                {1.1:varaston_ovi_alue}|
                {~ikkunavali_leveys : takaosan_ikkuna_aukko}* |
                ~0.5: takaosan_seina }

etuseina-->
    split(x){~0.5: sivu_seina |
                {oven_leveys: ovialue}*|
                1: sivu_seina |
                {oven_leveys: ovialue}*|~0.5: si-
vu_seina }

sivu_seina-->
    split (y) { sokkelin_korkeus:sokkeli
                |seinan_korkeus_sokkelista: seinä }

takaseina-->
    split (y) { sokkelin_korkeus:sokkeli
                |seinan_korkeus_sokkelista: seinä }

sivu_vasen-->
    split(x){~0.5: takaosan_seina |
                {~ikkunavali_leveys : takaosan_ikkuna_aukko}* |
                1.1:varaston_ovi_alue |
                ~0.5: takaosan_seina }

varaston_ovi_alue-->
    split (y) {0.2:sokkeli | 2.2: varaston_ovi | 0.6: seinä }
```

Liite 7 3(4)

```
takaosan_seina-->
    split (y) {sokkelin_korkeus:sokkeli | sei-
nan_korkeus_sokkelista: seinä }

takaosan_ikkuna_aukko-->
    split (y) {sokkelin_korkeus:sokkeli | 1.3:seinä
|0.6:taka_ikkuna_vali|0.8:seinä}

taka_ikkuna_vali-->
    split (x) {~2: seinä| 1:ikkuna | ~2:seinä}

//tehdään ikkuna-aukot
vali-->
    split (x) {~2: seinä| ikkuna_leveys:ikkuna_aukko |
~2:seinä}

ikkuna_aukko-->
    split (y){ 1.3:seinä |ikkuna_korkeus:ikkuna |~1.1:seinä }*

ikkuna-->
    setupProjection(0,scope.xy,scope.sx,scope.sy) projectUV(0)
    texture("ikkunat/P7211324.JPG")

//tehdään ovi ja tekstuuri oveen

ovialue-->
    split (y){0.1:sokkeli |2.2:ovi |0.7:seinä}

ovi-->
    setupProjection(0,scope.xy,scope.sx,scope.sy)
    projectUV(0)
    texture("ovet/autotallin_ovi_2.jpg")

varaston_ovi-->
    setupProjection(0,scope.xy,scope.sx,scope.sy)
    projectUV(0)
    texture("ovet/valk_ovi.jpg")

seinä-->
case seinan_varityys == "tumman_harmaa":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_8.jpg")
    tileUV(0,4,3)

case seinan_varityys== "vaalean_harmaa":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_3.jpg")
    tileUV(0,5,0.5)

case seinan_varityys== "tumman_keltainen":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
```

Liite 7 4(4)

```
texture("buildings/seinat/vari_5.jpg")
tileUV(0,2,2.26)

case seinan_varitys == "kirkkaan_keltainen":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_4.jpg")
    tileUV(0,5,0.7)

case seinan_varitys == "valkoinen":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_7.jpg")
    tileUV(0,1.4,1.405)

else: X.

sokkeli-->
    setupProjection(0,scope.xy,~0.8,~0.3)
    projectUV(0)
    texture("sokkeli/vari_4.jpg")
```

Liite 8 1(3) Uusien rivi- ja paritalojen autotallien mallinnusta varten muodostettu sääntö

```
/**
 * File:    rivitalo_autotalli.cga
 * Created: 23 Sep 2015 08:38:30 GMT
 * Author:  KANVK4
 */
version "2015.0"

@Group ("julkisivu",1)
attr Rakennuskorkeus = 3
@Order(1) @Range("tumman_harmaa","vaalean_harmaa","valkoinen",
,"tumman_keltainen", "kirkkaan_keltainen")
attr seinan_varitys = "tumman_harmaa"
attr ikkunavali_leveys = 2
attr oven_leveys = 2.2
attr sokkelin_korkeus = 0.5
attr seinan_korkeus_sokkelista = 2.5
attr ikkuna_leveys = 1.5
attr ikkuna_korkeus = 0.6
attr ikkunan_vari = fileRandom ("assets/ikkunat/ikkuna_*.jpg")

attr oven_korkeus = 2.1

@Group("Katon ominaisuudet",2)
@Order(3)
attr kattokaltevuus = 30
@Order(2) @Range("punainen","harmaa")
attr katon_varitys = "punainen"
@Order(1) @Range("harjakatto","aumakatto","pulpettikatto")
attr kattotyyppi = "harjakatto"
////////////////////////////////////
@StartRule
//Nostetaan rakennukset keksikorkeuteen
Lot -->
    extrude (Rakennuskorkeus) rakennukset
// nimetään rakennuksen seinät
rakennukset-->
comp(f){ street.front : etuseina | street.back: takaseina |
street.side : sivut | top: katto }

katto -->
case kattotyyppi == "harjakatto":
    roofGable(kattokaltevuus,0.5, 0.5)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}

case kattotyyppi == "aumakatto":
    roofHip(kattokaltevuus, 0.5)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}

case kattotyyppi == "pulpettikatto":
    roofShed(kattokaltevuus,2)
    comp(f) {bottom : NIL | top : kattotekstuuri | side: seinä}
else: X.
```

Liite 8 2(3)

```
kattotekstuuri-->
case katon_varitys == "punainen":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("katto/katto_1.jpg")
    tileUV(0,7, 10)
    comp(f) {side: katonreuna|bottom:katonreuna|top:X.}
case katon_varitys == "harmaa":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("katto/katto_2.jpg")
    tileUV(0,~20, ~10)
    comp(f) {side: katonreuna|bottom:katonreuna|top:X.}
else: X.

katonreuna-->
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_6.jpg")
//jaetaan seinät x ja y-suuntaisiin osiin
takaseina-->
    split (y) { sokkelin_korkeus:sokkeli
                |seinan_korkeus_sokkelista: takaosan_seina }

takaosan_seina-->
    split (x) {~2: seinä |
                {~ikkunavali_leveys : vali}* |
                ~2: seinä|{~ikkunavali_leveys : vali}*|~2: seinä}
etuseina-->
    split(x){~2: etuosan_seina |
             {oven_leveys: ovalue}*|~2: etuosan_seina|
             {oven_leveys: ovalue}*|~2: etuosan_seina }

etuosan_seina-->
    split (y) { sokkelin_korkeus:sokkeli
                |seinan_korkeus_sokkelista: seinä }

sivut-->
    split(x){~0.5: sivuosan_seina |
             {~ikkunavali_leveys : sivu_ikkuna_aukko}* |
             1.1:varaston_ovi_alue |~0.5: sivuosan_seina }
varaston_ovi_alue-->
    split (y) {0.2:sokkeli | 2.2: varaston_ovi | 0.6: seinä }
sivuosan_seina-->
    split (y) {sokkelin_korkeus:sokkeli | sei-
               nan_korkeus_sokkelista: seinä }
sivu_ikkuna_aukko-->
    split (y) {sokkelin_korkeus:sokkeli | 1.3:seinä
               |0.6:sivu_ikkuna_vali|0.8:seinä}
sivu_ikkuna_vali-->
    split (x) {~2: seinä| 1:ikkuna | ~2:seinä}
```

Liite 8 3(3)

```
//tehdään ikkuna-aukot
vali-->
    split (x) {~2: seinä| ikkuna_leveys:ikkuna_aukko |
~2:seinä}
ikkuna_aukko-->
    split (y){ 1.3:seinä |ikkuna_korkeus:ikkuna |~1.1:seinä }*

ikkuna-->
    setupProjection(0,scope.xy,scope.sx,scope.sy) projectUV(0)
    texture("ikkunat/P7211324.JPG")
//tehdään ovi ja tekstuuri oveen
ovialue-->
    split (y){0.1:sokkeli |2.2:ovi |0.7:seinä}
ovi-->
    setupProjection(0,scope.xy,scope.sx,scope.sy)
    projectUV(0)
    texture("ovet/autotallin_ovi_2.jpg")

varaston_ovi-->
    setupProjection(0,scope.xy,scope.sx,scope.sy)
    projectUV(0)
    texture("ovet/valk_ovi.jpg")
//seinän tekstuuri
seinä-->
case seinan_varitys == "tumman_harmaa":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_8.jpg")
    tileUV(0,4,3)
case seinan_varitys== "vaalean_harmaa":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_3.jpg")
    tileUV(0,5,0.5)
case seinan_varitys== "tumman_keltainen":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_5.jpg")
    tileUV(0,2,2.26)
case seinan_varitys == "kirkkaan_keltainen":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_4.jpg")
    tileUV(0,5,0.7)
case seinan_varitys == "valkoinen":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_7.jpg")
    tileUV(0,1.4,1.405)
else: x.

sokkeli-->
    setupProjection(0,scope.xy,~0.8,~0.3)
    projectUV(0)
    texture("sokkeli/vari_4.jpg")
```


Liite 9 1(4), Maankäytön mallintamista varten laadittu sääntö

```
/**
 * File:      maankaytto.cga
 * Created:   27 May 2015 20:11:29 GMT
 * Author:    KANVK4
 */
version "2015.0"
//////////////////////////////////ATRIBUUTIT//////////////////////////////////
@Group ("Maankäyttö tyyppi")@Order (1)
attr tyyppi                                     = ""
//Puiden attribuutit
@Group("Puut") @Order (2)
attr PUU_X_korkeus                             = rand(7,17)
@Group("Puut") @Order (3)
attr PUU_Y_korkeus                             = rand(9,20)
@Group("Puut") @Order(4)
attr Nurmen_puiden_korkeus                     = 13
@Group("Puut") @Order (5)
attr random_rotaatio                           = true
attr puiden_etaisyydet                         = 25
const randomRotation                           = case random_rotaatio:
rand(0,360) else: 0

//Vesialueiden attribuutit
@Group ("Vesi")@Order(1)
@Range("River", "Stream", "Pond", "Lake", "Ocean")
attr Water_Type                                = "River"
@Range(0,1.0)@Order(2)
attr r = .44
@Range(0, 1.0)@Order(3)
attr g = .55
@Range(0, 1.0)@Order(4)
attr b = .44
//////////////////////////////////Textures//////////////////////////////////
sora_tekstuuri      = "vegetation/natural_textures/sora.jpg"
ruoho_tekstuuri     = "vegetation/natural_textures/lawn.jpg"
asfaltti_tekstuuri  = "streets/textures/asphalt_brighter.jpg"
viheralue_tekstuuri = "vegetation/natural_textures/viheralue2.jpg"
hiekkat_ktektuuri   = "vegetation/natural_textures/hiekka.jpg"
joenreuna_tekstuuri = "vegetation/natural_textures/nurmi.jpg"
//////////////////////////////////Säännöt//////////////////////////////////

@StartRule
Lot -->
case tyyppi== "vesialue" :t(0,0.01,0) Vesi
case tyyppi== "viheralue" : t(0,0.01,0) Viheralue
case tyyppi== "uusi_puisto":t(0,0.01,0) Viheralue
case tyyppi== "uusi_tontti": t(0,0.01,0) Viheralue
case tyyppi== "nurmi" : t(0,0.01,0) Nurmi
case tyyppi== "pihakivetytys" : t(0,0.01,0)Pihakivetytys
case tyyppi== "meluvalli" : t(0,0.01,0) Meluvalli
case tyyppi== "asfalttitie" : t(0,0.01,0) Asfaltti
case tyyppi== "soratie" : t(0,0.01,0) Sora
case tyyppi== "jokivalli": t(0,0.01,0) Jokivalli
case tyyppi== "joenreuna":t(0,0.01,0) Joenreuna
case tyyppi== "leikkipaikka":t(0,0.01,0)Leikkipaikka
case tyyppi== "tekninen": t(0,0.01,0)Tekninen
else:NIL
```

Liite 9 2(4)

Vesi -->

```
// __waterparams_scale_speed
    case Water_Type == "River" :
        set(material.name, "riv-
er__water__waterparams_5_10")
        color(.44,.55,.44)
    case Water_Type == "Stream" :
        set(material.name, "riv-
er__water__waterparams_5_30")
        color(.44,.55,.44)
    case Water_Type == "Pond" :
        set(material.name, "riv-
er__water__waterparams_1_3")
        color(.2,.4,.45)
    case Water_Type == "Lake" :
        set(material.name, "riv-
er__water__waterparams_4_8")
        color(.2,.4,.45)
    case Water_Type == "Ocean" :
        set(material.name, "riv-
er__water__waterparams_15_20")
        color(.2,.4,.45)
    else: x.
```

Meluvalli-->

```
setupProjection(0,scope.xz,scope.sx,scope.sz)
projectUV(0)
roofGable(30,0,0,false,0)
split(y){3:Huippu}
```

Huippu-->

```
setupProjection(0,scope.xz,scope.sx,scope.sz)
projectUV(0)
texture(viheralue_tekstuuri)
tileUV(0,~100,80)
```

Nurmi-->

```
alignScopeToGeometry(yUp, 0, longest)
setupProjection(0,scope.xz,scope.sx,scope.sz)
projectUV(0)
comp(f) {top : Viheralue Nurmen_puut }
```

Nurmen_puut-->

```
alignScopeToGeometry(yUp, 0, longest)
t(0,0,0)
split(u,unitSpace,0){ ~ puiden_etaisyydet*0.5 : NIL
| {0.1: PUU_nurmi | ~ puiden_etaisyydet:NIL}*}
```

PUU_nurmi-->

```
alignScopeToAxes(y)
s(0,Nurmen_puiden_korkeus,0) center(xz)
r(0,rand(0,360),0)
i("/ESRI.lib/assets/Plants/Paper_Birch/Paper_Birch_Model_0.
obj")
set(material.opacity,1.0)
```

Liite 9 3(4)

Viheralue-->

```
alignScopeToGeometry(yUp, 0,0)
setupProjection(0,scope.xz,scope.sx,scope.sz)
projectUV(0)
texture (viheralue_tekstuuri)
tileUV(0,~100,60)
```

Jokivalli-->

```
setupProjection(0,scope.xz,scope.sx,scope.sz)
projectUV(0)
translateUV(0,0,0)
roofGable(15,0,0)
tileUV(0,~100,60)
split(y){ 3:Jokivallin_alue}
```

Jokivallin_alue-->

```
setupProjection(0,scope.xz,scope.sx,scope.sz)
projectUV(0)
tileUV(0,~100,60)
comp(f) {top : Viheralue Puut Havupuut }
```

Puut-->

```
scatter(surface, 1.5, uniform) {PUU_X }
```

Havupuut-->

```
scatter(surface, 2, uniform) {PUU_Y }
```

PUU_X -->

```
alignScopeToAxes(y)
s(0,PUU_X_korkeus,0) center(xz)
r(0,rand(0,360),0)
i("/ESRI.lib/assets/Plants/Norway_Spruce/Norway_Spruce_Mode
l_0.obj")
```

PUU_Y -->

```
alignScopeToAxes(y)
s(0,PUU_Y_korkeus,0) center(xz)
r(0,rand(0,360),0)
i("/ESRI.lib/assets/Plants/Paper_Birch/Paper_Birch_Model_0.obj")
```

Joenreuna-->

```
setupProjection(0,scope.xz,scope.sx,scope.sz)
projectUV(0)
translateUV(0,0,0)
texture(joenreuna_tekstuuri)
tileUV(0,~120,60)
```

Pihakivetys-->

```
setupProjection(0,scope.xz,scope.sx,scope.sz)
projectUV(0)
texture(sora_tekstuuri)
tileUV(0,2,2)
```

Liite 9 4(4)

Leikkipaikka-->

```
    alignScopeToGeometry(yUp, 0,0)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture (hiekkatekstuuri)
    tileUV(0,~40,5)
```

Ruoho -->

```
    alignScopeToGeometry(yUp, 0,0)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture(ruohotekstuuri)
    tileUV(0,~20,~20)
```

Sora -->

```
    alignScopeToGeometry(yUp, 0,0)
    setupProjection(0,scope.xz,scope.sx*6,scope.sz*10)
    projectUV(0)
    translateUV(0,0.31,0.62)
    texture(soratekstuuri)
    tileUV(0,~3,~3)
```

Asfaltti-->

```
    alignScopeToGeometry(yUp, 0,0)
    setupProjection(0,scope.xz,scope.sx*6,scope.sz*10)
    projectUV(0)
    texture(asfalttitekstuuri)
    tileUV(0,~1,~1)
```

Tekninen-->

```
    extrude (2)
    color (0.7,0.7,0.6)
```

Liite 10 Tonttien pensasaitojen mallintamista varten laadittu sääntö

```
/**
 * File:      tonttien_pensasaidat.cga
 * Created:   10 Jun 2015 10:00:14 GMT
 * Author:    KANVK4
 */
version "2015.0"

@Group("asetukset")
attr pensaiden_etaisyys          = 0.1
attr korkeus                     = 1.8
attr ROTATION_RANDOMIZE          = true
////////////////////////////////////
const randomRotation = case ROTATION_RANDOMIZE: rand(0,360) else: 0
puska_aita = ("/ESRI.lib/assets/Plants/Boxwood/Boxwood_Model_0.obj")

//////////puska-aidan ala-osan tekstuuri
@Hidden
attr katto_Ilmaukuva = ("images/ilmakuva_ETRSTM35fin3.tif")
@Hidden
attr ilmakeuvan_koko_X = 639.200
@Hidden
attr ilmakeuvan_koko_Z = 715.400
@Hidden
attr ilmakeuvan_offset_X = 224111.957
@Hidden
attr ilmakeuvan_offset_Z = -6831115.704
////////////////////////////////////

@StartRule
Lot-->
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    comp(f) {top :pohjatekstuuri pensaat}

pensaat-->
    alignScopeToGeometry(yUp, 0, longest)
    t(0,1,0)
    split(x)
    {~1: pensas | ~pensaiden_etaisyys:NIL
    |{ ~1: pensas | ~pensaiden_etaisyys:NIL}*}

pensas -->
    s(0,korkeus*1,0)
    center(xy)
    r(0,rand(0,360),0)
    i(puska_aita)
    set(material.opacity,1.0)

pohjatekstuuri-->
    setupProjection (0,world.xz,ilmakeuvan_koko_X,-           ilmakeu-
van_koko_Z,ilmakeuvan_offset_X,ilmakeuvan_offset_Z)
    projectUV(0)
    texture(katto_Ilmaukuva)
```

Liite 11 1(2) Esimerkki yhdestä kuvamallinnuksella laaditun rakennuksen säännöstä

```
/**
 * File:      Talojen_yhteissaanto.cga
 * Created:   29 Sep 2015 12:09:54 GMT
 * Author:    KANVK4
 */
version "2015.0"
@StartRule

@Group("Katon_ominaisuudet")@Order(3) @Range("gable","hip")
attr Rakennuksien_korkeudet      = 0
attr kattokaltevuus              = 25
attr Rakennustyyppi              = ""
attr Tyyppi                      = ""

import omakotita-
lo_1:"rules/mallinnus_rules/omakotitalo1_kuvamallinnus/omakotitalo_1.cga"
import omakotita-
lo_2:"rules/mallinnus_rules/omakotitalo2_kuvamallinnus/omakotitalo_2.cga"
import omakotita-
lo_3:"rules/mallinnus_rules/omakotitalo3_kuvamallinnus/omakotitalo_3.cga"
import omakotita-
lo_4:"rules/mallinnus_rules/omakotitalo4_kuvamallinnus/omakotitalo_4.cga"
import omakotita-
lo_5:"rules/mallinnus_rules/omakotitalo5_kuvamallinnus/omakotitalo_5.cga"
import autotal-
li_1:"rules/mallinnus_rules/omakotitalo1_kuvamallinnus/autotalli_keltainen.cga"
import autotal-
li_2:"rules/mallinnus_rules/omakotitalo4_kuvamallinnus/talo4_autotalli.cga"
import autotal-
li_3:"rules/mallinnus_rules/omakotitalo5_kuvamallinnus/autotalli_harmaa_punainen_katto.cga"
import autotal-
li_4:"rules/mallinnus_rules/omakotitalo3_kuvamallinnus/autotalli_valkoinen_2.cga"
import varas-
to_harmaa:"rules/mallinnus_rules/omakotitalo2_kuvamallinnus/varasto_harmaa.cga"
import varas-
to_harmaa_punainen_katto:"rules/mallinnus_rules/omakotitalo5_kuvamallinnus/varasto_harmaa_punainen_katto.cga"
import varas-
to_keltainen:"rules/mallinnus_rules/omakotitalo1_kuvamallinnus/varasto_keltainen.cga"
import rivita-
lo:"rules/mallinnus_rules/Rivitalojen_kuvamallinnus/Rivitalo_kuvamallinnus.cga"
```

Liite 11 2(2)

```
import rivita-  
lo_varasto:"rules/mallinnus_rules/Rivitalojen_kuvamallinnus/rivitalo_v  
arasto.cga"
```

```
import Parita-  
lo_kuvamallinnus:"rules/mallinnus_rules/Paritalo_kuvamallinnus/Parital  
o_kuvamallinnus.cga"
```

```
////////////////////////////////////
```

```
//Nostetaan rakennukset keksikorkeuteen
```

```
Lot -->
```

```
case Tyyppi == "varastoharmaa" :varasto_harmaa.Lot  
case Tyyppi== "varastohar-  
maa_punainen_katto":varasto_harmaa_punainen_katto.Lot  
case Tyyppi == "varastokeltainen":varasto_keltainen.Lot  
case Tyyppi == "rivitalovarasto":rivitalo_varasto.Lot  
case Tyyppi == "paritalovarasto": varasto_harmaa.Lot  
case Tyyppi=="omakotitalokeltainen": omakotitalo_1.Lot  
case Tyyppi=="omakotitaloharmaa": omakotitalo_2.Lot  
case Tyyppi=="omakotitaloharmaa2": omakotitalo_5.Lot  
case Tyyppi=="omakotitaloharmaa3": omakotitalo_4.Lot  
case Tyyppi=="omakotitalovalkoinen": omakotitalo_3.Lot  
case Tyyppi=="autotallikeltainen":autotalli_1.Lot  
case Tyyppi=="autotalliharmaa": autotalli_2.Lot  
case Tyyppi=="autotallivalkoinen": autotalli_4.Lot  
case Tyyppi=="autotalliharmaa_punainen_katto": autotalli_3.Lot  
case Tyyppi=="rivitalo": rivitalo.Lot  
case Tyyppi=="paritalo": Paritalo_kuvamallinnus.Lot  
else: NIL
```


Liite 12 1(3), Rakennuksien säännöt yhdistävä sääntötiedosto

```
/**
 * File:      omakotitalo_1.cga
 * Created:   5 Oct 2015 12:49:56 GMT
 * Author:    KANVK4
 */
version "2015.1"

@StartRule

@Group("Katon ominaisuudet")
attr Rakennuksien_korkeudet = 0
attr savupiippu_DimXZ       = 0.5
attr savupiippu_DimY        = 0.7
attr savupiippujen_etaisyys = 8
attr kattokaltevuus         = 25
@hidden
attr katto_tekstuuri = ("assets/katto/katto_1.jpg")@Order(2)
@Range("punainen", "harmaa")
@Order(2) @Range("punainen", "harmaa")
attr katon_varitys = "punainen"
@Order(3) @Range("punainen", "harmaa")
attr savupiipun_varitys = "punainen"
@Order(1) @Range("harjakatto", "aumakatto", "pulpettikatto")
attr kattotyypyi       = "harjakatto"

import f1: "rules/mallinnus_rules/omakotitalo1_kuvamallinnus/omakotitalo_etusivu.cga"
import f2: "rules/mallinnus_rules/omakotitalo1_kuvamallinnus/omakotitalo_vasen_sivu.cga"
import f3: "rules/mallinnus_rules/omakotitalo1_kuvamallinnus/omakotitalo_oikea_sivu.cga"
import f4: "rules/mallinnus_rules/omakotitalo1_kuvamallinnus/omakotitalo_takasivu3.cga"
import f5: "rules/mallinnus_rules/omakotitalo1_kuvamallinnus/omakotitalo_vasen_ylasivu.cga"
import f6: "rules/mallinnus_rules/omakotitalo1_kuvamallinnus/omakotitalo_oikea_ylä_sivu.cga"
import f7: "rules/mallinnus_rules/omakotitalo1_kuvamallinnus/sivu2.cga"
import f8: "rules/mallinnus_rules/omakotitalo1_kuvamallinnus/sivu3.cga"

Lot--> extrude (Rakennuksien_korkeudet)
comp (f){1: katto| 2: sivu2| 3: sivu3
        | 4: sivu4 | 5: sivu5| 6: sivu6 |
        7: sivu7| 8: sivu8| 9: sivu9 |
        10: sivu10 | 11: sivu11 |
        12: sivu12| 13: sivu13| 14: sivu14| 15: sivu15 | 16: sivu16
        | 17: sivu17| 18: sivu18 | 19: sivu19 | 20: sivu20}
```

Liite 12 2(3)

```
sivu2-->f4.Facade
sivu3-->f2.Facade
sivu4-->f1.Facade
sivu5--> f3.Facade
sivu6--> f7.Facade
sivu7-->f8.Facade
sivu8-->f1.Facade
sivu9-->f7.Facade
sivu10-->f4.Facade
sivu11-->f7.Facade
sivu12-->f7.Facade
sivu13-->f7.Facade
sivu14-->f7.Facade
sivu15-->f7.Facade
sivu16-->f8.Facade
sivu17-->f7.Facade
sivu18-->f8.Facade
sivu19-->f7.Facade
sivu20-->f7.Facade

//tehdään katto ja lisätään kattoväri ja savupiippu
katto -->
case kattotyyppi == "harjakatto":
    roofGable(kattokaltevuus,0.5, 0.5)
    comp(f) {bottom : NIL | top : kattotekstuuri |
street.right: sivu_vasen|street.left:sivu_oikea
    |street.front: sivu_oikea | street.back:sivu_vasen}
    split(y) { ~5 : NIL |~0.1 : savupiipun_alue }

case kattotyyppi == "aumakatto":
    roofHip(kattokaltevuus, 0.5,false)
    comp(f) {bottom : NIL | top : kattotekstuuri |
street.right: sivu_vasen|street.left:sivu_oikea
    |street.front: sivu_oikea | street.back:sivu_vasen}
    split(y) { ~5 : NIL |~0.1 : savupiipun_alue }

case kattotyyppi == "pulpettikatto":
    roofShed(kattokaltevuus,2)
    comp(f) {bottom : NIL | top : kattotekstuuri |
street.right: sivu_vasen|street.left:sivu_oikea
    |street.front: sivu_oikea | street.back:sivu_vasen}
    split(y) { ~5 : NIL |0.7 : savupiipun_alue }
else: X.

sivu_vasen-->f6.Facade
sivu_oikea-->f5.Facade

kattotekstuuri-->
case katon_varitys == "punainen":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("katto/katto_1.jpg")
```

Liite 12 3(3)

```
tileUV(0,~20, ~10)
comp(f) {side: katonreuna|bottom:katonreuna|top:X.}

case katon_varitys == "harmaa":
    extrude (0.1)
    t(0,0,-0.0465)
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("katto/katto_2.jpg")
    tileUV(0,~20, ~10)
    comp(f) {side: katonreuna|bottom:katonreuna|top:X.}
else: X.

katonreuna-->
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    projectUV(0)
    texture("buildings/seinat/vari_6.jpg")

savupiipun_alue -->

    split(x) {{~savupiippujen_etaisyys : NIL | 0.1 : savupiip-
    pu}*
    | ~savupiippujen_etaisyys : NIL}

savupiippu -->
case savupiipun_varitys== "punainen":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    texture("katto/katto_1.jpg")
    tileUV(0,~10, 20)
    s(savupiippu_DimXZ, savupiippu_DimY, savupiippu_DimXZ)
    center(xz)
    i("builtin:cube")

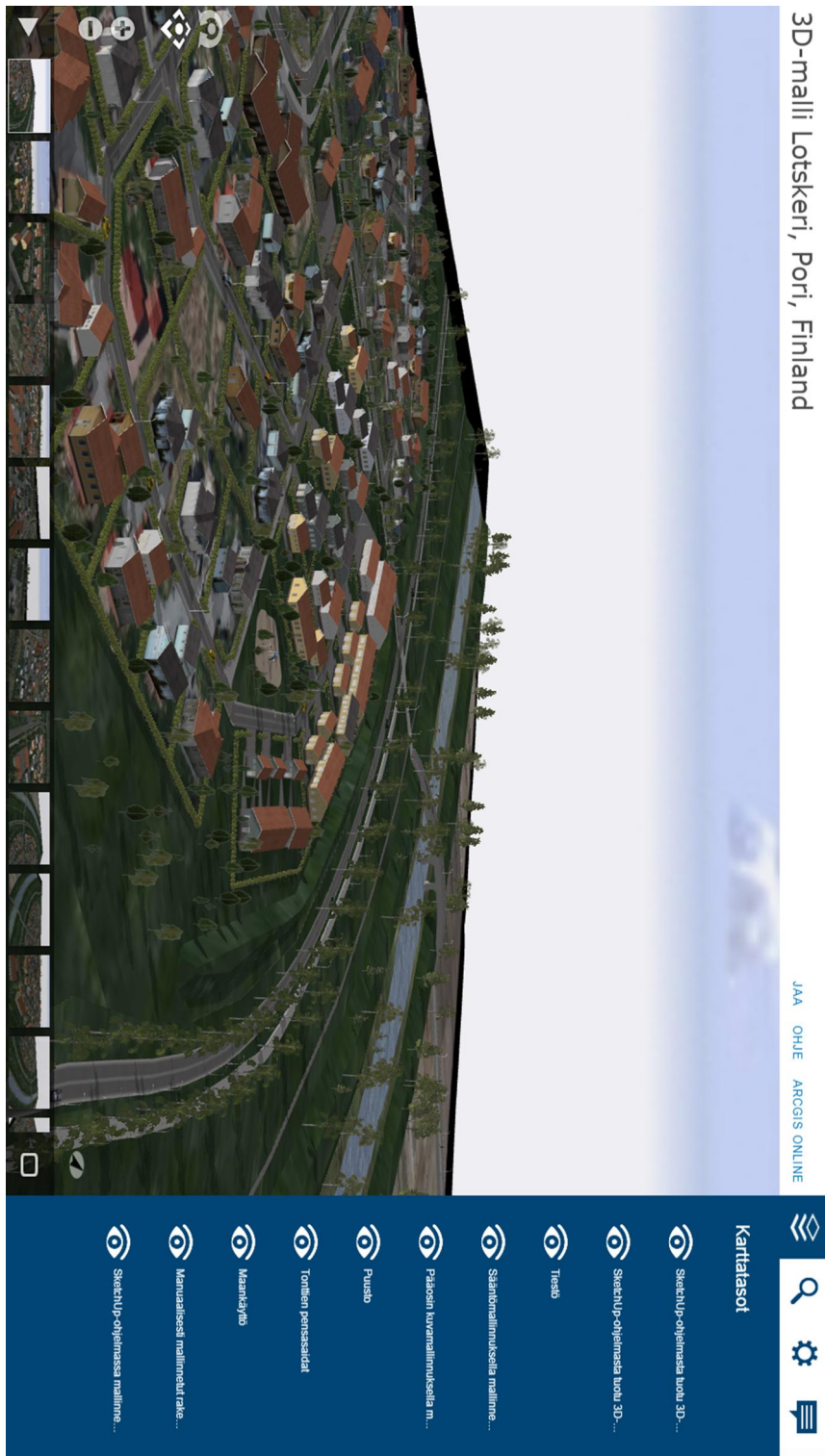
case savupiipun_varitys== "harmaa":
    setupProjection(0,scope.xz,scope.sx,scope.sz)
    texture("katto/katto_2.jpg")
    tileUV(0,~10, 20)
    s(savupiippu_DimXZ, savupiippu_DimY, savupiippu_DimXZ)
    center(xz)
    i("builtin:cube")

else: X.
```

Liite 13 CityEngine-ohjelmalla laadittu mallinnus

Mallinnus on katsottavissa seuraavassa osoitteessa (ei toimi Internet Explorer – selaimella):

<http://pori.maps.arcgis.com/apps/CEWebViewer/viewer.html?3dWebScene=d4acf50db1a64b4681dd5ee7b2c77532>



Liite 14 SketchUp-ohjelmalla laadittu mallinnus

