Eve Vainio

3D-KAUPUNKIMALLINNUKSEN AUTOMATISOINTI CITYENGINELLÄ

Opinnäytetyö
Tietojenkäsittely

2020

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkinto	Aika
Eve Vainio	Tradenomi (AMK)	Toukokuu 2020

Opinnäytetyön nimi

3D-kaupunkimallinnuksen automatisointi CityEnginellä

32 sivua

Toimeksiantaja

Vantaan Kaupunki

Ohjaajat

Esa Hannus & Jere Virolainen

Tiivistelmä

Viime vuosina kolmiulotteinen kaupunkimallinnus on yleistynyt Suomen kunnissa ja kaupungeissa, mutta vielä on pitkä matka, jotta kaikkialla Suomessa kaupunkimalli olisi yleinen standardi. Opinnäytetyö on tehty Vantaan kaupungille. Vaikka Vantaan kaupunkimallin tila on parempi kuin monelle muulla kunnalla, se ei ole estänyt heitä kehittämästä kaupunkimalliaan aina paremmaksi. Tämän opinnäytetyön tarkoituksena on selvittää, kuinka CityEngine sopii Vantaan kaupungin käyttöön ja pystyykö sillä luomaan prosessin, jolla 3D-mallin päivitystä voitaisiin automatisoida.

Teoriaosuudessa opinnäytetyö käsittelee ensin kaupunkimallinnusta sen juurilta tähän päivään, minkä jälkeen kerrotaan, mikä on kaupunkimallinnuksen nykytila. Seuraavassa osiossa käsitellään tarkkuustasoja, mitä ne ovat ja kuinka ne määrittyvät. Tämän lisäksi teoriaosuudessa perehdytään projektissa käytettyihin ohjelmistoihin ja tiedostomuotoihin.

Työn aluksi perehdytään FME-työtilaan ja Shapefile-tiedoston luomiseen, sekä CityEngine-ohjelmistoon. Tämän jälkeen tutustutaan CityEnginen mallinnusmenetelmiin ja sääntöpohjaiseen mallintamiseen eli CGA-kieleen. Työn edetessä tutustutaan ArcGIS Pro -ohjelmaan, jonne mallit viedään, kun ne ovat valmiita testattavaksi. Kun mallinnusprosessi on kunnossa, niin aloitetaan tutkimaan automatisointi prosessia ja sen menetelmiä. CityEngineen on luotu Python-ohjelmointia varten ominaisuuksia, jolla pystyy ohjelmoimaan ja testaamaan Python-koodia. Lopputuloksena on LOD1-tason malleja, jotka on luotu Esri File GeoDataBase-muotoon ja viety ArcGIS Pro -tuotantoympäristöön.

Asiasanat

CityEngine, 3D-mallinnus, opinnäytetyö, automatisointi

Author (authors)	Degree	Time
Eve Vainio	Bachelor of Business Administration	May 2020

Thesis title

Automatisation of 3D city modeling with CityEngine

32 pages

Commissioned by

City of Vantaa

Supervisors

Esa Hannus & Jere Virolainen

Abstract

In recent years, three-dimensional urban modeling has become more common in Finnish municipalities and cities, although the practice has yet to become a national standard. The purpose of this thesis was to find out if CityEngine would be suitable for the use of the City of Vantaa and whether it could create a process in which the 3D model update could be automated.

The first theoretical part dealt with urban modeling from its roots to the present day, after which the current state of urban modeling was explained. The next section discussed the levels of accuracy; what they were and how they were defined. In addition, the theoretical part introduced the software and file formats used in the project.

The work began with an introduction to the FME workspace and the creation of Shapefile, as well as the City Engine software. This was followed by an introduction to CityEngine's modeling methods and rule-based modeling utilizing the CGA language. As the work progressed, the thesis introduced ArcGIS Pro where the models were taken when they were ready. After that the thesis researched the automation process and its methods, as CityEngine had features for Python programming that were used to program and test Python code. The results were LOD1-level templates created in the Esri File GeoDataBase format and exported to the ArcGIS Pro tutorial environment.

Keywords

CityEngine, 3D modeling, thesis, automatisation

SISÄLLYS

KUVALUETTELO

KÄYTETYT OHJELMAT JA LYHENTEET

FME	Feature Manipulation Engine on datan integraatio-alusta.
CityEngine	Edistyksellinen 3D-mallinnusohjelmisto, jolla voi luoda vuorovaikutteisia kaupunkiympäristöjä.
CGA	CityEngine-ohjelman rakennusten muotoiluun tarkoitettu kieli (Computer Generated Architecture)
ArcGIS Pro	ArcGIS Pro on moderni paikkatieto-ohjelmisto työasemakäyttöön.
GIS	Paikkatietojärjestelmä (Geographical Information System)
Python	Monipuolinen ja käyttäjäystävällinen ohjelmointikieli
LOD	3D-kaupunkimallin tarkkuustaso eli Level of Detail
KML/KMZ	Keyhole Markup Language
Shapefile	Vektoridatan tallentamiseen tarkoitettu tiedostomuoto, johon voi tallentaa sijainnin, 3D-muotoja ja geologisia maastonmuotoja.
SQL	Structured Query Language. SQL on relaatiotietokanta ja standardoitu kyselykieli, joka on IBM:n kehittämä.
GML	(Geography Markup Language) GML on XML-kieleen perustuva kieli. Luotu määrittämään paikkatietodataa.
FGDB	Esri File GeoDataBase on tiedostopohjainen tietovarasto paikkatietodatalle

1 JOHDANTO

Kaupunkien 3D-mallintaminen on ollut puheen aiheena Suomen kunnilla ja kaupungeilla jo melkein kolmen vuosikymmenen ajan, mutta viimeisen vuosikymmenen aikana aihe on todella nostettu pintaan, jolloin edellä mainitut tahot ovat ryhtyneet vähitellen huomaamaan, mitä hyötyä kaupunkimalleista voi olla. Kolmiulotteinen kaupunkimalli edistää kaupunkien mahdollisuuksia suunnitella ja visualisoida rakennuksia, sekä käyttää sitä apuvälineenä ja edistää kaupungin toimivuutta.

Usein kunnat ja kaupungit tekevät tai teettävät malleja, joko käsin tai laserdataa muotoilemalla. Kaupungit ovat usein suosineet SketchUp-mallinnusohjelmaa tai Terrasolid-ohjelmistoja. Molemmat tapaukset vaativat paljon työtunteja, dataa ja ylläpitoa. Monet kaupungit pohtivat tähän ongelmaan ratkaisua.

Tämän opinnäytetyön toimeksiantajana toimii Vantaan kaupungin kaupunkimittausosasto. Vantaan kaupunkimalli on toimiva ja sitä päivitetään muutaman vuoden välein laserkeilausaineiston pohjalta. Kuitenkin Vantaa on kehittyvä kaupunki, jota rakennetaan ja kehitetään jatkuvasti. Tällöin myös kaupunkimallin olisi hyvä pysyä kehityksen perässä.

Opinnäytetyön tavoitteena on tutkia, voidaanko CityEngineä käyttää järkevästi vähintään LOD1-tason kaupunkimallien luomiseen niiden rakennusten osalta, joita ei saada mallinnettua pistepilvikeräysten välisinä aikoina. Työn tavoitteena on myös tutkia, voidaanko yllä mainittua prosessia automatisoida. Selvitystyö tehdään käyttämällä Vantaan kaupungin aineistoa ja tarvittavia ohjelmistoja.

Tässä raportissa kerrotaan työn eri vaiheista ja tuloksista. Raportin alussa kuvataan mitä kolmiulotteinen kaupunkimallintaminen on ja kerrotaan mitä tarkkuustasot ovat, sekä miten ne määrittyvät. Seuraavaksi kerrotaan tarkemmin mm. CityEngine- ja ArcGIS Pro -ohjelmistoista. Toteutus osiossa kerrotaan mallinnustavoista ja prosessin automatisoinnista sekä lopuksi esitetään omia pohdintoja ja kuvataan työn tuottamia tuloksia.

2 3D-KAUPUNKIMALLINNUS

Tässä osiossa käsitellään opinnäytetyön teoreettista osiota ja kerrotaan kaupunkimallinnuksesta. Ensimmäisessä osiossa käydään läpi kaupunkimallinnuksen historiaa, erilaisia kaupunkimallinnuksen tapoja ja millaista se on nykyään Suomessa. Seuraavassa osiossa kerrotaan tarkkuustasoista ja miten ne määrittyvät. Viimeisessä osiossa kerrotaan projektissa käytetyistä ohjelmistoista ja tiedostomuodoista, kuten CityEnginestä ja FME-työtilasta.

2.1 3D-Kaupunkimalli

Seuraavaa lukua koskevat osiot kaupunkimallinnuksesta perustuvat pääosin kaupunkimallinnuksen ohjekirjan (2016) sisältöön. Kaupunkimallinnusta ja suunnittelua on alettu kehittämään jo 1700-luvulla, kun arkkitehtonista representaatiota alettiin haastaa. Taiteellismieliset arkkitehdit, kuten esimerkiksi Jean-Laurent Legeay oli merkittävä hahmo kehityksen aloituksessa. Legeay toimi opettajana ja hän painotti usein näkemystään oppilailleen, että suunnitelma ei ole valmis ilman vähintään yhtä kuvaavaa perspektiivipiirrosta suunnitelmasta.

Kunnon muutos kuitenkin tapahtui vasta 1900-luvun vaihteessa, kun mukaan tulivat valokuvat, abstrakti taide, impressionismi ja ekspressionismi. Teknologian ja taiteellisten näkemysten kehittyessä ja yhteiskunnallisten muutosten aikana, myös arkkitehtuuri alkoi mullistumaan. Impressionismin ja expressionismin kehitys auttoivat kuvan havainnoinnissa ja vaikutelman kuvaamisessa, jota kubinismi paransi luomalla kolmiulotteista näkymää kasiulotteiseen formaattiin.

3D-grafiikan aika lähtee liikkeelle noin 1900-luvun puolivälistä. Sketchpad oli ensimmäisiä graafisella käyttöliittymällä toimivan koneella toimiva mallinnusohjelmia, joka julkaistiin vuonna 1962. Tästä lähti teknologian kehitys, jonka myötä 3D-mallinnus nousi pintaan.

Kaupunkimallinnusta tehdään moniin erilaisiin tarkoituksiin ja on olemassa erilaisia tapoja mallintaa. Kaupunkimallinnus ei aina välttämättä ole 3D-mallinnusta vaan se voi, myös olla visualisointia ja graafista ilmettä. Hyvinä esimerkkeinä graafinen malli, vektorimalli, visualisaatio ja tietomalli.

Graafinen malli eli perinteinen 3D-malli. Nimitys graafinen malli on annettu, jotta se erottuisi paremmin, ja ettei sitä ei sekoitettaisi tietomallinnuksen kanssa. Graafiseen kaupunkimallinnukseen on runsaasti erilaisia ohjelmia ja niitä käytetään usein erilaisiin tarkoitusksiin. Osa on parempia visualisointiin, osa on parempia suunnitteluun ja osaa käytetään keilausaineiston käsittelyyn.

Vektorimallit eli toiselta nimeltään pistepilvi mallit syntyvät yleensä laserkeilauksen ja ilmakuvauksen lopputuloksena. Pistepilveä voidaan hyödyntää sellaisenaan kaupunkimallina tai yhdessä vektorimuotoisen aineiston kanssa. Tyypillistä pisteiden jalostamista on värjäys pysty ja/tai viistokuva-aineiston perusteella. Hyvä luokittelu auttaa myös pistepilven suodattamisessa. Useimmat sovellukset, kuten ArcGIS Pro kykenevät jo hyödyntämään monipuolisesti pistepilveä suunnittelu- ja visualisointitehtävissä. Pistepilveä voidaan tuottaa eri tavoin kuten mobiilikeilauksella, perinteisellä ilmakeilauksella tai UAV-järjestelmällä.

Visualisaatio ei ole 3D-mallintamista, mutta on silti oleellinen osa kaupunkimallinnusta. Visualisointi voi olla kuvia tai videoita malleista, joiden tarkoituksena on avustaa suunnittelua ja luoda keskustelua. Malleista otetut kuvat voi omalla tavallaan ajattella virtuaalisina ikkunoina ja videot tavallaan kehystettyinä hetkinä. Ne pystyvä kuvaamaan todellisuutta melko tarkasti.

Tietomallinnus on monimuotoinen mallinnustapa, joka ei välttämättä ole vaikeampi tai monimutkaisempi kuin graafinen mallinnus. Hyvänä esimerkkinä rakennusten tietomallit (Building Information Model, BIM). Tietomallinnus eroaa graafisesta mallinnuksesta siten, että tietomallit ovat hyvin paljon tarkempia ja informatiivisempia kuin graafiset. Graafinen malli voidaan ajatella representaationa, kun taas tietomalli voidaan ajatella simulaationa, sillä tietomalli tehdään lähes täydelliseksi kopioksi tulevasta mallista 3D-ympäristöön. Semanttisen

datan ansiosta tietomallilla pystyy kuvastamaan myös muutoksia ja reaktioita, se on täydellinen realististen testien tekemiseen.

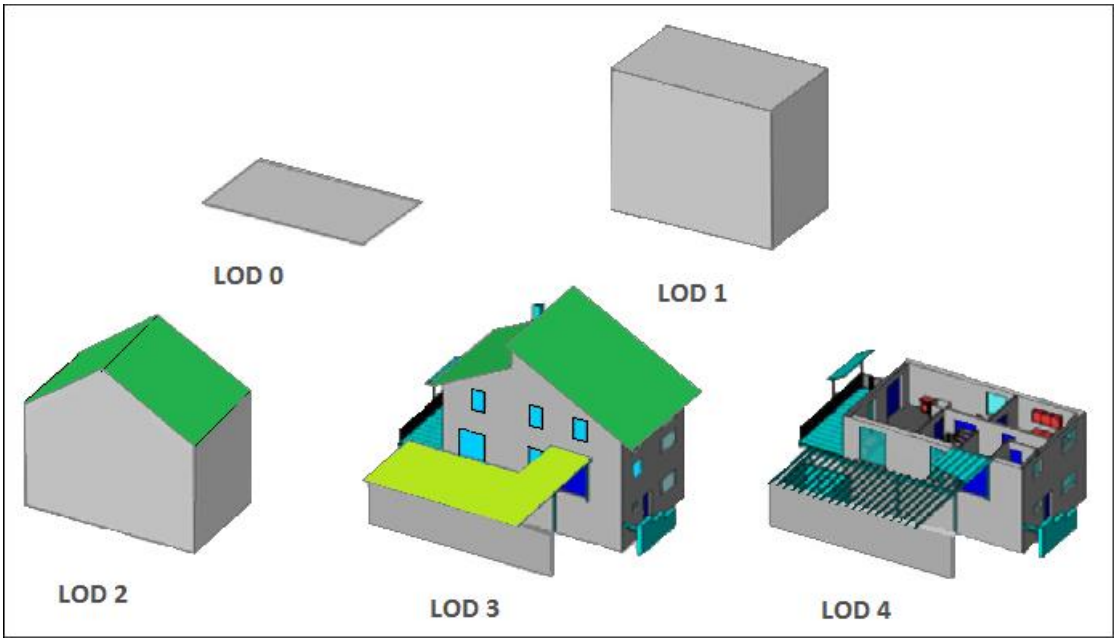
Edellä kerrottuun asiaan liittyen Valtteri Kettunen (2018) teki kyselytutkimuksen suomen kunnille vuonna 2018. Kettusen kyselyyn tuli vastauksia kolmestakymmenestä eri kunnasta. Kyselyn tuloksista käy ilmi, että suurimmalla osalla vastaajista oli käytössään vähintään LOD1-tason laatikkomalli ja seuraavaksi eniten LOD2-tason pintamalleja (Kettunen 2018.).

Kyselyyn vastanneista kolme käytetyintä kaupunkimallinnusohjelmistoa olivat Trimblen SketchUp ja Locus, sekä Terrasolid-ohjelmistot. Noin puolet vastaajista on julkaissut kaupunkimallin kaikille nähtäväksi, mutta vain pieni osa kaupungeista oli julkaissut sen avoimena aineistona. Tutkimuksessa käy myös ilmi, että suurimmat kehitysongelmat ovat riittävät resurssit ja mallin ylläpito. (Kettunen 2018.)

Lähteiden ja aiheeseen perehtymisen perusteella voidaan tulkita, että Suomen kunnat ja kaupungit kokevat kaupunkimallit pikkuhiljaa käteväksi osaksi kaupunkisuunnittelua, ja osa Suomen kaupungeista onkin tuottanut omia 3D-malleja. Suurimmat haasteet ovat osaamisen ja rahoituksen puute. Havaitsemieni esimerkkien perusteella kaupunkimalleja käytetään enimmäkseen suunnitteluun ja visualisointiin, sekä apuvälineenä vuorovaikutuksessa ja päätöksenteossa.

2.2 Tarkkuustasot

Tarkkuustasot eli LOD tulee englannin kielen sanoista Level Of Detail. Tarkkuustasot ovat tärkeä osa kaupunkimallinnusta. LOD-tarkkuustasoilla voidaan määritellä mallin tasoa ja yksityiskohtaisuutta (Kuva 1). Tarkkuustasoja on erilaisia, esimerkiksi tietomalleille on omat tarkkuustasot ja graafisille malleille omansa. Graafisille malleille tarkkuustasot on määritetty viiteen eri tasoon ja ne lukeutuvat nollasta neljään. Seuraavat LOD-tasojen kuvaukset olen koostanut useammasta eri lähteestä. (Viljanen 2016; Biljecki 2013; Harjuniemi 2015.)



Kuva 1: Level of detail

LOD0 voi olla yksinkertainen pohjapiirros tai esimerkiksi vain yksinkertainen maasto malli. Mallit yleensä ilmenevät 3D-ympäristössä tasaisina 2D-pintoina. Taso on vain tasainen pinta, vaikka mallissa olisikin mukana esim. korkeustiedot niin malli ei ole tarkoitettu 3D-malliksi.

LOD1 on yksinkertainen laatikkomalli, mikä vastaa rakennuksen korkeutta ja pohjamuotoa. Kaupunkimallina LOD1-tason kaupunki on kevyt ja helppo käsitellä, kaupungista saa myös helposti realistisen näköisen. Taso saattaa myös sisältää esimerkiksi yksinkertaisia puita ja pensaita.

LOD2 on talon ulkoisten muotojen yksinkertaistettu malli. Talossa on yleensä katto ja jotain yksinkertaisia viitteitä, kuten ulkoisia muotoja. Joissain tapauksissa malleissa saattaa olla ovia ja ikkunoita. Kaupunkimallina LOD2 on vähän raskaampi kuin LOD1, mutta on silti mukavan yksinkertainen ja toimii sulavasti esimerkiksi verkkoselaimessa.

LOD3 on kopio talosta ulkoapäin katsottuna. Talon muoto muuttuu hyvin paljon verrattuna LOD2-tason malliin. Talon ulkoiset mitat vastaavat oikeaa rakennusta ja taloon on lisätty ovien ja ikkunoiden lisäksi muitakin ulkoisia muotoja jotka vastaavat sitä kuvaavaa rakennusta. Joissain tapauksissa LOD3-malleilla on myös tehty sisäpuolelle muutoksia, esimerkiksi niin, että huoneet on tehty valmiiksi, mutta niitä ei ole kalustettu eikä niissä ole ovia. LOD3-tason

mallit ovat merkittävästi raskaampia kaupunkimalleissa, kuin LOD2-tason mallit.

LOD4 on kokonainen, tarkka malli talosta sisä- ja ulkopuolelta. Graafisena mallina talo on kalustettu ja teksturoitu näyttämään aidolta. Kaupunkimallinnuksessa käytetään hyvin harvoin LOD4-tason malleja, sillä ne ovat hyvin raskaita ja niissä on tietoa mikä ei ole tarpeellista yleisen kaupunkimallin kannalta. LOD4-tason malleja käytetään useinmiten peleissä tai esitelmissä.

2.3 Projektissa käytetyt ohjelmistot

Tässä kappaleessa esitellään lyhyesti projektissa käytetyt ohjelmistot ja tiedosto muodot. Tärkeimmät ohjelmistot olivat Esrin CityEngine ja Safe Softwarenn FME. Lisäksi kerrotaan mm. CGA-ohjelmointikielestä ja ArcGIS Pro -ohjelmasta.

CityEngine

CityEngine kuuluu Esrin tuoteperheeseen ja on tarkoitettu kaupunkimallintamiseen ja urbaanien maisemien visualisointiin. CityEngine on monipuolinen ohjelmisto, jotka toimii yhteen ArcGIS Pro:n kanssa (Wittner 2017; Wittner & Muller 2017.). CityEngine on tarkoitettu kaupunkien suunnitteluun ja kehittämiseen, ja tätä kehitystä avustavia työkaluja löytyy paljon(). Taloja pystyy muokkaamaan tavallisilla 3D-työkaluilla tai halutessaan koodilla. CityEnginessä on oma työtila koodin kirjoitusta varten, se tukee koodia kuten CGA ja Python. CityEngienssä työntekoa pystyy automatisoimaan ja kehittämään koodilla.

Erilaisten työkalujen lisäksi CityEngine sisältää erilaisia valmiita malleja esimekiksi katuvaloille ja aidoille, joilla voi helpottaa mallinnusprosessia sisäisiä malleja käyttämällä (Koverola 2014). CityEnginen käyttöä helpottaa myös se, että se lukee ja kirjoittaa monia erilaisia tedostomuotoja, kuten esimerkiksi Shapefilea. Shapefile (.shp) on vektoripohjainen tiedostomuoto, joka on luotu geospatiaalisen tiedon tallentamiseen erilaisissa paikkatietojärjestelmissä (Esri 1998). Shapefile on Esrin kehittämä ja toimii erinomaisesti myös ArcGIS Pros:ssa ja CityEngine:ssä.

CityEnginellä on oma ohjelmointikieli CGA (The Computer-Generated Architecture), josta kerrotaan Esrin verkkosivuilla (CGA refence 2019). Tietotekniikan opiskelijana on tullut opeteltua lukemaan ja kirjoittamaan muutamaakin ohjelmointikieltä. Kuitenkin on myönnettävä, että en ole vielä ennen tätä projektia päässyt tutustumaan CGA:n kaltaiseen koodiin. Seuraava CGA-kielen kuvaus perustuu omiin kokeiluihin ja havainnointeihin, sekä useammasta eri lähteestä löytyneeseen tietoon. (Geoff 2014; Viljanen 2016; Urban Analytics 2019; Wittner 2017; Esri CityEngine 2019.)

CGA on ohjelmointikielten joukossa hyvin erikoinen kieli, sillä CGA:ssa ei käytetä tavallisia funktionimiä, kuten If ja For. Funktioiden rakenne on myös hyvin erilainen esimerkiksi verrattuna C#-, JavaScript- tai Python- ohjelmointikieliin. Vaikka rakenne ja syntaksi ovat hyvin erilaisia verrattuna muihin kieliin, kuitenkin tutkiessani kieltä lisää, rupesin löytämään myös samankaltaisuuksia. CGA, kuten monet muutkin ohjelmointikielet, on tarkka ominaisuuksien järjestyksestä. Recursion-ominaisuus toimii samankaltaisesti kuin For vähän eri tekniikalla, mutta loppupelissä sama idea.

Rekursiolla on mahdollista luoda toistorakenteita, mutta toistorakenteet kaatavat CityEnginen herkästi. CityEnginellä on vain muutama toiminto mikä estää toistorakenteiden toimintaa – joten kannattaa olla varovainen, kun niitä tekee. Case-ominaisuus taas toimii melko samalla tavalla, kuin If funktio. Case-ominaisuus toimii Else-ominaisuuden kanssa.

Yritin tutkia, löytyykö CGA-koodille mitään vastaavaa koodia tai mitään koodia mikä toimisi samalla tavalla. Tutkiessani GML-koodia huomasin paljon yhteneviä piirteitä CGA:n kanssa. CGA vaikuttaa ottaneen vaikutteita GML-kielestä. Tämän takia voi olla myös mahdollista, että CityEnginellä tehdyt mallit toimivat paremmin CityGML-ohjelman kanssa.

ArcGIS Pro

Esrin verkkosivuilla kerrotaan (ArcGIS Pro 2020), että ArcGIS Pro on moderni paikkatieto-ohjelmisto työasemakäyttöön. Ohjelman edelläkävijä oli ArcMap. ArcGIS Pro on joissain asioissa edellä ArcMap-ohjelmistoa, mutta totuus on,

että ArcMap on toistaiseksi vielä parempi. ArcMap on hyvin hiottu ohjelmisto, jossa esiintyy melko vähän ongelmia tai bugeja.

ArcGIS Pro on luotu tehokkaaseen työskentelyyn, ohjelma vaatii melko paljon kapasiteettia tietokoneelta, jotta kaikkia ominaisuuksia voi käyttää tehokkaasti. Ohjelmaa kuitenkin päivitetään ja kehitetään koko ajan paremmaksi ja tehokkaammaksi. ArcGIS Pro sisältää tehokkaat ja kehittyneet toiminallisuudet tiedon visualisointiin, analysointiin ja hallintaan (Wittner 2017). Työkaluja on valmiiksi jo paljon, mutta ModelBuilder-työkalun ansiosta pystyy tekemään omia työkaluja ja nopeuttamaan toistuvien prosessien tekemistä. Tämä sama ominaisuus löytyi myös ArcMap:sta ja tämä helpottaa ModelBuilderin käyttöä ArcGIS Pro:ssa.

FME

FME tulee englannin kielen sanoista Feature Manipulation Engine ja on integraatiotyötila, joka tukee lukuisia erilaisia tiedostomuotoja ja mahdollistaa esimerkiksi tiedonsiirron automatisoinnin (Safe Software 2020). FME on monipuolinen ja laaja ohjelmisto, joka on yllättävän kevyt tietokoneen pyörittää, huolimatta siitä kuinka laaja ohjelmisto sen on.

FME:llä työskentely on kuin graafista koodaamista. Työtila on kuin tyhjä taulu, joka vain odottaa käyttäjän loogista mallinnusta. FME-työtilan avulla pystyy helpottamaan työskentelyä ja lyhentämään pitkiä käsin tehtäviä prosesseja. Se ei ole rajoitettu vain tietylle alalle sopivaksi, vaan toimii hyvin niin maanmittausosastolla, kuin lentoyhtiön tai vaikka myyntifirmankin käytössä.

3 AINEISTOT JA MENETELMÄT

Aineistot ja menetelmät-osio esittelee toimeksiantajan toimintaa ja tarvetta työlle. Siitä siirrytään itse työn tekoon ja alkuvalmisteluihin, kuten Shapefile-tiedoston luomiseen FME-työtilassa. Tämän jälkeen käsitellään itse mallinnusprosessia CityEnginellä, sekä CGA-kielen ominaisuuksia ja haasteita.

CityEngine-mallinnusprosessin kohdalla otin myös askeleen sivumpaan, ja tutkin mahdollisuuksia automatisoida laserkeilausaineistoa. Tutkimuksista selvinneet tulokset osoittivat, että laserkeilatusta aineistosta on mahdollista saada

automatisoitu prosessi, mutta mallien laatu ei vastannut haluttua. Toimeksiantajan kanssa todettiin, että kyseinen ratkaisu ei sovellu projektiin.

CityEnginellä mallinnuksen tutkimista jatkettiin, ja malleja testattiin myös ArcGIS Pro -työtilassa. Projekti eteni tämän jälkeen automatisointiprosessiin. Prosessissa selvitettiin automatisoinnin mahdollinen toteutustapa.

3.1 Vantaan kaupunki

Toimeksiantajana toimii Vantaan kaupungin kaupunkimittausosasto. Vantaan kaupunki sijaitsee Uudenmaan maakunnassa ja on väkiluvultaan Suomen neljänneksi suurin kaupunki. Vantaan naapurikuntia ovat Helsinki, Espoo, Nurmijärvi, Kerava, Tuusula ja Sipoo. Tilastojen mukaan Vantaa on kooltaan 240,35 km² ja väkiluku on väestörekisterin mukaan noin 233 383 asukasta eli väestön tiheys on noin 979,1 as./ km² (Vantaa tilastot ja tutkimukset 2020).

Kaupunkimittausosasto huolehtii mm. asemakaavan kiinteistöteknisestä toteutuksesta eli tonttijaot ja lohkomiset, kiinteistörekisterin ylläpitäminen ja kaupanvahvistuspalvelut, rakennusten paikkojen merkinnät ja sijaintikatselmukset kuuluvat kaikki kaupunkimittausosaston tehtäviin. Lisäksi kaupunkimittausosasto hoitaa maanalaisten johtojen sijaintitietopalveluita, karttojen, ilmakuvien, keilausaineistojen, kaupunkimallin ja kiinteistötietojen pitäminen ajantasaisina. Edellä mainittujen paikkatietojen jalostus tuotteiksi, sekä niiden myynti ja jakelu. (Maanmittauspalvelut 2020.)

Vantaan kaupungilla on 3D-kaupunkimalli, jonka mallinnetut talot ovat luokitukseltaan LOD1- ja LOD2-tason malleja. Vantaalla on oma karttapalvelu, missä pääsee tutkimaan erilaisia karttoja ja 3D-kaupunkimallia (Kettunen 2018). Kaupunkimallia päivitetään 3 vuoden välein laserkeilausaineiston pohjalta, kuitenkin kaupunki haluaisi päivittää malliaan useammin.

Tämä opinnäytetyö keskittyy nimenomaisesti keilausten välillä tapahtuvaan mallinnukseen. Tavoitteena on tutkia, voidaanko CityEngineä käyttää järkevästi vähintään LOD1-tason kaupunkimallin luomiseen niiden rakennusten

osalta, joita ei saada mallinnettua pistepilvikeräysten välillä, sekä perehtyä onko mahdollista luoda talojen mallinnuksesta automatisoitu prosessi.

3.2 Alkuvalmistelut ja materiaalit

Aineisto on Vantaan kaupungin keräämää tietoa, joka on tallennettu SQL-tietokantoihin. Käytetyt tietokannat sisältävät runsaasti erilaista dataa, josta vain pientä osaa haluamme käyttää tähän projektiin. Parhaaksi tiedon tallennusmuodoksi todettiin Shapefile-tiedosto. Tiedostoa varten tietoa tarvitaan kahdesta eri taulusta, tieto haetaan, sekä suodatetaan FME-työtilassa.

Rakennusten tietokantarakenne

Seuraavassa on listattu palvelutietokannasta haetut tarvittavat sisällöt ja niiden linkitykset. Kyseiset näkyvät taulut ovat osa Shapefile-tiedostoa. Listaus on ote toimeksiantajan tietokantakuvauksesta.

ESRI_palvelutietokanta:
RAKENNUS: abstrakti kohde, joka sisältää yksittäisten rakennuksen tiedot abstraktina oliona mikä ei sisällä geometriaa.

RAKENNUKSEN_OSA: 2D-geometria rakennuksesta. Sisältää geometriatiedon rakennuksen pohjapiirroksesta mitatun datan perusteella. Kohdeluokkina on itse rakennus, sekä rakennukseen kuuluvat osat, kuten terassi, katos, parveke ja niin edelleen. Rakennus ja sen osat ovat siis omia geometrioitaan.

RAKENNUKSEN_RUNGON_OSA: LOD1-muotoinen geometria rakennuksesta. Geometria esitetään rakennuksen osista (2D-geometrian) 3D korkeuteen pursotettuina geometrioina. Katot ja terassit ym. on "sulautettu" yhdeksi rakennukseksi, ja koko rakennus osineen on yksi solid tyyppinen geometria.

RAKENNUKSEN_RUNKO: LOD2-muotoinen geometria rakennuksesta. Linkittyy vain RAKENNUS-tauluun. Tuotettu laserkeilauksesta ensin CityGML:ksi ja konvertoitu ArcGIS:iin. Ei suoraa yhteyttä 2D-geometriaan samalla tavalla kuin LOD1-muotoisessa mallissa.

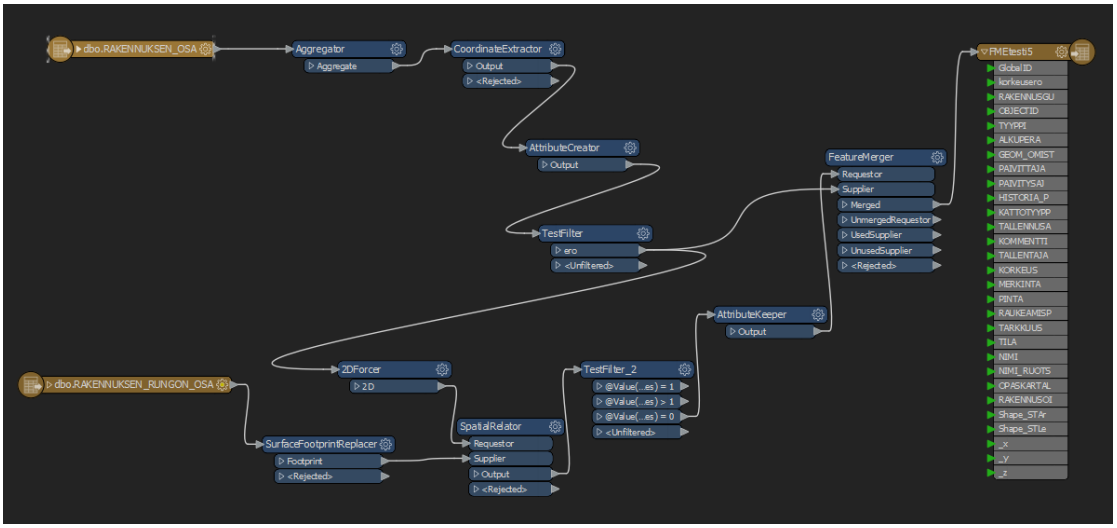
Linkitykset:
RAKENNUS -taulussa on GlobalID-tunnus jokaiselle kohteelle. Tämä tunnus on uniikki ja periytyy geometriaesityksille.

Kaikissa geometriatauluissa on RAKENNUSGUID-tunnus, ja lisäksi myös omat GlobalID:t. GlobalID:t yksilöivät näitä yksittäisiä geometriarepresentaatioita. Jokainen yksittäinen geometria kaikissa LOD-tasojen represtaatioitauluissa linkittyy RAKENNUS-tauluun, siten että RAKENNUSGUID vastaa kunkin rakennuksen RAKENNUS-taulun GlobalID:tä

Täten tuo RAKENNUS-taulu toimii yksittäisen rakennuksen "parent-tauluna", abstraktina oliona, jolla on linkitys kaikkiin olemassa oleviin geometriarepresentaatioihin GlobalID = RAKENNUSGUID avaimilla. Siten jokainen geometria tietää mihin rakennukseen se kuuluu.

Shapefile-tiedoston luonti FME-työtilassa

Shapefile-tiedosto luotiin käyttämällä FME-työtilaa (Kuva 2), koska tarkoituksena on luoda LOD1-tason malleja niiltä osin mistä niitä ei olla vielä luotu. FME-työtilassa haettiin ESRI_palvelutietokannasta kaikki siellä olevat LOD1-tason mallit, sekä kahdesta taulusta 2D-geometriat. RAKENNUKSEN_OSA-taulusta haetaan 2D-geometria ja RAKENNUKSEN_RUNGON_OSA-taulusta haetaan LOD1-mallit.



Kuva 2: Shapefile-tiedoston luonti FME-työtilassa

Näiden taulujen sisällöille tehtiin spatiaalinen vertailu, jonka avulla tunnistettiin sellaiset 2D-kohteet, joiden kohdalla ei ole LOD1-tason geometriaa. FME-työtilassa tehdyn suodatuksen jälkeen suodatettu Shapefile-tiedosto tallennetaan halutun kansioon, mistä sitä pääsee helpoiten käyttämään, ja näin saadaan tarvittava data projektia varten.

3.3 Talojen mallinnus

Talojen mallinnus aloitettiin käynnistämällä CityEngine-ohjelmisto ja luomalla tyhjä maisema menemällä tuonti asetuksiin. Työ alkoi tuomalla tyhjään maisemaan Shapefile-tiedosto. Tiedostoa tuodessa oli tärkeää muistaa asettaa oikea koordinaattijärjestelmä, se on tärkeää rakennusten sijaintitietojen sisällyttämisen vuoksi.

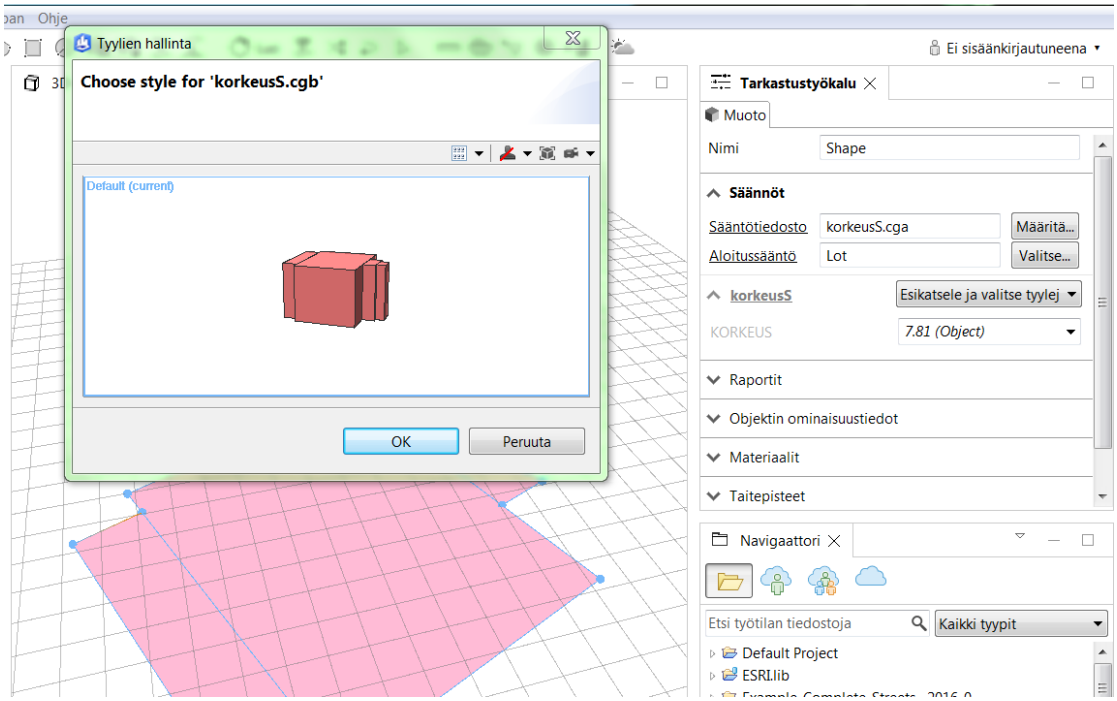
Tuonnin jälkeen talojen pohjat ilmaantuivat tyhjään maisemaan, missä niiden oletus väri oli valkoinen. Väri vaihdettiin, jotta niitä oli helpompi tutkia kauempaa. Talot olivat hyvin hajautettuja, kun ne tuotiin työtilaan. Joten niihin vaihdettiin kirkkaan vihreä väri, jotta ne erottuivat selkeästi.

Projektin kohdalla Shapefile-tiedoston kääntämisessä oli ollut ongelmia, joten tuonnin jälkeen varmistettiin, että tiedosto oli kääntynyt FME-työtilassa oikein, ja kaikki tarvittavat tiedot löytyivät objektin ominaisuustiedoista. Varmistuksessa huomattiin, että tiedot olivat oikeat, joten luotiin CGA-tiedosto.

Kuva 3: Objectin ominaisuustiedot ja siihen yhdistetty CGA-koodi

CGA-koodin sisältö lukee dataa suoraan shapefile-ominaisuustiedoista. Koodille annetaan arvo ’KORKEUS’, koska se on sama nimi kuin objektin ominaisuustiedoissa (Kuva 3). Koodi lukee objektin tiedoista rakennuksen korkeuden. Attribuutille ’KORKEUS’ annetulla arvolla ei ole väliä (Kuva3), numero

voisi olla yhtä lailla 2, 20, 50, 5 tai -10. Attribuutilla pitää vain olla aina jokin arvo annettuna. ’Lot’ toimii aloitussääntönä rakennukselle ja extrude() nostaa rakennuksen haluttuun korkeuteen (Kuva 3). Sulkujen sisällä löytyy korkeus attribuutti, mutta sen lisäksi ’world.up.flatTop’. Tämä ominaisuus nostaa rakennuksen suoraan maanmuodoista riippumatta.



Kuva 4: 2D:stä 3D

CGA-koodi lisättiin rakennusten sääntötietoihin ja rakennukset nostettiin tiedoissa olevaan oikeaan korkeuteen. Näin saatiin luotua muutamalla yksinkertaisella rivillä CGA-koodia LOD0-tason malleista LOD1-tason malleja. Mallit eivät toki muutu suoraan 3D-malleiksi, vaan koodin lisäyksen jälkeen pitää valita oletustyyli kohdasta ”Esikatsele ja valitse tyylejä…”. Siitä aukeaa uusi ikkuna, missä näkyy talosta luotu 3D-malli (Kuva 4). Painetaan vain ’Ok’ ja talo näkyy myös työtilassa 3D-mallina. Tapa on hieman kömpelö, mutta kun mallit on kerran nostettu, niin jokaisen työtilan käynnistyksen yhteydessä CityEngine kysyy, halutaanko että mallit nostetaan 3D-malleiksi.

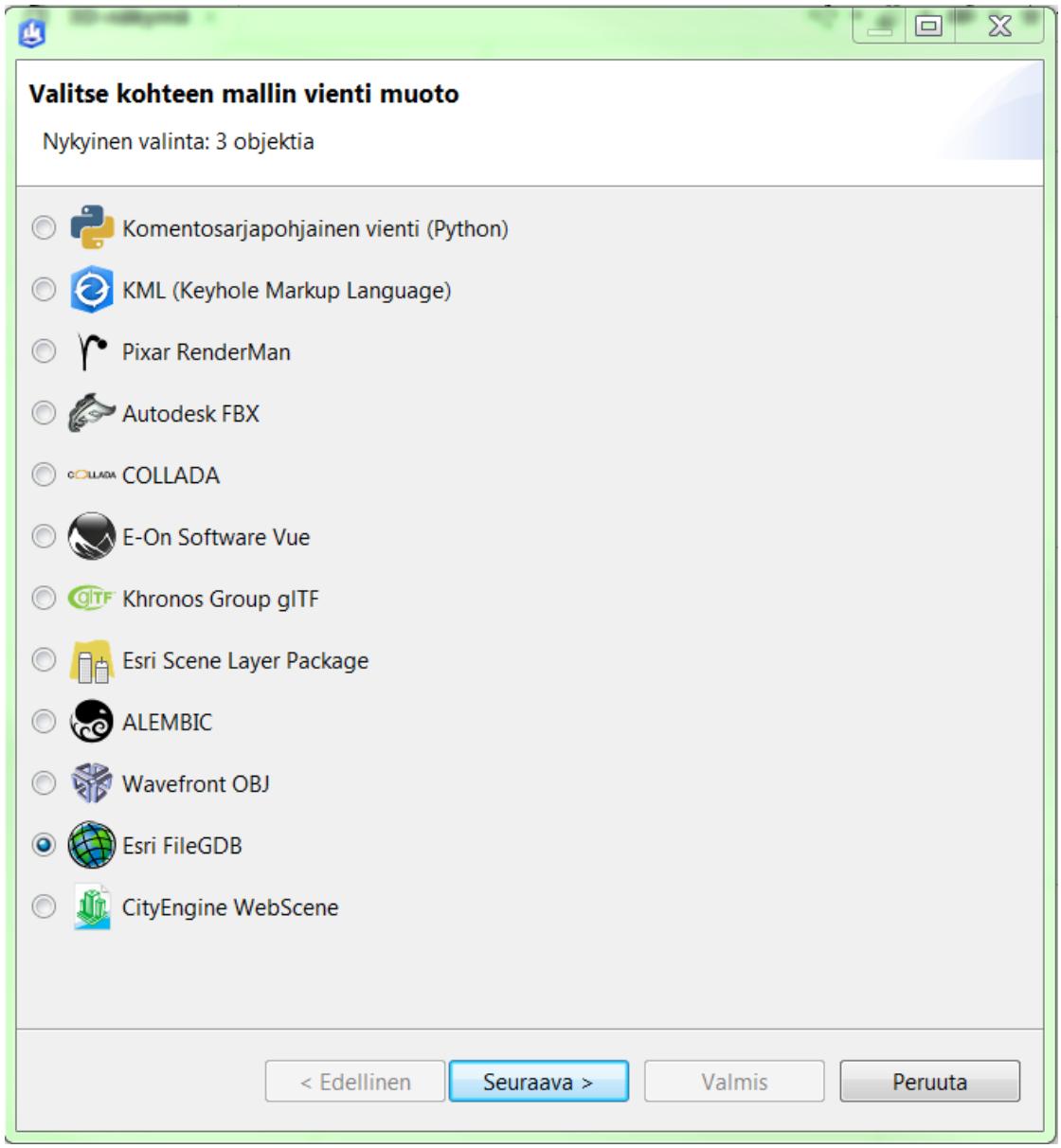
Samasta Shapefile-tiedostosta kokeiltiin myös luoda LOD2-tason malleja, mutta se osoittautui hiukan haastavammaksi kuin oli odotettu. Rakennuksista ei yksinkertaisesti ollut tarpeeksi tietoa, jotta rakennuksista olisi voinut luoda tarkkoja LOD2-malleja. Kokeilu kuitenkin osoitti, että se voisi olla mahdollista.

Shapefile-tiedostossa tuotujen talojen ominaisuustiedoissa (Kuva 3) näkyy kattotyyppi. Samalla tavalla, kun talot nostettiin LOD1-tasoon, saatiin luotua katot talojen päälle LOD2 kokeilussa. Huomattiin kuitenkin, että rakennusten korkeus ei ollut enää tämän jälkeen oikea. Netistä löytyi keskiarvoisen seinäkorkeus, tällöin pystyttiin laskemaan kerrosten lukumäärän ja paljonko seinän korkeus olisi ilman kattoa. Seinän kokonaiskorkeus saatiin laskettua, jolloin saatiin myös selville paljonko katon tulisi olla korkea. Katon korkeus ei kuitenkaan auttanut, sillä piti selvittää kattokulma. Jotta taloista tulisi oikean korkuisia laskettiin paljonko keskiarvoisen kattokulman tulisi olla. Tämä ratkaistiin vertaamalla annettua korkeutta siihen, minkä korkuisia kattokulmilla katoista tulisi (kuva 5).

```
version "2019.0"

attr KORKEUS = 10
attr Skorkeus = KORKEUS * 0.7


@Range ("Shed", "Gable", "Flat")
attr kattotp = "Shed"

@StartRule
Lot -->
    extrude(Skorkeus)
    comp (f) {top: Pinta | side:Wall}


Pinta -->
    case kattotp == "Shed" : roofShed(8, 0.4)
    case kattotp == "Gable" : roofGable(21, 0, 0)
    case kattotp == "Flat" : FlatRoof
    else:PrintRoof

FlatRoof -->
    extrude(KORKEUS-Skorkeus)

PrintRoof -->
    print("Katto on" + kattotp)
    extrude(KORKEUS-Skorkeus)
```

Kuva 5: Katon lisäys rakennuksiin

Vertailujen perusteella tultiin siihen tulokseen, että harjakattojen osalta keskiarvoinen kulma on 21 ja pulpettikattojen osalta kulma on 8. Näillä kulmilla osa rakennuksista saatiin täysin oikean korkuisiksi, osasta tuli vähän liian matalia ja osasta vähän liian korkeita, mutta yli tai ali mennyt ero ei ollut kuin enintään 20 cm. Tämän jälkeen talomallit vietiin ArcGIS Pro -ohjelmaan, jotta ne voitaisiin siitä siirtää eteenpäin. CityEnginessä on monia vaihtoehtoja, miten viedä malleja eri ohjelmiin ja mihin tiedostomuotoihin niitä voi kääntää (Kuva 6).



Kuva 6: CityEngine vientivaihtoehdot

Ensimmäisissä kokeiluissa testattiin melko lailla kaikki mahdolliset vienti vaihtoehdot mitkä saattaisivat toimia ArcGIS Pro:ssa, kuten:

- KML
- FBX
- GTiff
- ESLP (Esri Scene Layer Package)
- Esri File GDB

Joissain testeissä oli unohtunut, että koordinaattijärjestelmä piti asettaa viennin yhteydessä ja talot löytyivät aivan vääristä paikoista, eivätkä välttämättä olleet lainkaan kartalla. Tähän auttoi se, kun muutti vienti asetuksista, että talon tarkat koordinaatit pitää lukea. Näistä testatuista tietotyypeistä KML ja ESLP olivat ainoat, jotka ArcGIS Pro saatiin hyväksymään. Näistä kahdesta ainut, joka toimi kunnolla oli KML. KML-tiedostot saatiin toimimaan siten, että mallit ensin vietiin 2D-näkymään, josta ne vietiin 3D-sceneen.

ArcGIS Pro on monimuotoinen ja monitoiminen ohjelma, jota voi olla toisinaan haastavaa käyttää. Aiemmissa kokeiluissa huomattiin tapahtuneen virheitä, jotka johtuivat heikosta ohjelman ymmärryksestä. Suurin ongelma oli, että mallit oli yritetty viedä maisemaan, ei sceneen. Tästä seurasi se, että talot näkyivät kartalla vain punaisina neliöinä maanpinnassa. Kuitenkin tämä oli aikaisemmin saatu kierrettyä 2D-näkymän kautta, mutta se hidasti mallien tuontia hyvin merkittävästi. Uusien tietojen valossa ja paremmassa ymmärryksessä miten ArcGIS Pro toimii, ryhdyttiin tekemään uusia kokeiluja. KML toimi kuten aikaisemminkin, ei uusia ongelmia. ESLP ei tällä testi kerralla toiminut lainkaan. Seuraavaksi kokeilussa oli Esri File GDB ja joka toimi paremmin kuin oli odotettu.

Esri File GDB -mallien tuonti piti tehdä eri tavalla kuin KML-mallien tuonti. CityEnginessä Esri File GDB -tiedoston vienti loi kaksi Shapefile-tiedostoa. Ensimmäinen oli tavallinen Shapefile ja toinen oli Prosedually Generated Multipatches Shapefile (PGM Shapefile). Näiden tiedostojen ero oli siinä, että tavallinen Shapefile toi vain talojen pohjat ja talojen tiedot, mutta PGM Shapefile toi mallin ja sen tiedot. Mallit piti tuoda ArcGIS Pro :n ’Map’ välilehdeltä löytyvältä ’Add Data’ kohdan kautta. Add Data -kohdasta valittiin kohta ’data’, jonka jälkeen näytölle aukesi resurssinhallinta, josta pystyi valita halutun tiedoston.

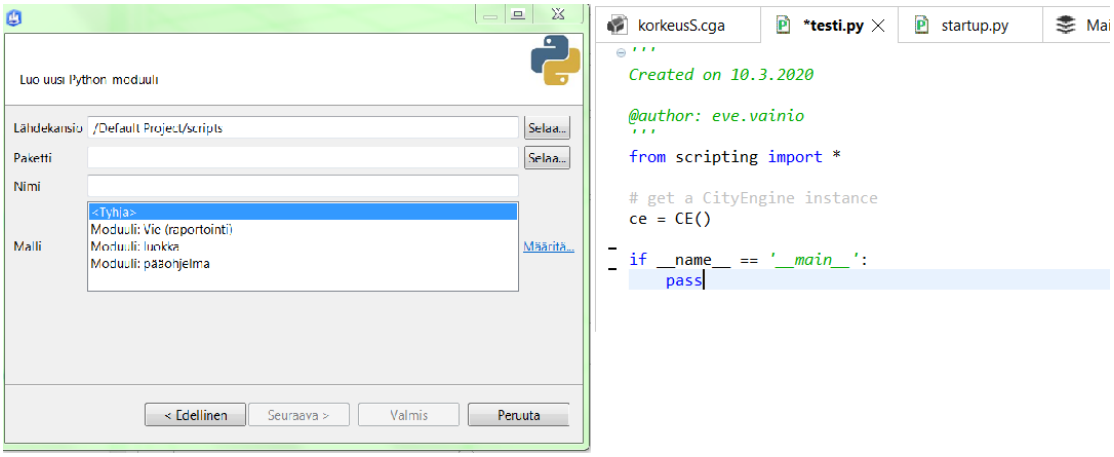
Huomattiin myös, että osa malleista oli väärässä korkeudessa, kun ne tuotiin ArcGIS Pro -ohjelmaan. Tämä ongelma pystyttiin korjaamaan helposti mallien asetuksista, kohdasta ’elevation’. Mallit piti vain asettaa maanpinnan korkeuteen, jotta ne tulivat näkyviin.

3.4 Prosessin automatisointi

Prosessin automatisoinnista oli puhuttu läpi koko projektin, ja se oli yksi työn päätarkoituksista. Ajatuksena oli myös, että automatisointi toteutettaisiin Python-koodia käyttäen, jos mahdollista. Tämä idea on osoittautunut toimivaksi. Python on kätevä ohjelmointi kieli, hyvin monipuolinen ja yksinkertainen. Python on myös yhteensopiva CityEnginen ja ArcGIS Pro :n kanssa.

Netistä löytyi hyvin paljon ohjeita Python-ohjelmointiin ja reilusti erilaisia tutoriaaleja. Kuitenkin todelliset vastaukset osoittautuivat löytyvän CityEnginen omista ohjeista. Esri on luonut CityEnginen Python ohjelmointiin tutoriaalin, ’Tutorial 10: Python programming’, josta löytyy mm. ohje kuinka mallit saa exportattua KML-muotoon. Tämän projektin automatisointi on toteutettu kyseisen tutoriaalin pohjalta (Tutorial 10: Python programming 2019). Vaikka Esrin tutorial 10 -ohjetta on noudatettu, niin koodiin kuitenkin tehtiin muutoksia. Projektin osalta oli päätetty, että mallit halutaan FGDB (Esri File GeoDataBase) muotoon, joten piti etsiä CityEnginen ohjeista keino muuttaa vientitoimintoa eri muotoon (Offline-ohjekirja- ja käyttöopas 2019). CityEnginen ohjekirjasto antaa hyvin rajallisen määrän tietoa ja ei sisällä esimerkkejä, mikä vaikeutti koodin ymmärtämisestä.

Automatisointi aloitettiin miettimällä mitä koodiin tarvitaan ja mitä sen halutaan tekevän. Koodin haluttiin valitsevan Shapefile-tiedostossa tuodut talojen pohjat, lisäävän niihin jo aikaisemmin tehty CGA-koodi ja luovan pohjat talomalleiksi. Tämän jälkeen koodin haluttaan valitsevan mallit ja exporttaavan ne FGDB-muotoon, mistä ne voidaan sittemmin siirtää ArcGIS Pro:n työtilaan.



Kuva 7: Python-tiedoston luonti

Koodin teko aloitettiin luomalla uusi Python-moduuli CityEngine projekti kansioon (Kuva 7). Tiedoston malliksi valittiin ’Moduuli: Pääohjelma’ ja sille annettiin nimeksi testi.py. CityEngineen on kehitetty oma alusta koodin kirjoitusta ja juoksuttamista varten. Vasempaan reunaan aukeaa ikkuna koodin kirjoitusta varten ja oikeasta reunasta saa konsolin auki. Nämä sijainnit vaihtelevat sen mukaan, miten ne halutaan asetella työtilaan.

```
select = ce.setSelection(ce.getObjectsFrom(ce.scene, ce.withName("*Muodot*")))

#If takertuu valittuihin taloihin, antaa niille sääntötiedoston ja
#nostaa rakennukset 3D malleiksi.
#varmistukseksi on laitettu else joka keroo jos jostain syystä tapahtuu error

if ce.withName("*Shape*"):
    ce.setRuleFile(ce.selection(), 'korkeusS.cga')
    ce.generateModels(ce.selection())
else:
    print" Ei oikea muoto"
```

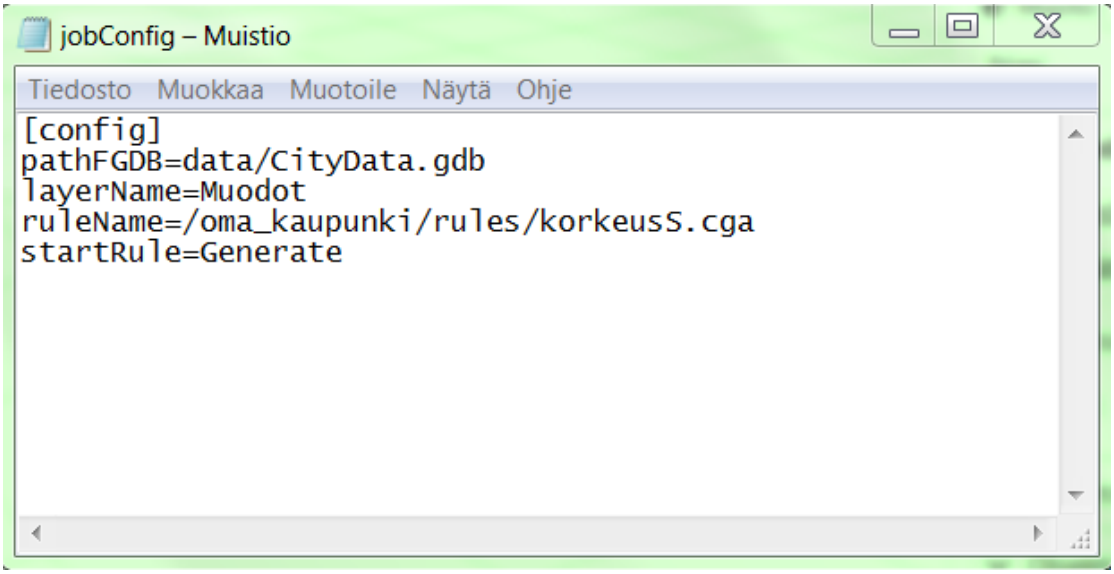
Kuva 8: Koodi valitsee scenen, lisää säännön ja tekee niistä malleja

Koodin ensimmäinen osuus (Kuva 8) valitsee työtilassa (scenessä ’Muodot’) olevat objektit. If-lause suodattaa niistä objektit, joiden nimi on ’Shape’ ja lisää niihin CGA-säännön ja nostaa niistä 3D malleja. Koodin saa testattua yläreunassa olevan valikon kohdasta ’Komentosarjat’. Pitää vain valita haluttu Python-tiedosto. Koodin testausta varten on myös pikanäppäin F9.

```
Created on 10.3.2020

@author: eve.vainio
'''
from scripting import *

# get a CityEngine instance
ce = CE()

# https://doc.arcgis.com/en/cityengine/2019.0/tutorials/tutorial-10-python-scripting.htm


def fgdbToKml(pathFGDB,layerName,ruleName,startRule = "Generate"):
    # open scene in the automation project
    ce.openFile('/oma_kaupunki/scenes/Maisema2.cej')

    # assign rule file based on the layer name
    select = ce.setSelection(ce.getObjectsFrom(ce.scene, ce.withName("*Muodot*")))

    #If takertuu valittuihin taloihin, antaa niille sääntötiedoston ja
    #nostaa rakennukset 3D malleiksi.
    #varmistukseksi on laitettu else joka keroo jos jostain syystä tapahtuu error

    if ce.withName("*Shape*"):
        ce.setRuleFile(ce.selection(), 'korkeusS.cga')
        ce.generateModels(ce.selection())
    else:
        print" Ei oikea muoto"

    # export models to FGDB
    exportSettings = FGDBExportModelSettings()
    exportSettings.setOutputPath(ce.toFSPath("/oma_kaupunki/models/kooditestidata") + "/")
    exportSettings.setGeodatabaseName("CityData.gdb")
    exportSettings.setTerrainLayers(exportSettings.TERRAIN_NONE)
    ce.export(ce.selection()[0], exportSettings)

     # close CityEngine
    ce.waitForUIIdle()
    ce.closeFile()
```

Kuva 9: Export-koodi osa1

Toinen osuus koodista eli export, on tehty kuten jo aikaisemmin mainitsin CityEnginen Tutorial 10 -pohjalta. Tein koodiin omia muutoksia, mutta suurimmilta osin se on Esrin koodia. Koodi pakkaa mallit FGDB -muotoon ja tallentaa ne ’kooditestidata’ nimiseen kansioon, jonka jälkeen koodi sulkee CityEnginen. Koodin (Kuva 9) saa ajettua suoraan CityEnginen sisällä ja se toimii loistavasti ilman virheilmoitusta. Mallit sai haettua ’kooditestidata’ kansiosta ja vietyä ArcGIS Pro -työtilaan. Työtilassa mallit ilmaantuivat kuitenkin jostain syystä väärässä korkeudessa. Ongelma saatiin kuitenkin korjattua sillä, että mentiin mallien asetuksiin ja korjattiin ’elevation’ kohdasta korkeus maanpinnalle.



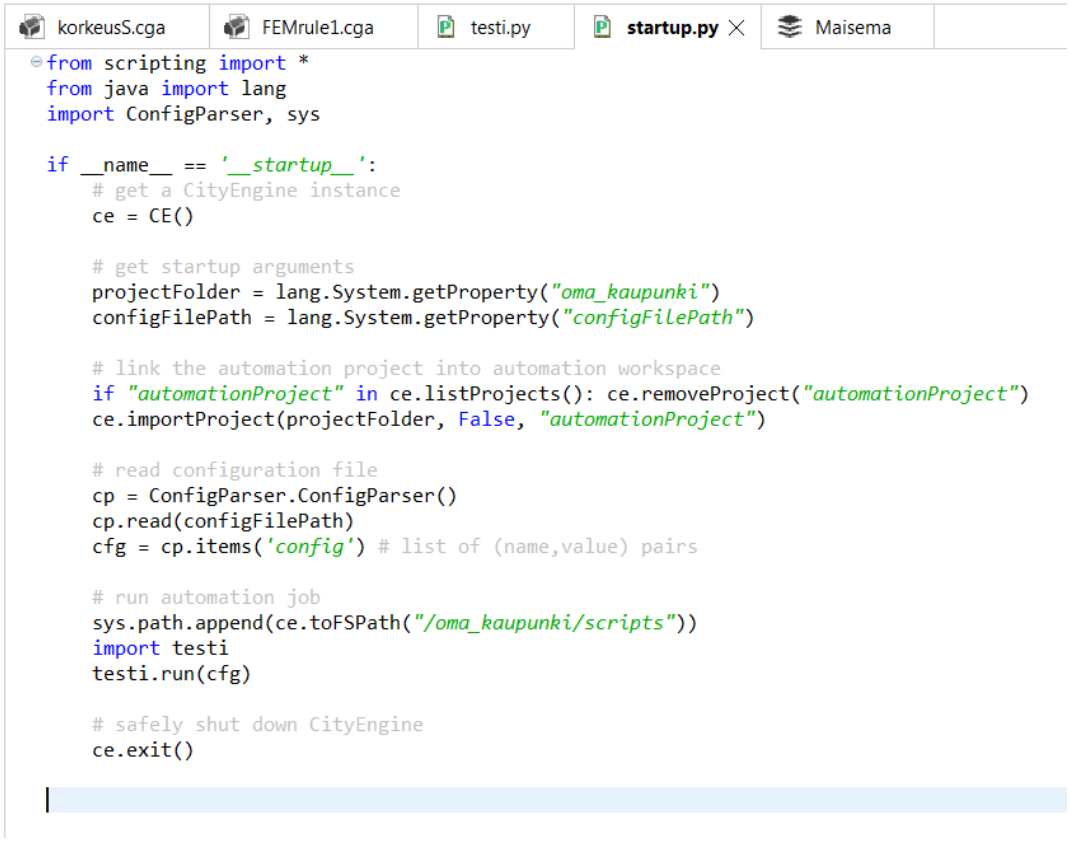
Kuva 10: jobConfig.cfg

JobConfig.cfg on osa koodia (Kuva 10), jonka tehtävänä on toimia linkkinä kahden Python-koodin välillä eli testi.py ja startup.py. JobConfig sisältää ohjeistuksen startup.py koodille ja kertoo sille polut, mistä etsiä CGA-sääntö, mistä etsiä geo databas, mikä on halutun scenen nimi ja kertoo sille aloitus sääntö. Kun jobConfig.cfg on saatu luotua ja nimettyä oikein, niin testi.py koodiin lisätään (kuva 9) kaksi funktiota (kuva 11), näiden funktioiden tehtävänä on yhdistää jobConfig-tiedoston testi.py koodiin.

```
    ce.closeFile()

def getCfgValue(cfg,name):
    for c in cfg:
        if  c[0] == name: return c[1]
    return None

def run(cfg):
    pathFGDB = getCfgValue(cfg,'pathfgdb')
    layerName = getCfgValue(cfg,'layername')
    ruleName = getCfgValue(cfg,'rulename')
    startRule = getCfgValue(cfg,'startrule')

    fgdbToKml(pathFGDB, layerName, ruleName, startRule)


if __name__ == '__main__':
    fgdbToKml("data/CityData.gdb", "NewShapes", "/oma_kaupunki/rules/korkeusS.cga", "Generate")
    pass
```

Kuva 11: Export-koodi osa2

Esrin ohjeiden mukaan Startup.py nimensä mukaisesti (Kuva 12) kuuluu käynnistyä samaa aikaa, kuin työtila missä se käynnistetään. Startup.py tiedosto toimii linkkinä työtilojen välillä ja välittää tehtävän eteenpäin testi.py koodille. Kun koodit on käsitelty ja FGDB-tiedosto on valmis, niin CityEnginen sulkee automaattisesti työtilan ja sammuttaa ohjelmiston.



Kuva 12: Export-koodi osa3 startup.py

Koodin seuraava osuus käsittelee sen käynnistystä komentoriviltä. CityEnginen tutoriaali 10 ohjeistaa kuinka tämän kuuluisi toimia. Alapuolella näkyvä polku ei ole tutoriaalista, mutta on hyvin saman kaltainen. Alapuolella näkyvä polku (Kuva 13) lisätään komentoriville ja aktivoi startup.py koodin, sekä ohjeistaa koodia jobConfig.cfg tiedostoon, josta se hakee yhteyden testi.py tiedostoon.

```
"C:\Program Files\Esri\CityEngine2019.0\CityEngine.exe" -data "C:\Apps\Automation_Workspace" -vmargs -DprojectFolder="C:\Apps\oma_kaupunki" -DconfigFilePath="C:\Apps\oma_kaupunki\data\jobConfig.cfg"
```

Kuva 13: Tiedostopolku

Valitettavasti projektin automatisointi ei ole vielä täysin valmis, sillä koodi pysähtyy jonkinlaiseen Python-koodissa tapahtuvaan virheeseen. Kuitenkin testasimme toimeksiantajan kanssa, että kyseinen Tutorial 10 -koodi toimii ja sitä hyödyntämällä on mahdollista saada työ valmiiksi. Olimme kuitenkin yhtä mieltä toimeksiantajan kanssa, että Tutorial 10 -ohjeistuksessa on puutteita.

Vaikka automatisointia ei vielä saatu loppuun, projektissa on kuitenkin todistettu, että automatisointi prosessin luominen on mahdollista ja kehitys työtä jatketaan vielä. Todennäköisesti kun automatisointikoodi saadaan toimimaan, niin se tullaan jatkossa toteuttamaan serverikoneella ja bat-ajastuksella (Rousku, K 2000). Bat:iin lisätään käskyjä, jotka tulisivat automaattisesti käynnistämään FME-työtilan prosessin ja tallentaisi päivitetyn Shapefile-tiedoston suoraan haluttuun kansioon, mistä voidaan käynnistää CityEngine-työtilan ja mahdollisesti vielä automatisoida valmiiden mallien vienti FME-työtilan kautta tuotantokantaan. Asian jatkotutkimuksen raportoinnin ei katsottu olevan relevanttia.

4 JOHTOPÄÄTÖKSET

Kaiken kaikkiaan Vantaan kaupungin tilanne kaupunkimallin kanssa on todella hyvä, kaupunki on paljon edellä monia suomen kaupunkeja/kuntia. Vantaan kaupunki laserkeilaa kaupunkia 3 vuoden välein ja päivittää mallia aina uuden tiedon tai keinon valossa. Kuitenkin kaupunkimallin talojen tekeminen ei välttämättä vaatisi edes laserdataa.

Talot voitaisiin mallintaa suoraan CityEnginessä pelkkien tietokantaan tallennettujen tietojen pohjalta, joita kaupunki kerää kaikista taloista. Ajantasaisia malleja voitaisiin saada käytännössä sitä mukaa mitä tietokantaan lisätään taloista tietoa. Tällä samalla idealla voitaisiin, myös luoda helposti LOD2-tason malleja, millä saisi helposti ilmettä kaupunkimalliin. CGA-koodin rajallisuus riippuu vain annetusta datasta.

Projektin aikana opiskelin myös laserkeilausaineistoa ja pistepilvimallinnusta. Tutkin pystyisikö laserkeilausaineiston mallinnusprosessia automatisoimaan tai helpottamaan mitenkään. Huomasin myös että mallinnuksen automatisointi olisi mahdollista mm. ArcGIS Pro:ssa olevalla ’ModelBuilder’ työtilalla. Kuitenkin asiasta keskusteltiin toimeksiantajan kanssa, että mallien laatu ei ollut halutun kaltaista, jollin todettiin ettei asiaa kannata tutkia enempää, joten sitä ei myöskään lisätty opinnäytetyöhön.

Automatisointiosuus oli mielestäni mielenkiintoisin osa projektia. Koin että siinä oli riittävästi haastetta ja oli hienoa päästä oppimaan uusi ohjelmointikieli. Vaikka automatisointi osio ei ole vielä täysin valmis ja työtä jatketaan vielä, niin olen iloinen tähänkin asti saavutetuista tuloksista.

Projektissa oli muutama haastava osuus, joista ensimmäinen ja vaikein oli opeteltavien asioiden määrä ja tarpeellisen tiedon tunnistaminen ja löytäminen. Toinen haastava osuus oli CGA-koodi. CGA on mielenkiintoinen ja hyvin erilainen koodi, jonka ymmärtämisessä meni pitkä aika ja tarpeellisen tiedon löytäminen vei vielä vähän pidemmän. Kolmas haastava osuus oli Python. CityEnginellä Python ohjelmointiin liittyviä ohjeita ei meinaa löytää oikeastaan lainkaan. Tiedon löytämisessä menee kauan ja saattaa olla että et silti löydä etsimääsi. Parhaiten tietoa pystyi löytämään CityEnginen offline-ohjeista.

LÄHTEET

Biljecki, F. 2013. The concept of level of detail in 3D city models. Delft University of Technology. PhD Research Proposal. PDF-dokumentti. Saatavilla: http://www.gdmc.nl/publications/reports/GISt62.pdf[viitattu 8.4.2020]

Dahmen, C. 2018. Data Conversion to I3S foe 3D Modeling from CityGML. PDF-dokumentti Saatavilla: http://proceedings.esri.com/library/user-conf/proc17/tech-workshops/tw_2582-457.pdf [viitattu 15.1.2020]

Esri Finland. 2020. ArcGIS Pro. WWW-dokumentti. Saatavilla: https://www.esri.fi/fi-fi/tuotteet/arcgis-pro/yleiskuvaus[viitattu 24.2.2020]

Esri CityEngine. 2019. CGA reference. WWW- dokumentti. Saatavilla: https://doc.arcgis.com/en/cityengine/2019.0/cga/cityengine-cga-introduction.htm[viitattu 18.12.2020]

Esri. 1998. Esri Shapefile techical description. PDF-dokumentti. Saatavilla: https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf[viitattu 2.5.2020]

Esri. 2018. Tutorial 10: Python scripting. Automate CityEngine tasks with startup.py. WWW-dokumentti. Saatavilla: https://doc.arcgis.com/en/cityengine/2019.0/tutorials/tutorial-10-python-scripting.htm[viitattu 9.4.2020]

Esri R&D Center Zurich. 2019. Offline-ohjekirja- ja käyttöopas. CityEngine Help. [viitattu 15.12.2020]

Geoff, T. 2014. Creating a Smart 3D City Model from Start to Finish. Julkaistu 27.3.2014. Youtube video. Saatavilla: https://www.youtube.com/watch?v=MxT39BK1_iU[viitattu 4.2.2020]

Harjuniemi, P. 2015. 3D-Kaupunkimallin tuottaminen ja hyödyntäminen yrityksissä. PDF-dokumentti. Saatavilla: https://www.theseus.fi/bitstream/handle/10024/94580/Harjuniemi_Paivi.pdf?sequence=1&isAllowed=y[viitattu 12.3.2020]

Kettunen, V. 2018. 3D-kaupunkimallinnuksen nykytila kunnissa kyselytutkimuksen perusteella. Metropolian Ammattikorkeakoulu. PDF-dokumentti. Saatavilla: https://www.theseus.fi/bitstream/handle/10024/148497/Kettunen_Valteri.pdf?sequence=1&isAllowed=y[viitattu 2.4.2020]

Koverola, O. 2014. Karttapohjainen kaupunkimallinnus CityEnginellä. Mikkelin Ammattikorkeakoulu. PDF-dokumentti. Saatavilla: https://www.theseus.fi/bitstream/handle/10024/79765/Koverola_Olli-Pekka.pdf?sequence=1&isAllowed=y[viitattu 12.3.2020]

Laatunen, K. 2015. Kaupunkimallit mahdollistavat teknologioiden ja osaamisresurssien tuottavan hyödyntämisen. Verkkolehti. Saatavilla: https://www.rakennuslehti.fi/blogit/kaupunkimallit-mahdollistavat-teknologioiden-ja-osaamisresurssien-tuottavan-hyodyntamisen/[viitattu 12.2.2020]

Lappalainen, P. 2016. BuildingSmart. Kapunkimallinnuksen ohjekirja. WWW-dokumentti. Saatavilla: https://buildingsmart.fi/kaupunki/kaupunkimallinnuksen-ohjekirja/ [viitattu15.12.2019]

Safe Software. 2020. FME- The Simple Solutions for Complex Integration. WWW-dokumentti. Saatavilla: https://www.safe.com/[viitattu 1.5.2020]

Rousku, K. 2000. Pelastava mausta laatikko. Verkkolehti. PDF-dokumentti. Saatavilla: http://mikropc.net/nettilehti/pdf/pc0709200062.pdf[viitattu 29.4.2020]

Tietoa Vantaan väestöstä. Vantaa, Tilastot ja tutkimukset. WWW-dokumentti. Päivitetty 27.1.2020. Saatavilla: https://www.vantaa.fi/hallinto_ja_talous/tietoa_vantaasta/tilastot_ja_tutkimukset/vaesto_ja_ennuste[viitattu 17.1.2020]

Urban Analytics. 2019. T5: Learn to code with CityEngine. Julkaistu 24.4.2019. Youtube video. Saatavilla: https://www.youtube.com/watch?v=BddSNl8aWas[viitattu 12.1.2020]

Vantaa. 2020. Maanmittauspalvelut. WWW-dokumentti. Saatavilla: https://www.vantaa.fi/asuminen_ja_ymparisto/rakentaminen/maanmittauspalvelut [viitattu 24.4.2020]

Viljanen, K. 2016. CityEngine- ja SketchUp-mallinnusohjelmien käyttö Porin kaupungin suunnittelussa. Lapin Ammattikorkeakoulu. PDF-dokumentti. Saatavilla: https://www.theseus.fi/bitstream/handle/10024/104647/opinnayte-tyo.pdf?sequence=1&isAllowed=y [viitattu 12.1.2020]

Virolainen, J. 2018. Vantaan 3D-kaupunkimallin ylläpito tietokannassa. Metropolian Ammattikorkeakoulu. PDF-dokumentti. Saatavilla: https://www.theseus.fi/bitstream/handle/10024/149369/Virolainen_Jere.pdf?sequence=1&isAllowed=y[viitattu 12.1.2020]

Wittner, E. 2017. ArcGIS Pro and CityEngine. PDF-dokumentti. Saatavilla: https://proceedings.esri.com/library/userconf/proc17/tech-workshops/tw_672-205.pdf [viitattu 17.3.2020]

Wittner, E. & Muller, P. 2017. Creating Rule Packages for ArcGIS pro and CityEngine with CGA. Youtube video. Saatavilla: https://www.youtube.com/watch?v=6oiiXkwnnJc [viitattu 12.1.2020]