Sisällys

Johdanto 2
1. CityGML-lähdeaineiston valinta 3
1.1 Berliini 4
1.2 Helsinki 5
1.3 Montreal 6
1.4 Rotterdam 8
1.5 Valittu aineisto 9
2. Ohjelmistot ja menetelmät CityGML:n muuntamiseen 12
2.1 CityGML2OBJs 12
2.2 FME Desktop 12
2.2.1 FME Quick Translator 12
2.2.2 FME Workbench 13
2.3 3D City Database 18
2.3.1 Asentaminen 18
2.3.2 CityGML-tuonti 21
2.3.3 Collada-viennin asetukset 25
2.3.4 Collada-vienti 26
2.3.5 Viennin lopputulos 27
3. Visualisointiympäristön luominen 30
3.1 Collada-tiedostojen muuntaminen FBX-tiedostoiksi 30
3.2 Tiedostojen käsittely 3D-mallinnusohjelmassa 31
3.2.1 FBX-tiedostojen tuonti mallinnusohjelmaan 31
3.3 Tekstuurien leipominen 36
3.3.1 Johdanto 36
3.3.2 Kopion tekeminen ja siivoaminen 37
3.3.3 UV-koordinaattien generoiminen 39
3.3.4 Väri-informaation leipominen 41
3.4 Tekstuurien valaistusinformaation poistaminen 46
3.4.1 Erikoistekstuurien leivonta valaistuksenpoistotyökalua varten 46
3.4.2 Valaistuksenpoisto 52
3.4.3 Lopputuloksen arviointi 54
4. Esimerkkivisualisoinnit 57

4.1 3ds Max -versio 57
4.2 Unreal Engine 4 -versio 65
5. Loppusanat 72

Johdanto

Tämä dokumentti on osa Kira-digi kärkihankkeeseen kuuluvan kokeiluhankkeen ”Kaupunkivisualisoinnin kehitysprosessin vauhdittaminen avoimen kaupunkidatan avulla” dokumentointia. Kokeiluhankkeen toteuttaja on Raahen kaupunki, Hanketoiminta ja kehittäminen -yksikkö. Kokeilussa tutkittiin CityGML-datan hyödynnettävyyttä 3D-kaupunkivisualisoinnin ympäristön luomisessa.

Dokumentti sisältää tietoa hankkeen aikana kokeilluista asioista. Mukana on sekä ohjeita että havaintoja. Dokumenttia tulee ajatella vapaasti kirjoitetun muistion kaltaisena – sitä ei ole viimeistelty rakenteeltaan tai sisällöltään vastaamaan esimerkiksi julkaisukelpoista tutkimusta.

Dokumentin tarkoituksena on tuoda esille hankkeessa kokeillut asiat niin, että muutkin voivat ottaa oppia hankkeen havainnoista – niin ratkaisuista kuin ongelmista.

Hankkeessa on luotu erikseen lyhyempi ohjeistus, jossa käsitellään CityGML-datan tuonti visualisointiohjelmiin. Lisäksi on asetettu saataville kaksi avointa esimerkkivisualisointia, joista toinen on toteutettu Unreal Engine 4 -alustalla ja toinen Autodesk 3ds Max -ohjelmalla. Kaikki dokumentit ja aineistot on saatavilla avoindata.fi-sivustolta.

1. CityGML-lähdeaineiston valinta

Lähdeaineiston valinnassa on selvää, että CityGML-data pitää olla teksturoitua ja vähintään LOD2-tasoista. Heikomman lähdeaineiston (LOD0, LOD1) käytössä ei ole järkeä, koska vastaavia yksinkertaisia laatikoita voidaan helposti tehdä ilmaortokuvien perusteella suoraan 3D-mallinnusohjelmassa. Teksturoiduissa LOD2 ja paremman tasoisissa rakennuksissa on huomattavasti enemmän informaatiota, jota voidaan hyödyntää visuaalisen taustaympäristön luonnissa.

Osoitteesta "citygml.org/3dcities/" löytyy virallinen listaus kaupungeista, joista on tarjolla avointa CityGML-dataa. Kun näistä karsitaan alle LOD2-tasoiset ja teksturoimattomat, jää jäljelle Berliini, Montreal, Helsinki ja Rotterdam.

Näitä vaihtoehtoja haluttiin verrata visuaalisesti. Mielenkiinnon kohteena oli erityisesti tekstuuriprojektioiden laatu.

CityGML-tiedostoja tarkasteltiin ilmaisella Elyx 3D -ohjelmalla.

1.1 Berliini

Berliinin aineiston pääsee lataamaan osoitteesta www.businesslocationcenter.de/en/downloadportal. Aineiston valintaan ja lataukseen on tehty käytännöllinen työkalu, jossa voi vapaasti valita ladattavan alueen ja aineiston formaatin.

Berliinin kaupunkidata noudattaa CityGML 1.0 -standardia. Se ei sisällä maastonmuotoja, ainoastaan rakennukset.

Visuaalisesti rakennusten teksturointi on vallitsevaan tasoon nähden kohtuullisen hyvä. Suurin osa tekstuureista on riittävän teräviä, jotta niistä erottaa selkeästi yksityiskohdat kuten pienet ikkunat, ovet, parvekkeet ja mainoskyltit. Ilma- ja satelliittikuvista muodostetulle aineistolle tyypilliseen tapaan rakennusten seiniin piirtyy puiden latvoja. Tekstuureille ei ole tehty minkäänlaista valaistuksen korjausta – riippuen kuvaushetkellä vallinneesta säästä, aineistoon on tallentunut voimakkaita varjoja ja valon kontrastia.

Berliinin 3D-kaupunkidataa

1.2 Helsinki

Helsingin aineiston saa hankittua samaan tapaan katselusovelluksen kautta kuin Berliinin aineiston, osoitteesta kartta.hel.fi/3d/.

Helsingin aineisto noudattaa uusinta CityGML 2.0 -standardia. Se ei sisällä maastonmuotoja, ainoastaan rakennukset.

Rakennusten tekstuuriprojektiot ovat kohtuullisen hyviä. Yksityiskohdat on pääasiassa erotettavissa. Puiden latvat piirtyvät rakennusten seiniin. Valaistusta ei ole poistettu, mutta tekstuurien värit ovat tasaisemmat kuin Berliinin aineistossa. Valkoisten rakennusten puhki palaminen on ongelma.

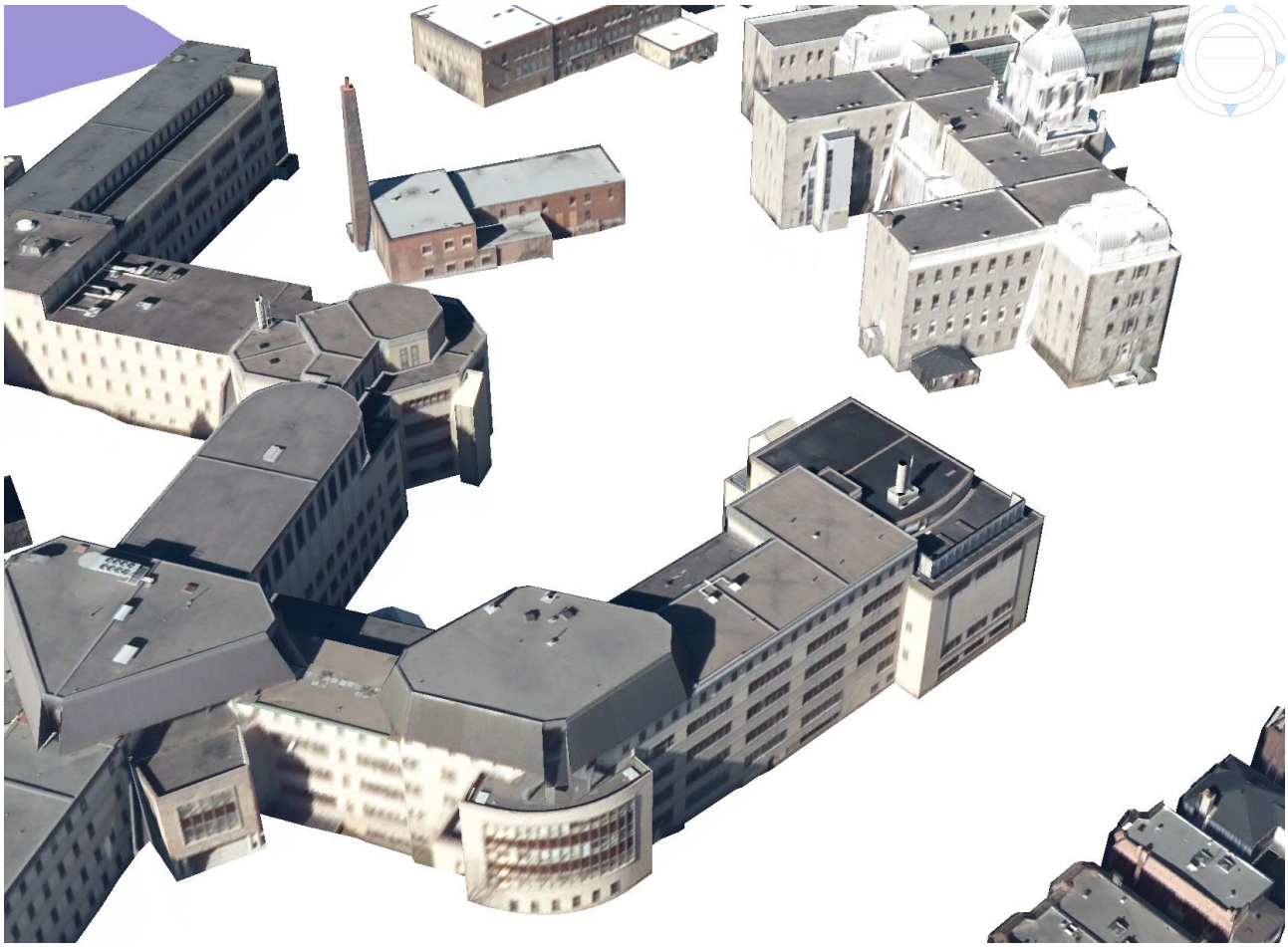
Helsingin CityGML-aineiston maisemia

1.3 Montreal

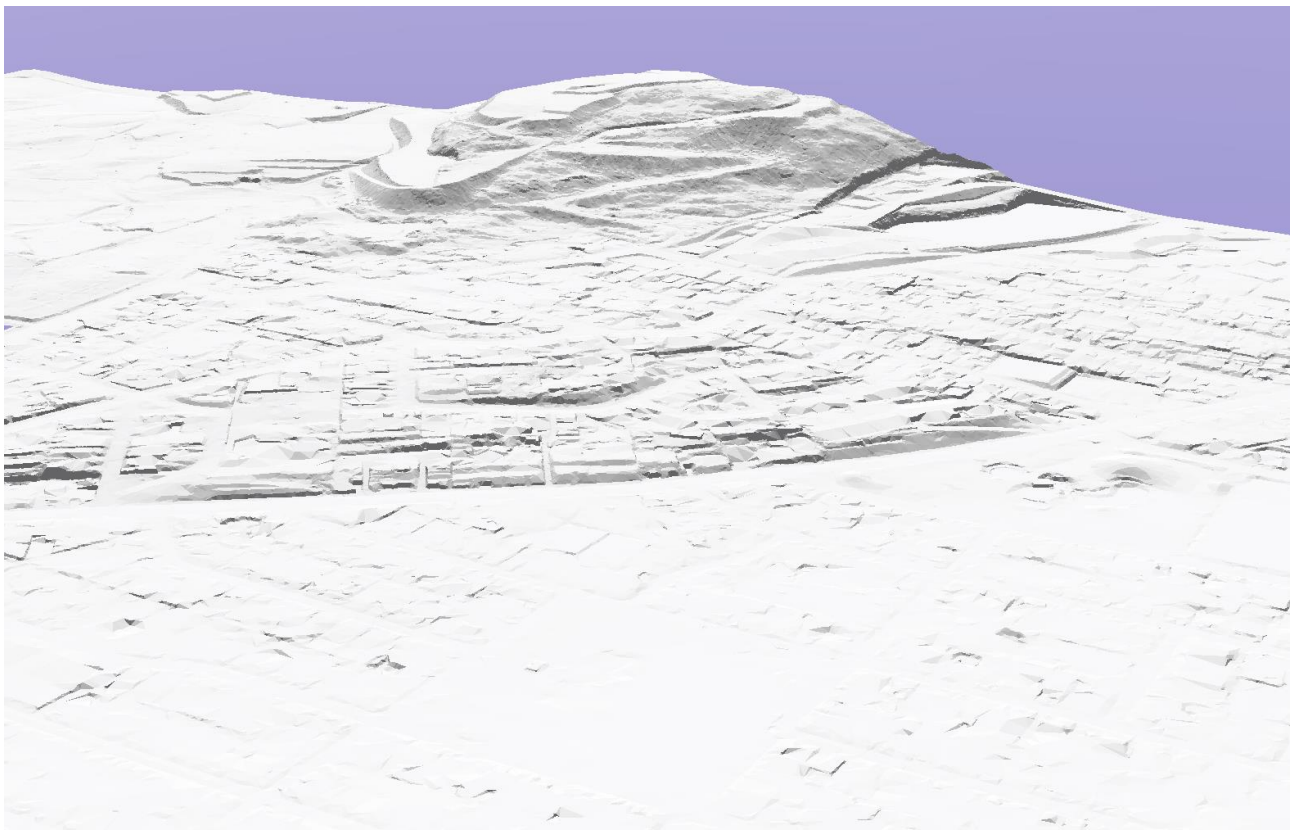
Montrealin aineistot löytyvät vain ranskankielisiltä sivuilta, osoitteesta donnees.ville.montreal.qc.ca/dataset/maquette-numerique-plateau-mont-royal-batiments-lod2-avec-textures. Aineiston standardi on CityGML 1.0. Erikoisuutena Montrealin sivuilta on mahdollista ladata myös maastonmuodot erillisenä CityGML-tiedostona, suora linkki on donnees.ville.montreal.qc.ca/dataset/modele-numerique-de-terrain-mnt.

Rakennusten tekstuuridatan laatu on keskimäärin tavallista heikompi. Tekstuurit ovat usein sumeita ja niissä on pahoja projektiovirheitä. Valaistusta ei ole poistettu. Valkoiset värit ovat puhki palaneita, ja myös tässä tapauksessa puut projisoituvat rakennusten seiniin.

Maasto ei sisällä väri-informaatiota. Se on kohtuullisen tarkka. Mäet, talojen paikat ja tiet erottuvat maastosta selkeästi. Keskimäärin polygonien koko näyttää olevan 1-10 metriä. Tarkkuus näyttää olevan suhteutettu maaston korkeusvaihteluiden jyrkkyyteen. Tasaisilla alueilla on käytetty vähemmän geometriaa.



Montrealin rakennuksia

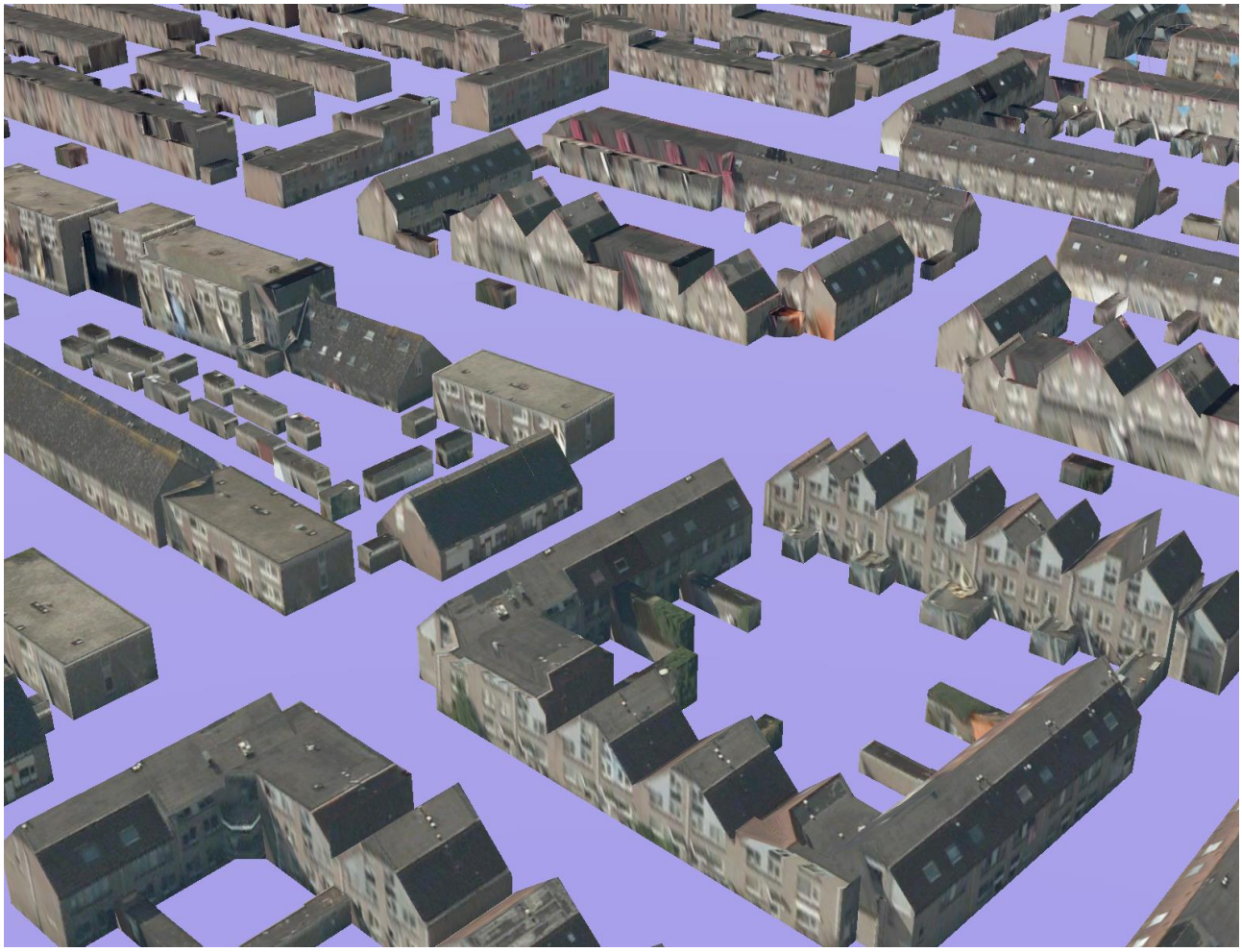


Montrealin CityGML-maastoa

1.4 Rotterdam

Aineisto ladattavissa: www.web3d.org/competition/city-modeling-2015/rotterdam-information. CityGML-standardi: 1.0. Aineisto ei sisällä maaston muotoja.

Tekstuuriprojektioiden laatu on heikoin tutkituista vaihtoehdoista, ja käytännössä käyttökelvoton visuaaliseen työhön. Sävyltään tekstuurit ovat luonnottoman ruskeaa massaa, jopa viherpuista projisoituneet latvat näyttävät harmahtaville.



Rotterdamin kaupunkidata on visuaalisesti heikkolaatuisin kaikista

1.5 Valittu aineisto

Aineiston valintaan vaikutti erityisesti tekstuuriprojektioiden laatu. Jos jokin vaihtoehdoista olisi sisältänyt teksturoidut maastonmuodot, sitä oltaisiin painotettu voimakkaasti. Tässä tapauksessa vain Montreal antoi mahdollisuuden ladata maastodatan CityGML-muodossa, mutta se ei sisältänyt lainkaan väri-informaatiota. Tietysti joissain tapauksissa pelkästä maastonmuodostakin voi olla visualisoinnissa hyötyä, jos suunnittelualueella on voimakkaita korkeusvaihteluita. Nyt päätettiin keskittyä rakennuksiin, ja käyttää maastona yksinkertaista ilmakuvalla teksturoitua, käsin kevyesti muokattua tasopintaa. Tulevaisuudessa olisi toivottavaa, että kaupunkidata sisältäisi aina laadukkaan, teksturoidun maaston.

Tekstuuriprojektioiden heikon laadun vuoksi Rotterdam ja Montreal oli helppo jättää laskuista pois. Berliinin ja Helsingin välillä ei ole valtaisaa laatueroa, mutta Helsingin aineisto on paikoitellen tarkempi ja värimaailmaltaan tasaisempi. Berliinin aineistossa on enemmän epäloogisuuksia valaistusinformaatiossa, esimerkiksi kaksi vierekkäistä rakennusta voivat olla värikirkkaudeltaan täysin poikkeavia. Koska hankkeen aikana haluttiin kokeilla myös tekstuurien valaistusinformaation poistoa, on tekstuurien väreillä väliä.

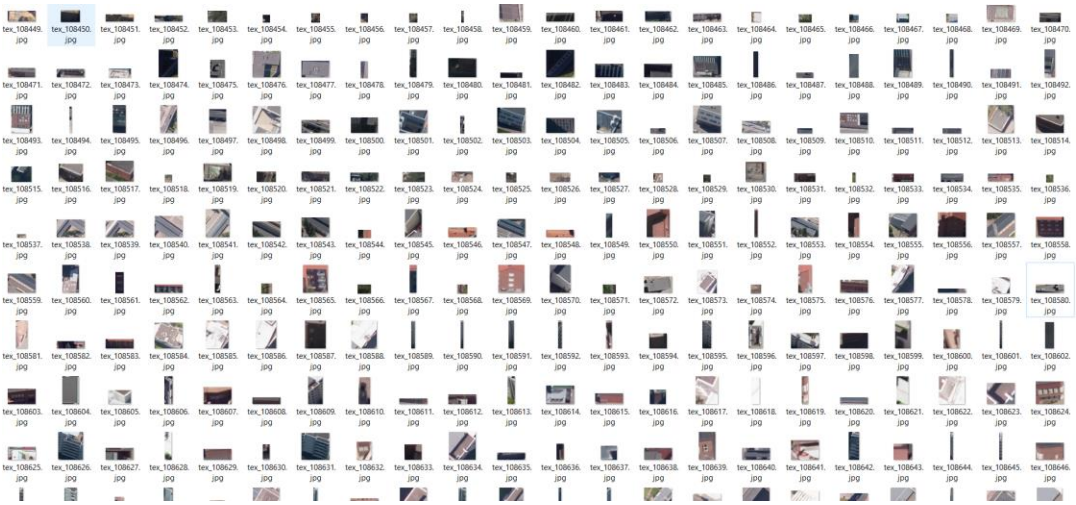
Berliinin aineistossa on ongelmia tekstuurien värikirkkauden tasaisuudessa

Laadun perusteella päädyttiin valitsemaan Helsingin CityGML-aineisto kokeilun demoaineistoksi.

Mitenkään ihanteellinen ei Helsingin kaupunkidata kuitenkaan ole. Tulevaisuudessa datan luomisessa tarvitaan aivan uudenlaista ajattelua, jotta kaupunkidatan käytettävyys visuaalisessa työssä olisi korkea. Satelliittikuva tuskin koskaan tulee riittämään lähtöaineistoksi jo siitä syystä, että kuvausta ei voi suorittaa pilvisellä säällä. Paras laatu saavutettaisiin todennäköisesti yhdistämällä kuivalla, pilvisellä säällä suoritettua lennokkikuvausta katutasolta otettuun aineistoon. Lennokeilla voi jo suorittaa automatisoitua kuvausta fotogrammetriasovellutuksissa, ja maan tasolta voitaisiin ottaa Googlen tapaan automatisoidusti auton kyydistä kuvia tarkoitukseen soveltuvalla

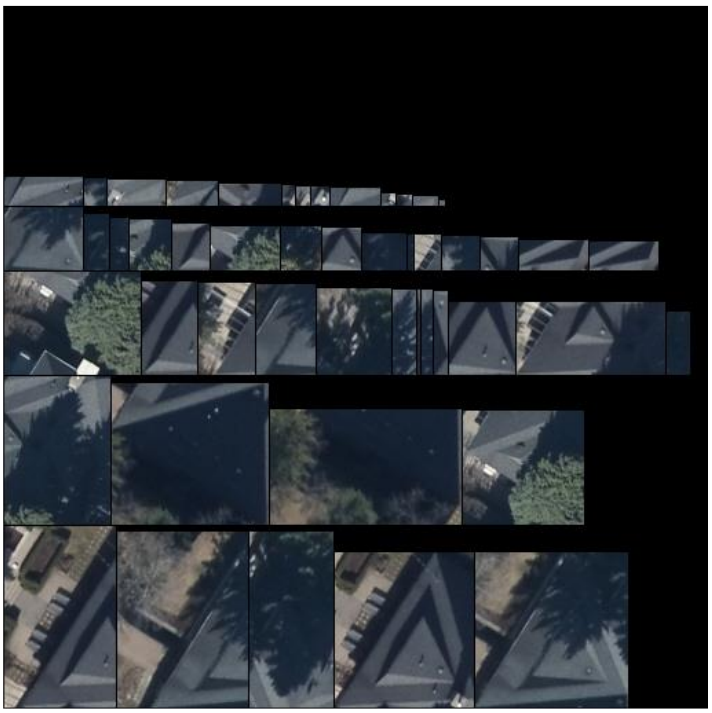
optiikalla.

Demoaineistossa havaittiin myös ongelma tavassa, jolla tekstuurit on luotu. Yhden CityGML-tiedoston sisältämät väri-informaatiot saattavat olla tallennettuna 16 000 pieneen kuvatiedostoon. Tämä aiheuttaa lukuisia ongelmia tiedoston käyttämisessä. Näin monen tiedoston lukeminen kiintolevyltä on ihan levytekniikasta (SSD/HDD) riippumatta hidasta verrattuna muutaman sadan ison tiedoston lukemiseen. Lisäksi 3D-mallinnusohjelmassa jokainen uusi tekstuuri luo uuden materiaalipaikan (material slot) malliin. Normaalisti yhdessä mallissa saattaa olla muutamasta kymmenestä korkeintaan muutamaan sataan materiaalipaikkaa. Tuhannet materiaalipaikat aiheuttavat vakavia suorituskykyongelmia, kun mallinnusohjelma yrittää piirtää teksturoidun mallin näytölle. Suorittimen ja näytönohjaimen välille syntyy valtavasti liikennettä, kun jokainen pieni tekstuuri täytyy erillisellä piirtokomennolla syöttää näytönohjaimelle.



Helsingin aineiston tekstuurit on tallennettu pienissä sirpaleissa

Montrealin datassa teksturointi on hoidettu mallikkaasti. Pienet kuvat on yhdistetty suurempiin tekstuurikokonaisuuksiin (texture atlas). Kuvien sivumitat ovat lisäksi näytönohjaimen kannalta suotuisasti toisen potensseja, esimerkiksi 512 tai 1024 pikseliä.



Montrealin tekstuurit on yhdistetty suurempiin kokonaisuuksiin

2. Ohjelmistot ja menetelmät CityGML:n muuntamiseen

2.1 CityGML2OBJs

CityGML2OBJs on avoimen lähdekoodin Python-sovellus, jota käytetään komentoriviltä. Se muuntaa CityGML-tiedostot yleiseen 3D-mallinnusohjelmien käyttämään OBJ-muotoon. Sovelluksen skriptit löytää GitHubista osoitteesta github.com/tudelft3d/CityGML2OBJs.

Tämä vaihtoehto jätettiin nopeasti pois. Sovellusta on päivitetty viimeksi vuonna 2015, eikä se pysty käsittelemään tekstuureja. Sen asentaminen havaittiin työlääksi, koska Pythonin asentamisen lisäksi täytyi hankkia eri lähteistä useita lisäkirjastoja, joita CityGML2OBJs vaatii.

Sovelluksella yritettiin muuntaa muutamaa CityGML -tiedostoa, mutta sovellusta ei saatu toimimaan oikein. Tässä vaiheessa päätettiin jatkaa hedelmällisempiin kokeiluihin, koska vaikka CityGML2OBJs saataisiin toimimaan, siinä ei ole edes teoreettista tukea tekstuureille, eikä siksi tule auttamaan kokeilun edistämisessä.

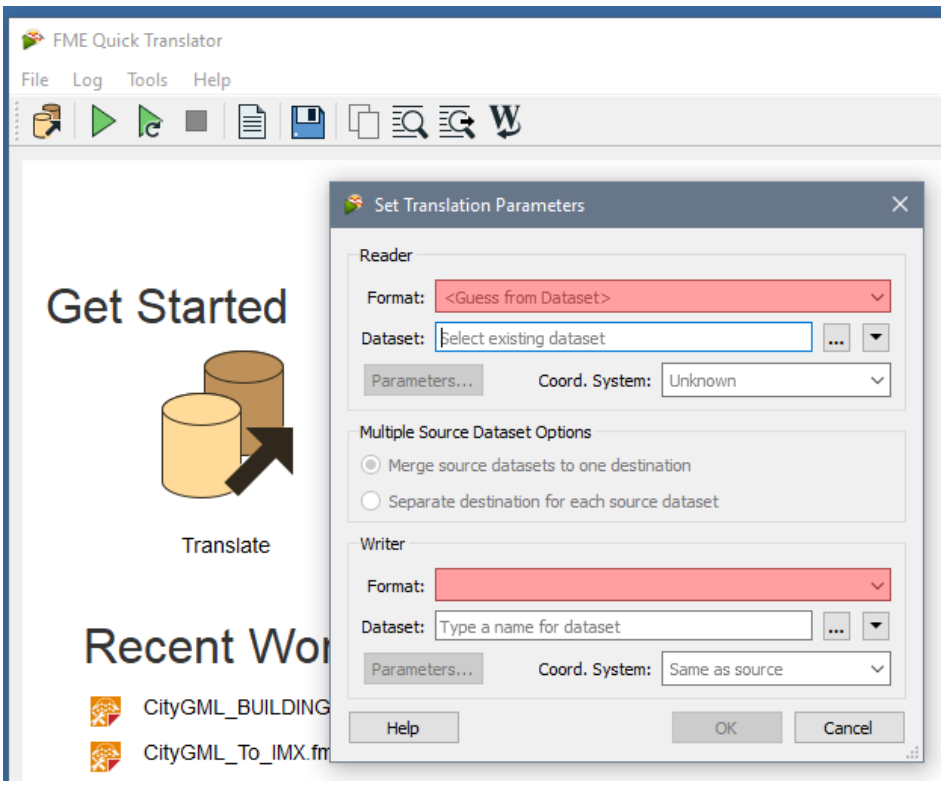
2.2 FME Desktop

Safe Software myy FME-ohjelmistoa "Feature Manipulation Engine", joka on tarkoitettu monipuoliseen datan manipulointiin ja muuntamiseen formaatista toiseen. FME Desktop Professional Edition maksaa tämän dokumentin kirjoitushetkellä 2000 euroa. Ohjelmisto lienee tuttu paikkatietojärjestelmien kanssa työskenteleville yrityksille, mutta graafisen 3D-tuotannon ammattilaisilla tuskin on normaalisti kokemusta tästä ohjelmasta.

Tämän hankkeen aikana käytettiin FME Desktop -pakettiin kuuluvaa Workbench ja Quick Translator versiota 2017.1.2.1 (koonti 17725).

2.2.1 FME Quick Translator

Quick Translator havaittiin periaatteessa toimivaksi CityGML -> OBJ -muunnokseen. Ohjelman käyttö on helppoa. Valitaan vain aloitusruudusta "Translate", valitaan lähdetiedosto, kohdetiedosto ja kohdetiedoston formaatti (Wavefront OBJ). Lähdetiedoston formaatin voi jättää automaattitunnistuksen varaan, tai valita manuaalisesti CityGML.



Näkymä Quick Translator -ohjelman muunnosruudusta

80 megatavun kokoisen CityGML-tiedoston muuntamisen havaittiin kestävän Core i5 6600 -suorittimella tyypillisesti 20-30 minuuttia. Muunnokseen kuluva aika on pitkä suhteutettuna siihen, että vastaavan verteksimäärän sisältävä OBJ tai FBX -tiedosto avautuu yleensä sekunneissa. Tietysti CityGML:n sisältämän semantiikan voi olettaa hidastavan tiedoston käsittelyä ja muuntamista, mutta siitäkin huolimatta prosessin hitaus herätti epäilyksiä muunnoksen toteutuksesta.

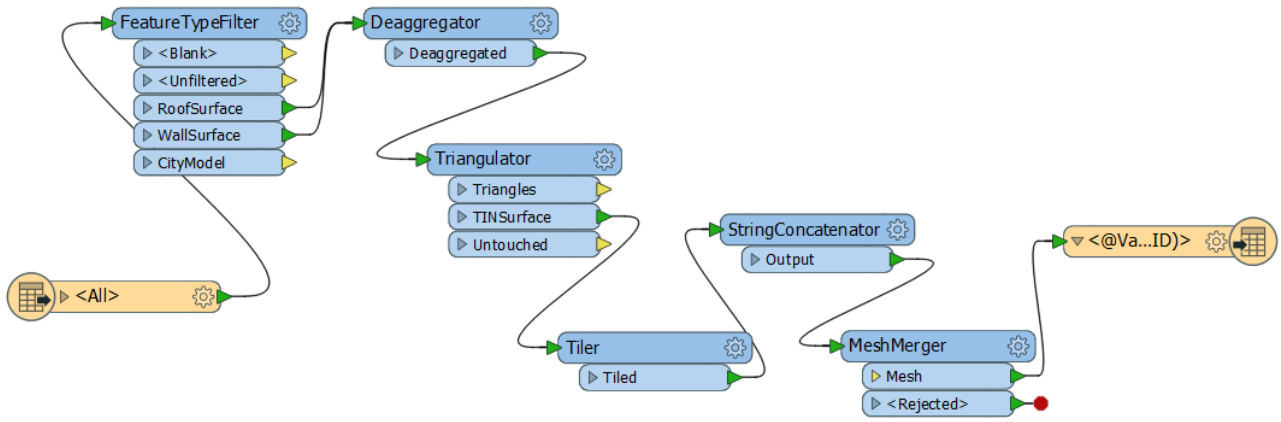
QuickTranslatorin lopputuloksena syntyi yksi OBJ-tiedosto ja tekstuurikansio. Testidatana käytettiin Helsingin keskustaan sijoittuvaa CityGML-tiedostoa. Tässä kohtaa törmättiin ongelmaan. Tiedoston avaamiseen yritettiin käyttää sekä Blender 2.79 että 3ds Max 2009 -ohjelmia. OBJ-tuonti keskeytyi tai jumiutui aina kesken kaiken. Yritettiin myös muuntaa OBJ-tiedosto FBX-tiedostoksi Autodesk FBX converter -ohjelmalla, kuten myös viedä QuickTranslatorista suoraan FBX-tiedosto. Nämä kokeilut eivät tuottaneet parempaa tulosta.

Lopulta kokeiltiin muuntaa Helsingin merialueelta otettua CityGML-tiedostoa, joka sisälsi vain muutaman rakennuksen. Tästä tiedostosta luotu OBJ-versio saatiin avattua normaalisti. Heräsi epäily, että muunnettuun tiedostoon syntyy liikaa materiaalipaikkoja, jotta mikään ohjelma sitä pystyisi avaamaan. Syy voi olla myös jokin muu, mutta tiedoston sisältämän informaation määrään se tuntui liittyvän. Tämän vuoksi kokeiltiin mutkikkaampaa tapaa suorittaa muunnos FME Workbench -ohjelmalla.

2.2.2 FME Workbench

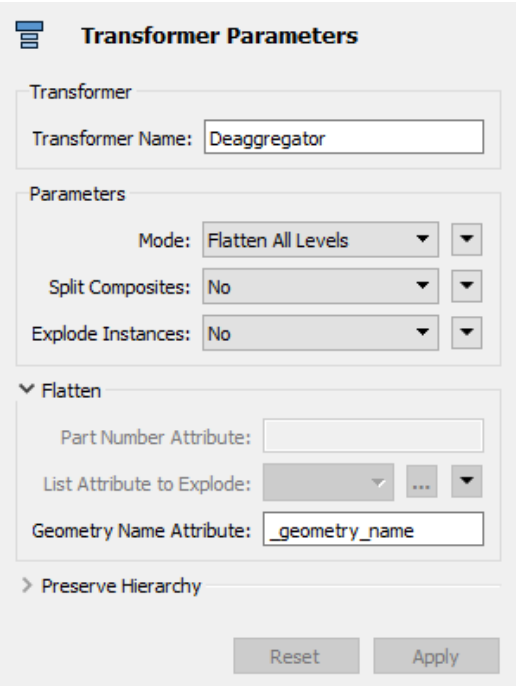
Workbenchiä käytetään eräänlaisella visuaalisella ohjelmoinnilla. Yhdistelemällä eri toimintoja, on

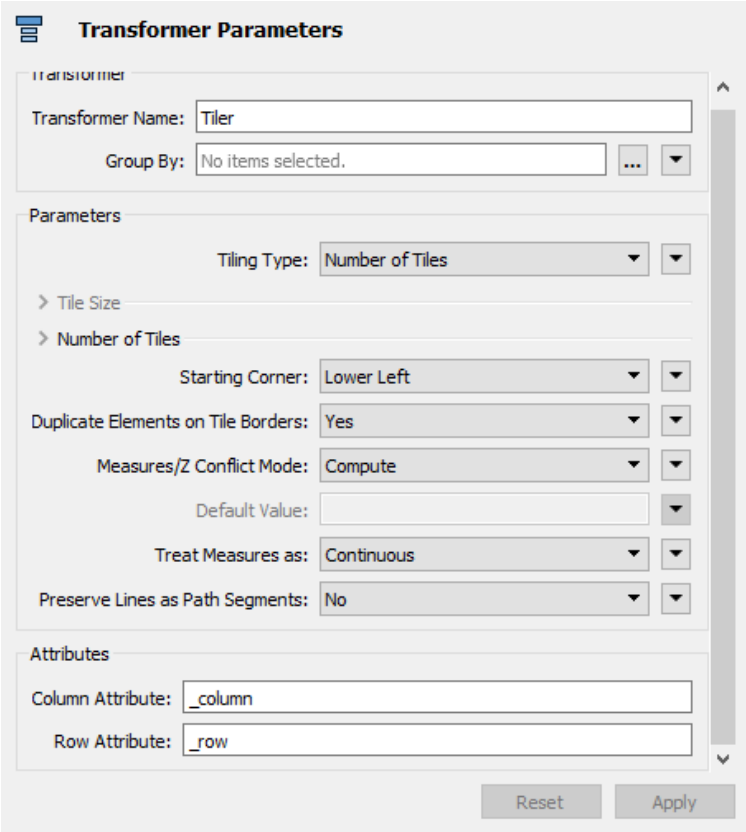
mahdollista luoda juuri omaan käyttöön räätälöity muunnosohjelma. Ohjelmagraafiin lisätään ensin datan lukija (reader). Sen jälkeen luodaan haluttu toiminnallisuus erilaisilla muunninpalikoilla (transformer). Muuntimilla muokataan dataa tai sen rakennetta. Lopuksi muokattu data syötetään ulos ohjelmasta kirjoittimella (writer).

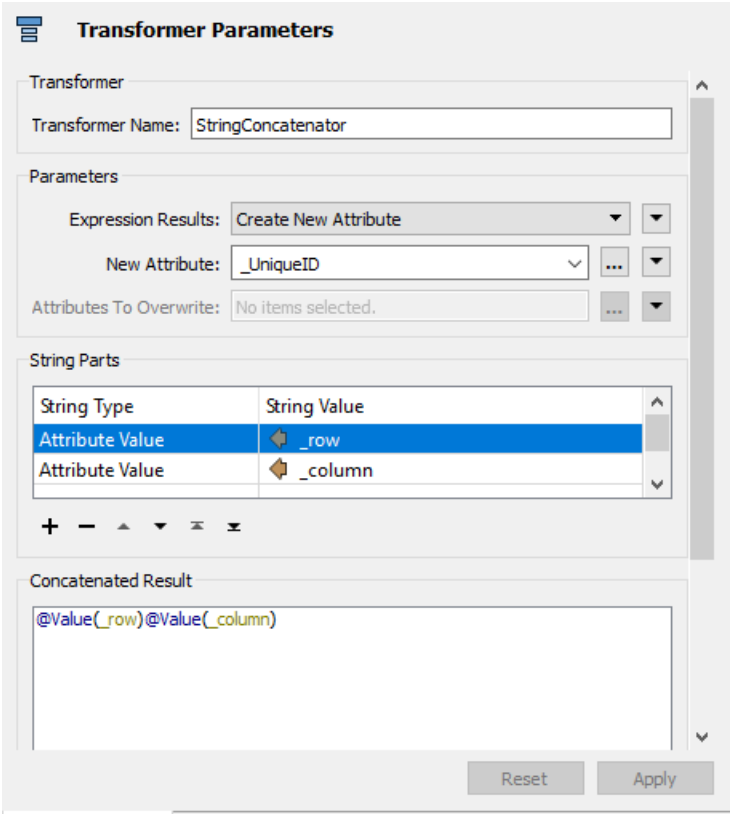


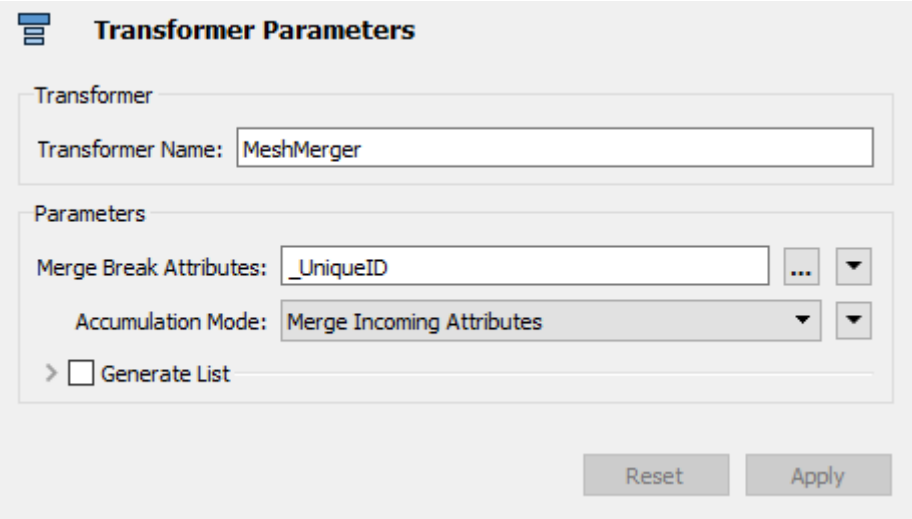
Kokeilussa kehitetty FME Workbench -muunningraafi

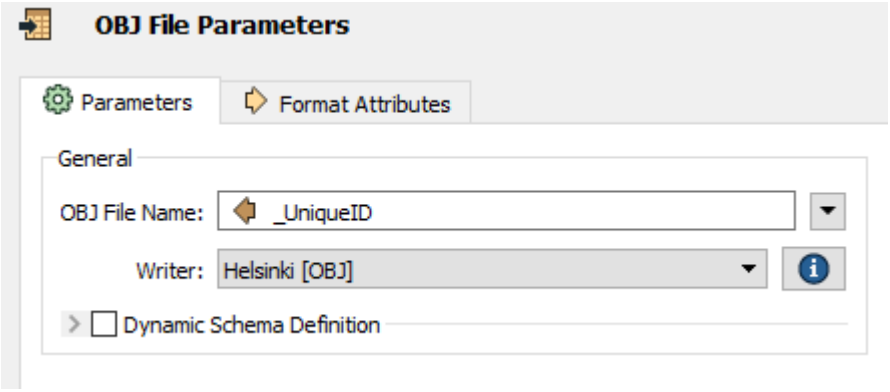
Perusajatus muunnosgraafissa on, että lähtödatasta otetaan rakennukset, muunnetaan ne kolmioverkoksi ja pilkotaan pienempiin palasiin. Tiler-muunnin luo dataan lisätunnisteet _row ja _column, joiden avulla luodaan yhdistetty _UniqueID-tunniste StringConcatenator-muunninta käyttämällä. Lopuksi MeshMerger yhdistää palaset tämän uniikin tunnisteen avulla kokonaisuuksiksi, ja lopputulos tallennetaan OBJ-kirjoittimella tiedostoihin, jotka on myös nimetty uniikin tunnisteen perusteella.




Transformer Parameters
Transformer
Transformer Name: Tiler
Group By: No items selected.
Parameters
Tiling Type: Number of Tiles
Tile Size
Number of Tiles
Starting Corner: Lower Left
Duplicate Elements on Tile Borders: Yes
Measures/Z Conflict Mode: Compute
Default Value:
Treat Measures as: Continuous
Preserve Lines as Path Segments: No
Attributes
Column Attribute: _column
Row Attribute: _row
Reset
Apply


Transformer Parameters
Transformer
Transformer Name: StringConcatenator
Parameters
Expression Results: Create New Attribute
New Attribute: _UniqueID
Attributes To Overwrite: No items selected.
String Parts
String Type
String Value
Attribute Value
_row
Attribute Value
_column
Concatenated Result
@Value(_row)@Value(_column)
Reset
Apply

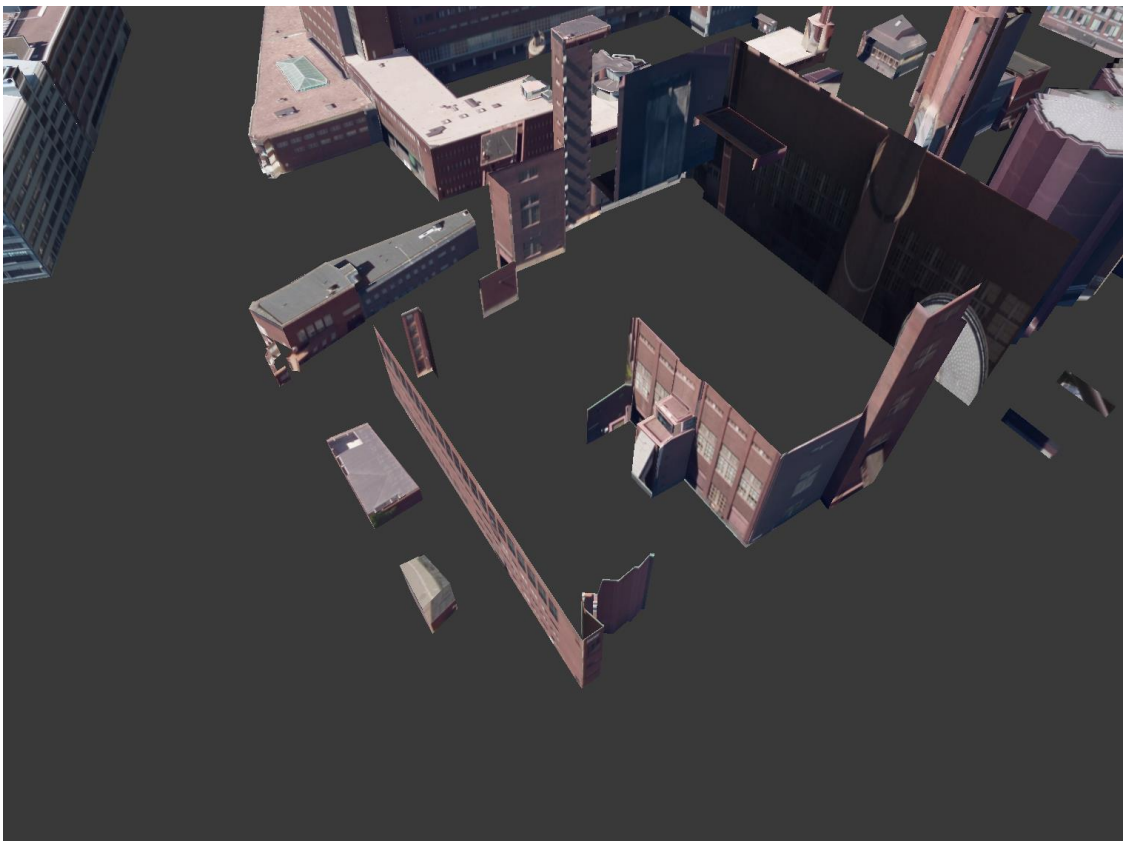




Näin luotu muunnosgraafi saatiin toimimaan lähes täysin oikein. Lopputuloksena oli paloittelun myötä erilliset, alkuperäistä pienemmät OBJ-tiedostot, jotka saatiin auki tekstuureineen Blenderissä. Palaset säilyttivät koordinaatit suhteessa toisiinsa, eli niitä ei tarvinnut enää Blenderissä asetella kohdalleen.

Ongelmaksi muodostui tiler-muuntimen toiminta. Se rikkoi jakoruudukon rajoilla olevat rakennukset käyttökelvottomiksi. Kun palaset yhdistettiin Blenderissä, kävi ilmi, että rakennuksista jäi pysyvästi puuttumaan osia. Tiler-muunninta yritettiin säätää ja otettiin käyttöön ”Duplicate elements on tile borders”-vipu, mutta tällä ei ollut minkäänlaista vaikutusta. Lopulta epäiltiin, että kyseessä on FME-ohjelmiston bugi, mutta varmuudella tätä ei pystytty toteamaan.

Tämä ainoa puute prosessissa aiheutti sen, että FME-ohjelmiston käyttö jouduttiin hylkäämään. Jos ongelmaan löytyy tulevaisuudessa ratkaisu, on FME Workbench hyvin varteenotettava vaihtoehto, ainakin jos se löytyy valmiiksi yrityksen työkalupakista tai hankkimisen kustannus ei ole este.



Rakennus on menettänyt muunnosoperaatiossa seinien ja katon osia pysyvästi

2.3 3D City Database

3D City Database on ilmainen ohjelma, joka on pääasiassa tarkoitettu CityGML-datan muuntamiseen tiettyihin katseluohjelmiin, kuten Google Earth ja ArcGIS. Ohjelman kotisivu on osoitteessa www.3dcitydb.org.

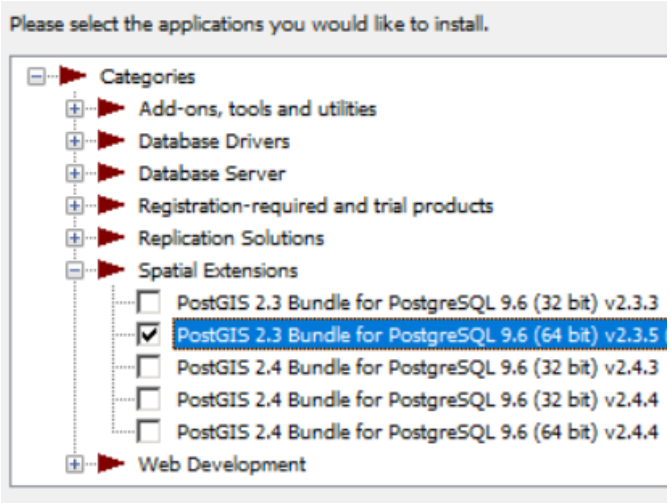
2.3.1 Asentaminen

Ennen ohjelman asentamista on syytä varmistaa, että tietokoneelle on asennettu uusin Java Runtime Environment, versio 8 tai uudempi. Kokeilussa 3D City Database -ohjelmaa ei aluksi saatu toimimaan, tämän huomattiin myöhemmin johtuvan 32-bittisestä Java-ympäristöstä. 64-bittinen Javan asennuspaketti täytyy valita erikseen Javan lataussivulta.

Tämän dokumentaation kirjoitushetkellä 3D City Database oli versionumerossa 3.3.2. Ohjelman kotisivuilta ladataan .jar-päätteinen asennuspaketti. Koska ohjelma on tehty Javan päälle, asennusohjelma ei välttämättä pysty asentamaan sovellusta Windowsin tavanomaiseen kansioon. Tällöin voi käyttää Windows-käyttäjätilin dokumentit-kansiota, yleensä ”C:\Users\Käyttäjä\Documents\”. Valitaan asennettavaksi ainakin ominaisuudet ”core application files” ja ”3D city database”.

Ohjelman käyttäminen vaatii myös Oracle tai PostgreSQL -tietokantapalvelimen asentamisen. Tietokantaan tuodaan CityGML-data, josta sen voi viedä halutussa formaatissa ja laajuudessa ulos ohjelmasta. Tässä kokeilussa käytettiin avoimen lähdekoodin PostgreSQL:n versiota 9.6.

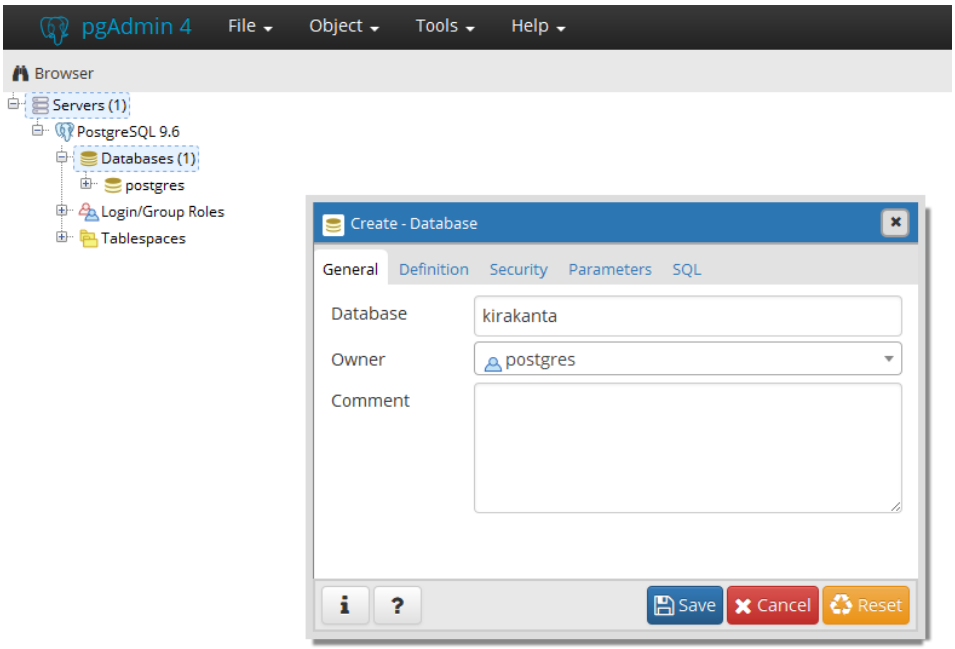
PostgreSQL:n asentamisen jälkeen on asennettava mukana tulevalla stack builderilla PostGIS-lisäosa. Tämän kokeilun aikana käytettiin versiota 2.3.



3D City Database vaatii PostgreSQL-lisäosan PostGIS

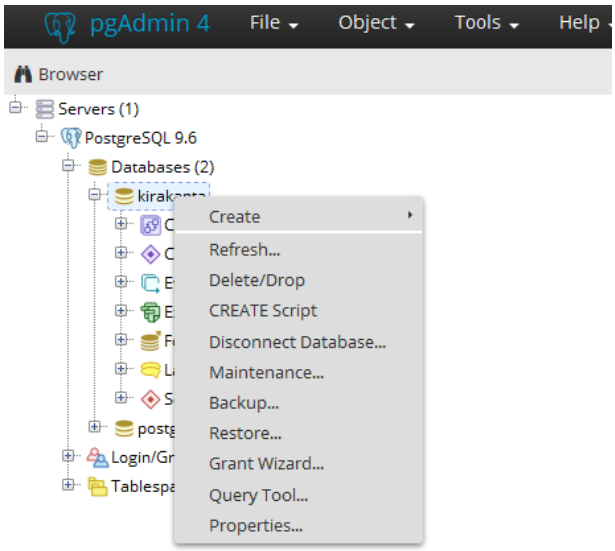
Tämän jälkeen luodaan uusi tietokanta pgAdmin-hallintapaneelilla. Klikkaa hiiren oikealla

Browser-ikkunassa kohtaa Databases, valitse Create > Database ja anna tietokannalle nimi.

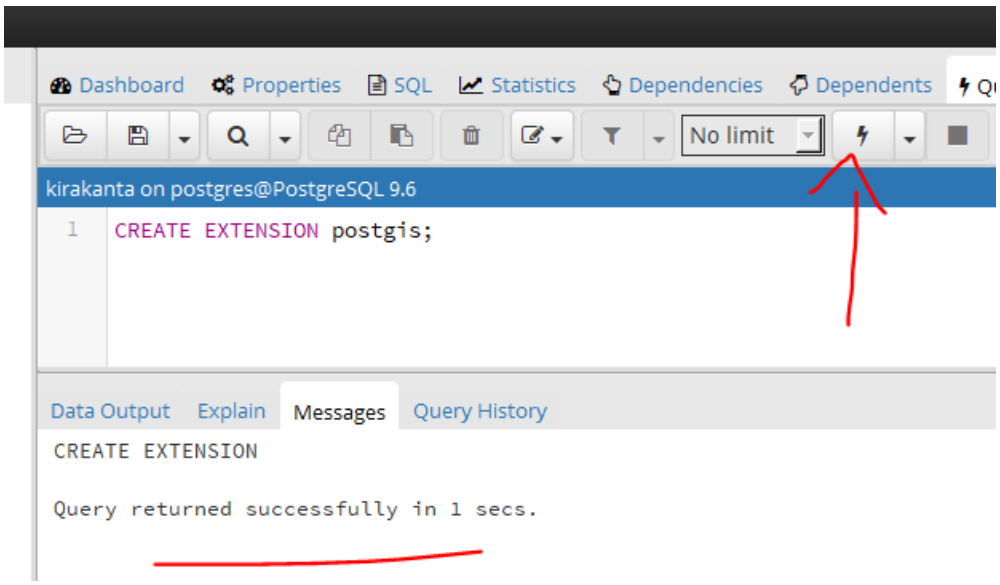


Tietokannan luominen

Seuraavaksi kerrotaan tietokannalle PostGIS-lisäosan olemassaolosta. Suoritetaan tietokannalle kysely ”CREATE EXTENSION postgis”. Klikkaa hiiren oikealla tietokannan nimestä ja valitse Query Tool. Lisää kysely avautuneeseen tekstikenttään ja suorita se Execute-napista (salaman kuva).



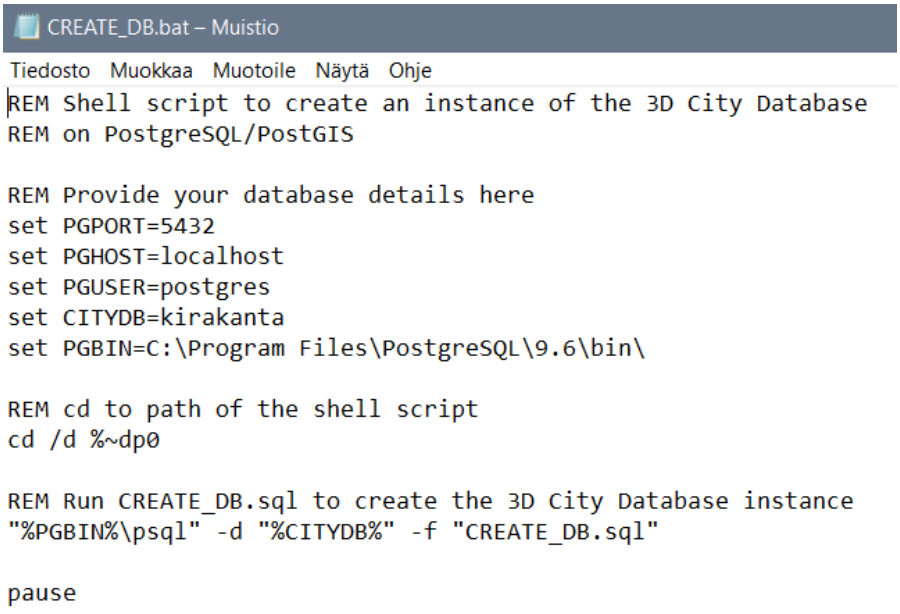
Tästä valikosta etsitään Query Tool



Kyselyn suorittaminen

Jos kysely palautti ”query returned succesfully”, tietokanta on valmis jatkoon.

Seuraavaksi valmistellaan tietokanta käytettäväksi CityGML-datan tallennukseen. 3D City Databasen asennuskansiosta löytyy tarkoitukseen valmistellut komentosarjat: ”asennuskansio\3dcitydb\postgresql”. Muokkaa CREATE_DB-komentosarjaan seuraavat tiedot:
tietokannan portti (yleensä 5432)
host (localhost jos asennettu samalle tietokoneelle)
käyttäjä (vakiona postgres)
tietokannan nimi (tämän valitsit edellisessä vaiheessa)
polku psql.exe-tiedostoon (esim. C:\Program Files\PostgreSQL\9.6\bin\)



Esimerkki toimivasta CREATE_DB-komentosarjasta

Komentosarjan suorittaminen luo tietokannan, kun vastaat komentokehotteessa esitettyihin kysymyksiin. Helsingin kaupunkidata käyttää koordinaattijärjestelmää EPSG:3879 (ETRS89 / GK25FIN). SRID-kohtaan ilmoitettiin näin ollen 3879 ja SRSName kohtaan crs:EPSG::3879. Havaittiin, että jos tietokannan ja kaupunkidatan koordinaattijärjestelmät eivät täsmää, niin tuotua aineistoa ei voi valita kartalta ja datan vienti ulos ohjelmasta voi epäonnistua.

Tietokanta on nyt valmis. Vielä ennen 3D City Database Importer / Exporter -ohjelman käyttöä on syytä varmistaa, että Java-suoritusympäristölle on määritelty tarpeeksi keskusmuistia käytettäväksi, jotta suurten CityGML-tiedostojen käsittely ei kaada ohjelmaa. Kokeilussa yritettiin aluksi lisätä Javalle keskusmuistia globaaleilla parametreilla, mutta näillä ei ollut vaikutusta. Lopulta löydettiin ohjelman käynnistykseen käytettävä komentosarja, josta selvisi, että se ohittaa globaalit Java-parametrit. Näin ollen muokkaukset on tehtävä suoraan tähän komentosarjaan.

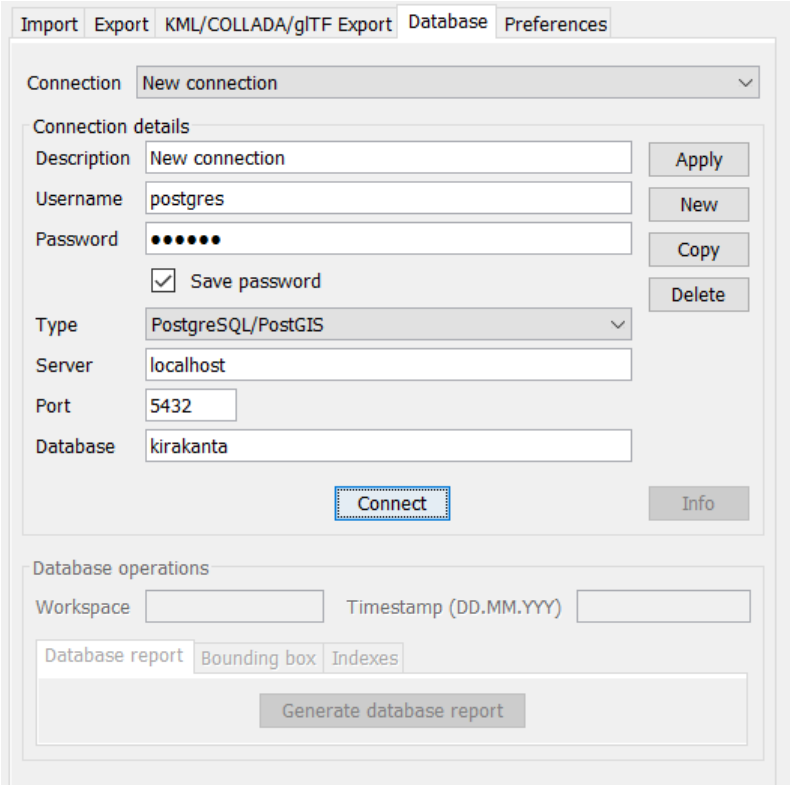
Etsi komentosarja ”3DCityDB-Importer-Exporter.bat” 3D City Databasen asennuskansiosta. Muokkaa komentosarjasta seuraavaa osuutta:

```
REM The following command executes the GUI version
start /min cmd /c "%JAVA% -jar -Xms128m -Xmx1024m lib/3dcitydb-impexp.jar"
```

Parametri -Xms määrittää vähimmäismuistimäärän jonka Java saa käyttöönsä, ja -Xmx määrittää suurimman sallitun määrän. Jos tietokoneessa on vähintään 16 GB keskusmuistia, parametrit voi muuttaa huoletta muotoon -Xms8192m -Xmx8192m. Näin Javalla on varmasti käytössä 8 GB keskusmuistia. Kokeilussa jouduttiin ison kaupunginosan viennissä nostamaan arvoksi 12288 eli 12 GB, joka vielä toimi 16 GB keskusmuistilla varustetussa koneessa. Javan keskusmuistin käyttöä kannattaa seurata tehtävienhallinnasta, sillä testien perusteella on mahdollista, että vientiprosessi onnistuu liian pienellä muistimäärälläkin, mutta vie paljon enemmän aikaa. Jos Java käyttää viennin aikana 8 GB muistia ja rajaksi on asetettu 8 GB, se kielii siitä, että muistia olisi syytä luovuttaa Javan käyttöön vielä enemmän.

2.3.2 CityGML-tuonti

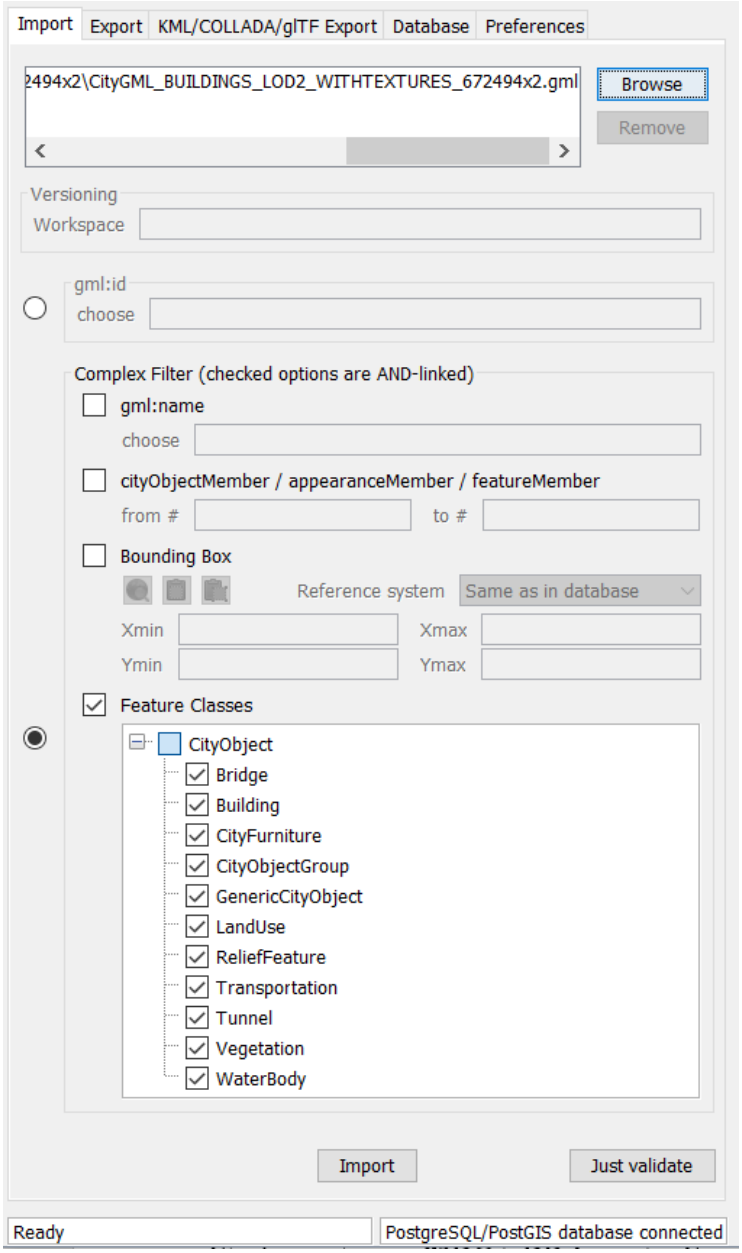
Kaupunkidatan tuonti on yksinkertainen prosessi. Käynnistä 3D City Database Importer / Exporter ja aseta Database-välilehdellä tietokannan tiedot oikein.



Tietokantayhteyden määrittäminen

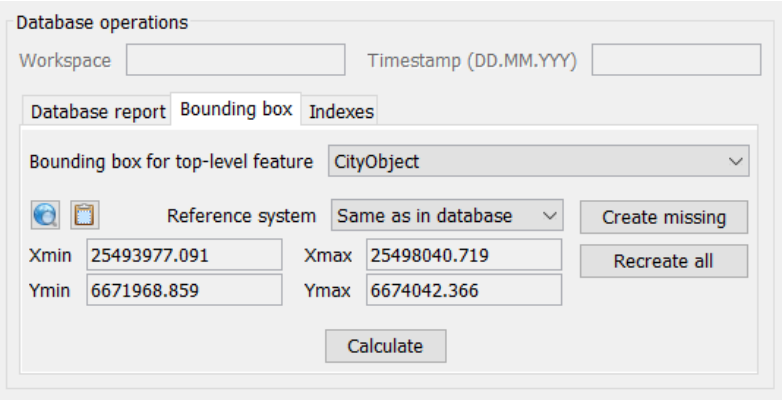
Yhteys muodostuu, jos tiedoissa ei ole virheitä ja PostgreSQL-palvelin on päällä. Normaalisti palvelin käynnistyy aina tietokoneen mukana.

Import-välilehdellä valitaan tuotava CityGML-tiedosto ja ominaisuudet. Tietokantaan kannattaa tuoda kaikki data mitä tiedosto sisältää, sillä vientivaiheessa voi halutessaan karsia ominaisuuksia. Helsingin demodata sisältää vain rakennuksia. Datan tuonti kestää yleensä alle viisi minuuttia.

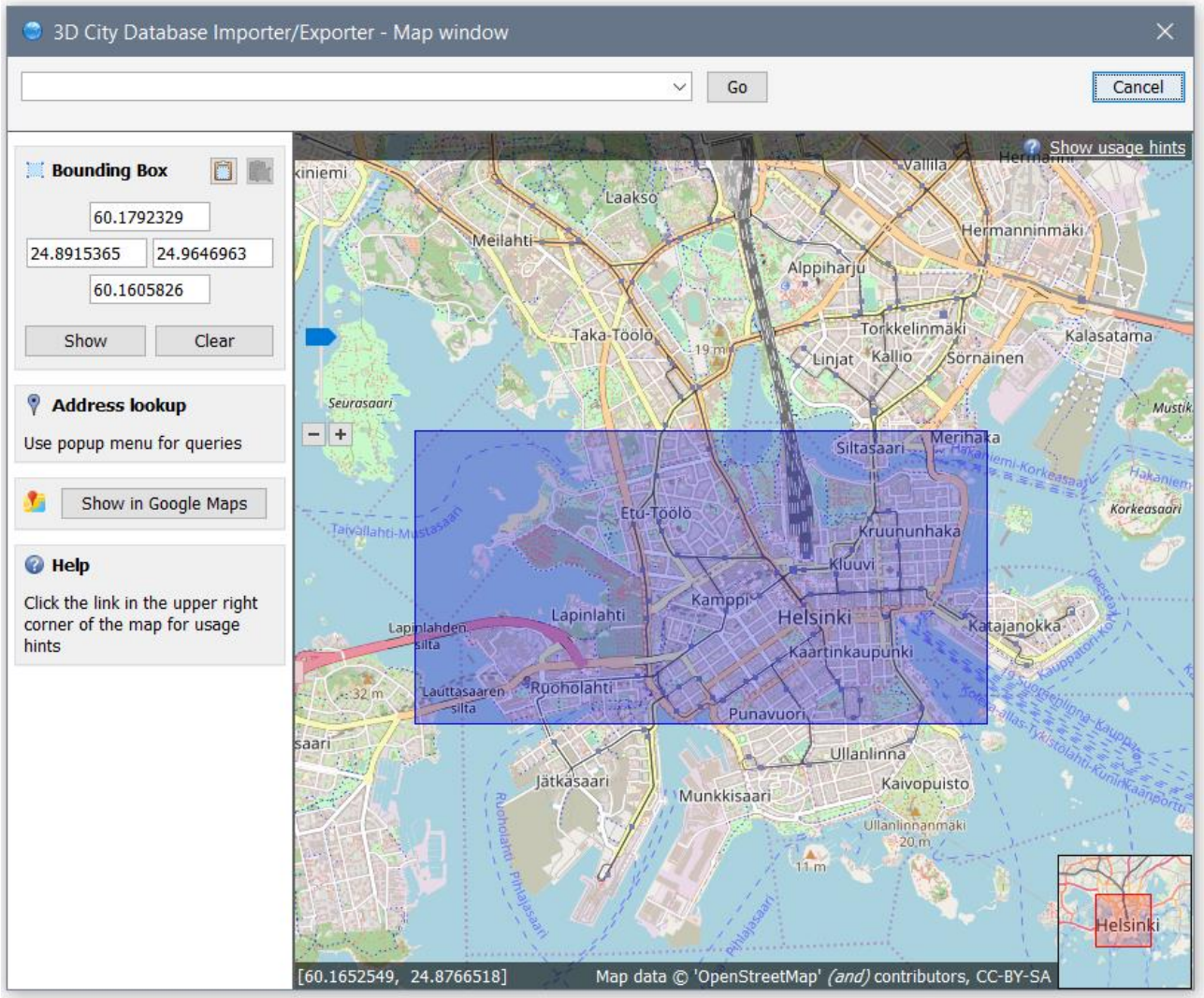


CityGML-tuonti

Tässä vaiheessa kannattaa varmistaa, että tietokantaan tuotu aineisto näkyy rajauksenvalintakartalla Database-välilehdellä. Database operations -osiossa on Bounding box -välilehti, jonka avulla voidaan laskea tietokannassa olevan datan rajakoordinaatit. Kaikki data on näiden koordinaattien sisällä. Alueen pystyy havainnollisesti tarkistamaan maapallon kuvaa klikkaamalla. Ennen kartalle menoa, on muistettava klikata ”Calculate”.



Lasketaan bounding box



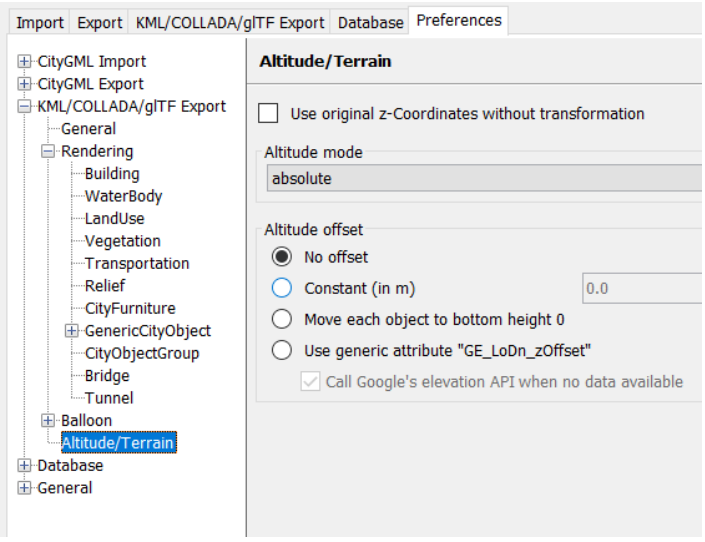
Kaikki tietokantaan tuotu data on tämän alueen sisällä

Tietokantaan voi tuoda useampia CityGML-tiedostoja peräjälkeen. Ne kaikki tallentuvat tietokantaan, ja viennin aikana on valittavissa, mitkä alueet halutaan ulos tietokannasta.

2.3.3 Collada-viennin asetukset

Ennen kaupunkidatan vientiä on syytä tarkistaa Preferences-välilehdeltä muutama asetus. Kaikki kiinnostavat asetukset löytyvät ”KML/COLLADA/glTF Export”-osiosta.

Hyväksi todetut Altitude/Terrain-asetukset löytyvät kuvakaappauksesta. Erityisen tärkeää on muuttaa ”Altitude offset” kohtaan ”No offset”. Jos käytetään vakioasetusta, vientiprosessissa ohjelma tekee kyselyitä Googlen ohjelmistorajapintaan, mikä hidastaa vientiä todella paljon. Ilmeisesti ohjelma tekee jokaiselle rakennukselle erillisen API-kyselyn, ja jää odottamaan verkon kautta tulevaa vastausta ennen seuraavan rakennuksen vientiä. Jos rakennuksia on satoja, voi tästä johtuva hidastuminen lisätä vientiprosessin kestoa kokeilujen perusteella jopa 30-60 minuuttia.



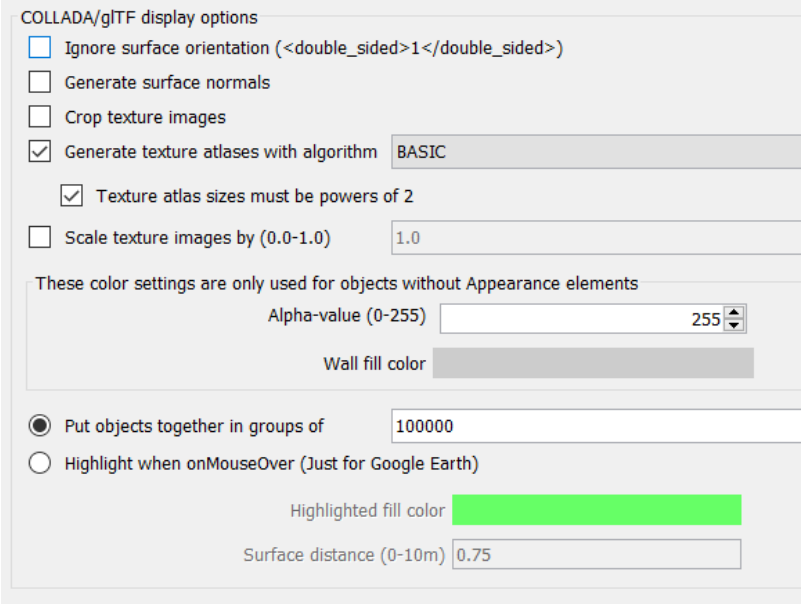
Korkeusasetukset

Rendering-osiosta löytyy oma kohta kaikille mahdollisille vietäville objektityypeille. Testidata sisältää vain rakennuksia, joten tarpeen on muuttaa asetuksia vain kohdasta ”Building”.

Yhdistelmätekstuurien luonti on ehdottomasti otettava käyttöön. Generaatiotyyppinä ”Basic” toimii kokeilujen perusteella aivan hyvin. Dokumentaation perusteella ”TPIM” käyttää enemmän prosessointitehoa viennin aikana, mutta luo paremmin pakattuja yhdistelmätekstuureja. Käytännössä kokeiltiin eri asetuksia, eikä niillä nähty mainittavaa eroa. Jatkoprosessoinnin kannalta on järkevää jättää päälle asetus, joka pakottaa tekstuurit standardikokoihin ”powers of 2”.

Esimerkkinä erään testidatan vienti texture atlas algoritmillä ”Basic” tuottaa 158 kpl tekstuuritiedostoja, joiden yhteiskoko on 24 MB. Ilman yhdistelyä syntyy 8525 kpl tekstuureja joiden yhteiskoko on 29 MB. Oleellista tässä on tekstuurien määrän vähentyminen n. kahteen prosenttiin alkuperäisestä.

Jotta lopputuloksena muodostuisi mahdollisimman vähän erillisiä tiedostoja joita joudutaan käsin yhdistelemään, on syytä asettaa päälle ”Put objects together in groups of” ja numerokenttään jokin suuri lukema. Näin jokaisesta talosta ei muodostu erillistä Collada-tiedostoa.



Nämä asetukset muutetaan jokaisen vietävän objektityypin kohdalta erikseen

General-osiosta käytännössä kaikki ruksit on pois päältä. glTF-mallin vienti voi olla tulevaisuudessa hyvä vaihtoehto, jos glTF-formaatista muodostuu standardi jota kaikki ohjelmat tukevat. Tällä hetkellä se on niin lastenkengissä, että sitä ei saa tuotua kaikkiin ohjelmiin ongelmitta. Myöskään kmz-tiedostoa ei haluta viedä, se on käytännössä sama kuin normaali kml/Collada, mutta pakattuna.

2.3.4 Collada-vienti

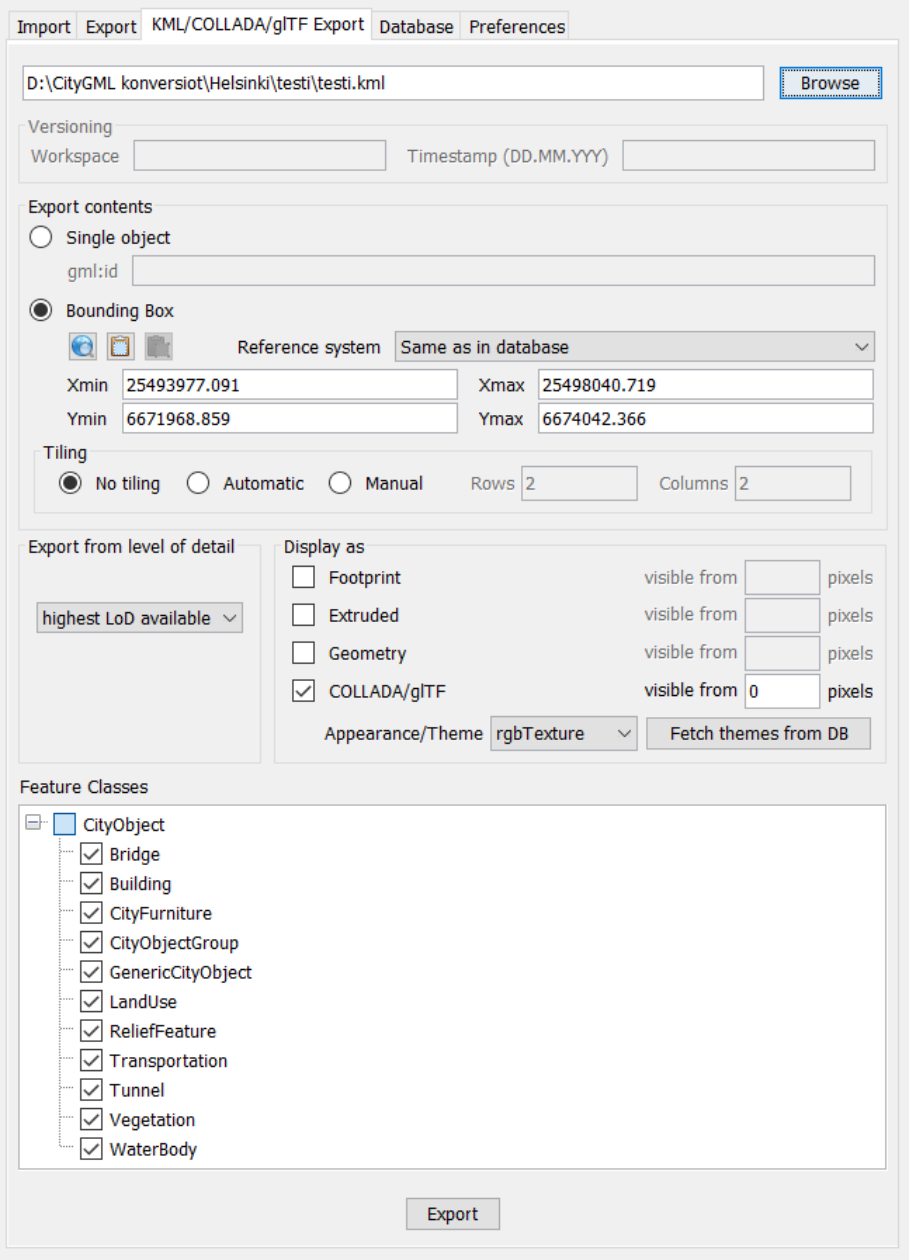
Itse vientiprosessi on helppo, kun kaikki on valmiina. Tässä vaiheessa ei kannata säikähtää sitä, että kohteeksi määritetään .kml-päätteinen tiedosto. Sen viereen muodostuu alikansioita, joista löytyy tarvittavat Collada- ja tekstuuritiedostot.

”Bounding box”-kohdasta voi valita vietävän alueen karttakuvasta, joten pakko ei ole viedä kaikkea tietokantaan tuotua aineistoa.

Tiling-kohdasta valitaan ”No tiling”, jotta vienti ei sirpaloituisi satoihin tiedostoihin. Asetuksesta huolimatta viennissä muodostuu muutama erillinen Collada-tiedosto, mutta niiden määrä on siedettävä käsin yhdistämistä ajatellen. Tätä koskien kysyttiin apua 3D City Database -ohjelman yhteyshenkilöltä. Vastaukseksi saatiin, että koska ohjelman suunniteltu käyttötarkoitus on vienti Google Earth -ohjelmaan, viennin pilkkoutumista ei voi täysin estää.

Level of detail valitaan käsin halutuksi tai jätetään tähän ”highest LoD available”. Jos tekstuurit halutaan vientiin mukaan, haetaan ensin visuaaliset teemat tietokannasta (”Fetch themes from DB”) ja valitaan sitten haluttu teema, tässä tapauksessa ”rgbTexture”. Huomaa, että teeman nimi tulee kaupunkidatasta, ja voi olla eri riippuen datan tuottajasta.

Lopuksi valitaan vietävät ominaisuudet ja klikataan Export. Jos kaikki menee hyvin, viennissä kuluu aikaa muutamasta minuutista muutamaan kymmeneen minuuttiin.



Vientiruutu

2.3.5 Viennin lopputulos

Vientikansioon muodostuu .kml ja .json -tiedostot. Näitä tiedostoja ei käytetä, joten siirrytään Tiles-alikansioon.

CityGML konversiot > Helsinki > testi

Nimi	Muokkauspäivä	Tyyppi	Koko
Tiles	16.4.2018 14.40	Tiedostokansio	
testi.kml	16.4.2018 15.03	KML-tiedosto	2 kt
testi_collada_MasterJSON.json	16.4.2018 15.03	JSON-tiedosto	1 kt

Vientikansion juuri

Alikansiosta löytyy lopulta kansio, joka sisältää oman kansion jokaiselle viedylle osatiedostolle. Nämä kansiot sisältävät varsinaiset Collada-tiedostot ja tekstuurikuvat, jotka täytyy yhdistää takaisin yhdeksi kokonaisuudeksi.

CityGML konversiot > Helsinki > testi > Tiles > 0 > 0

Nimi	Muokkauspäivä	Tyyppi	Koko
2	16.4.2018 14.51	Tiedostokansio	
44	16.4.2018 14.53	Tiedostokansio	
5408	16.4.2018 15.03	Tiedostokansio	
5455	16.4.2018 14.53	Tiedostokansio	
7076	16.4.2018 15.03	Tiedostokansio	
testi_Tile_0_0_collada.kml	16.4.2018 15.03	KML-tiedosto	4 kt

Kaupunkimalli on jakautunut viiteen eri osaan

Tässä kansiossa sijaitsevaa .kml-tiedostoa hyödynnetään osien yhdistämisessä. Sitä voi tarkastella tavallisella tekstinkäsittelyohjelmalla tai helpommin ohjelmointiin tarkoitetulla tekstieditorilla, kuten Programmer's Notepad.

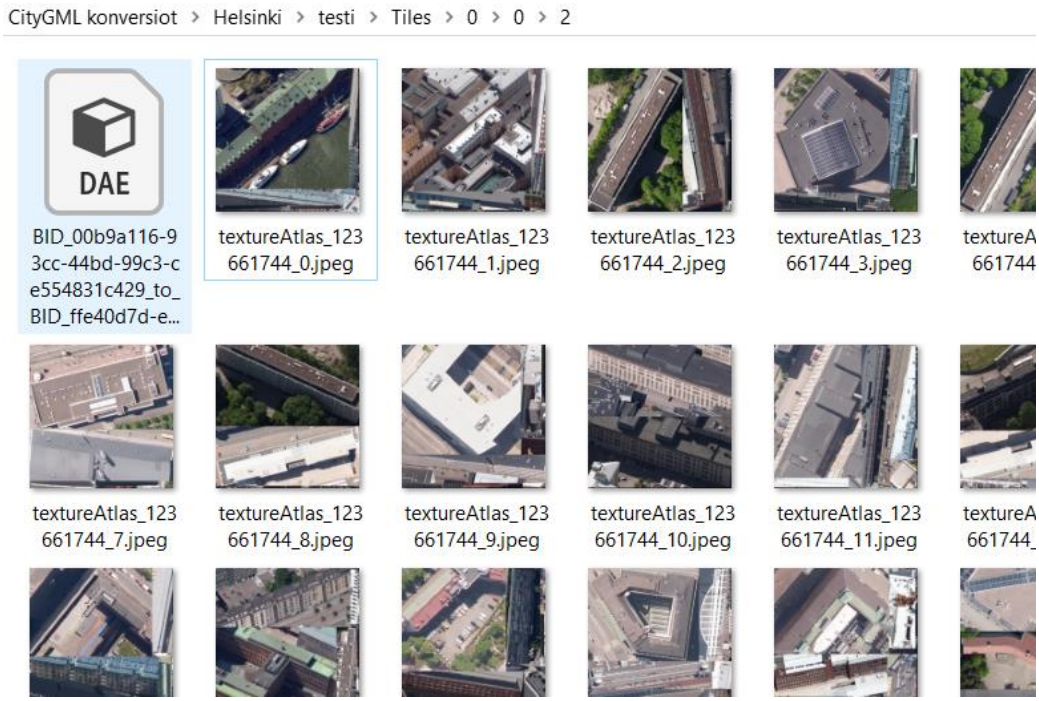
Kml-tiedosto noudattaa xml-standardia, ja sen pääasiallinen tarkoitus on sitoa alikansioissa sijaitsevat Collada-tiedostot toisiinsa. Placemark-tagi tarkoittaa aina yhtä Collada-tiedostoa. Name-tagi kertoo tunnisteen, jonka perusteella alikansioista löytää oikean tiedoston, johon Placemark-tagilla viitataan.

Placemark sisältää Location-tagin, jonka sisältä löytyy osatiedoston koordinaatit ja korkeus. Näistä korkeus on mielenkiintoisin, koska se on kokeilujen perusteella suoraan metreissä. Altitude-arvoa voi siis käyttää suoraan osatiedostojen sijoittamiseen pystyakselilla oikeaan kohtaan.

```
<open>false</open>
<Placemark id="COLLADA_BID_00c42d1b-0578-4302-8ffc-85d398e201ee_to_BID_ffe48ac8-4c7b-453a-a0c8-7df74cf885d7">
  <name>BID_00c42d1b-0578-4302-8ffc-85d398e201ee_to_BID_ffe48ac8-4c7b-453a-a0c8-7df74cf885d7</name>
  <Model>
    <altitudeMode>absolute</altitudeMode>
    <Location>
      <longitude>24.9213862</longitude>
      <latitude>60.1698023</latitude>
      <altitude>8.03</altitude>
    </Location>
    <Orientation>
      <heading>359.9272005</heading>
    </Orientation>
    <Link>
      <href>5408/BID_00c42d1b-0578-4302-8ffc-85d398e201ee_to_BID_ffe48ac8-4c7b-453a-a0c8-7df74cf885d7.dae</href>
    </Link>
  </Model>
</Placemark>
<Placemark id="COLLADA_BID_007e1c12-c3d5-4e79-9f5b-590a1d6ddc8c_to_BID_ff661856-fb1e-426a-8c7c-4f46c0e8a9af">
  <name>BID_007e1c12-c3d5-4e79-9f5b-590a1d6ddc8c_to_BID_ff661856-fb1e-426a-8c7c-4f46c0e8a9af</name>
  <Model>
```


Ote .kml-tiedostosta

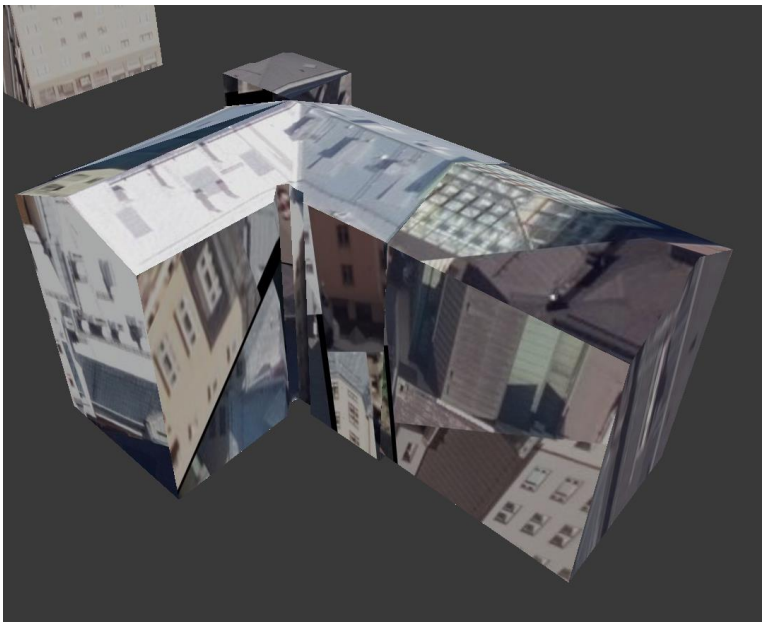
Jokaisen alikansion sisältö on sama: yksi Collada-tiedosto, jonka tiedostonimi on kml-tiedoston name-tagin mukainen. Lisäksi kansio sisältää tiedoston vaatimat tekstuurit.



3. Visualisointiympäristön luominen

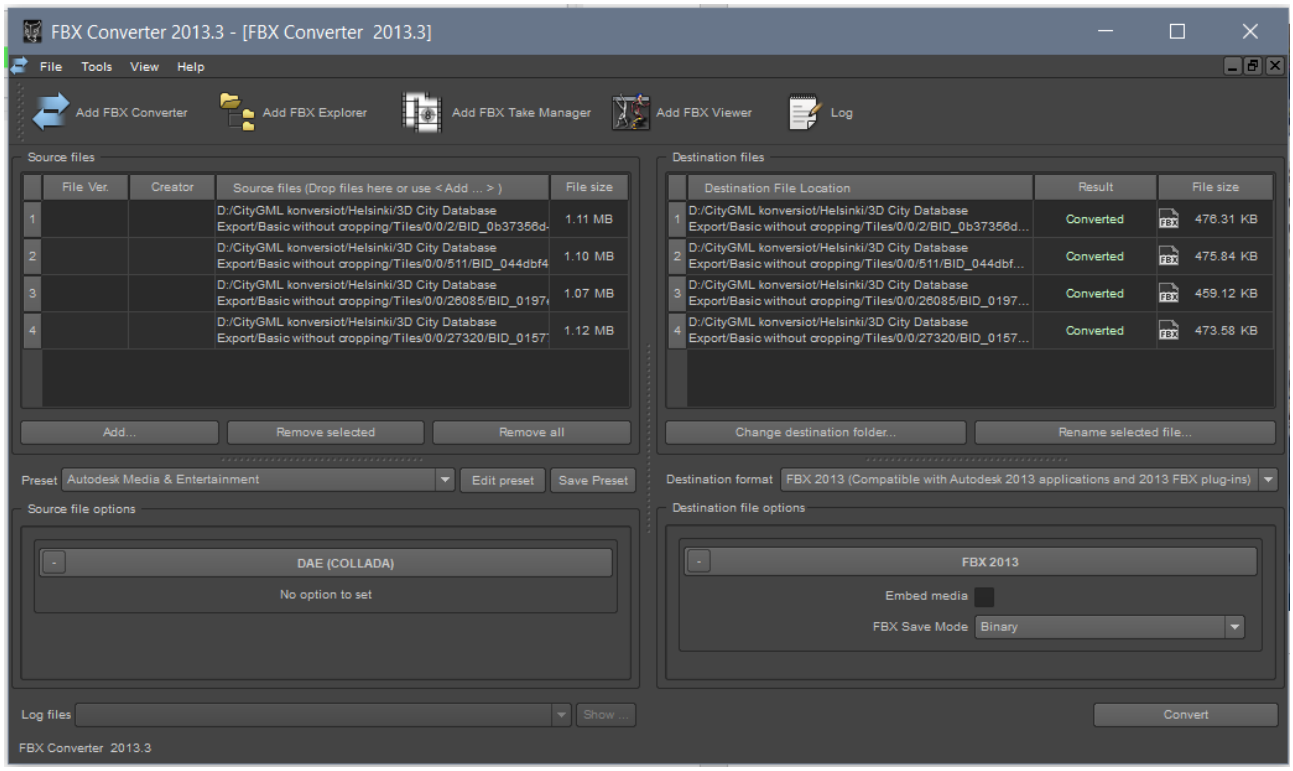
3.1 Collada-tiedostojen muuntaminen FBX-tiedostoiksi

3D City Database -ohjelmasta vietyjä Collada tiedostoja yritettiin aluksi avata suoraan sekä Blenderillä että 3ds Maxilla. Tiedostot avautuivat, mutta jostain syystä rakennusten UV-koordinaatit olivat aina rikki. Näin ollen tekstuurit projisoituivat täysin vääriin paikkoihin rakennusten pinnalle.



Näin käy, kun UV-koordinaatit eivät täsmää

Mitään asetuksia muuttamalla (viennissä tai tuonnissa) tilannetta ei saatu korjattua. Lopulta Collada-tiedostot ajettiin Autodesk FBX 2013.3 Converter -ohjelman läpi, koska aiemman kokemuksen perusteella tiedettiin tämän joskus auttavan ongelmatiedostojen kanssa.



Autodesk FBX Converter

Muunnoksessa muodostuneet FBX-tiedostot avautuivat virheettä ja UV-koordinaatit olivat oikein.

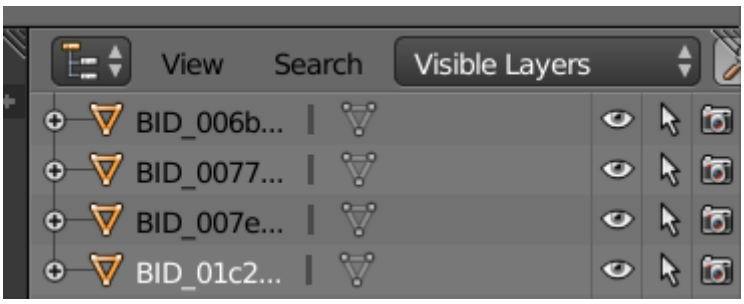
3.2 Tiedostojen käsittely 3D-mallinnusohjelmassa

3D-mallinnusohjelmassa Colladaksi viedyt (FBX-muotoon muunnetut) osat yhdistetään yhdeksi kokonaisuudeksi. Lisäksi malli valmistellaan tekstuurien leipomista varten. Tässä kokeilussa käytettiin Blender-ohjelmaa, mutta periaatteet ovat samat kaikilla mallinnusohjelmilla.

3.2.1 FBX-tiedostojen tuonti mallinnusohjelmaan

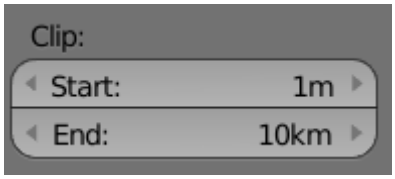
Tuodaan mallinnusohjelmaan kaikki FBX-tiedostot, jotka aiemmin muunnettiin Collada-formaatista. Blenderiä käytettäessä kannattaa vaihtaa renderöintimoottoriksi Cycles ennen objektien tuontia. Näin Blender osaa tehdä tuonnin aikana Cycles-materiaalit objekteille. Tästä on hyötyä, kun halutaan leipoa väritekstuurit uusiksi Blenderin sisällä.

Tuontivaiheessa kannattaa nimetä objektit mallinnusohjelmassa niin, että yhteys Collada-tiedostojen nimeämiseen säilyy. Tämä helpottaa seuraava vaihetta, kun objektien korkeudet asetetaan kml-tiedoston arvojen mukaan.



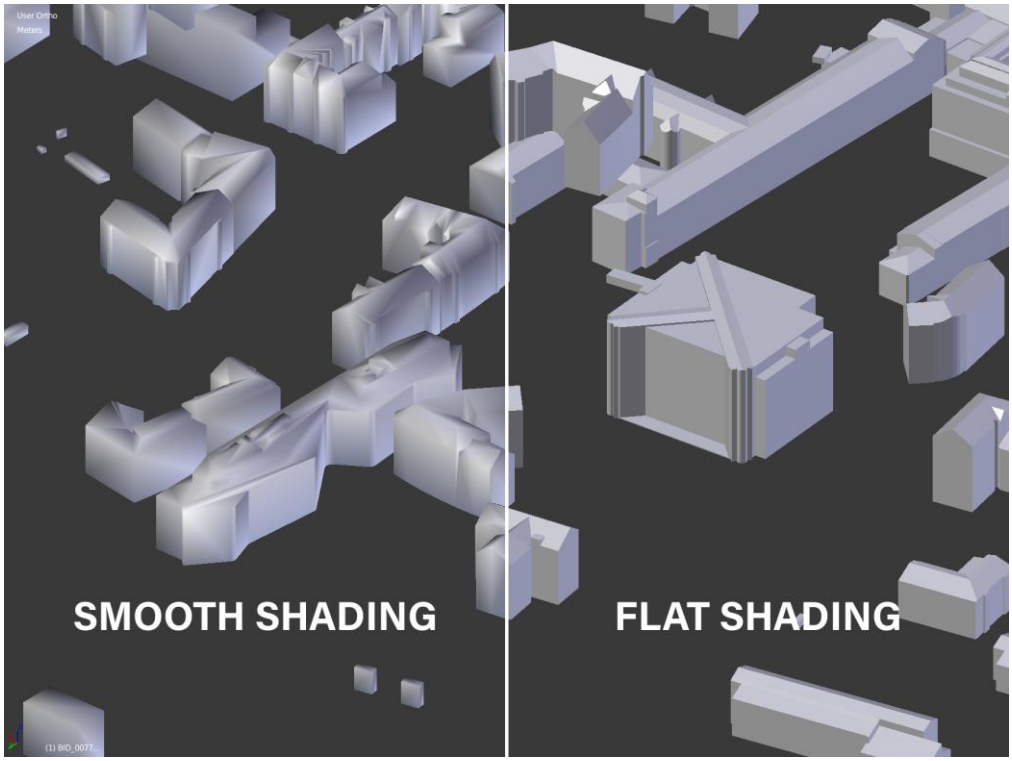
Objektit on nimetty Blenderissä alkuperäisten tiedostonimien mukaan

Joissain mallinnusohjelmissa voi olla tarpeen muuttaa editorin näkökentän asetuksia. Blenderissä muutettiin näkyvyys 10 kilometriin, käytettäessä metrijärjestelmän asetuksia. Near clip plane voi olla 1 m, koska mallia ei tulla tarkastelemaan senttimetrietäisyyksiltä.



Blenderissä asetettiin näkyvyys 10 km asti

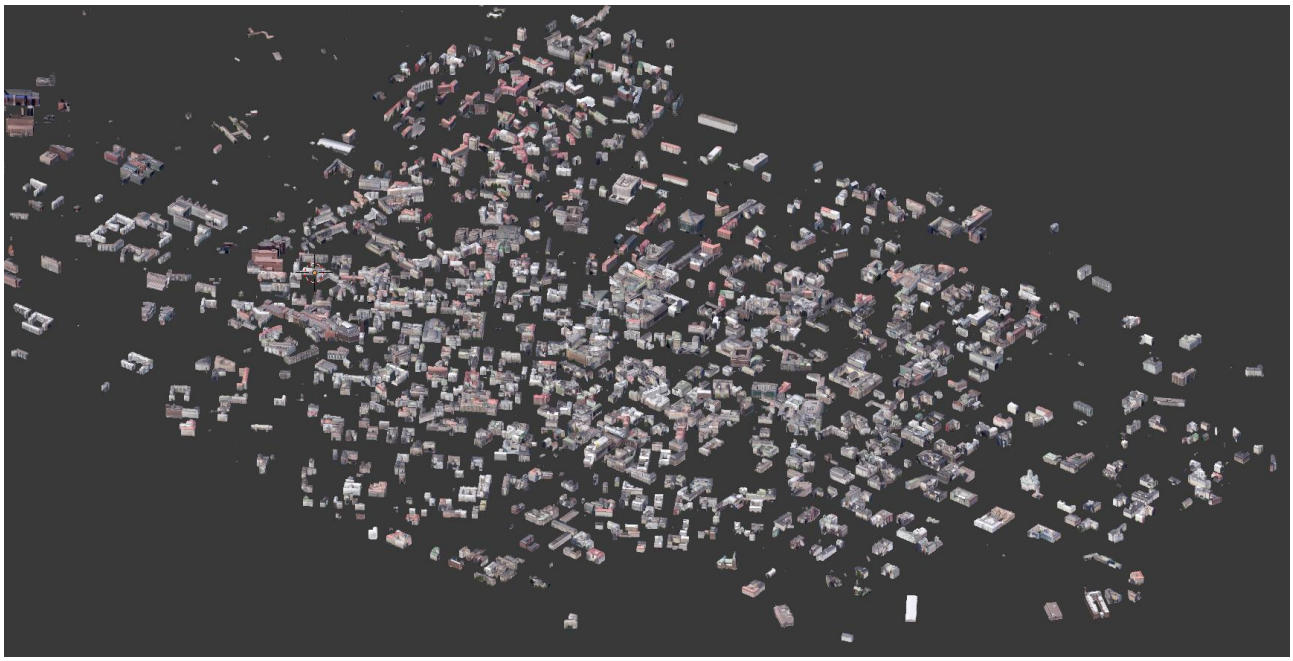
Lisäksi asetetaan tuotujen objektien varjostus / normaalit tasavarjostukselle.



Tasavarjostus on suositeltavaa tässä tapauksessa

Asetetaan vielä tekstuurit näkyville editorin perusnäkymässä. Blenderissä Cycles-moottoria

käytettäessä, tämä käy yksinkertaisesti vaihtamalla ”Viewport shading” -moodiksi textured.



Rakennukset ovat väärillä paikoilla

Tuodut objektit ja näin ollen rakennukset eivät ole vielä oikeassa suhteessa toisiinsa. Aiemmin mainitusta kml-tiedostosta löydetään objekteille oikeat korkeudet Z-akselille, mutta X- ja Y-akseleiden sijoitus käy helpoiten silmämääräisesti asettelemalla. Periaatteessa kml-tiedoston koordinaatit voitaisiin muuntaa metreiksi, mutta käytännön kokeilun perusteella silmämääräinen asettelu onnistuu alle 10 minuutissa.

Asetetaan kaikkien objektien Z-sijainti alla näkyvien kuvakaappausesimerkkien mukaisesti.

```
<Placemark id="COLLADA_BID_006baa64-603e-4c
  <name>BID_006baa64-603e-4c21-817c-82b17d9
  <Model>
    <altitudeMode>absolute</altitudeMode>
    <Location>
      <longitude>24.9275805</longitude>
      <latitude>60.1647454</latitude>
      <altitude>9.71</altitude>
    </Location>
```

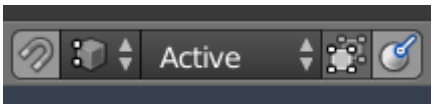
Kml-tiedostosta etsitään kutakin objektia vastaava Placemark ja sen altitude

Objektin korkeussijainti asetetaan kml-tiedoston mukaisesti

Kun korkeudet on asetettu, kannattaa siirtyä ortografiseen ylhäältä alas -näkymään. Näkyviin jätetään kaksi objektia, joista toista liikutetaan X- ja Y-akseleilla. Visuaalisesti etsitään oikea paikka rakennuksille. Tässä huomattiin, että kannattaa katsoa katujen linjoja ja erikoisen muotoisia rakennuskokonaisuuksia. Lopulta oikea paikka löytyy.

Rakennukset aseteltu oikein jokaisella akselilla

Objektien tarkassa sijoittelussa voi hyödyntää ”vertex snapping”-ominaisuutta. Tämä voi olla hyvin erilainen eri ohjelmissa. Blenderissä tapahtuu niin, että mennään edit-moodiin, snapping asetuksista snap elementiksi vertex ja snap targetiksi active.



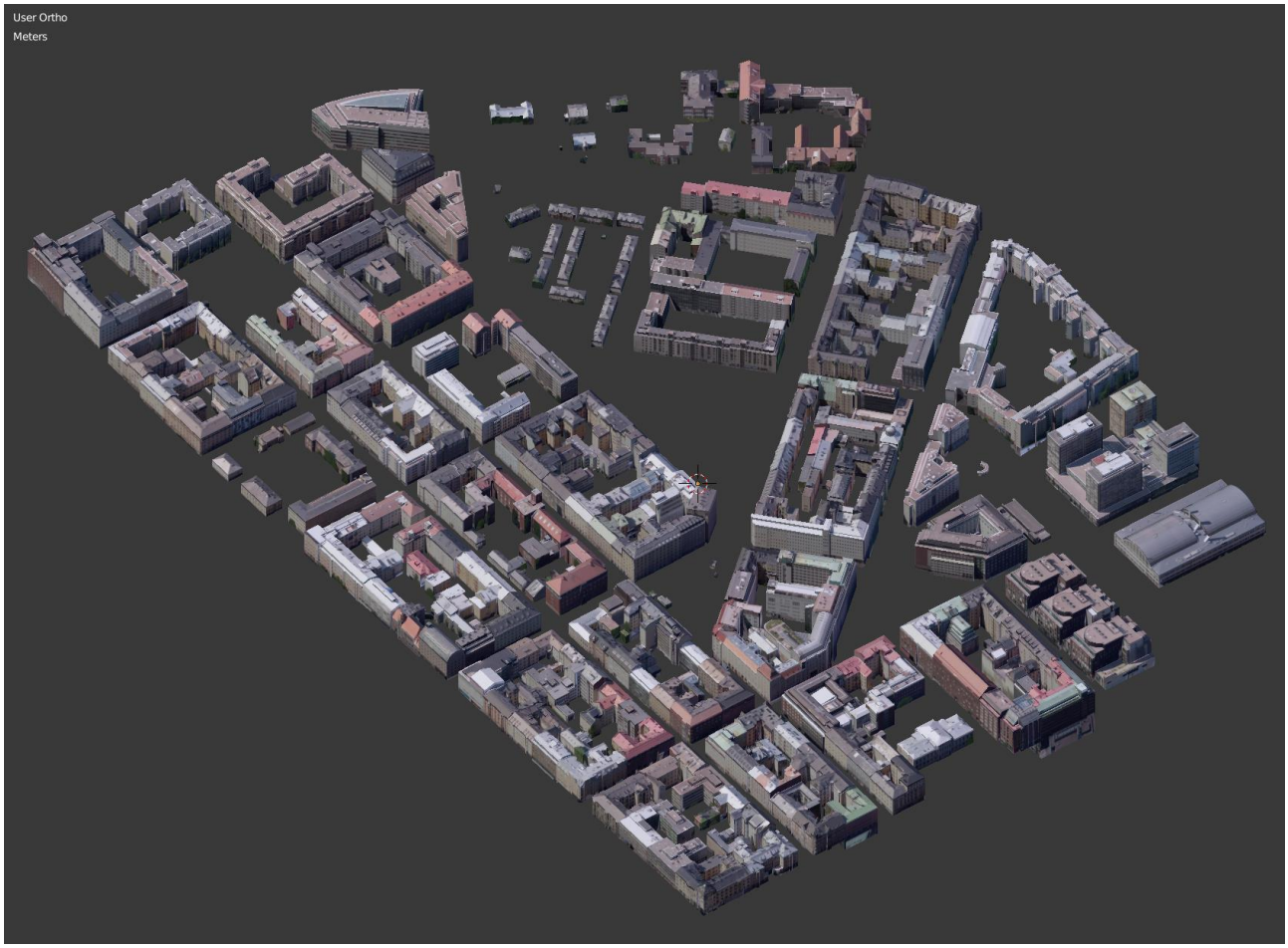
Nyt valitaan kaikki objektin verteksit (A), klikataan haluttu verteksi aktiiviseksi (2x shift + hiiren oikea), aktivoidaan liikutus (G), lukitaan Z-akseli (shift+Z) ja napsautetaan aktiivinen verteksi haluttuun kohdeverteksiin.

Tässä tulee olla tarkkana siinä mielessä, että jos vierekkäin napsautetut rakennukset eivät oikeasti ole vierekkäin, niin kaikkiin muihinkin objektin rakennuksiin tulee sama systemaattinen virhe.

Rakennusten kulmat ovat suunnilleen kohdakkain

Rakennusten kulmat on napsautettu X- ja Y-akseleilla kohdalleen

Nyt objektit voi yhdistää yhdeksi objektiksi. Yhdistetystä objektista on nyt helppo poistaa rakennukset, joita ei tarvita lopullisessa mallissa. Siistimisen jälkeen objektin origon voi siirtää sen keskelle ja objektin sijainnin nollata. Näin uusi siistitty malli on koordinaatiston keskikohdassa.



Nämä rakennukset haluttiin jättää lopulliseen malliin

3.3 Tekstuurien leipominen

3.3.1 Johdanto

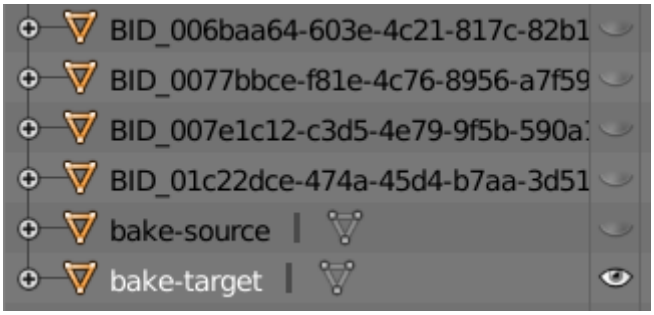
Jos mallia halutaan käyttää pelkästään Blenderin ja 3ds Maxin kaltaisella mallinnusohjelmalla eikä tekstuuridatan valaistuksenpoistoa haluta tehdä, voi tekstuurien leipomisen halutessaan jättää tekemättä. Jos tarkoituksena on käyttää mallia pelimoottorissa ja/tai tehdä valaistuksenpoisto, on tämä vaihe tehtävä.

Alkuperäinen malli on sirpaloitunut useaan tekstuuritiedostoon. 3D City Databasen viennillä määrä saatiin laskettua alle kahteen sataan, mutta tämä on edelleen aivan liikaa. Delighting-työkalu voi käsitellä vain yhden yhtenäisen tekstuurin, ja vaikka sitä ei käytettäisiin, olisi kymmenien tai satojen materiaalien luominen käsin pelimoottorissa täysin järjetön tehtävä.

Periaatteena on tehdä 3D-mallinnusohjelmassa kopio aiemmin yhdistellystä ja siistitystä objektista. Tästä kopiosta siivotaan vielä ylimääräisiä pintoja, jotka eivät näy renderöinneissä, eli ne veisivät turhaa tilaa leivotulta tekstuurilta. Siistitylle kopiolle lasketaan automaattialgoritmilla uudet UV-koordinaatit. Objektilla on siis yksi UV-kartta, ja jokaisella objektin kolmiolla on uniikit koordinaatit U 0-1 ja V 0-1 akseleiden sisällä. Lopuksi siirretään alkuperäisen mallin väri-informaatio tähän kopioon, jolloin muodostuu yksi tekstuuri, joka sisältää kaiken objektin tarvitseman väri-informaation.

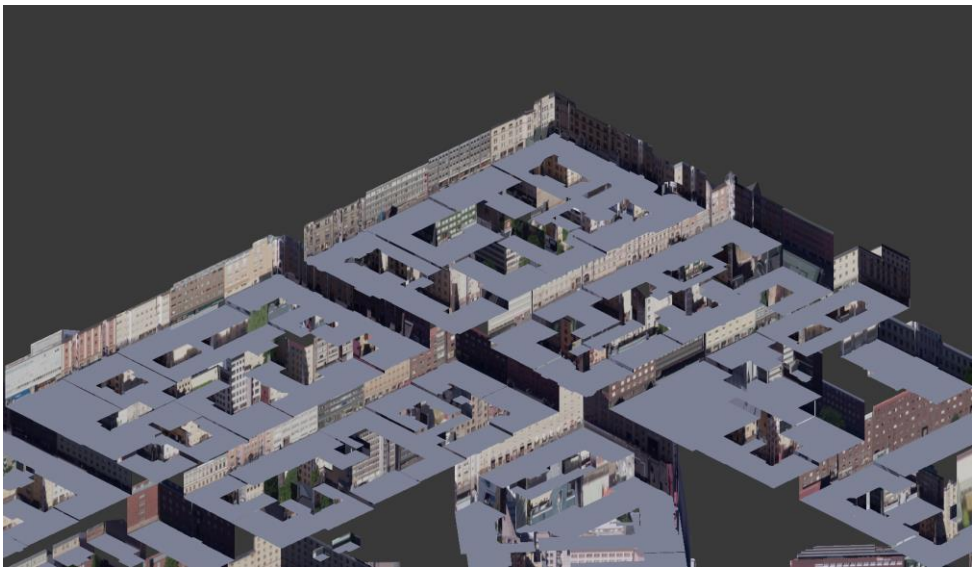
3.3.2 Kopion tekeminen ja siivoaminen

Aiemmin yhdistellyn mallin voi nimetä esim. ”bake-source”. Tästä tehdään kopio (huom. ei linkitetetyä kopiota, vaan täysin itsenäinen kopio). Kopion voi nimetä ”bake-target”. Bake-sourcen voi piilottaa näkyvistä toistaiseksi.



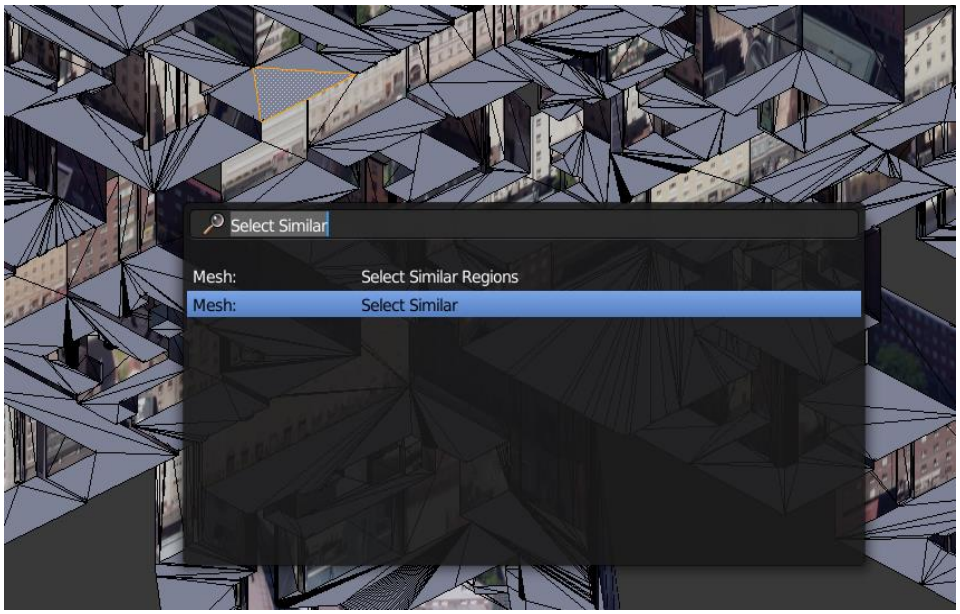
Alkuperäiset objektit (BID_...), siistitty yhdistelmäobjekti (bake-source)
ja siitä tehty kopio (bake-target)

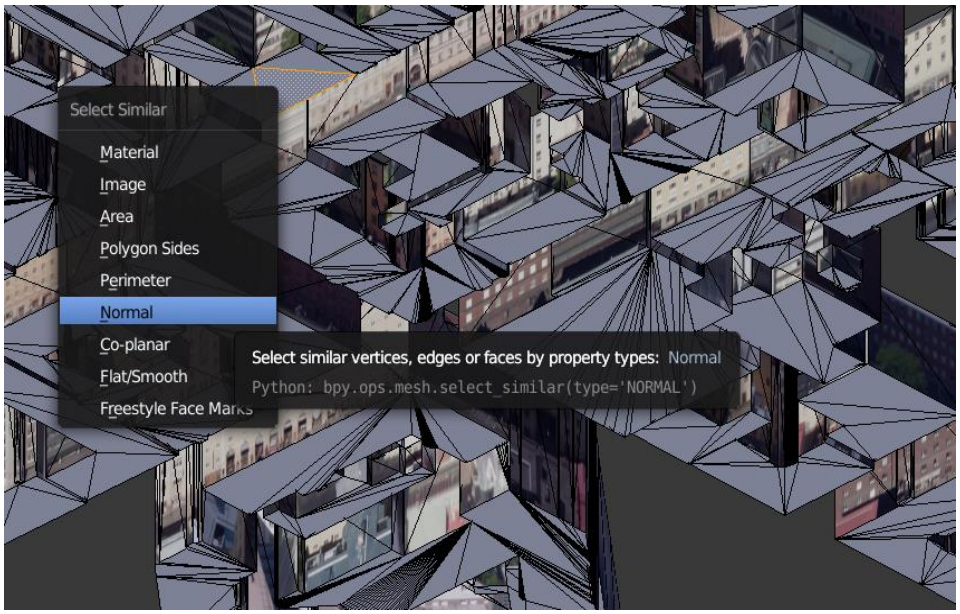
Nyt poistetaan ylimääräiset pinnat leivonnan kohdeobjektista. Periaatteessa mitään piiloon jääviä pintoja ei kannata säästää, ne vain vievät tekstuuritilaa tärkeämmiltä pinnoilta. Käytännössä on löydettävä kompromissi, jotta siistimiseen ei kulu liikaa aikaa.

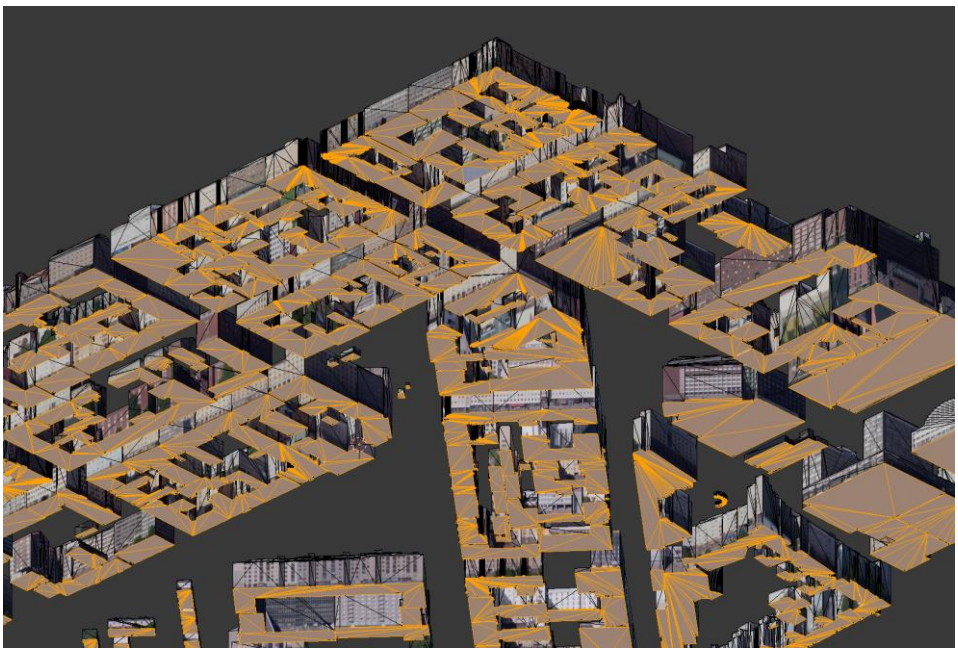


Rakennusten pohjissa on visualisoinnin kannalta turhia pintoja

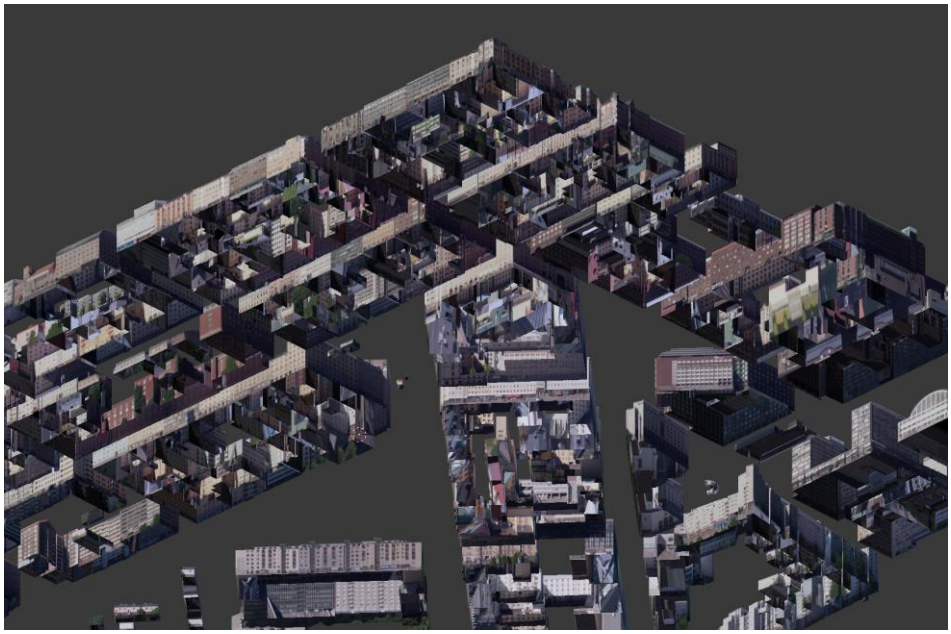
Blenderissä on kätevä toiminto, jonka avulla voidaan poistaa rakennusten pohjista pinnat, jotka eivät tule visualisoinneissa näkymään. Valitaan edit-moodissa yksi face jonkin rakennuksen pohjasta. Tämän pinnan normaalin tulisi osoittaa alaspäin. Nyt voidaan käyttää toimintoa select similar by normal. Tämä valitsee kaikki pinnat, joiden normaali osoittaa käyttäjän määrittelemän toleranssin rajoissa samaan suuntaan kuin valitun pinnan normaali. Rakennuksissa ei pitäisi olla näkyvissä osissa pintoja, joiden normaali osoittaa suoraan alaspäin.







Valittuna kaikki pinnat, joiden normaali osoittaa alaspäin



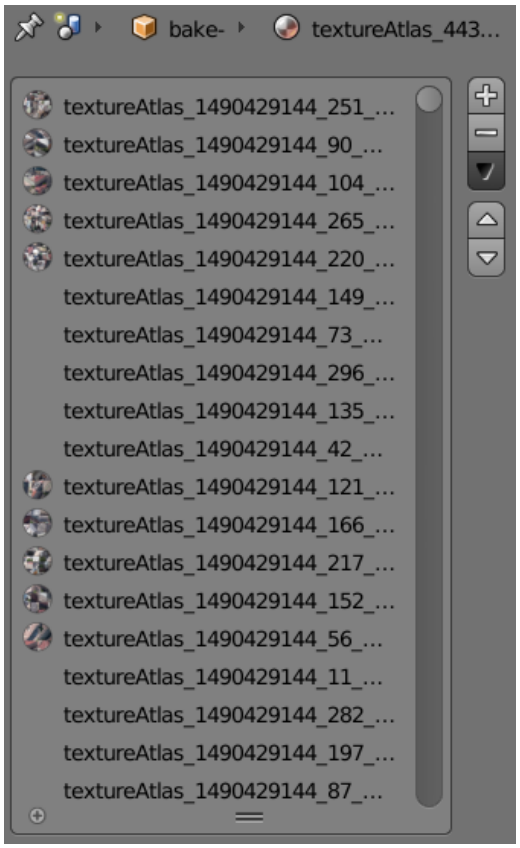
Alapinnat poistettu

Alapintojen poistamisen jälkeen on mahdollista käsin poistaa turhia pintoja esimerkiksi vierekkäisten rakennusten seinien välistä. Ajankäytöllisesti tämä on hyvin hidasta saavutettuun hyötyyn nähden, joten edes tässä kokeilussa ei lähdetty siivoamaan tämän pidemmälle.

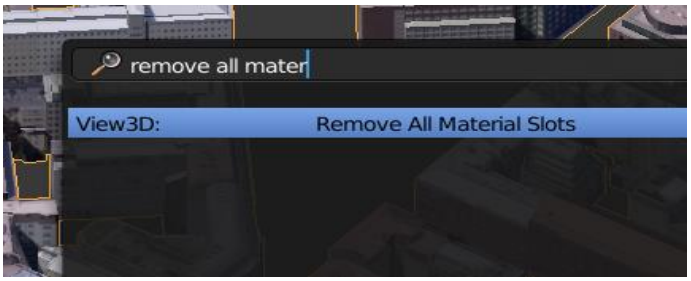
3.3.3 UV-koordinaattien generoiminen

Geometriaan ei tehdä enää muutoksia. Tarvitaan vain uudet UV-koordinaatit, jotka kannattaa generoida automaattista algoritmia käyttäen. Aluksi poistetaan objektista kaikki materiaalit, joita on

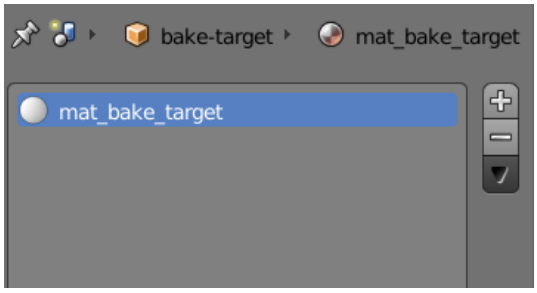
yhtä monta kuin tekstuuriakin. Blenderissä materiaalien poistaminen käy ”Remove all material slots”-toiminnolla. Lopuksi luodaan yksi materiaali, joka kattaa koko objektin.



Jokaista tekstuuria kohden on yksi materiaali



Poistetaan kaikki materiaalit leivonnan kohdeobjektista



Luodaan yksi materiaali, joka kattaa koko objektin

UV-projektioon havaittiin toimivaksi Blenderin automaattinen ”Smart UV Project”. Vakioasetukset toimivat muuten hyvin, mutta ”Island Margin” on syytä muuttaa esimerkiksi arvoon 10 cm. Näin projisoidut erilliset UV-saarekkeet eivät ole täysin toisissaan kiinni. 8k-tekstuurilla saarekkeiden väliin havaittiin jäävän minimissään n. 2-4 pikseliä tällä asetuksella.

Automaattisen UV-projektion asetukset

Algoritmin suorittamisessa voi kestää 2-5 minuuttia.

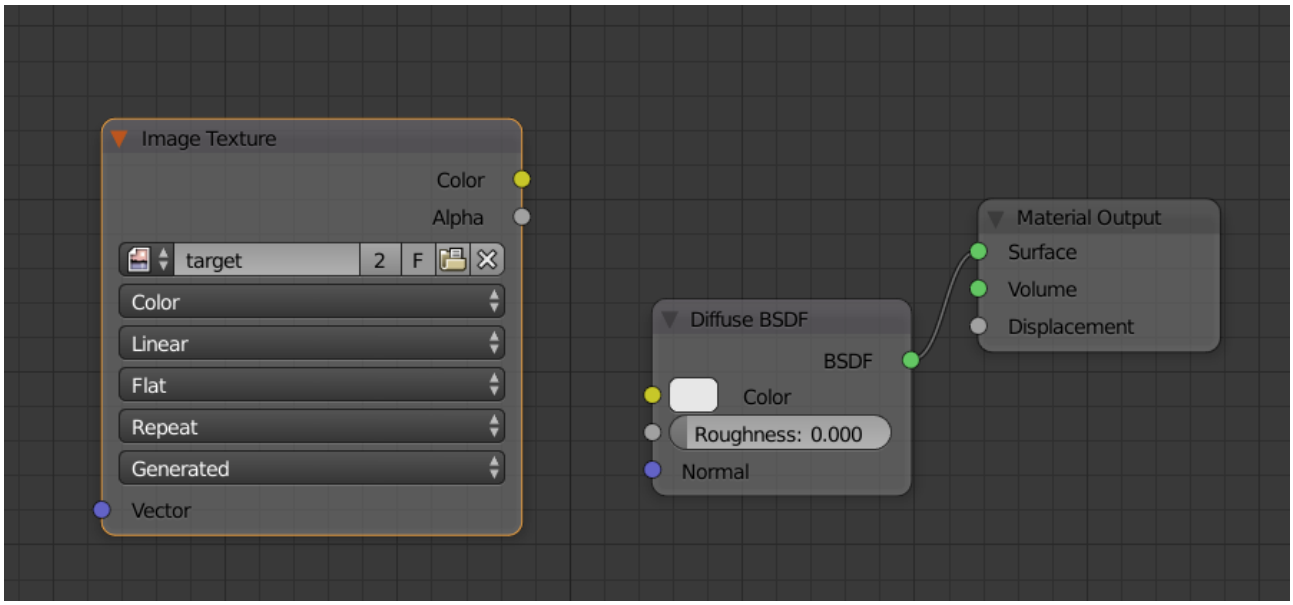
Nyt tekstuurileivonnan kohdeobjekti on geometrisesti siisti ja sillä on vain yksi materiaali sekä toimivat UV-koordinaatit.

3.3.4 Väri-informaation leipominen

Kokeilussa käytettiin väritekstuurin leivontaan Blenderin Cycles-renderöintimoottoria. Myöhemmin leivotaan valaistusinformaation poistamiseen liittyen muita erikoistekstuureja Substance Designer -ohjelmalla.

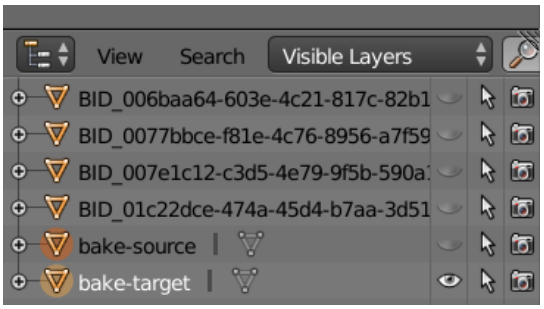
Blenderissä luodaan halutun kokoinen kuva, johon värit leivotaan lähdeobjektista. Tässä kokeilussa käytettiin 8192 * 8192 pikselin kokoista kuvaa, joka riitti kattamaan kaikki halutut rakennukset riittävällä tarkkuudella.

Leivonnan kohdetekstuurin valitseminen on epäintuitiivinen operaatio Blenderissä. Blender tallentaa leivonnan tuloksen kuvaan, joka on valittuna material node -editorissa, mutta kuva ei saa olla kytkettynä materiaalissa mihinkään muuhun nodeen. Katso kuva. Huomaa oranssi reunus Image Texture -noden ympärillä, väri kertoo että node on viimeksi valittu eli tässä tapauksessa aktiivinen baking target.



Blenderin epäintuitiivinen tapa valita tekstuurileivonnan kohde

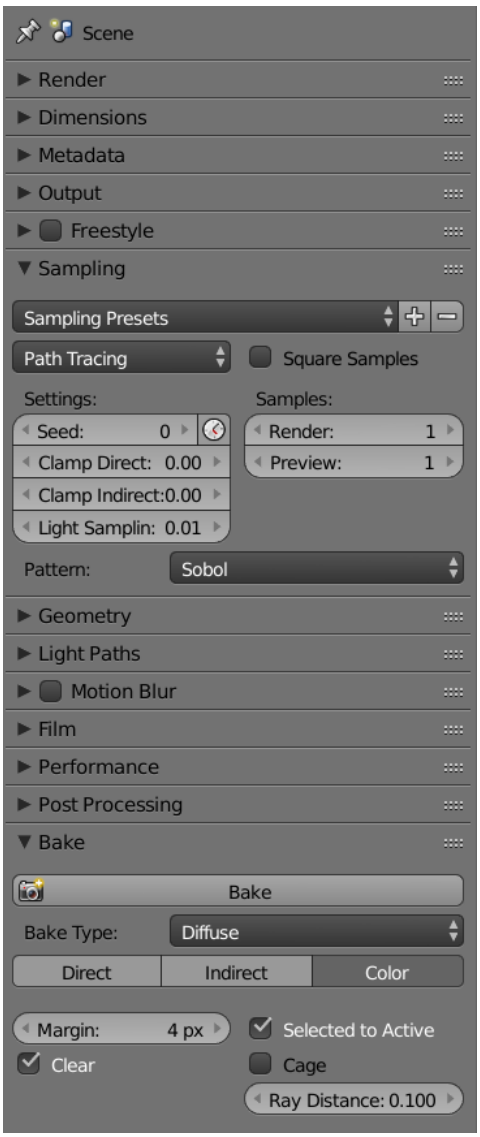
Leivonta tehdään käyttäen ”Selected to Active”-asetusta, eli Blender leipoo valitut objektit aktiiviseen objektiin. Valitaan siis Outliner-ikkunasta ensin bake-source, sitten shift-näppäintä samalla painaen bake-target. Katso kuva. Molemmat objektit ovat valittuina, mutta bake-target on näistä viimeksi valittu eli aktiivinen. Näin Blender tietää leipoa tekstuurit bake-source-objektista bake-target-objektiin.



bake-source ja bake-target ovat valittuina, näistä bake-target on aktiivinen

Voidaan siirtyä leivonnan asetuksiin. Sampling-asetuksista voidaan laskea render-samplet yhteen prosessin nopeuttamiseksi. Koska tässä siirretään vain väri-informaatio objektista toiseen, ei synny myöskään vähäisistä sampleista johtuvaa kohinaa.

Bake-asetuksista valitaan leivontatyypiksi diffuse, jätetään valituksi ainoastaan color, ruksitetaan selected to active ja asetetaan ray distance -arvoksi 0.1. Marginaali tulee muokata sopivaksi, 8k-tekstuurille neljä pikseliä on sopiva.

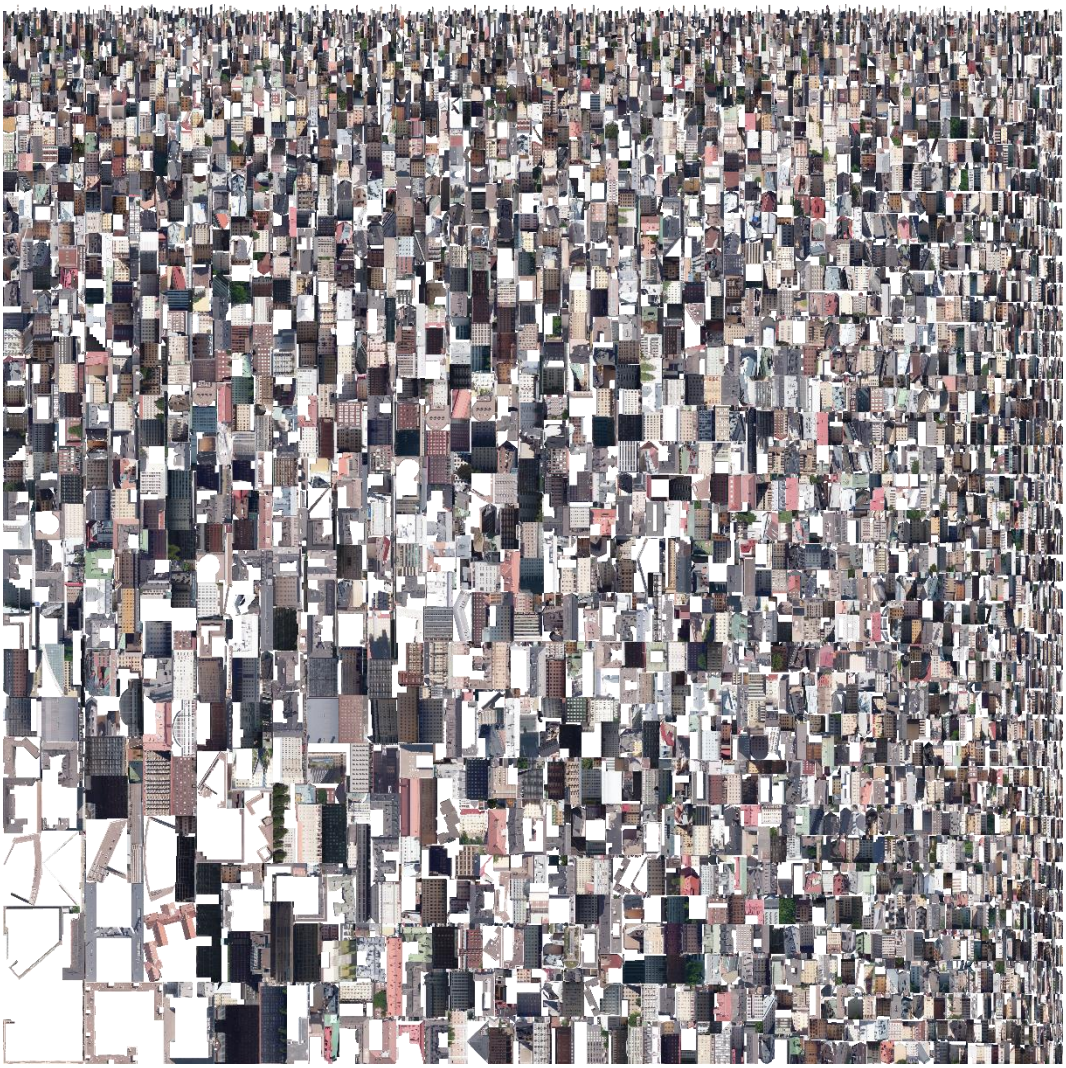


Hyväksi havaitut leivonta-asetukset

8k-tekstuurin leivontaan meni keskimäärin noin viisi minuuttia. Blender ei toistaiseksi voi tehdä leivontaa näytönohjaimella, eli prosessin kesto riippuu täysin tietokoneen suorittimesta.

Esimerkin mukaisen rakennusmassan sovittaminen yhteen 8k-tekstuuriin ei aiheuttanut merkittävää häviötä tekstuurien laadussa. Jos alkuperäisen tekstuurin resoluutio on korkeampi tai halutaan suurempi määrä rakennuksia, on joko nostettava leivottavan tekstuurin kokoa tai jaettava alue useampaan osaan. Tämän kokeilun ajanhetkellä yli 8k-tekstuurien käyttöä ei voi suositella, sillä esimerkiksi pelimoottorit eivät osaa käyttää 16k-tekstuureja. Jos rakennusmassa jaetaan vaikka kahteen osaan, suoritetaan leivonta erikseen molemmille osille.

Rakennukset leivotulla tekstuurilla



Yksi tekstuuri sisältää kaikkien rakennusten pintavärit

Leivottu tekstuuri lähempää tarkasteltuna

3.4 Tekstuurien valaistusinformaation poistaminen

Satelliittikuvista yhdistettyihin tekstuureihin on tallentunut sen hetken valot ja varjot kiinteästi rakennusten pintaan palaneina. Jos kaupunkidatasta tehtyä 3D-mallia käytetään visualisoinnissa, sitä todennäköisesti halutaan valaista vapaasti eri suunnista. Näin ollen on syytä poistaa tai edes vähentää tekstuureihin tallentunutta valaistusta.

Tässä kokeilussa käytettiin uutta Unity-pelimoottorin päälle rakennettua työkalua, Unity De-Lighting Tool. Sen löytää dokumentaation kera GitHubista: github.com/Unity-Technologies/DeLightingTool

Työkalun asennukseen ja käyttöön on hyvät ohjeet GitHubissa, joten tässä dokumentaatiossa siirrytään suoraan ohjelman käyttöön.

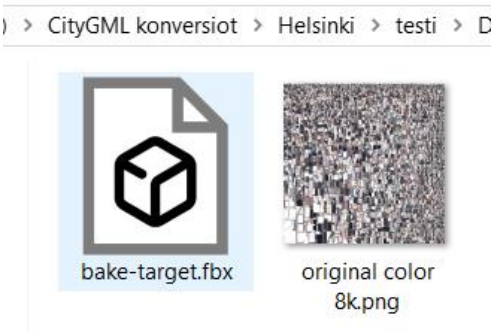
3.4.1 Erikoistekstuurien leivonta valaistuksenpoistotyökalua varten

De-lighting tool käsittelee vain tekstuureja, se ei ota syötteenä 3D-mallia lainkaan. Pakollisia syötteitä työkalulle ovat:

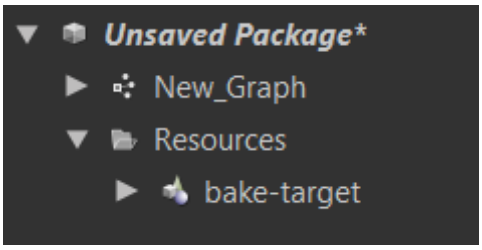
- Perusvärikartta (jossa poistettava valaistusinformaatio on)
- Normaalikartta (object space)
- Taittunut normaalikartta (bent normals, object space)
- Ambient occlusion

Näiden tekstuurien luomiseen käytetään tässä esimerkissä maksullista Substance Designer -sovellusta, joka on todettu laadukkaaksi ja luotettavaksi ratkaisuksi tekstuurileivontaan. Ohjelma sisältää myös paljon muuta toiminnallisuutta. Vaihtoehtoja Substance Designerille on mm. Knald ja xNormal.

Blenderistä viedään ”bake-target”-objekti FBX-muodossa ulos vakioasetuksilla. Tässä vaiheessa on siis syytä olla kaksi tiedostoa, viimeistelty rakennukset sisältävä FBX-tiedosto ja leivottu värikartta.



Substance Designerissa luodaan uusi Substance-graafi, sen asetuksilla ei ole merkitystä. FBX-tiedosto linkitetään graafin resurssiksi vetämällä se kansiosta ohjelmaan tai valitsemalla Unsaved package > Link > 3D Mesh.

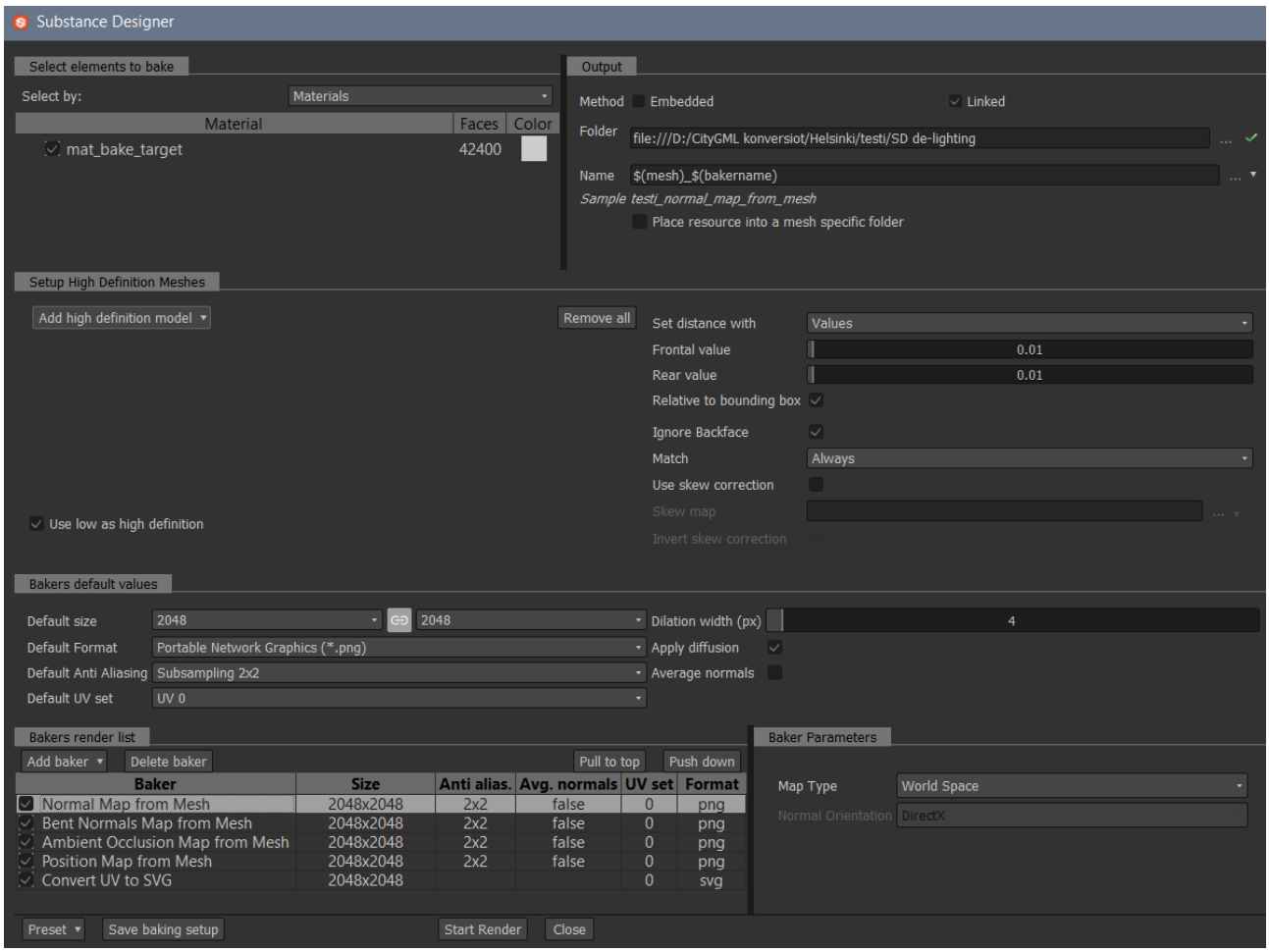


FBX-tiedosto on linkitetty Substance-graafiin

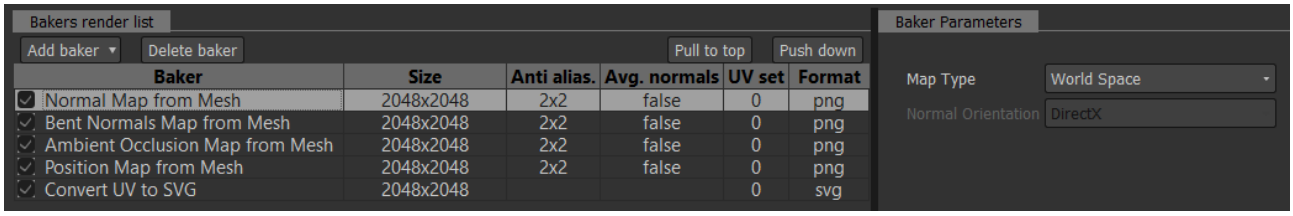
Tekstuurileivonnan asetukset löytyvät klikkaamalla hiiren oikealla linkitettyä FBX-tiedostoa > Bake model information. Valaistuksenpoistoa varten leivotaan siis normal map, bent normals map, ambient occlusion ja position map. Lisäksi tarvitaan UV-koordinaateista luotu tiedosto, josta luodaan myöhemmin maski joka erottaa värikartasta alueet, jotka tarvitsevat valaistuksenpoiston.

Leivonnan lopputulokseen vaikuttavia asetuksia on paljon, ja kokeilussa meni huomattavasti aikaa oikeiden asetusten löytämiseen. Väärillä asetuksilla valaistuksenpoisto tuotti yleensä täysin kelvottoman lopputuloksen. Hyväksi havaitut asetukset on esitetty alla kuvankaappauksina.

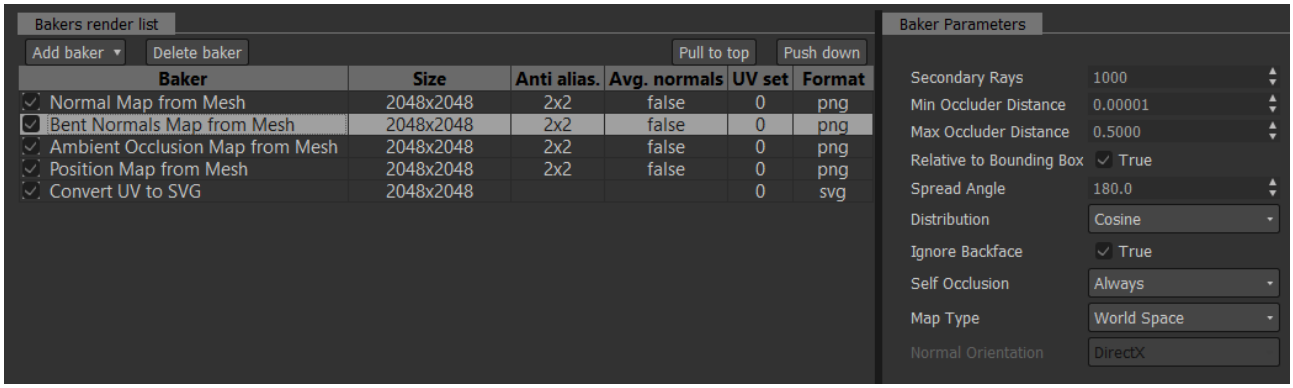
Huomattavaa on, että vaikka värikartta on 8k-resoluutiolla, leivottiin nämä kartat 2k-resoluutiolla. Tähän on kaksi syytä. 4k ja 8k bent normals ja ambient occlusion -tekstuurien leivontaan kuluu hyvillä asetuksilla todella paljon aikaa, jopa useita tunteja yhtä tekstuuria kohden. Lisäksi Unity De-lighting tool vaikutti toimivan epävarmasti suurilla tekstuureilla. Siksi päädyttiin kokeilemaan, että voiko pienemmän valaistusinformaatiosta siivotun 2k-tekstuurin yhdistää kuvankäsittelyohjelmassa lakuperäiseen 8k-tekstuuriin, niin että molempien hyvät puolet saadaan yhdistettyä.



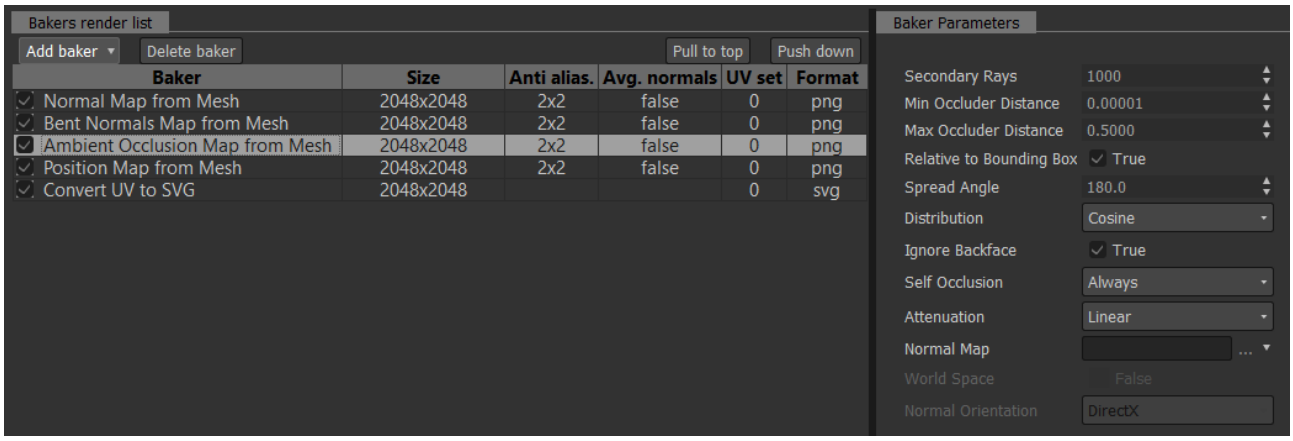
Yleiskatsaus Substance Designer texture baking -asetuksiin



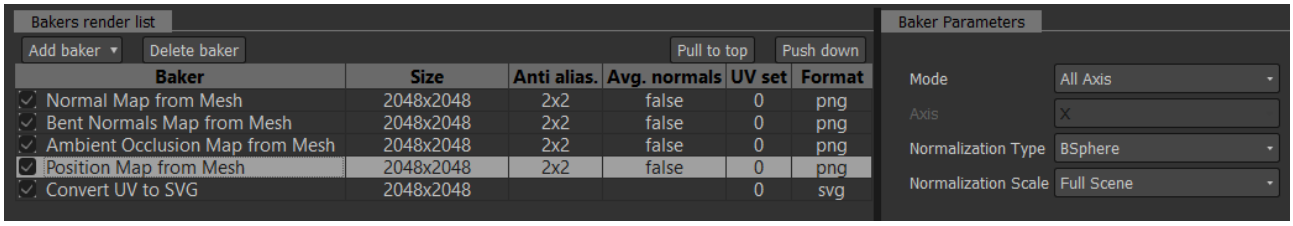
Normal Map from Mesh



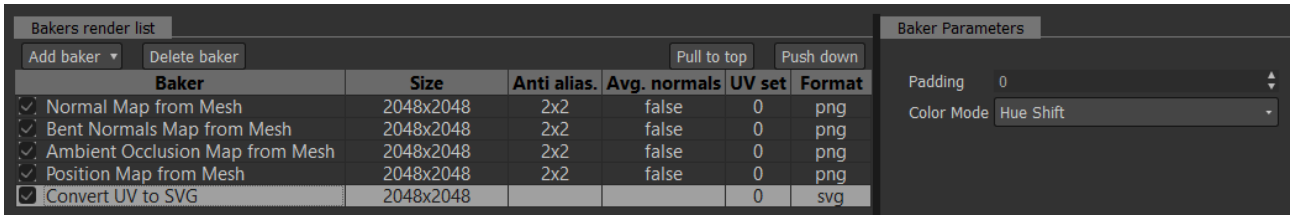
Bent Normals Map form Mesh



Ambient Occlusion Map from Mesh



Position Map from Mesh



Convert UV to SVG

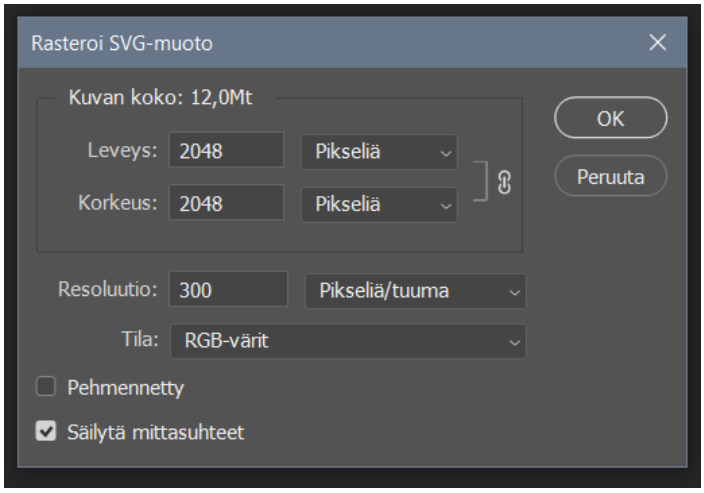
Tekstuurit tallennettiin PNG-muodossa, jonka jälkeen ne avattiin Adobe Photoshopilla ja tallennettiin TIFF-formaatissa. Valaistuksenpoistotyökalu ei jostain syystä aina lue PNG-tiedostoja oikein, ja toisaalta jos Substance Designerissa tallennetaan suoraan TIFF-muodossa, tallentuvat tekstuurit 32-bittisellä värisyvyydellä, kun Unity pystyy avaaamaan vain 8- tai 16-bittisiä värejä.

Kartat kannattaa nimetä seuraavasti, jotta Delighting tool osaa valita niille suoraan oikeat asetukset. Nimen alkuosa (tässä tapauksessa ”target”) voi olla itse valittu.

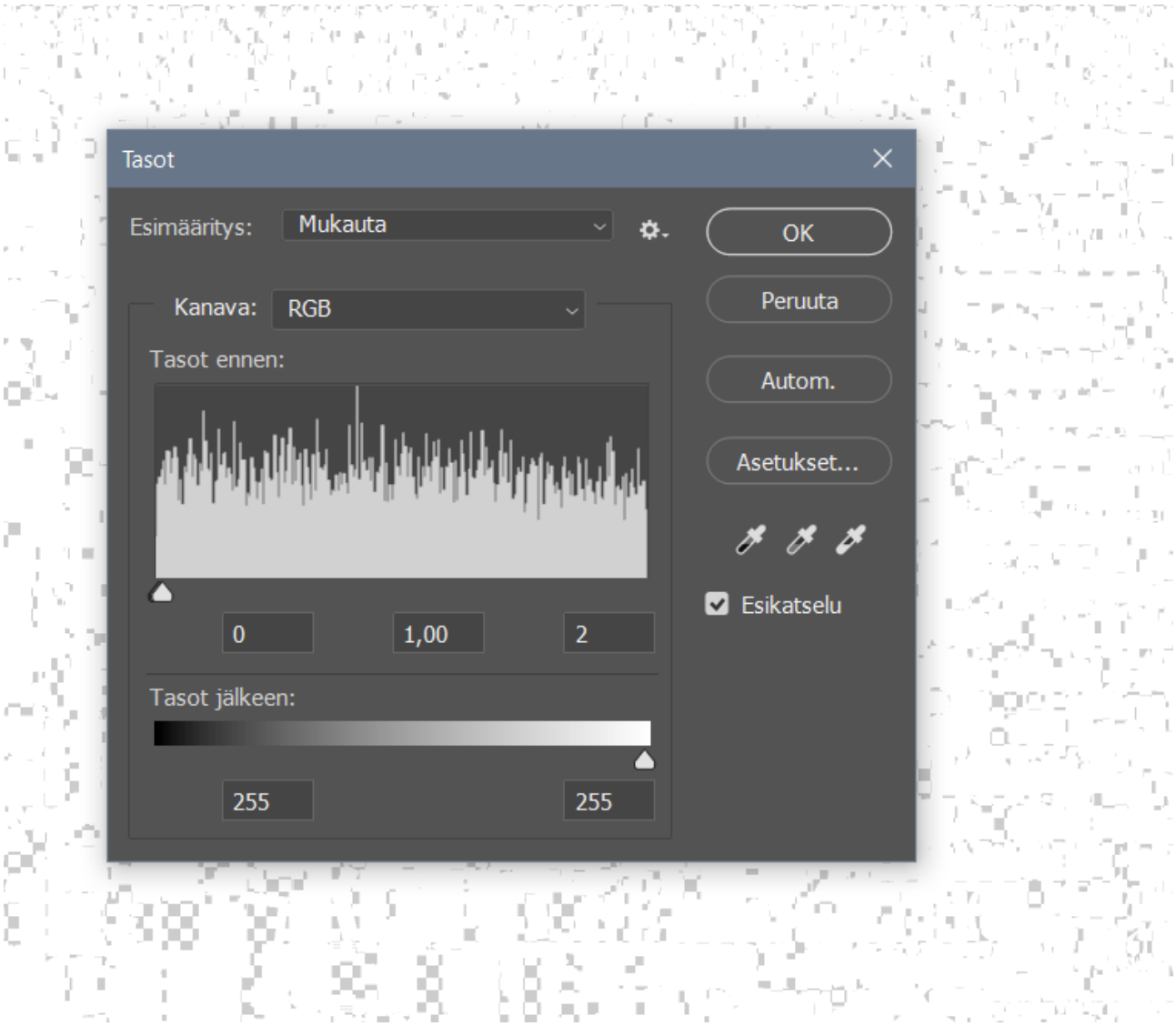
Normal map =	target_dln
Bent normals map =	target_dlbn
Ambient occlusion map =	target_dlao
Position map =	target_position

Vielä tarvitaan yksi tekstuuri, jossa on yhdistettynä värikartta ja maski. Avataan aiemmin leivottu

8k-väritekstuuri, ja muunnetaan se 2048 px kokoon käyttäen bilineaarista interpolaatiota. Avataan myös leivottu UV SVG -tiedosto ja rasteroidaan se resoluution 2048. Säädetään kuva levels-työkalulla valkoiseksi.

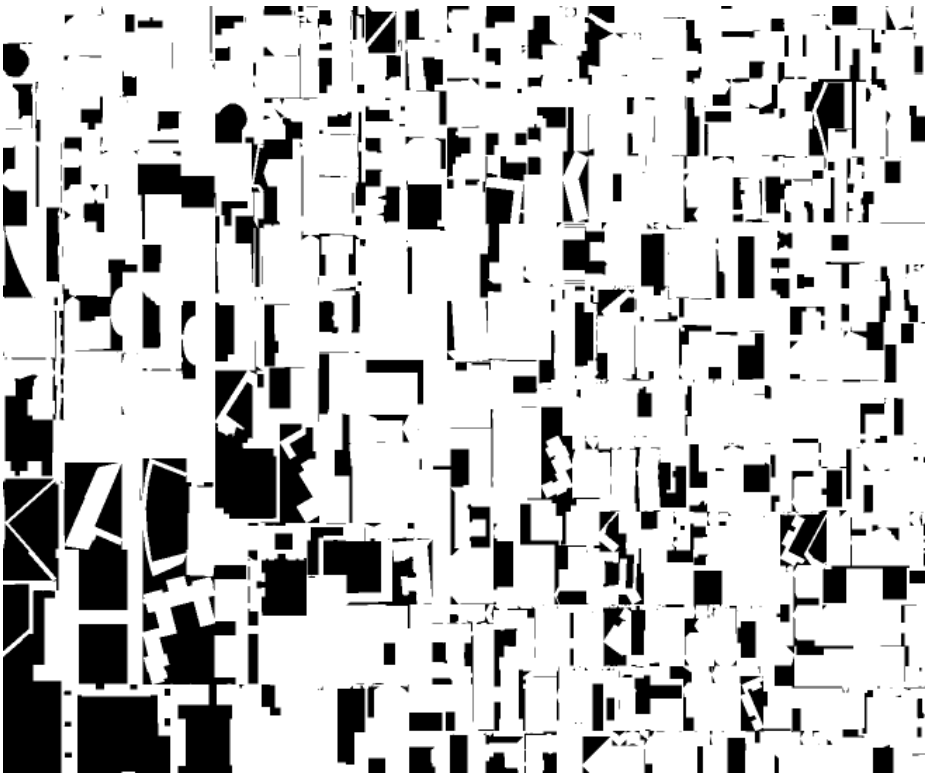


UV SVG:n rasterointi



Kaikki pikselit, jotka eivät ole läpinäkyviä, on säädetty valkoiseksi

Lopuksi lisätään pohjalle uusi taso, joka täytetään mustalla ja yhdistetään kaikki tasot. Lopputuloksena on mustavalkoinen maski, jossa mustalla on alueet jotka eivät sisällä oleellista väri-informaatiota.



Mustavalkoinen maski

Tämä mustavalkoinen maski lisätään 2048 pikselin kokoon muutetun värikartan alfa-kanavaan.

Maski on värikartan alfa-kanavassa

Tiedosto tallennetaan TIFF-muodossa, ja nimetään esim. ”target-dlbc.tiff”.

Kaikki valaistuksenpoistoon tarvittavat tekstuurit on luotu

3.4.2 Valaistuksenpoisto

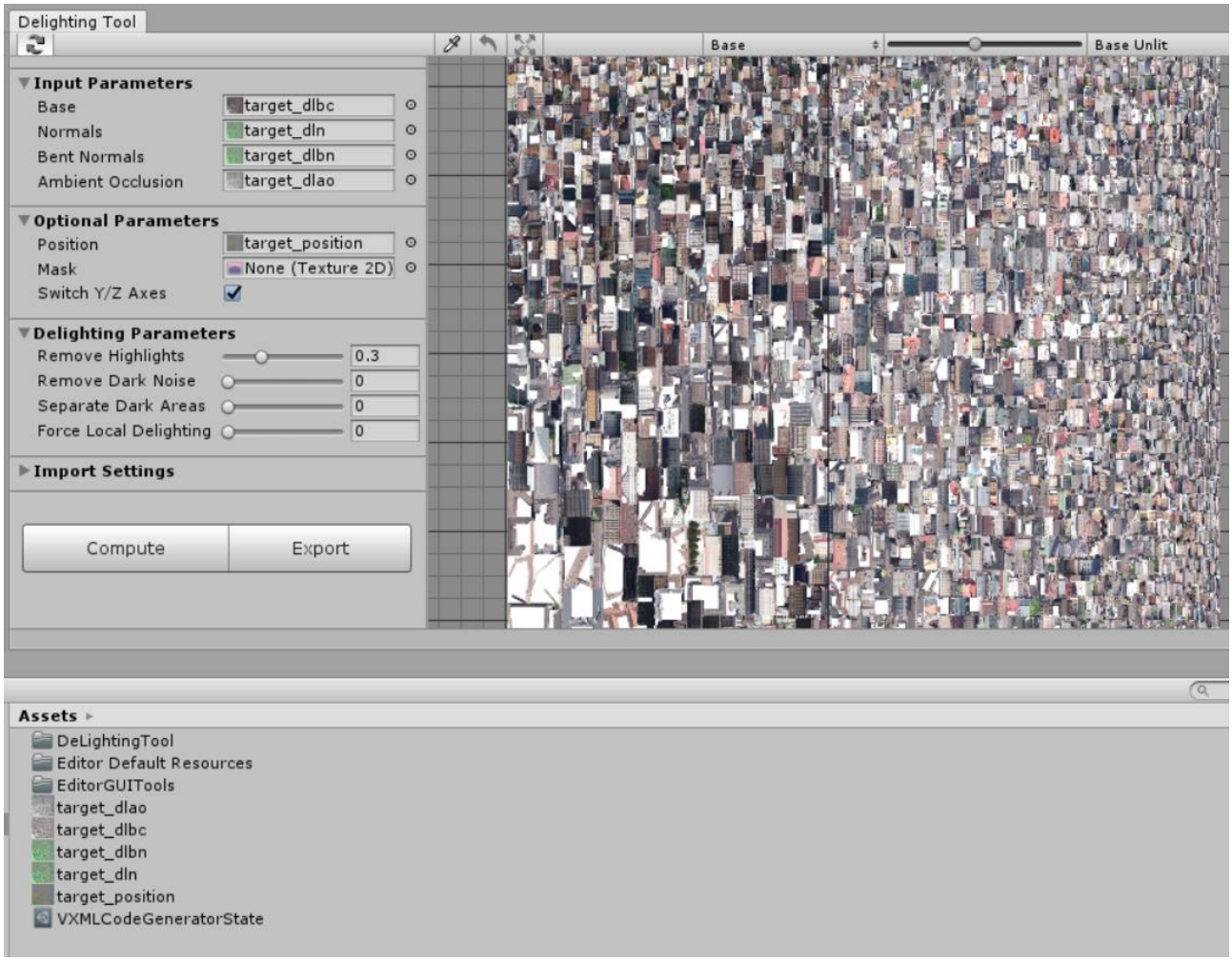
Unity Delighting tool on tarkoitettu melko tasalaatuisten materiaalien valaistuksenpoistoon. Kohdekäyttäjiä ovat esimerkiksi fotogrammetriamalleja tuottavat tahot. Työkalu ei käytä tekoälyyn pohjautuvaa algoritmia, ja siksi oli epäilys, että työkalu ei välttämättä toimi ison monimuotoisen rakennusmassan valaistuksenpoistossa. Lisäksi sen havaittiin olevan hyvin tarkka leivottujen tekstuureiden asetuksista. Kokeilussa löydettiin kuvainnollisesti kymmenen tapaa luoda huono lopputulos, ja yksi tapa luoda hyvä lopputulos. Tässä vaiheessa menetelmän toistettavuus jää arvoitukseksi. Kokeilun testiaineistolla saatiin aikaan onnistuminen, mutta ei tiedetä, miten erilaisella aineistolla tulee käymään.

GitHubista löytyvä dokumentaatio on varsin kattava, eikä tässä ole syytä käsitellä Unityn tai

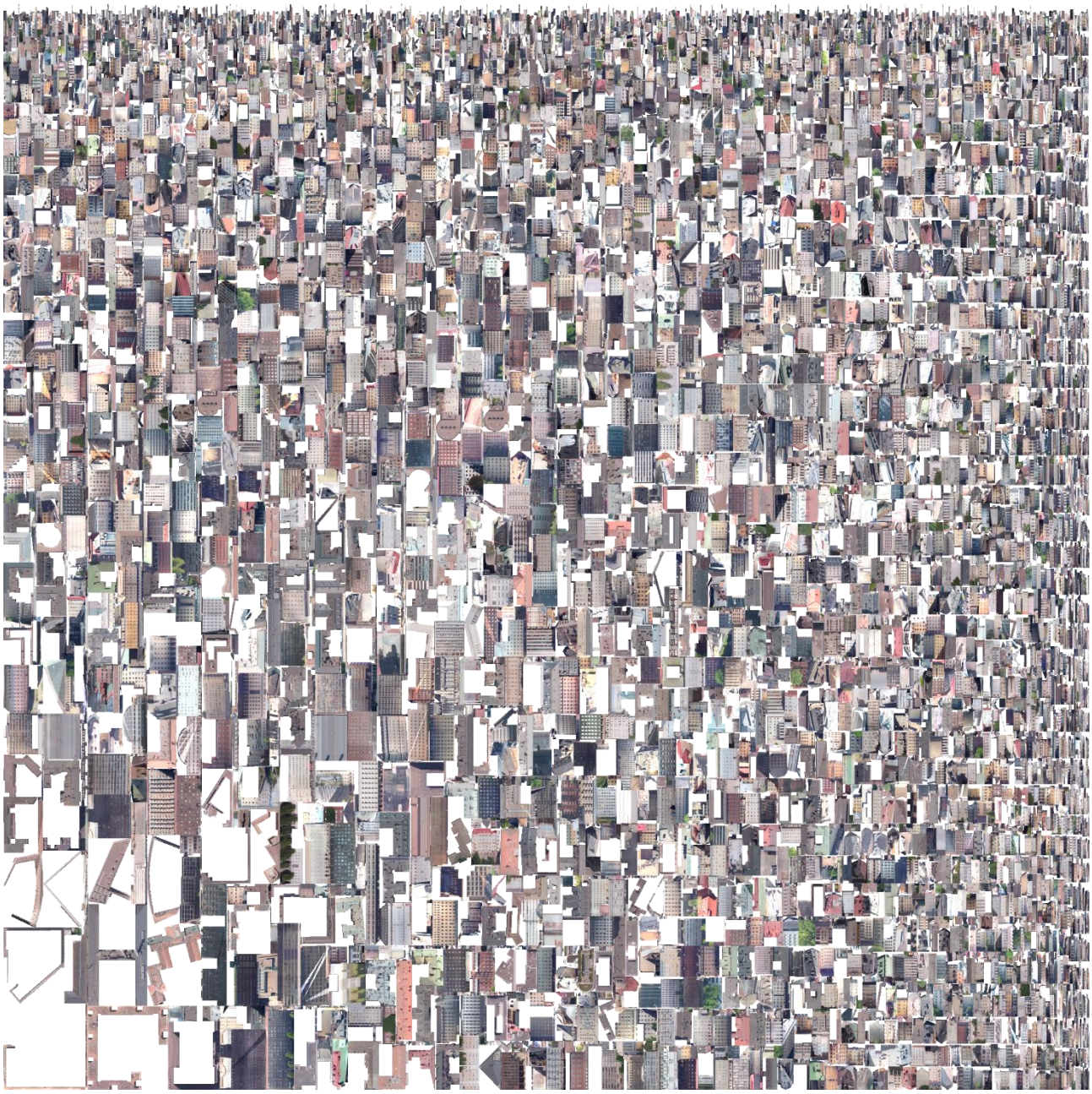
Delighting työkalun asennusta. Siirrytään suoraan valaistuksenpoistoon, joka on varsin suoraviivainen prosessi, kun leivotut tekstuurit on luotu ja Unityssä on auki uusi tyhjä delighting-projekti.

Jos tekstuurit nimettiin kuten aiemmin ohjeistettiin, ne voi tuoda projektiin ja Delighting tool -lisäosa tunnistaa ne automaattisesti. Näin niiden asetukset tulevat automaattisesti oikein. Oleellista on, että kaikista muista paitsi base color tekstuurista on sRGB ruksi otettu pois päältä, pakkaus otettu pois päältä, resize algorithm on ”Mitchell” ja maksimikoko 8192 tai vähintään leivottujen tekstuurien resoluutio.

Tuodut tekstuurit vedetään Delighting tool -välilehdellä oikeisiin paikkoihin, jolloin ohjelma muodostaa yleensä automaattisesti valaistusinformaatiosta siivotun tekstuurin.



Tässä kokeilussa vaadittiin vielä ”Switch Y/Z Axes” -toiminnon ottaminen käyttöön, jolloin lopputuloksena syntyi tasalaatuinen tekstuuri ilman merkittäviä värivirheitä.

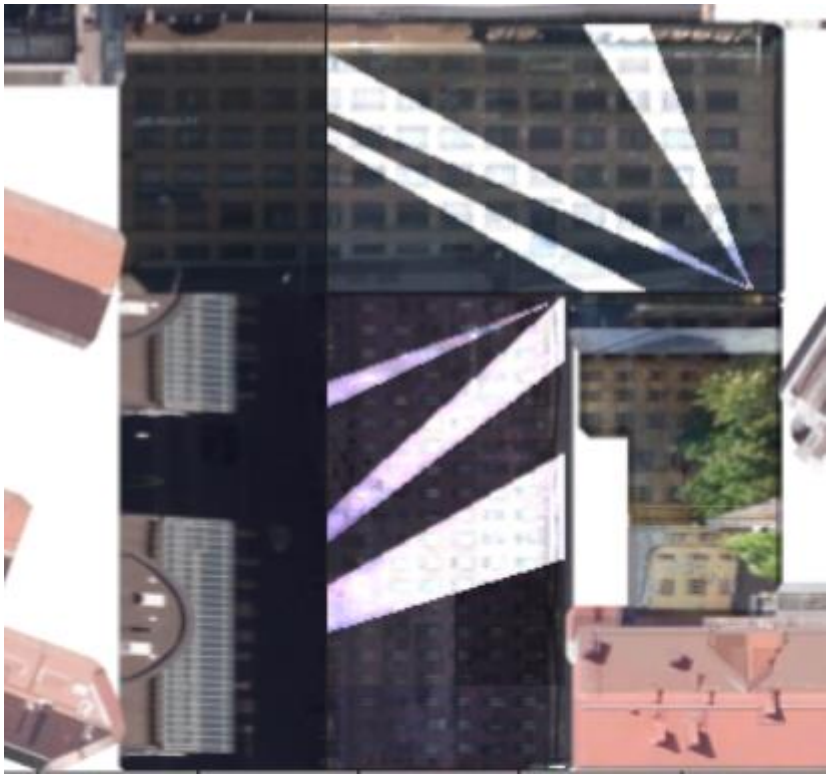


”Delighted” perusväritekstuuri

3.4.3 Lopputuloksen arviointi

Pikaisesti katsottuna valaistuksenpoiston lopputulos vaikuttaa hyvältä: varjot poistuvat todella tehokkaasti ja kokonaisuudessaan tekstuurin värikirkkaus vaikuttaa tasaiselle. Lähempää tarkasteltuna joukossa on muutamia yksittäisiä rakennuksia, joiden sävy tai kirkkaus vaikuttaa virheelliselle, mutta suurin osa on hyväksyttävissä.

Valitettavasti kokeilussa törmättiin yhteen vakavaan ongelmaan, johon ei onnistuttu löytämään ratkaisua. Joihinkin pintoihin ilmestyy vaaleita kolmioita, jotka ovat todella näkyviä ja käytännössä estävät lopputuloksen hyödyntämisen täysin.



Ratkaiseva ongelma: vaaleita kolmioita lopputuloksessa

Ongelman ratkaisemiseksi kokeiltiin mm.

- leivonta-asetusten muuttamista, erityisesti normaaleihin liittyviä asetuksia
- tarkistettiin geometrian normaalit ja virheettömyys
- luotiin uusi lähdegeometria erilaisella uv-kartoituksella
- toista ohjelmaa tekstuurien leivontaan (Marmoset Toolbag 3)
- suurempaa resoluutiota delighting-tekstuureille

Minkään muutoksen ei havaittu auttavan ongelmaan. On tietysti mahdollista, että kyseessä on delighting-työkaluun liittyvä virhe, koska kyseessä on hyvin uusi ja kokeellinen ohjelma. Voi myös olla, että näin monimutkaisen datan valaistuksenpoisto ei ole käytännöllistä perinteisillä ohjelmilla, vaan vaadittaisiin tekoälyyn pohjautuvaa sovellusta.

Ongelmaa ratkaistaessa havaittiin sattumalta, että Marmoset Toolbag 3 -ohjelmaan oli juuri tullut päivitys 3.0.4, joka mahdollistaa kaikkien delighting-prosessin tekstuurien leipomisen myös tällä ohjelmalla. Marmoset havaittiin paljon nopeammaksi ja yhtä laadukkaaksi työkaluksi tekstuurien leipomiseen. Se on myös hyvin edullinen. Jos prosessi saadaan joskus toimivaksi, tai tarvitaan muuhun käyttöön hyvää ohjelmaa leivontaan, voidaan tämän kokeilun kokemusten perusteella suositella Marmoset Toolbagia, koska sillä saavutetaan huomattavia aikasäästöjä verrattuna Substance Designeriin. Suurin ero vaikuttaa tulevan siitä, että Marmoset käyttää leivontaan näytönohjaimen laskentatehoa, kun Substance Designer pelkästään prosessoria. Eroa ei siis tule, jos käytössä ei ole tehokasta näytönohjainta. Kokeilussa meni paljon aikaa tekstuurien leivontaan, kunnes huomattiin siirtyä käyttämään Marmoset Toolbagia. Jopa 4k ja 8k tekstuurien leivonta-ajat

muuttuivat mahdollisiksi tällä ohjelmalla.

Esimerkiksi 4k ambient occlusion leivonta 1000 näytteellä, visuaalisesti toisiaan vastaava lopputulos. Käytössä NVidia GTX 1070 -näytönohjain ja Intel Core i5 7400 -suoritin:

Marmoset Toolbag: 5 minuuttia
Substance Designer: 72 minuuttia (!)

Vaikka valaistuksenpoisto lopulta epäonnistui, päästiin yhtä estettä vaille hyvään lopputulokseen. Nopean leivontatyökalun löytäminen auttaa huomattavasti iterointivauhdissa, jos tätä menetelmää halutaan kokeilla uudestaan.

4. Esimerkkivisualisoinnit

Esimerkkikokeilut toteutettiin käyttäen Helsingin CityGML-aineistoa. Kohteena oli Lastenlehdon puisto (Ruoholahdenkatu, 00180 Helsinki), jonne sijoitettiin kuvitteellinen 3D-mallinnettu uudisrakennus kaupunkidatan ympäristöön. Puistoon mallinnettiin käsin maaston muodot ja yksinkertaiset tiet ja reunakiveykset, sekä lisättiin kasveja.

Paikan valinta tai uudisrakennuksen ulkomuoto ei ole sinänsä oleellinen hankkeen kannalta, mutta tässä koitettiin luoda realistisen oloinen esimerkkiskenaario. Lisäksi haluttiin luoda visualisointi paikasta, jossa on hiukan liikkumatilaa. Tämä johtuu siitä, että nykyisten CityGML-kaupunkien visuaalinen laatu tiedettiin heikoksi, eikä niitä ole mielekästä kuvata läheltä ahtailla kaduilla.

Tekstuureista ei poistettu valaistusinformaatiota, mutta niitä säädettiin Photoshopilla aavistuksen miellyttävämmäksi, eli kohotettiin varjoja ja laskettiin valkoisia sävyjä hieman. Koska lähtötekstuurit ovat jo pakatussa formaatissa, tämä ei varsinaisesti tuo tekstuureihin mitään lisäinformaatiota tai suuresti paranna niiden visuaalista laatua.

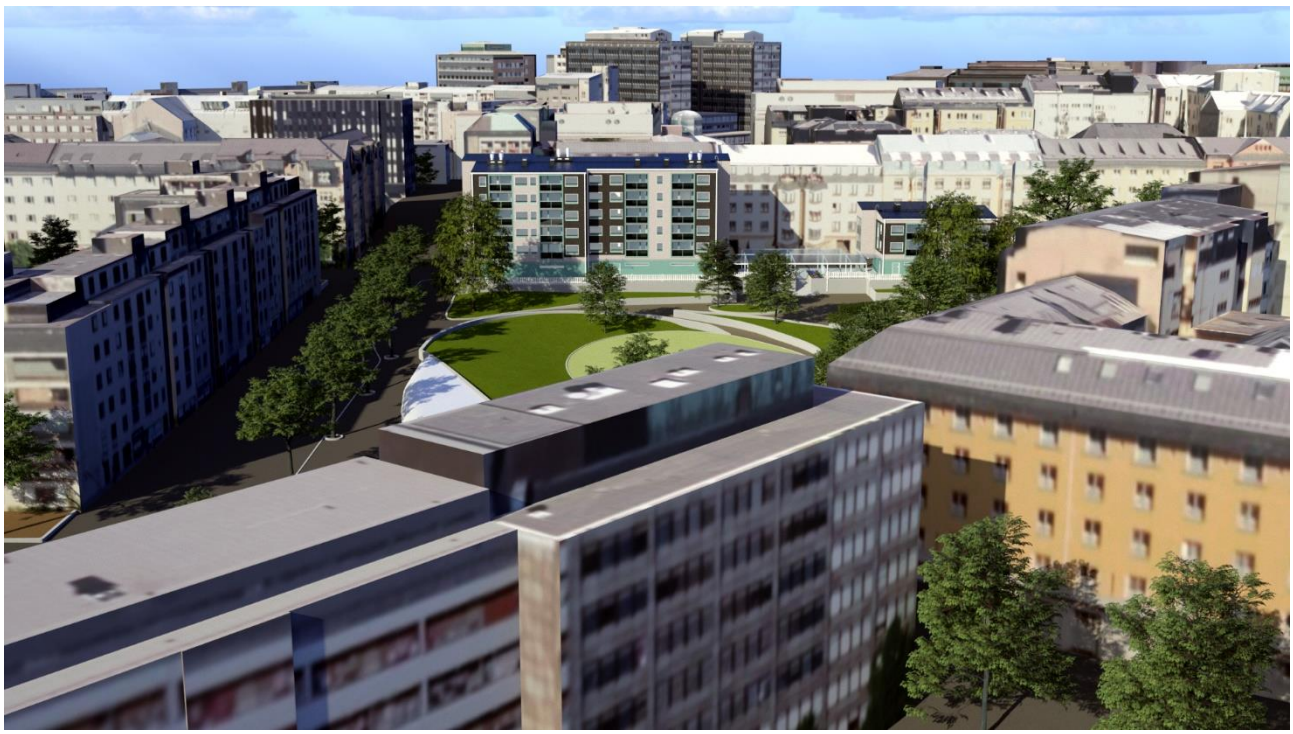
4.1 3ds Max -versio

Visualisoinnin toteuttaminen 3ds Maxilla kävi suoraviivaisesti, eikä 3D City Database ohjelmalla muunnettujen ja Autodesk FBX converterin läpi vietyjen FBX-mallien tuonnissa ollut ongelmia.

Haasteita oli uskottavasti valaistun lopputuloksen luomisessa. Tekstuuridata sisältää paljon tummia varjopaikkoja ja toisaalta myös puhkipalaneita auringon valaisemia vaaleita pintoja. Tämä vaikeuttaa huomattavasti mallin valaisua. Jos luodaan realistinen, kesäisen auringonpaisteen kaltainen valaisu mallille, CityGML-mallit voivat palaa pahoin puhki siinä vaiheessa, kun käsin luotu ympäristö näyttää vielä tasapaksulle ja heikosti valaistulle. Toisaalta kaupunkidatan varjonpaikat jäävät todella helposti tuplasti varjoon, eli tekstuureihin leivottu tumma väri jää mallissa varjoon ja näin värin tummuus kertaantuu.

Valaistuksen suuntaa eli auringon sijaintia on vaikea poikkeuttaa uskottavasti verrattuna tekstuureihin tallentuneen informaation kuvaamaan tilanteeseen. Jos 3D-mallissa aurinko valaisee suoraan rakennuksen pintaa, joka on ollut satelliittikuvauksen aikaan varjossa, muodostuu ristiriitainen kokonaisuus, jossa auringossa oleva pinta näyttää liian tummalle (katso kuva 5, taustalla olevat rakennukset). Toisaalta varjossa olevat pinnat saattavat hohtaa liian kirkkaina.

Käytännössä on mahdollista luoda nykyiselläkin kaupunkidatan laadulla käyttökelpoisia kuvia esimerkiksi päätöksentekoa varten. CityGML-ympäristö luo todella hyvän kontekstin



3ds Max kuva 1

3ds Max kuva 2

3ds Max kuva 3

3ds Max kuva 4

3ds Max kuva 5

3ds Max kuva 6

3ds Max kuva 7

3ds Max kuva 8

3ds Max kuva 9

3ds Max kuva 10

3ds Max kuva 11

3ds Max kuva 12

3ds Max kuva 13

4.2 Unreal Engine 4 -versio

Unreal Engine 4 mahdollistaa joko täysin reaaliajassa laskettavan valaistuksen, tai etukäteen laskettavien valaistuskarttojen käytön (lightmass baking). Näitä tekniikoita voidaan myös yhdistää. Tässä käytettiin kokonaan reaaliajassa laskettavaa valaistusta, koska haluttiin tehdä ero 3ds Max –versioon.

Käytettiin Unreal Enginen versiota 4.19. Toteutuksesta ei ole muuta erityisesti mainittavaa, kuin 8k-tekstuurien käyttö. Niitä UE4 ei avaa ainakaan 4.19 versiossa täydessä tarkkuudessa ilman muutoksia. Tämä korjataan muuttamalla pelimoottorin asennuskansiossa (”Engine\Config”) BaseDeviceProfiles.ini-tiedostoa. Sieltä etsitään kohta ”MaxLODSize=4096” ja se muutetaan ”MaxLODSize=8192”. Tämän jälkeen pelimoottori on käynnistettävä uudestaan.

Käytännössä tässä tapauksessa ongelmat reaaliaikaisessa grafiikassa ovat täysin samat kuin ray tracing –toteutuksessa. Lähtöaineiston tekstuurien laatu on rajoittava tekijä. Jonkin verran ongelmaa helpottaa se, että reaaliaikainen renderöinti ei tuota täysin realistisia varjoja, minkä vuoksi tässä tapauksessa tekstuureihin tallentuneet varjot eivät korostuneet niin paljon, kuin 3ds Max versiossa.

Unreal Engine 4 kuva 1

Unreal Engine 4 kuva 2

Unreal Engine 4 kuva 3

Unreal Engine kuva 4

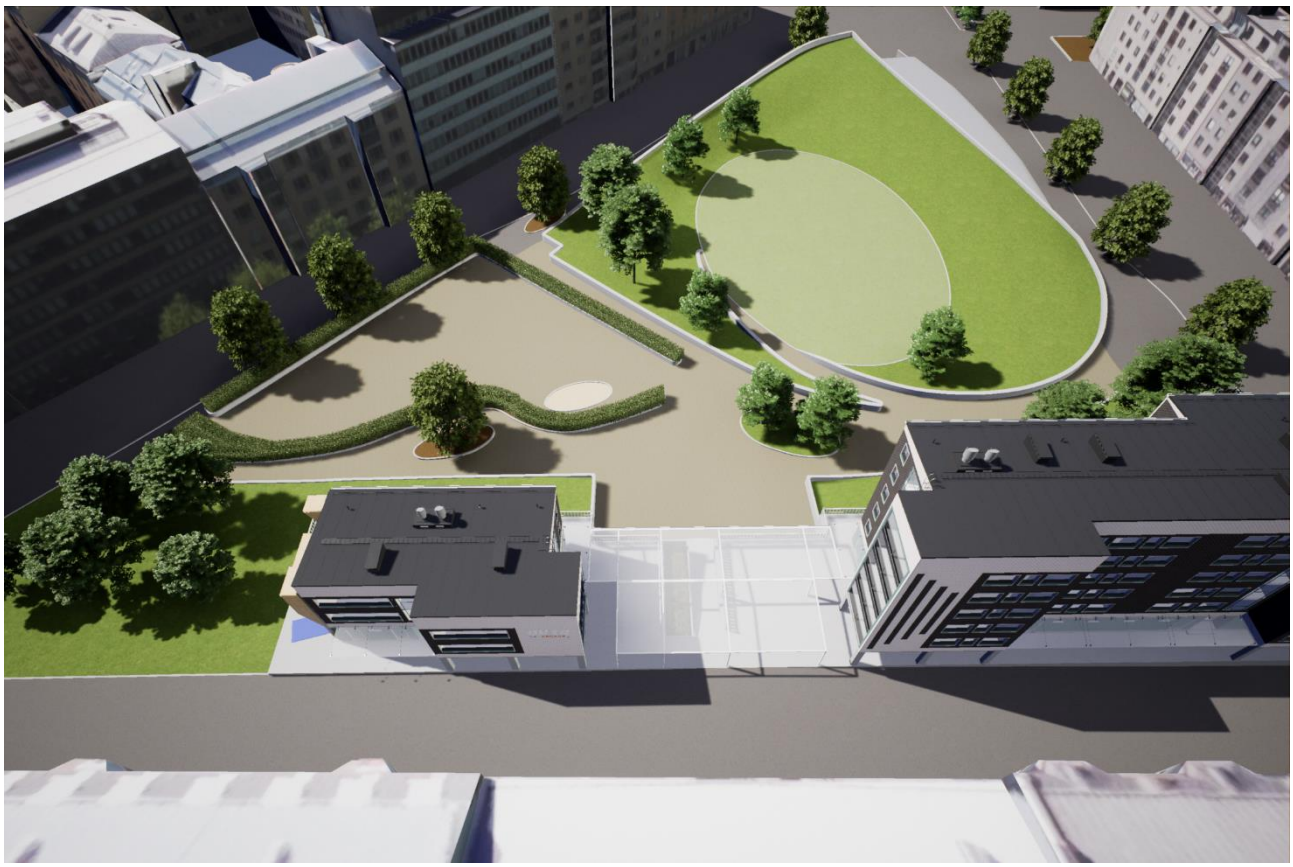
Unreal Engine kuva 5

Unreal Engine kuva 6

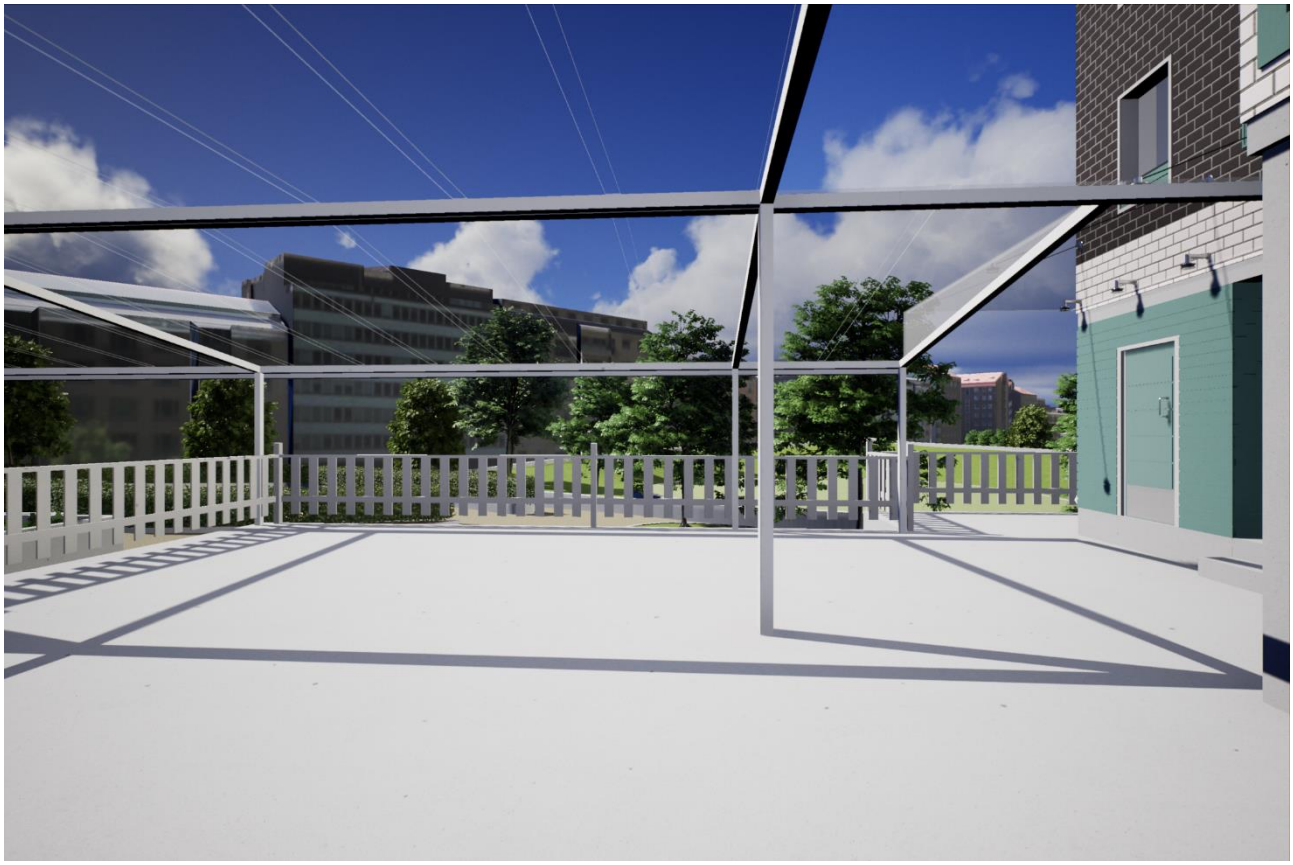
Unreal Engine kuva 7

Unreal Engine kuva 8

Unreal Engine kuva 9



Unreal Engine kuva 10



Unreal Engine kuva 11

Unreal Engine kuva 12

5. Loppusanat

Avoimen kaupunkidata avaa mahdollisuuksia myös visuaalisen työn nopeuttamiseen. Nykyisellään datan hyödyntäminen ei ole aivan suoraviivaista, mutta kuitenkin mahdollista. Laatu jättää toivomisen varaa, mutta joihinkin käyttötarkoituksiin avoimet kaupunkiympäristöt ovat riittäviä. Todellinen ympäristö antaa hyvän kontekstin visualisointikohteelle, jolloin päätöksenteossa saadaan paremmin todellisuutta vastaava kuva suunnitellun kohteen visuaalisista vaikutuksista.

Tulevaisuudessa avoimia kaupunkiaineistoja toivotaan kehitettävän laadullisesti parempaan suuntaan. Erityisesti rakennusten tekstuuridata jättää paljon toivomisen varaa. Satelliittikuvista tehtävä fotogrammetria ei yksinään riitä laadukkaaseen lopputulokseen. Mahdollisuuksia voisi tulevaisuudessa olla esimerkiksi puoliautomaattisen lennokkikuvauksen ja autoihin liitettävien kameroiden käyttäminen fotogrammetrian lähteinä.

Hankkeen tärkeimmät tulokset olivat lopulta tämä dokumentti ja erityisesti ohjeet CityGML-datan muuntamiseen. Tulevaisuuden mahdollisuuksien tutkijoille valaistuksenpoisto, joka nyt epäonnistui, on mielenkiintoinen jatkotutkimuksen aihe. Lisäksi koevisualisointien kuvat ja projektitiedostot ovat erittäin hyödyllisiä niille, jotka pohtivat, voisiko avoin kaupunkidata tarjota heille mahdollisuuksia.