SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

RAKENNUKSEN YMPÄRISTÖN HAVAINNOLLISTAMINEN VIRTUAALIMALLIN AVULLA

TEKIJÄ: Juho-Markus Niemi

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Rakennustekniikan koulutusohjelma	
Työn tekijä(t) Juho-Markus Niemi	
Työn nimi Rakennuksen ympäristön havainnollistaminen virtuaalimallin avulla	
Päiväys 22.4.2013	Sivumäärä/Liitteet 65
Ohjaaja(t) Lehtori Viljo Kuusela & Yliopettaja Janne Repo	
Toimeksiantaja/Yhteistyökumppani(t) Capisso Oy / Rakennusliike U. Lipsanen Oy, Admino Technologies	

Tiivistelmä

Opinnäytetyön tavoitteena oli tehdä Capisso Oy:lle rakennuksen ympäristöön keskittyvä virtuaalimalli ja samalla tutkia miten virtuaalimalli käytännössä tehdään. Opinnäytetyö on toteutettu siitä syystä että nykyiset rakennusalalla käytetyt ohjelmistot eivät pysty suoraan tarjoamaan tarpeeksi tehokasta ympäristönmallinnustyökalua, niinpä ratkaisua etsittiin virtuaalimallista, jonka teossa hyödynnetään peli- ja rakennusalan ohjelmistoja. Työn toimeksiantajan Capisso Oy:n lisäksi työssä yhteistyökumppaneina toimi Rakennusliike U. Lipsanen Oy ja Admino Technologies.

Aluksi opinnäytetyössä tutkittiin, millaisia ohjelmistoja oli tarjolla virtuaalimallien tekoon. Oikeiden ohjelmistojen löydyttyä aloitettiin varsinainen virtuaalimallin tekeminen. Tässä työssä virtuaalimalli tehtiin käyttämällä ArchiCAD 16-, Blender 2.65-, Rocket client 2.4.1.2 -ohjelmistoja ja Meshmoon-palvelua.

Työn tuloksena saatiin aikaan virtuaalimalli sekä virtuaalimallin toteutuksen prosessikuvaus. Työn tuloksena voitiin todeta, että virtuaalimalleissa mallien koko yleensä kasvaa helposti liian suuriksi, mikä on rajoittava tekijä, pyrittäessä näyttävään lopputulokseen. Lisäksi todettiin, että virtuaalimallien toteutukseen löytyy monia eri ohjelmistoja ja että tässä työssä käytetyt ohjelmistot eivät olleet kaikkein edistyksellisempiä.

Avainsanat
Virtuaalimalli, Blender, ArchiCAD, Rocket client, Meshmoon

Salainen

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme In Construction Engineering			
Author(s) Juho-Markus Niemi			
Title of Thesis Using a Virtual Model to Visualize Building Enviroment			
Date	22 April 2013	Pages/Appendices	65
Supervisor(s) Mr. Viljo Kuusela, Lecturer & Mr. Janne Repo, Principal Lecturer			
Client Organisation /Partners Capisso Oy / Rakennusliike U. Lipsanen Ltd, Admino Technologies			

Abstract

The aim of the thesis was to implement a virtual model which focuses on the building enviroment for Capisso Oy and also to investigate how to create a virtual model. This thesis was done because the current construction industry software does not provide effective environment modeling tools, so the solution was to try to seek results through a virtual model which combines game and construction industry software. The thesis was commissioned by Capisso Oy and partners were Rakennusliike U. Lipsanen Ltd and Admino Technologies.

The first thing was to investigate what kind of software was available. When the right software was found the making of the virtual model actual began. In this thesis the virtual model was produced with ArchiCAD 16, Blender 2.65, Rocket client 2.4.1.2 and Meshmoon application.

The result of this thesis was a fully working virtual model and also a process description of how to implement a virtual model. The conclusion is that there are many different software combinations of how to create a virtual model and the software that were used in this thesis were not the advanced ones. It was also discovered that usually the model size of a virtual model could easily increase too big, which is a limiting factor if the virtual model tries to have a realistic outcome.

Keywords
Virtual model, ArchiCAD, Blender, Rocket client, Meshmoon

Secret

ESIPUHE

Haluan aluksi kiittää työn toimeksiantajaa Capisso Oy:tä ja yhteistyökumppaneita Rakennusliike U. Lipsasta ja Admino Technologiessia. Erityisesti haluan kiittää Capisso Oy:n projekti-insinööri, Jenni Kaukosta hyvästä avusta työn eri vaiheissa sekä Admino Technologiesin Jonne Nauhaa hyvistä neuvoista.

Kuopiossa 22.4.2013

Juho-Markus Niemi

SISÄLLYS

LYHENTEET JA MÄÄRITELMÄT

APPLIKAATIO	Tulee englanninkielen sanasta *application*, joka tarkoittaa sovellusta.
AVATAR	Hahmo, joka edustaa virtuaalimallissa käyttäjää.
BUMP MAPPING	Kuvastaa tyyliä luoda objektien pinnoille rosoisuutta, jolloin saadaan objektista realistisempi.
DRAG AND DROP	Tarkoittaa tapaa, jossa tiedosto klikataan hiiren vasemmalla painikkeella aktiiviseksi ja siirretään tiedosto painamalla painiketta pohjassa toiseen ikkunaan.
DWG	*DragWinG*, tallennusmuoto useille 2D- ja 3D-tiedostoille. Käytetään useissa rakennusalan ohjelmistoissa tallennusmuotona.
EXPORTAUS	Tulee englanninkielen sanasta *export*, joka tarkoittaa vientiä.
FPS	Tulee englanninkielen sanoista *frames per secound*, joka tarkoittaa näytön päivitysnopeuden yksikköä.
HEIGHT MAPPING	*Bump mapping* -tyyliin perustuva tapa luoda objektien pinnoille rosoisuutta. Toteutetaan harmaasävy (*grayscale*) ja värikuvan avulla. Harmaasävy kuvassa musta väri edustaa minimikorkeutta ja valkoinen väri maksimikorkeutta.
HOSTING PLATFORM	”Hotelli”, josta käyttäjä ostaa tilan (*spacen*) virtuaaliympäristölle.
MB	Tulee englanninkielen sanasta *megabyte*, joka tarkoittaa suomeksi megatavua. Kuvaa tiedoston kokoa.
MESH	Objekti, joka koostuu pisteistä, reunoista ja pinnoista, jotka määrittelevät (monikulmion) objektin kolmiulotteisessa tietokonegrafiikassa.
MESHMOON	Usean samanaikaisen käyttäjän mahdollistava virtuaaliympäristöjen *hosting platform.*
NORMAL MAPPING	*Bump mapping* -tyyliin perustuva tapa luoda objektien pinnoille rosoisuutta. Toteutetaan RGB- ja värikuvan avulla.
OGRE ENGINE	Avoimeen lähdekoodiin perustuva 3D engine (rendering engine), joka on laadittu C++-ohjelmointikielellä. Tuottaa reaaliaikaista grafiikkaa.
PLANE	Tarkoittaa Blenderissä luotua meshiä, jolla on vain x-, ja y-koordinaatisto arvot mutta ei z-koordinaattiarvoa.
POW²	*Power of two*, kuvastaa lukua, jossa kantaluku kaksi korotetaan potenssiin 1,2,3 jne.
RGB	Kuva, joka koostuu punaisesta, vihreästä ja sinisestä värikanavasta.
REALXTEND	Virtuaalisten 3D Internet-sovellusten kehitysalusta.
SCENE	Virtuaaliympäristö.
SPACE	Tila virtuaaliympäristölle.
UV MAPPING	Teksturointitapa, jossa 3D-objekti paketoidaan kuvalla.

1 JOHDANTO

Opinnäytetyön aihe on saatu yritykseltä nimeltä Capisso Oy, joka toimii lähinnä tietomallikoordinaattorina rakennushankkeissa. Yrityksen hallituksen puheenjohtajalta, TkT Jarmo Laitiselta tuli kysely, olisiko minulla mielenkiintoa alkaa tutkia, kuinka saisi rakennettua mahdollisimman näyttävän ympäristön virtuaalimalliin ja samalla opetella, miten virtuaalimalli toteutetaan. Tiesin, että virtuaalimallin teossa hyödynnetään peli- ja rakennusalan ohjelmistoja, joten olin valmis tarttumaan haasteeseen, koska minulla riittää mielenkiinto erilaisten ohjelmistoalustojen yhdistelyyn ja opetteluun. Lisäksi katsoin, että on etu, että oppii käyttämään uusia ohjelmistoja rakennusalan ohjelmistojen ulkopuolelta.

Työn tavoitteena on saada aikaan Capisso Oy:lle rakennuksen ympäristöön keskittyvä virtuaalimalli ja samalla tutkia, miten virtuaalimalli käytännössä tehdään, sekä tehdä virtuaalimallin toteutusvaiheesta prosessikuvaus. Tässä työssä virtuaalimallin pääpaino on rakennuksen ulkopuolen- ja ympäristön mallintamisessa. Opinnäytetyö on tehty siksi, että nykyiset rakennusalalla käytetyt ohjelmistot eivät pysty suoraan tarjoamaan tarpeeksi tehokasta ympäristönmallinnustyökalua. Tässä työssä etsitään ratkaisua juuri tähän ongelmaan, miten saadaan rakennuksen ympäristö mahdollisimman näyttäväksi ja kustannustehokkaasti toteutetuksi menemättä liian pikkutarkkaan näpertelyyn.

Virtuaalimallin toteutukseen käytetään rakennusalan mallinnusohjelmistoista ArchiCAD 16:ta, pelialan ohjelmistoista Blender 2.65:tä ja virtuaalimallin jako toteutetaan käyttämällä Rocket clientiä ja Meshmoon-palvelua.

Toimeksiantajan Capisso Oy:n lisäksi työssä yhteistyökumppaneina toimii Rakennusliike U. Lipsanen Oy ja Admino Technologies.

Tiettyjä teknisiä yksityiskohtia on jouduttu karsimaan salassapitosyistä.

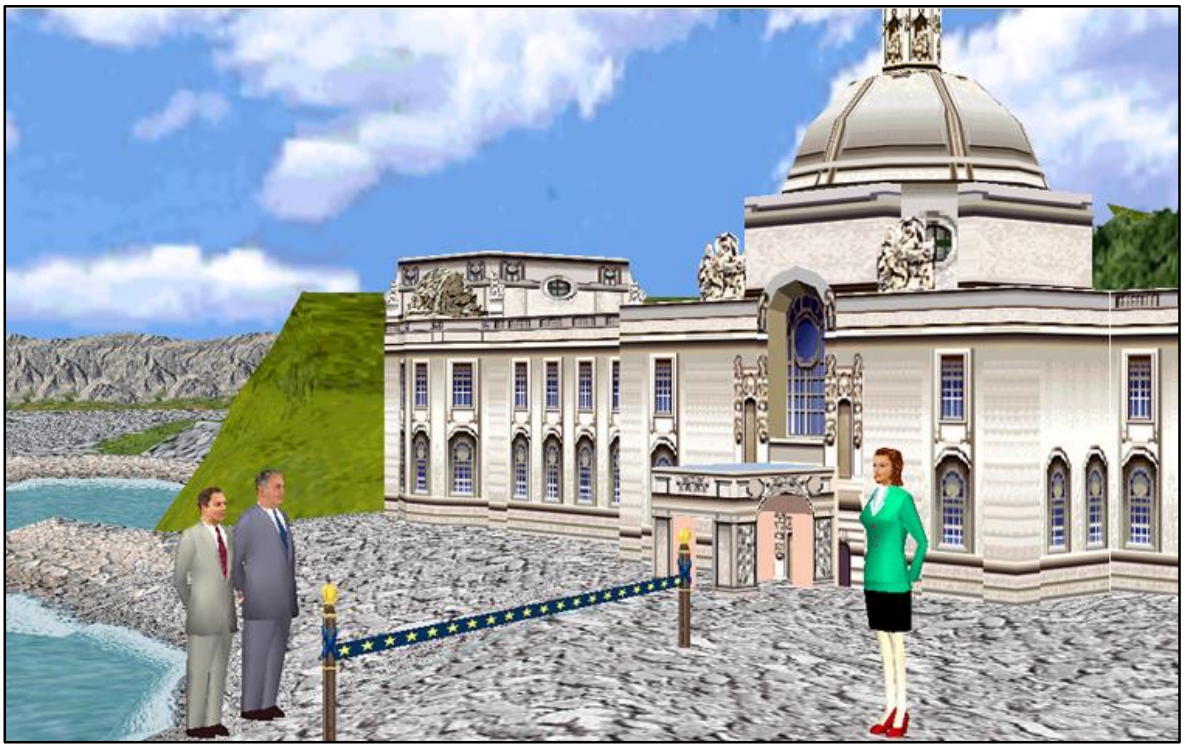
2 VIRTUAALIMALLI

2.1 Virtuaalimallin määritelmä

Virtuaalimalli-käsitteellä tarkoitettaan tässä työssä kolmiulotteista mallia/ympäristöä, jossa katselija voi vapaasti liikkua avatarilla tietokonepeleistä tutuilla näppäinkomennoilla. Virtuaalimallien teossa yhdistetään pelialalla käytetyt ohjelmistot rakennusalalta — tässä tapauksessa — tuttujen 3D-mallinnusohjelmistojen kanssa. Yleensä myös virtuaalimalleissa katselija voi vaikuttaa mallin vuorokaudenaikoihin. Virtuaalimallien teossa pääajatuksena on se, että valmiit mallit pystytään julkaisemaan Internetiin muiden ihmisten katseltavaksi ja varsinainen katselu tapahtuu internetselainten laajennuskomponettejen avulla, jotka lisäävät internetselaimeen mallin katselukyvyn. Mallia voidaan katsella myös erikseen ladattavilla ohjelmistoilla. Yhdistävä tekijä on se, että mallien katselu onnistuu vain nettiyhteyden välityksellä.

2.2 Virtuaalimallin historiaa lyhyesti

Lyhyesti voidaan sanoa, että virtuaalimallien historia alkaa jo vuodesta 1995, kun Internet kaupallistettiin ja se tuli suurempien ihmismassojen käyttöön. Ensimmäisenä yleisenä tiedostomuotona julkaistiin *VRML* (*Virtual Reality Modeling Language*), jonka avulla pystyttiin tekemään kolmiulotteista vektorigrafiikkaa. *VRML* on kehitetty etenkin Internetiä varten. Nykyään *VRML*-tiedostomuotoa ei enää ole käytössä vaan se on korvattu X3D:llä.



KUVA 1. Esimerkki kuva VRML-mallista (Driver-Inter)

Toinen huomionarvoinen virtuaalimallien teossa käytetty tiedostomuoto on *3DMLW* (*3D Markup Language for Web*), jolla pystytään myös tekemään kolmiulotteista grafiikkaa Internetiin samalla tavoin kuin *VRML*:llä pystytään. Tämä tiedostomuoto on kehitetty vuonna 2009 ja se tarjoaa paljon kehittyneemmän *rendering enginen* kuin *VRML*. *3DMLW* käyttää hyväkseen *OpenGL*:ää (*Open Graphics Library*), jolla tehdään 2D- ja 3D-grafiikkaa. Alla olevaa kuvaa klikkaamalla (kuva 2) pääsee tutustumaan Tallinnan vanhaan kaupunkiin, jonka tekoon on käytetty kyseistä tiedostomuotoa. Katseluohjelmana toimii Unity Web Player.

KUVA 2. Tallinnan vanha kaupunki (Kuvankaappaus, 3D Technologies R&D)

Virtuaalimallien ja niiden toteutustapoihin löytyy monia lähestymistapoja, jotka ovat jalostuneet vuosienvarrella. Edellä mainitut kaksi tiedostomuotoa ovat sieltä yksinkertaisimmasta päästä ja näiden perusta rakentuu yksinkertaisiin tekstitiedostoihin. Sekä ohjelmistot- ja tietokoneiden laskentatehot ovat kehittyneet ja näin on saatu entistä näyttävämpiä- ja laajempia virtuaalimalleja aikaan.

2.3 Virtuaalimalleihin liittyviä teknisiä termejä

Virtuaalimallien teossa hyödynnetään perinteisten rakennusalan mallinnusohjelmistojen lisäksi pelialalla käytettäviä ohjelmistoja, jolloin erilaisten teknisten käsitteiden määrä kasvaa huomattavasti. Seuraavassa luvussa selostetaan muutamia teknisiä termejä, jotka liittyvät muun muassa materiaaleihin ja tekstuureihin. Nämä termit esiintyvät yleisesti pelialan ohjelmistoissa ja ovat muutenkin tärkeä ymmärtää virtuaalimalleja tehtäessä.

2.3.1 Materiaalit ja tekstuurit

Kolmiulotteisissa malleissa materiaaleilla ja tekstuureilla on suuri merkitys. Niiden määrittäminen antaa visuaalisen ilmeen malliin ja katselijalle vaikutteen siitä, miten realistiseksi virtuaalimalli koetaan. Ilman materiaalien ja tekstuurien määritystä koko malli on pelkkää harmaata massaa. Materiaalien ja tekstuurien erot on hyvä ymmärtää.

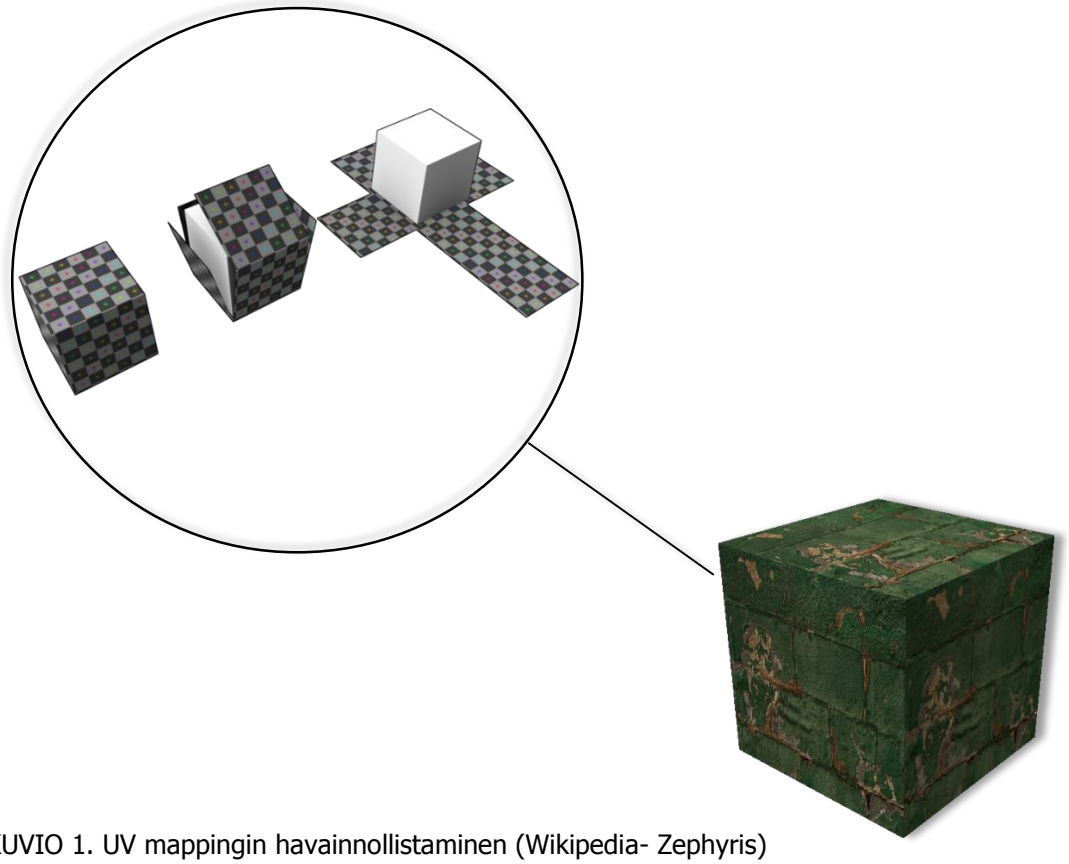
Materiaalit (materials) määrittelevät optiset ominaisuudet objekteille/mesheille, kuten värin, himmeyden ja kirkkauden.

Tekstuuri (texture) antavat objekteille/mesheille tietyn pintakuvion. Tätä voi havainnollistaa sillä keinolla, että lähes kaikilla reaalimaailman esineillä tai tavaroilla on tietyn näköinen pintakuvio esimerkiksi tiiliharkko, jossa pinnan kuvioinnin näkee ja myös tuntee pideltäessä tiiltä kädessä.

Seuraavissa kohdissa kerrotaan kaksi yleisintä materiaalien ja tekstuurien määritystapaa. Näistä tavoista *Bump mapping* esiintyy myös rakennusalan mallinnusohjelmistoissa mutta yleensä ominaisuuksiltaan se on karsitumpi kuin mitä pelialan ohjelmistot tarjoaa.

2.3.2 UV mapping

UV mappingin perusidea on yksinkertainen, 2D-kuvalla pyritään ”paketoimaan” objektiin/meshiin haluttu pinta, jolloin lopputuloksena saadaan pinta joka pyrkii antamaan kuvan reaalimaailman vastaavasta objektista. Alla oleva kuvio (kuvio 1) havainnollistaa *UV mapping* -tekniikkaa.



KUVIO 1. UV mappingin havainnollistaminen (Wikipedia- Zephyris)

2.3.3 Bump mapping

Bump mapping on tekniikka, jossa objektien/meshien pinnoille luodaan kuvien avulla rosoisuutta, jolloin pinnasta saadaan realistisempi kuin mitä se olisi pelkällä tasaisella kuvalla. *Bump mapping* perustuu valojen, heijastuksen sekä kuvien värierojen laskentaan. *Bump mapping* voidaan toteuttaa joko *Height mappina* tai *Normal mappina*. *Height mappingissa* käytetään hyväksi harmaasävy tekstuureja (kuvio 2) kun taas *normal mappingissa* RGB-tekstuureja. (kuvio 3.) Erona näille teksturointi tavoille on se, että *height mappingia* käytetään ainoastaan renderoitujen still-kuvien tekoon ja *normal mappingia* reaaliaikaisen renderoinnin tekoon.

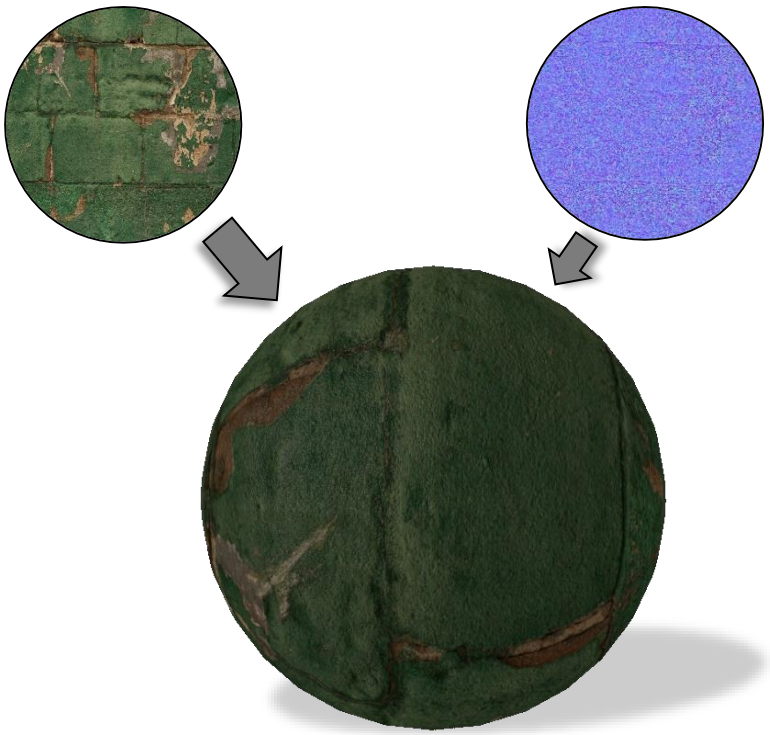
Height mapping -tekniikan kuvaus:

Tekoon käytetään harmaasävykuvaa ja tavallista värikuvaa, jolla on saatu seuraavanlainen lopputulos aikaiseksi. (kuvio 2.) Pinnasta voi havaita rosoisuuden ja syvyysefektin, jonka *height mapping* -tekniikka saa aikaiseksi. Näin pinnasta saadaan realistisen näköinen.

KUVIO 2. Height mapping -tekniikka

Normal mapping -tekniikan kuvaus:

Tekoon käytetään RGB-kuvaa ja tavallista värikuvaa, jolla on saatu seuraavanlainen lopputulos aikaiseksi. (kuvio 3.) Myös tässä tekniikassa pinnasta voi havaita rosoisuuden ja syvyysefektin, jonka *normal mapping* -tekniikka saa aikaiseksi. Tekniikaltaan *normal mapping* on kehittyneempi.



KUVIO 3. Normal mapping -tekniikka

2.3.4 Renderointi

Renderoinniksi kutsutaan toimenpidettä jossa 3D-mallista tuotetaan still-kuva. (kuva 3.) Tässä kuvassa tulee mallissa asetetut tehosteet voimaan, kuten esimerkiksi varjostukset, kuvan pehmennys, tekstuurit, materiaalit jne. Virtuaalimalleissa pyritään jatkuvaan renderoitujen kuvien ”virtaan” siksi tätä kutsutaankin pelkän renderoinnin sijaan reaaliaikaiseksi renderoinniksi.

KUVA 3. Blenderissä renderoitu kuva

Reaaliaikaisella renderoinnilla tarkoitetaan prosessia, jossa tietokone tuottaa tarpeeksi nopeasti kuvavirtaa katselijansilmien eteen, jotta katselijalla on tunne siitä, että hän on vuorovaikutuksessa virtuaaliympäristöön. Tarpeeksi nopea -käsite pitää sisällään kirjainyhdistelmän *FPS*, joka tulee sanoista *frames per secound*. Tämä tarkoittaa sitä, että kuinka nopeasti tietokone pystyy tuottamaan uutta kuvaa katselijansilmien eteen. Alarajana pidetään lukua 25 eli katselinjan tulisi saavuttaa 25 ruutua sekunnissa tai enemmän. Tämän alle mentäessä katselija ei koe, että olisi vuorovaikutuksessa virtuaaliympäristöön vaan katselu tuntuu tökkivän. Jotta arvoon 25 tai enemmän päästään vaatii se koneelta laskentatehoa tai vastaavasti kevyempää mallia. Hyvänä arvona pidetään lukua 60, jolloin se vastaa samaa tasoa kuin katsoisi suoraa televisiolähetystä tai DVD-elokuvaa.

Reaaliaikaista renderointia ei käytetä lähes lainkaan rakennusalan mallinnusohjelmistoissa vaan sen käyttö painottuu suurimmaksi osaksi pelialalle. Renderoinnin käyttötapa onkin juuri se suurin erottava tekijä rakennusalan mallinnusohjelmistojen ja pelialan ohjelmistojen välillä.

2.3.5 Game-, Graphics -, Rendering- ja 3D engine

Game engine -käsitteellä tarkoitetaan järjestelmää, joka on kehitetty pelien kehitykseen. Kun taas *graphics enginellä* tarkoitetaan teknologiaa, joka tuottaa 2D/3D-grafiikkaa *game engineen*. Useasti puhutaan myös *graphics enginen* yhteydessä karsitusta *game enginestä* joka tarkoittaa sitä, että siitä puuttuu joitakin pelin kehitykseen liittyviä komponentteja. Jotkut *game enginet* tarjoavat vain reaali-aikaisen renderointi mahdollisuuden sen sijaan, että tarjoaisivat pelialalla tarvittavia komponentteja. Tämän tyyppisiä enginejä kutsutaankin yleisesti *graphics engineiksi*, *rendering engineiksi* tai *3D engineiksi* näistä onkin alettu käyttää nimeä *3D engines*.

Lista tunnetuimista game engineistä:

- CryEngine
- Frostbite Engine
- Unreal Engine
- Unity Engine.

Lista yleisistä 3D engineistä:

- Crystal Space
- Irrlicht Engine
- Truevision3D
- OGRE (Object-oriented Graphics Rendering Engine).

Tämän opinnäytetyön virtuaalimalli pohjautuu *OGRE engineen*. Se on avoimeen lähdekoodiin perustuva *3D engine* (*rendering engine*), joka on laadittu C++-ohjelmointikielellä. Tarkempia yksityiskohtia ei tässä opinnäytetyössä tarkastella.

2.4 Virtuaalimallien hyödyntäminen maailmalla

Tässä luvussa esitellään muutamia projekteja maailmalta, joissa on hyödynnetty virtuaalimallien potentiaalia.

Virtuaalikaupungit

Virtuaalimallien avulla on toteutettu kokonaisia virtuaalikaupunkeja, joita on tehty paljon maailmalla kuin myös Suomessakin. Virtuaalikaupungit tarjoavat mahdollisuuden esitellä muun muassa kaupungin nähtävyyksiä, upeita rakennuksia tai kaupungin historiaa. Virtuaalikaupungin avulla voidaan myös tarjota turisteille mahdollisuus tutustua kaupunkiin etukäteen. Samalla virtuaalikaupungit tarjoavat mahdollisuuden mainostaa kauppoja virtuaalisesti, jotta turistit osaisivat tulla juuri sinne ostoksille.

3D-simulointimalli

Ranskassa sijaitsevaan Dijonin kaupunkiin virtuaalimallin avulla on luotu 3D-simulointimalli, jossa simuloidaan raitiovaunuverkoston toimivuutta. Haasteena oli mallintaa kaupungin aluetta 20 km²:n alueelta ja pyrkiä samalla mallintamaan koko raitiovaunuverkosto, sekä itse vaunut sisältä että ulkoa. Malli: http://transfert.virtuelcity.com/VStory/WebResources/DijonWeb2/Vweb.html

Second Life

Kyseessä on tunnetuin virtuaalimaailma, jossa käyttäjä aluksi luo avatar-hahmon ja voi näin virtuaalisesti tavata tuttuja ja muita ihmisiä ympärimaailmaa tietokoneen välityksellä. *Second Life* on siis peli jota voisi kutsua elämäsimulaattoriksi. *Second Life* on yksi suurimmista virtuaaliseen ajatteluta-paan perustuvista projekteista, palvelulla on jopa miljoonia käyttäjiä.

3 OHJELMISTOT

Ohjelmistot-luvussa tarkastellaan ohjelmistoja joiden avulla virtuaalimalleja pystytään tuottamaan ja katselemaan. Erilaisia ohjelmistopolkuja on useita, mutta tässä pääpaino säilyy ohjelmistoissa joita on rakennusalalla käytetty. Lopuksi listataan opinnäytetyössä käytetyt ohjelmistot ja niiden valintaan vaikuttaneet tekijät.

3.1 Virtuaalimallien tuottamisohjelmistoja

3.1.1 Novapoint Virtual Map

”Novapoint Virtual Map työkalu rakennushankkeiden suunnitelmien virtuaaliseen mallintamiseen ja visualisointiin. Ohjelmaa on käytetty kaupunkien, satamien, meriväylien, lentokenttien, rautateiden ja teiden mallintamiseen.” (Vianova.fi)

KUVA 4. Novapoint Virtual Mapilla tuotettu malli (Vianova.dk)

3.1.2 RENDERLights

RENDERLights-ohjelmisto tarjoaa tehokkaan työkalun arkkitehdeille, artisteille, suunnittelijoille ja muille ammattilaisille virtuaalimallien tekoon. Ohjelmisto tarjoaa käyttöön hyvin kehittyneen *3D enginen* ja näin ohjelmistolla pystytään tuottamaan hyvinkin näyttäviä malleja.

KUVA 5. RENDERLightilla tuotettu malli (Kuvankaappaus, YouTube)

3.1.3 Unity

Unity-ohjelmisto on varsinaisesti kehitetty pelialalle pelien tekoon, mutta sitä voidaan myös soveltaa virtuaalimallien teossa. Tässä ohjelmistossa on pelialan yhteyden takia hyvin kehittynyt *game engine*, joka tarjoaa käyttäjälle mahdollisuuden tehdä virtuaalimalleista visualisesti näyttäviä.

KUVA 6. Kuva Espoon Tapiolan virtuaalimallista (Kuvankaappaus, Sito)

3.2 Virtuaalimallien katselu- ja jakamistavat

Virtuaalimallin valmistuttua se pitää saada yleensä Internetiin ja siellä katseltavaan muotoon ja tähän tarvitaan erillistä katseluohjelmaa ns. clientiä/vieweriä.

Virtuaalimallin katselu tapahtuu yleensä niin, että käyttäjä löytää virtuaalimallin netistä ja lataa katseluohjelman koneelle tai asentaa nettiselaimeen tarvittavan komponentin, joka mahdollistaa virtuaalimallin katselun suoraan nettiselaimessa. Virtuaalimallien katselu on aina ilmaista eikä se vaadi käyttäjältä minkäänlaista maksua.

Lista muutamista clienteistä/viewereistä:

- Unity - Unity Web Player
- Novapoint Virtual Map - ActiveX
- Rocket Client (opinnäytetyössä käytetty sovellus).

Virtuaalimallien jakaminen perustuu joissain tapauksissa ns. pilvipalvelumalliin, jossa käyttäjät hakevat virtuaalimallin palveluntarjoajan serveriltä ja kirjautuvat palveluun, jolloin samassa virtuaalimallissa pystyy olemaan samaa aikaan useita ihmisiä reaaliaikaisesti. Yksinkertaisempi tapa on tehdä virtuaalimallista versio, jonka käyttäjä lataa lokaalisti omalle koneelle tai avaa nettiselaimeen, tällöin katselija on itse virtuaalimallissa eikä samaan virtuaalimalliin voi tulla muita käyttäjiä. Tässä opinnäytetyössä käytetään juuri pilvipalveluun perustuvaa mallia.

3.3 Opinnäytetyön virtuaalimallin toteutuksessa käytetyt ohjelmistot

Opinnäytetyön tarkoituksena oli tehdä Capissolle rakennuksen ympäristöön keskittyvä virtuaalimalli ja samalla tutkia miten virtuaalimalli käytännössä toteutetaan. Seuraavat kappaleet kertovat taustoja miten päädyin työssä käytettyihin ohjelmistoihin.

Rakennusalan mallinnusohjelmiston valintaan vaikutti se, että rakennusalan mallinnusohjelmistoista Capissolla oli käytössä ArchiCAD. Myös Revit Architecturen käytöstä keskusteltiin, mutta loppujen lopuksi tämä idea hylättiin. Syy Revitin hylkäykselle oli se, että case-kohteen tietomalli oli toteutettu ArchiCAD:llä ja siksi ei haluttu enää siirtää malleja ohjelmistoista toisiin jo muuten kirjavassa ohjelmistovalikoimassa.

Virtuaalimallin katseluohjelmaksi (clientiksi) oli jo ennalta sovittu Rocket client, jolla päästään Meshmooniin. Meshmoon on suomalaisen yrityksen nimeltä Admino Technologies kehittämä usean samanaikaisen käyttäjän mahdollistava virtuaaliympäristöjen *hosting platform*. Meshmooniin pääsee Rocket clientillä, joka on kehitetty avoimen lähdekoodiin perustuvan *realXtend*-ohjelmiston pohjalta. *RealXtend* taas perustuu *OGRE engineen*, josta on mainittu aiemmin tekstissä.

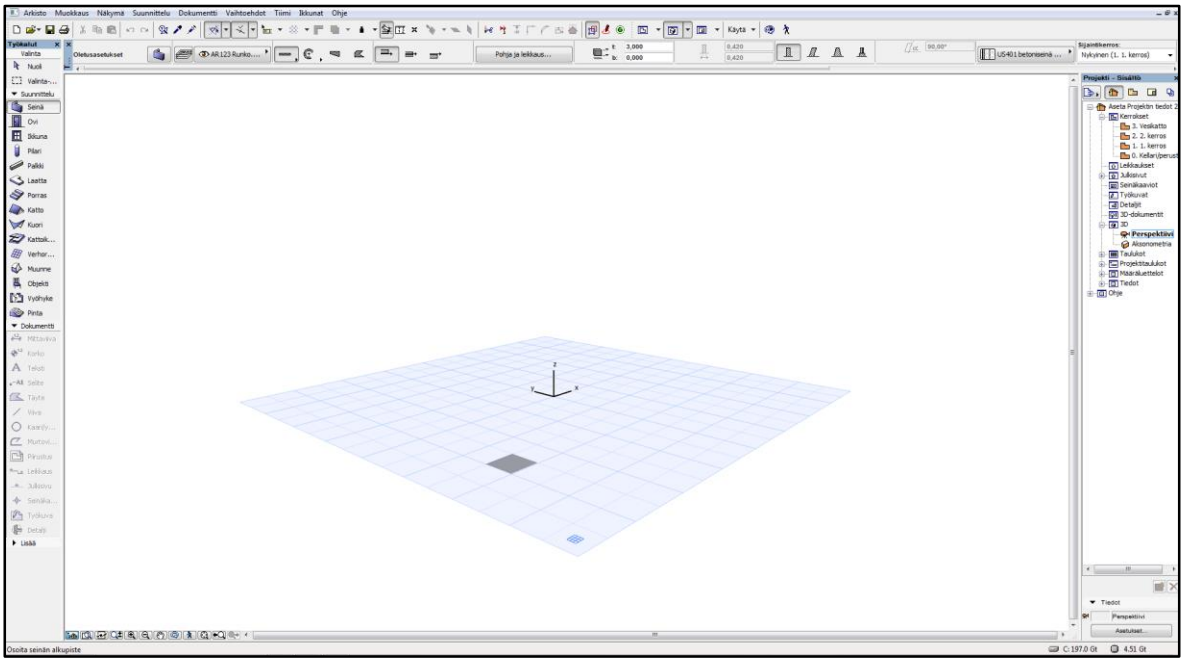
Seuraavaksi tutkittiin minkä muotoisissa tiedostomuodoissa ArchiCAD:stä malleja voi exportata, jotka sitten saataisiin seuraavassa vaiheessa vietyä pelialan ohjelmistoihin. Kävi ilmi, että ArchiCAD:stä voi exportata tiedostoja *3ds*-muodossa jota useat pelialan ohjelmistot tukevat. Tämän käydessä ilmi piti vielä selvittää pelialan ohjelmisto, jolla saadaan virtuaalimalli tehtyä Meshmooniin ja joka tukee ArchiCAD:in tuottamaa *3ds*-tiedostoa. Käytettäväksi pelialan ohjelmistoksi valikoitui Blender.

Blender on avoimeen lähdekoodiin perustuva 3D-grafiikan tekoon tarkoitettu ohjelmisto, jolla voidaan muun muassa tehdä animaatioita, visuaalisia tehosteita tai pelejä. Blenderin laajoihin ominaisuuksiin kuuluu muun muassa 3D-mallinnus, UV mapping, teksturointi ja monipuoliset renderointi mahdollisuudet. Ohjelmiston valintaan vaikutti ratkaisevasti myös se, että ohjelmisto on ilmainen ja siihen saa useita laajennus komponentteja, kuten tässä tapauksessa piti löytyä tuki *OGRE engineen* ja *realXtendiin*.

Seuraavat kohdat kertovat lyhyesti valittujen ohjelmistojen ominaisuuksista.

3.3.1 ArchiCAD 16

ArchiCAD on rakennusalalla yleisesti käytetty mallinnusohjelmisto. ArchiCAD tarjoaa mahdollisuuden 2D-piirustusten ja 3D-mallien lisäksi myös visualisointiin. ArchiCAD:stä löytyy myös mahdollisuus erilaisten laajennuskomponenttien käyttöön. Sen etuna kilpailijoihin on se, että se on lokalisoitu vastaamaan suomalaisia suunnittelutarpeita. Kuvassa (kuva 7) näkymä ArchiCAD 16 käyttöliittymästä.



KUVA 7. ArchiCAD 16:n käyttöliittymä

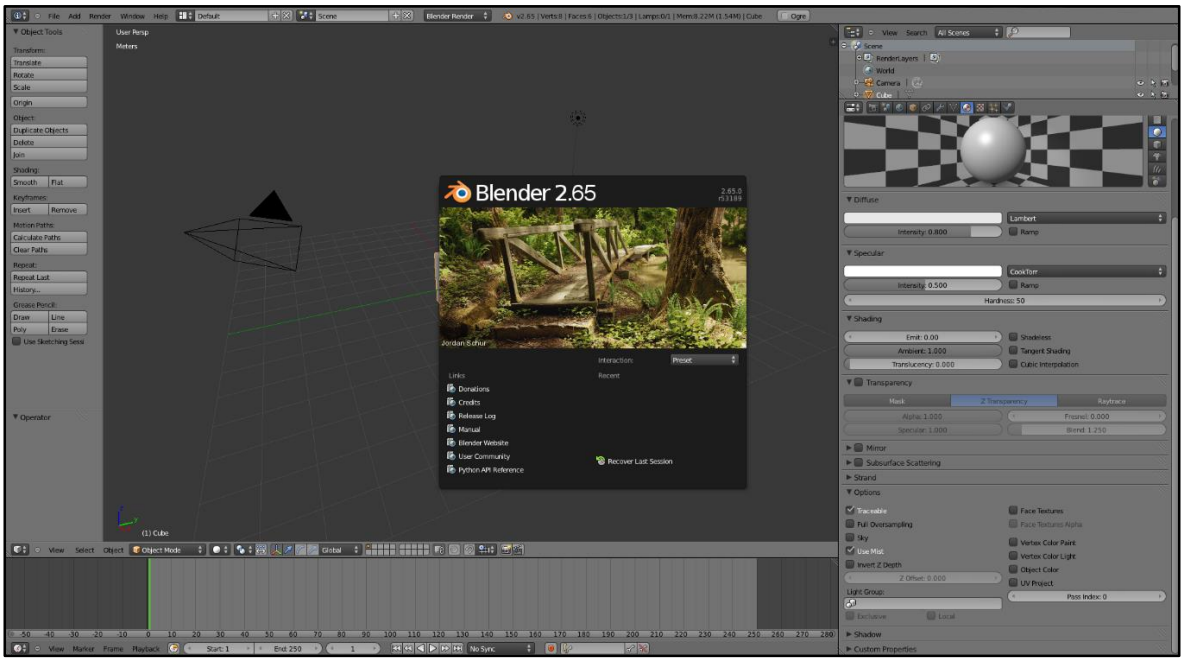
3.3.2 Blender 2.65

Blender on ilmainen avoimeen lähdekoodiin perustuva 3D-grafiikan tekoon tarkoitettu ohjelmisto, jolla voidaan muun muassa tehdä animaatioita, visuaalisia tehosteita tai pelejä.

Blenderin laajoihin ominaisuuksiin lukeutuvat muun muassa:

- animaation teko -ominaisuus
- fysiikka ja partikkeli -ominaisuudet
- hahmonmallinnus-ominaisuus
- luonnollisten varjojen teko -ominaisuudet
- laajennus-ominaisuudet
- monipuoliset renderointi -ominaisuudet
- pelien teko -ominaisuus
- Ray tracing-renderointi -ominaisuus
- tekstuurien- ja materiaalien muokkaus -ominaisuudet
- UV mapping -ominaisuus.

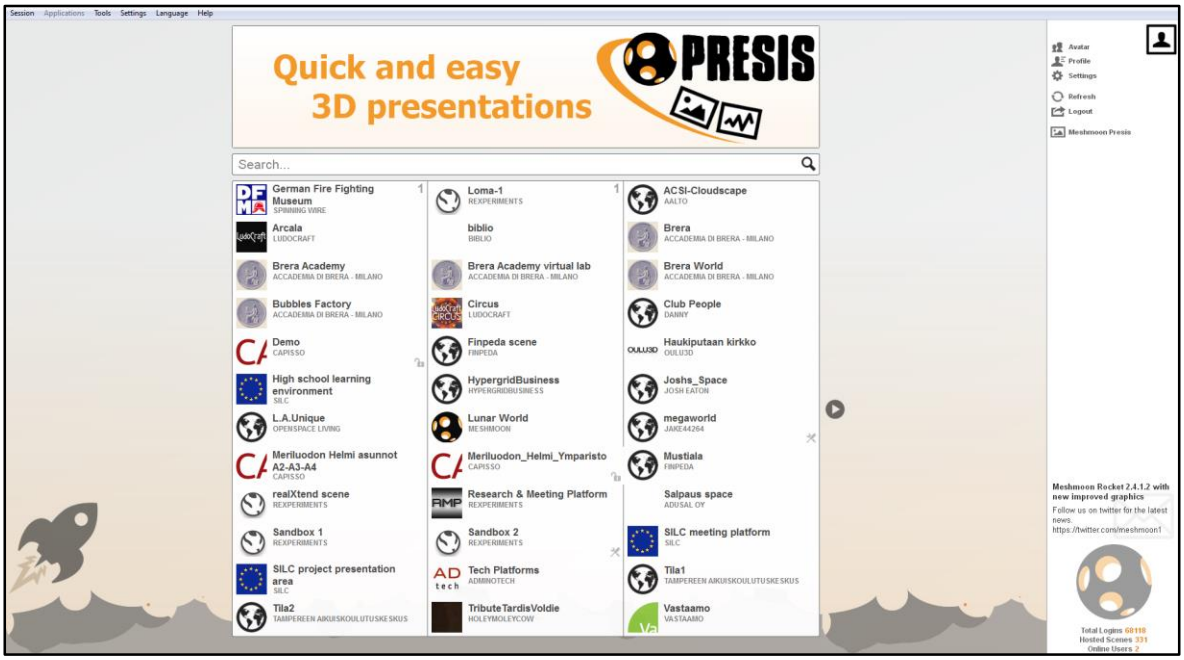
Blenderillä on hankalan ohjelmiston maine verrattuna muihin 3D-grafiikkan tekoon tarkoitettuihin ohjelmistoihin. Blenderin käyttö perustuu hyvin pitkälti näppäinkomentojen käyttöön. Alla olevassa kuvassa (kuva 8) perusnäkymä Blender 2.65 käyttöliittymästä.



KUVA 8. Blender 2.65 käyttöliittymä

3.3.3 Rocket client 2.4.1.2

Rocket client on suomalaisen Admino Technologies kehittämä avoimeen lähdekoodiin perustuvan *realXtendin* pohjalta luotu katseluohjelma, jolla varsinaisen virtuaalimallin katselu onnistuu. Rocket client käyttää *OGRE engineä* grafiikan tuottamiseen. Alla olevassa kuvassa (kuva 9) perusnäkymä Rocket clientin 2.4.1.2 käyttöliittymästä.



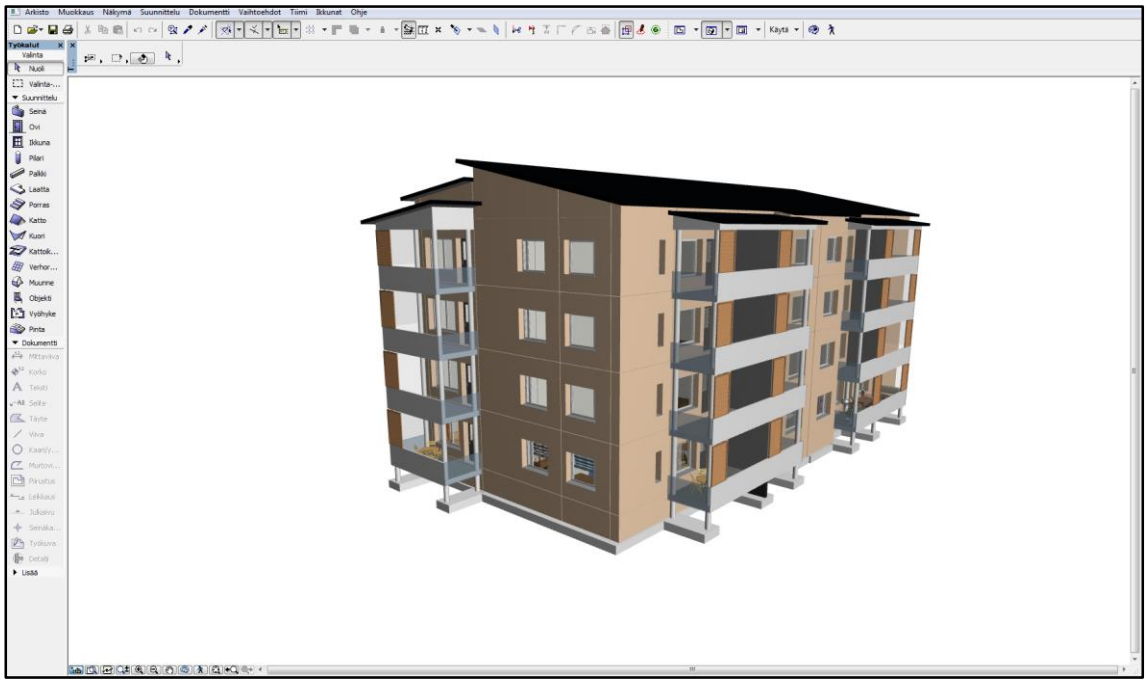
KUVA 9. Rocket client 2.4.1.2 käyttöliittymä

4 TUTKIMUKSEN PROSESSIKUVAUS – VIRTUAALIMALLI MERILUODON HELMI

Virtuaalimallin toteutusvaihetta käsittelevässä luvussa Meriludon helmen virtuaalimallin toteutusvaiheet jaetaan kolmeen pääosaan. Osat on jaoteltu ohjelmisto- sekä toteutusjärjestys perustein. Ensimmäinen vaihe käsittelee virtuaalimalli Meriluodon helmen käsittelyä ArchiCAD 16:ssa. Toinen vaihe käsittelee ArchiCAD-mallin jatkojalostusta Blenderissä ja kolmas vaihe käsittelee mallin työstöä Rocket clientiin ja Meshmooniin.

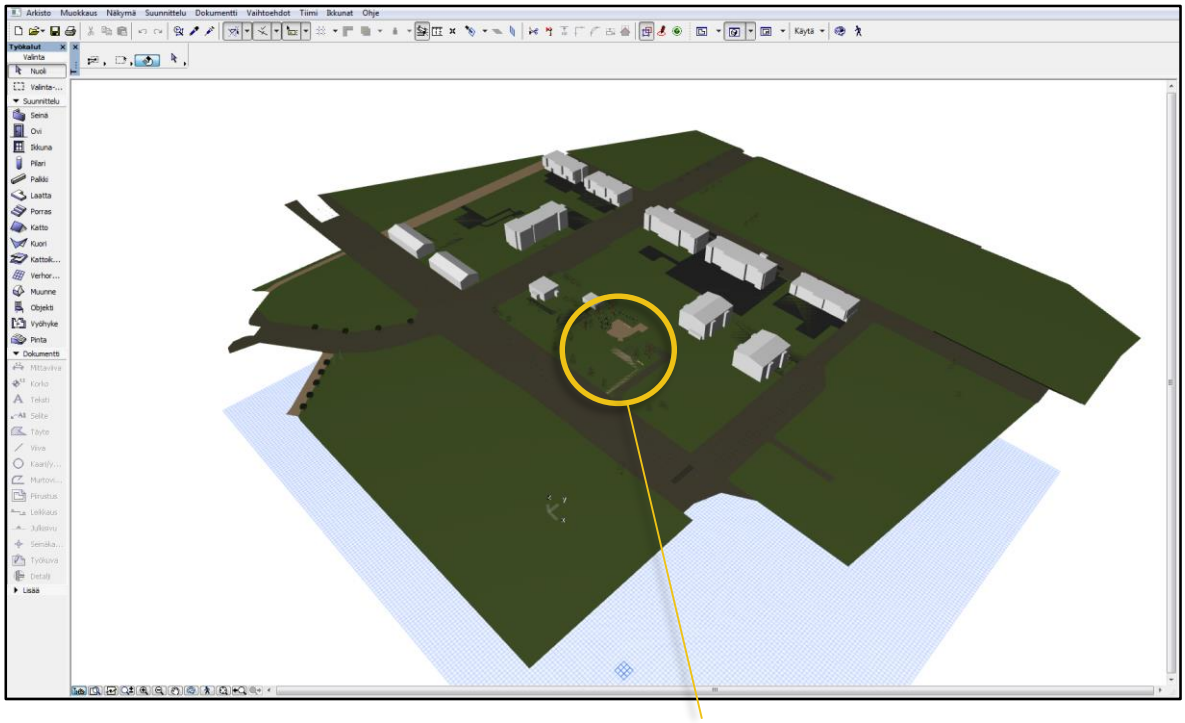
4.1 Ensimmäinen vaihe – ArchiCAD 16

Virtuaalimalli rakentui ArchiCAD:ssä tuotetun tietomallin pohjalle. Rakennusliike U. Lipsanen luovutti heidän suunnitteluvaiheessa olevan Pieksänmäelle rakennettavan As Oy Meriluodon helmen -tietomallin minulle käyttöön case-kohteeksi. Rakennuksen tietomalli oli jo työstetty siihen pisteeseen, että siitä löytyi koko rakennus sekä tontti mallinnettuna oikeaan korkoon. Alla olevassa kuvassa (kuva 10) As Oy Meriluodon helmen -tietomalli avattuna ArchiCAD:ssä.



KUVA 10. As Oy Meriluodon helmen -tietomalli

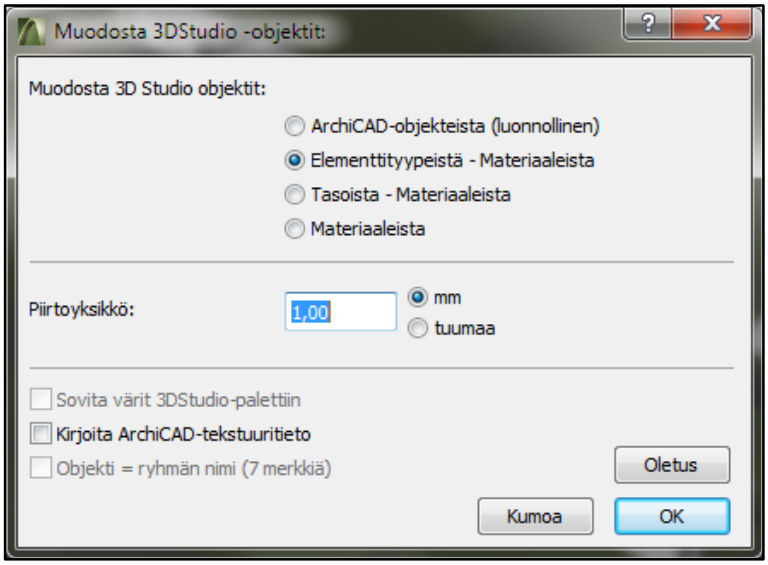
Tietomallin työstö aloitettiin karsimalla turhat kuormittavat tekijät pois. Kuormittavilla tekijöillä tarkoitetaan asunnoissa olevia huonekaluja. Seuraava vaihe oli tehdä ympäröivä maasto, koska pelkkä rakennuksen tontin maasto ei tässä tapauksessa riittänyt. Maaston tekoon käytettiin ArchiCAD 16:ssa tullutta uutta työkalua nimeltä *Muunne*, jolla pystytään tekemään kevyitä luonnosteluja ja vapaita muotoja. Ympäröivän maaston tekoon käytettiin apuna alueen *DWG*-muodossa olevaa kantakarttaa, jolla ympäröivät tiet ja kadut oli helppo toteuttaa. *DWG*-muodossa oleva kantakartta tuotiin ArchiCAD:in pohjalle ja kadut ja tiet luonnosteltiin sen päälle muunne-työkalua apuna käyttäen. Myös lähiympäristössä olevat rakennukset massoiteltiin kantakartan pohjalta muunne-työkalulla oikeille paikoille. Alla olevassa kuvassa (kuva 11) näkymä ArchiCAD:ssä mallinnetusta alueesta.



KUVA 11. Mallinnettu ympäristö As Oy Meriluodon Helmen tontti

Kun rakennus ja ympäristö oli saatu valmiiksi malli piti tallentaa Blenderiä tukevaan tiedostomuotoon. Seuraava kappale selvittää tietomallin tallennusta *3ds*-muotoon ja tallennusvaiheessa tehtäviä asetuksia.

Mallin tallennus *3ds*-muotoon onnistui ArchiCAD:ssä *Tallenne nimellä* -komennolla. Listasta valittiin tallennusmuodoksi 3DStudio-tiedosto. Seuraavaksi valittiin *3ds*-tiedoston tallennus asetukset eli asetukset, jotka määrittelevät miten *3ds*-tiedosto tuodaan ulos ohjelmasta. Alla olevassa kuvassa (kuva 12) asetukset, jotka totesin toimiviksi tähän tarkoitukseen.



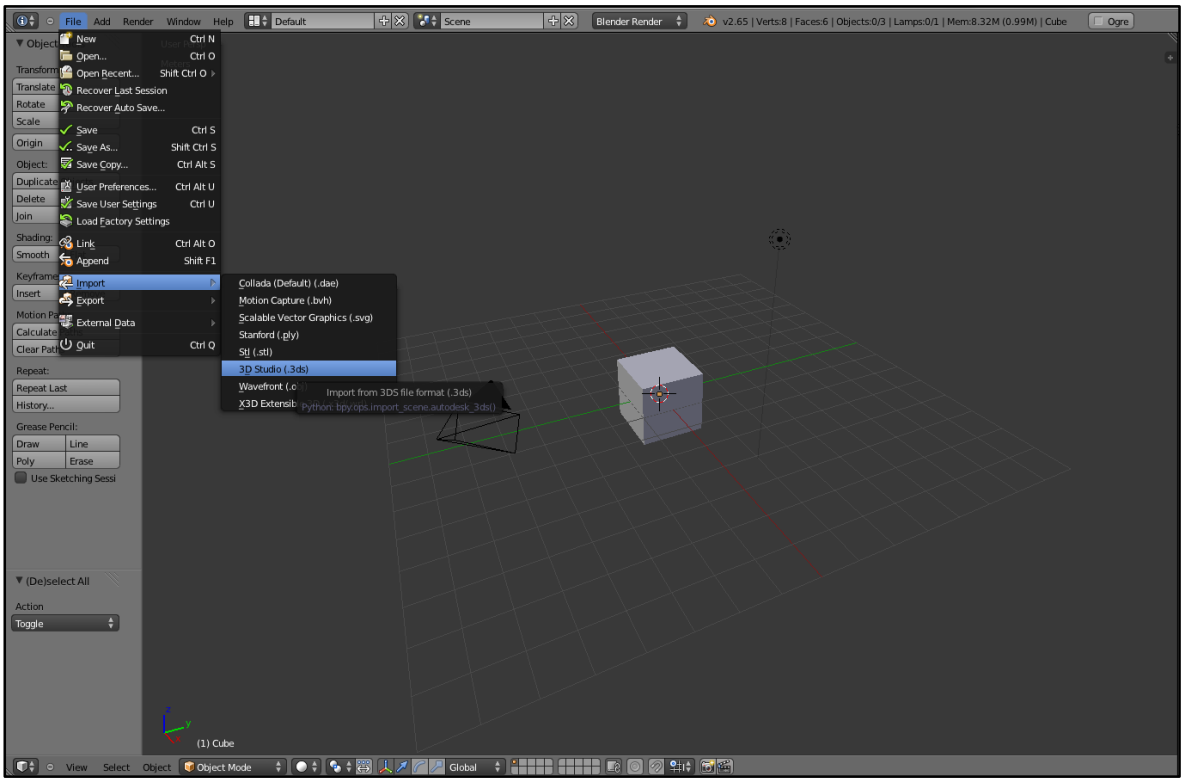
KUVA 12. Asetukset 3ds-tiedoston exportaukseen ArchiCAD:stä

Nyt oli saatu valmiiksi *3ds*-tiedosto jota voitiin alkaa jatkojalostamaan Blenderissä. Siihen vaiheeseen keskittyy seuraava luku.

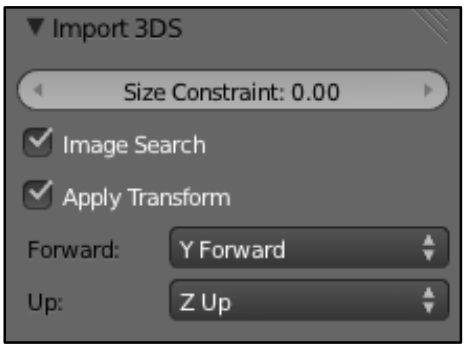
4.2 Toinen vaihe – Blender 2.65

Ensimmäiseksi Blenderiin piti asentaa *blender2ogre* -exporteri, joka mahdollistaa objektien exportauksen Rocket clientiin. Asennuksen jälkeen päästiin varsinaiseen työhön, virtuaalimallin tekoon.

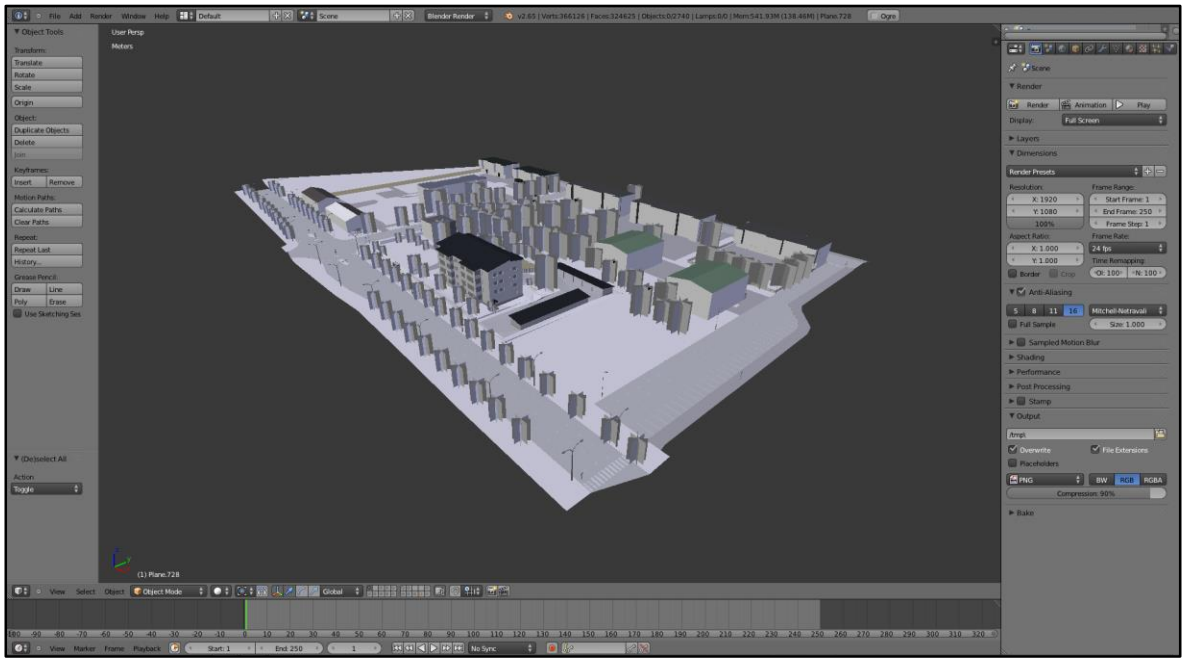
Blenderissä ensimmäiseksi avattiin *3ds*-tiedosto. Tiedosto tuotiin Blenderiin *Import*-käskyllä. (kuva 13.) Tuonti vaiheessa *Size Constraint* tuli asettaa vakioarvosta 10, nollaan. Tämä siksi, että malli skaalautuu oikein. (kuva 14.) Malli avautui Blenderiin seuraavan näköisenä, tässä kuvassa puut on korvattu yksinkertaisilla *planeilla*, jotka toteutettiin Blenderissä jälkikäteen. (kuva 15). Tämän jälkeen muutettiin Blenderissä olevat mittayksiköt metriseen järjestelmään. Se onnistui oikeassa laidassa olevasta *Scene* välilehdestä kohdasta *Units*. (kuva 16.)



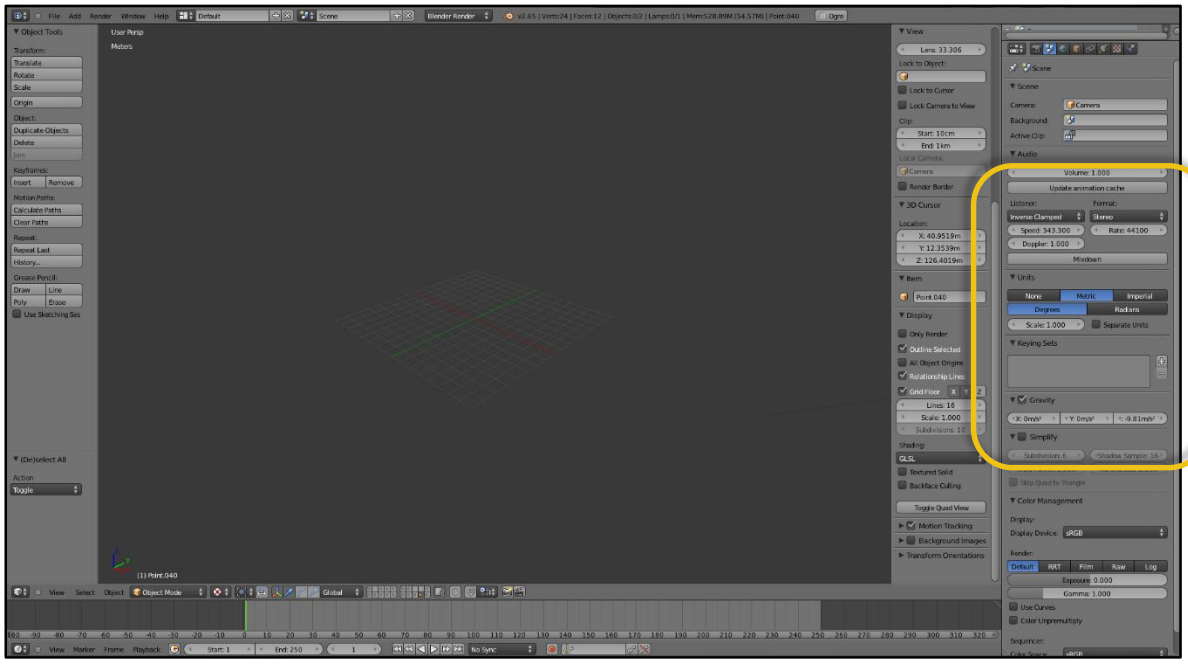
KUVA 13. 3ds-tiedoston tuonti Blenderiin



KUVA 14. Size Constraint -asetukset



KUVA 15. Malli Blenderissä



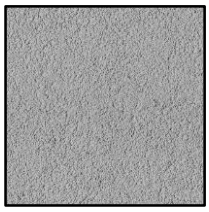
KUVA 16. Mittayksiköt

Kun malli oli tuotu Blenderiin oikein, alkoi materiaalien ja tekstuurien työstäminen. Siinä vaiheessa kun tiedosto tallennettiin ArchiCAD:ssä *3ds*-muotoon asetuksista ei valittu objekteille muuta kuin materiaalit eikä lainkaan tekstuureja eli kuvapintoja. Tämä siksi, koska Blenderin teksturointityökalu on paljon monipuolisempi mitä se on ArchiCAD:ssä ja näin tekstuureista tulee parempia. Seuraavaksi tarkastellaan miten materiaalit ja tekstuurit asetetaan Blenderissä, esimerkkinä rakennuksen ulko-seinä.

4.2.1 Materiaalien ja tekstuurien määritys

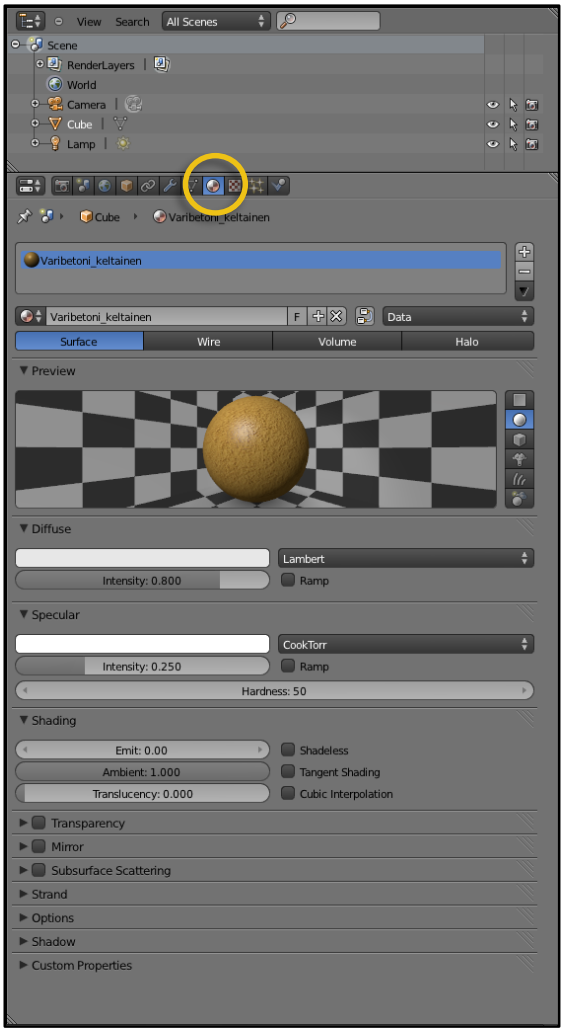
Seinän materiaalien ja tekstuurien asettaminen aloitettiin sillä, että valittiin seinään halutun lainen tekstuuri eli kuva. Tekstuureita voi ostaa ns. tekstuuripaketteina joita visualiseen 3D-suunnitteluun erikoistuneet sivustot tarjoavat tai hakea ilmaiseksi Internetistä tekstuureihin erikoistuneilta foorumeilta. Seinään valittiin tekstuuri maalausselosteen mukaan, joka oli tässä tapauksessa keltainen väribetoni. (kuva 17.)

KUVA 17. Varibetoni_seamless_col



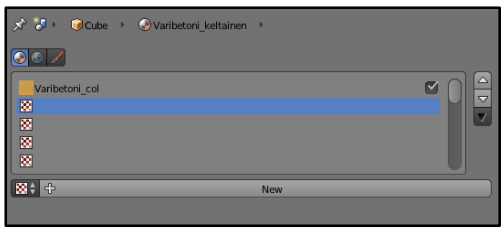
KUVA 18. Varibetoni_seamless_bump

Blenderissä materiaalien ja tekstuurien asennus aloitettiin *Materials* välilehdestä. (kuva 19.) Valikossa määriteltiin ensin materiaalin nimi. Nimeämisessä ei saa käyttää ä:tä, ö:tä, eikä välilyöntiä. Nimeämisessä hyväsääntö on se, että välilyönnit korvataan alaviiva-merkillä. Kuvassa näkyvät muut asetukset voidaan vaihtaa tilanteen mukaan ja vaikutukset selviää kokeilemalla.

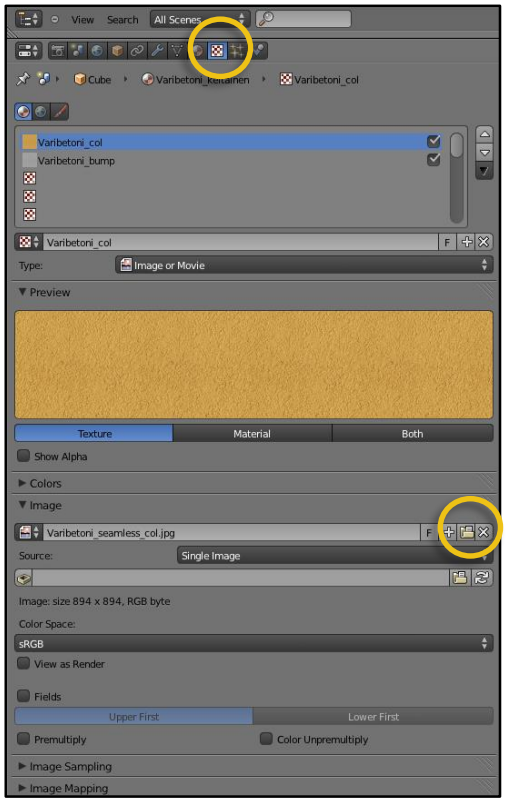


KUVA 19. Materiaaliasetukset

Textures välilehdestä asetettiin pinnoille haluttu kuva. Tässä vaiheessa nimeäminen toteutettiin niin, että varsinaisen kuvan nimenä käytettiin *Varibetoni_col*, joka viittaa värikuvaan ja *height mapping* -kuvana joka luo pinnalla rosoisuuden käytettiin *Varibetoni_bump* nimeä. Uuden tekstuurin teko samalle pinnalle onnistui painamalla uutta riviä ja valitsemalla kohta *New*. (kuva 20.) *Height mapping* -kuvan eli harmaasävykuvan voi tehdä esimerkiksi kuvankäsittelyohjelmalla, jossa värikuvan sävyt muutetaan harmaaksi. (kuva 18.) Jotta saatiin kuvat tekstuureihin tuli valita tekstuurityypiksi, *Image or Movie*. *Image*-valikosta haettiin *Open Image* -kohdasta haluttu kuvatiedosto. (kuva 21.) Kuvien kooksi suositellaan pow² -tyylisiä resoluutioita mm. 256x256, 512x512, 1024x1024 ja 2048x2048. *Varibetoni_col* ja *Varibetoni_bump* -kuvissa oli käytössä resoluutio 894x894, joka ei sinänsä ole ongelma, mutta jos virtuaalimallissa on paljon poikkeavia kuva kokoja se tekee näytönohjaimelle kuvien laskemisen raskaaksi. Kuvissa 22 ja 23 tarkemmat asetukset *Varibetoni_col* ja *Varibetoni_bump* -kuville. Muut asetukset voi jättää vakioiksi.

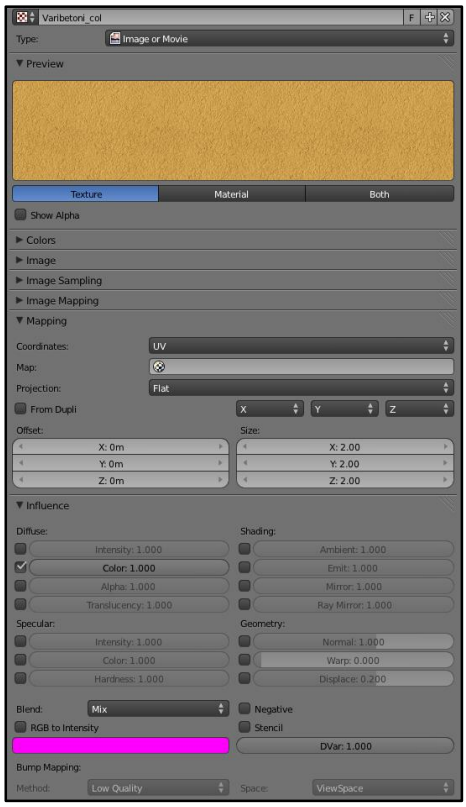


KUVA 20. Uusi tekstuuri



KUVA 21. Open image

Varibetoni_col tekstuurinasetukset:

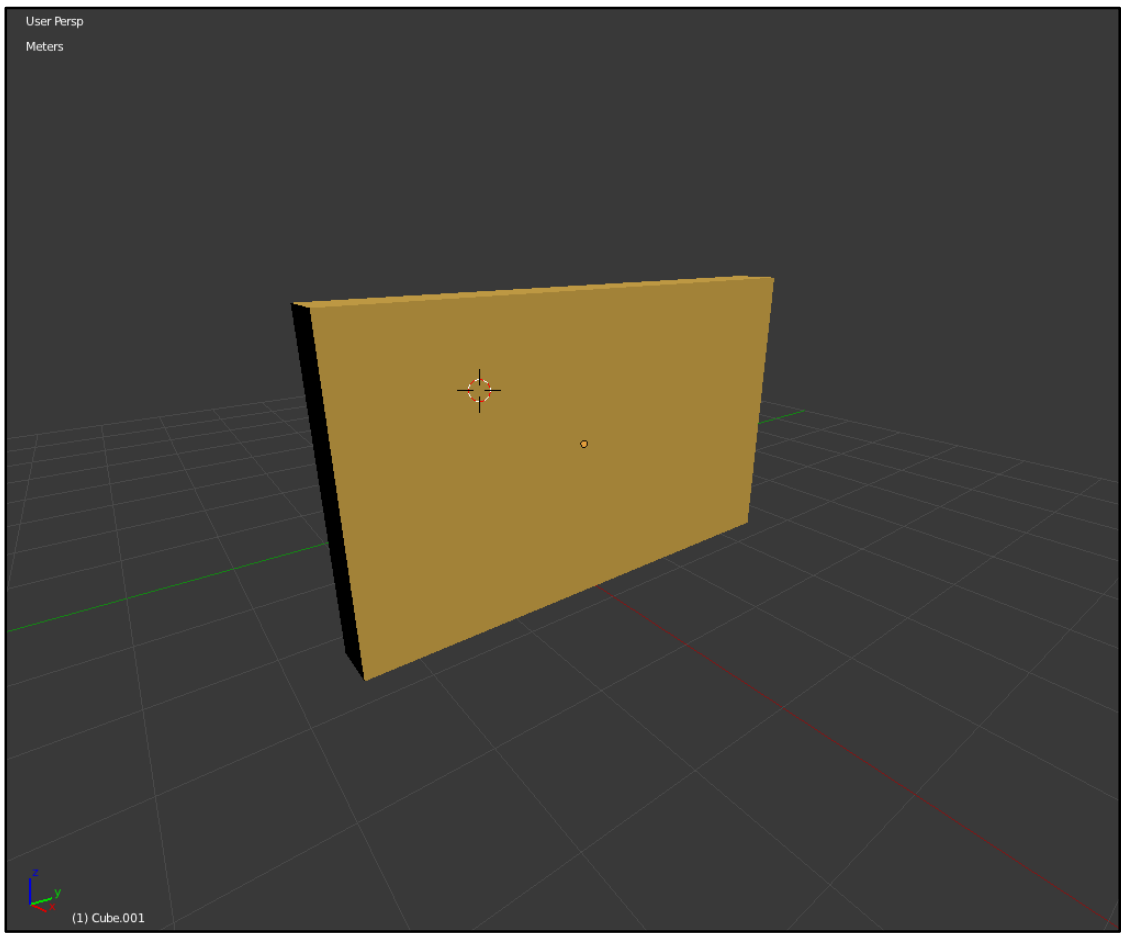


KUVA 22. Asetukset Varibetoni_col

Varibetoni_bump tekstuurinasetukset:

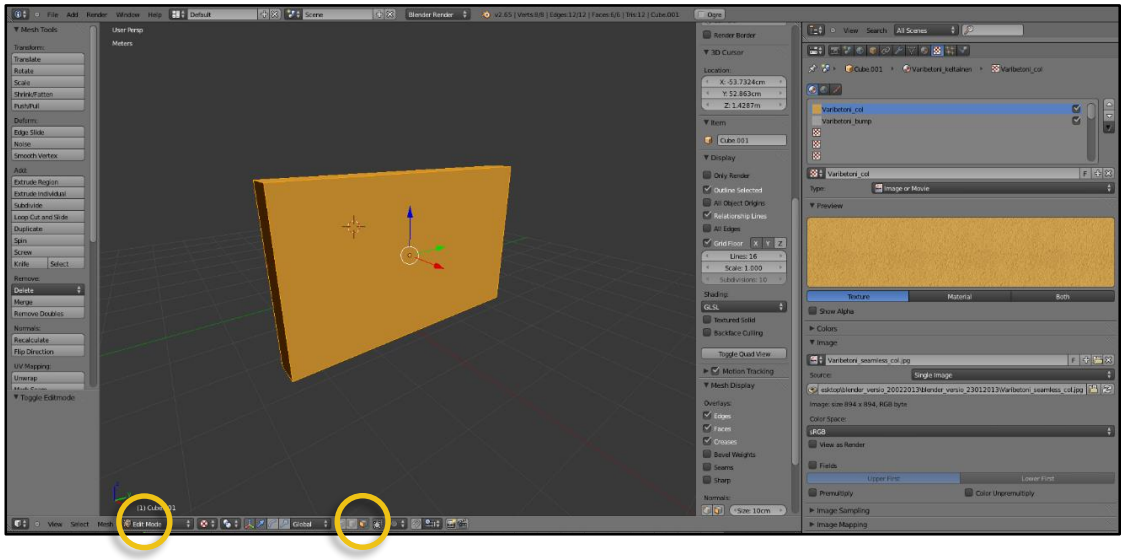
KUVA 23. Asetukset Varibetoni_bump

Kun meshille oli määritelty tekstuurit ja materiaalit mitä sen tulisi käyttää, piti se vielä *UV mapata*. Ennen *UV mappingia* seinä näytti tältä. (kuva 24.)



KUVA 24. Seinä ennen UV mappingia

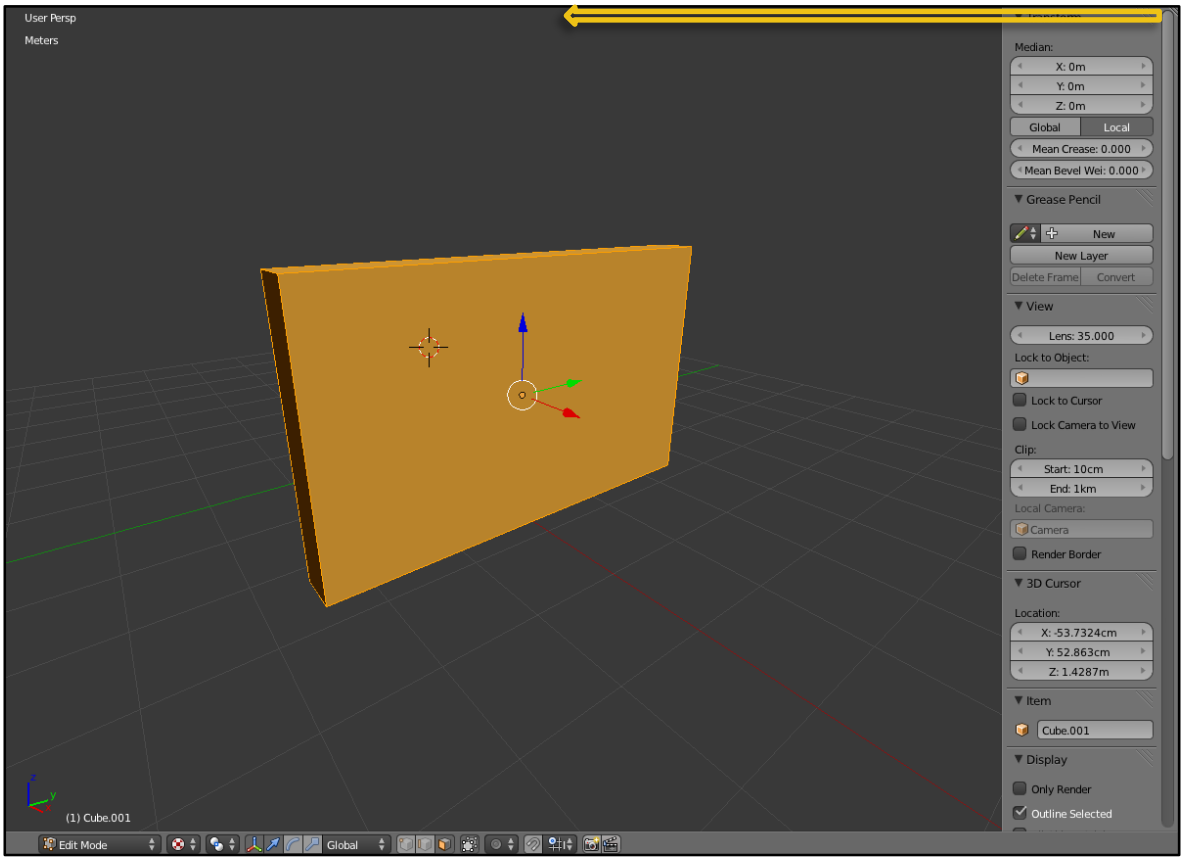
UV mapping Blenderissä toimi seuraavasti. Valittiin meshi — tässä tapauksessa seinä —, painettiin *tab*-näppäintä (vaihtoehtoisesti valikosta), jolloin Blenderissä siirryttiin objektitilasta (*Object Mode*) editointitilaan (*Edit Mode*), jossa voitiin muokata meshejä. Avautui seuraavanlainen näkymä. (kuva 25.)



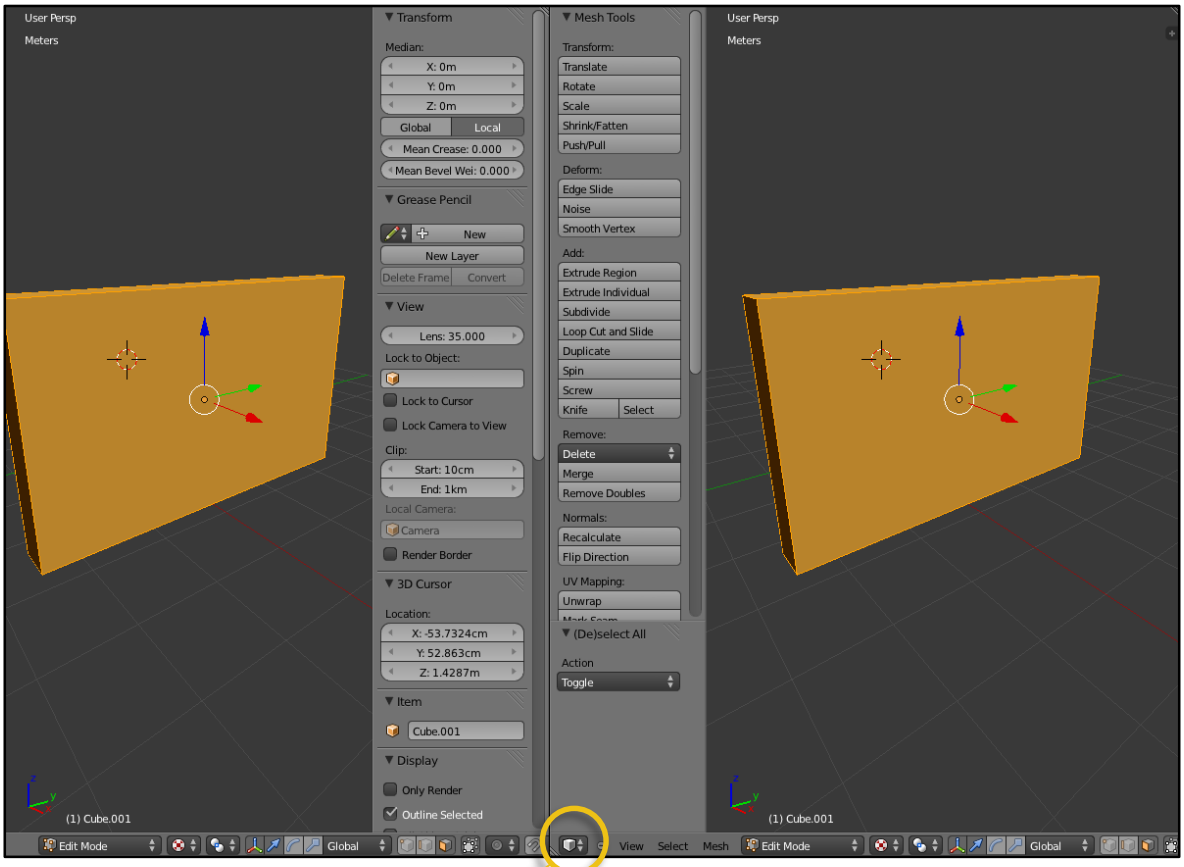
KUVA 25. Edit mode

Hiiren oikeata painiketta klikkaamalla sai seinänpintoja valituksi, jotta pystyi valitsemaan pintoja, piti valita alanauhasta *Face select.* (kuva 25.) Vaihtoehtoisesti voitiin painaa A-näppäintä jolloin kaikki pinnat valittiin, valitut pinnat muuttuivat oranssin värisiksi. Tässä tapauksessa kaikki seinäpinnat valittiin.

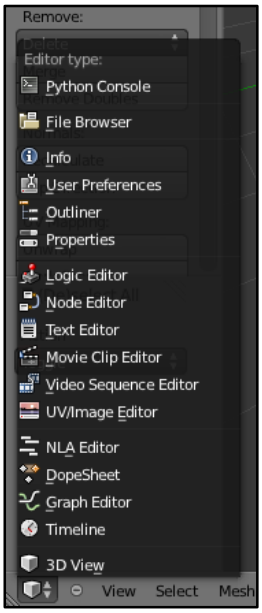
Seuraavaksi jaettiin näkymä kahteen osaan, oikeasta ylälaidasta raahattiin kursoria vasen hiiren painike pohjassa (kuva 26) ja uusi ikkuna ilmestyi käyttöön. (kuva 27.)



KUVA 26. Raahaus

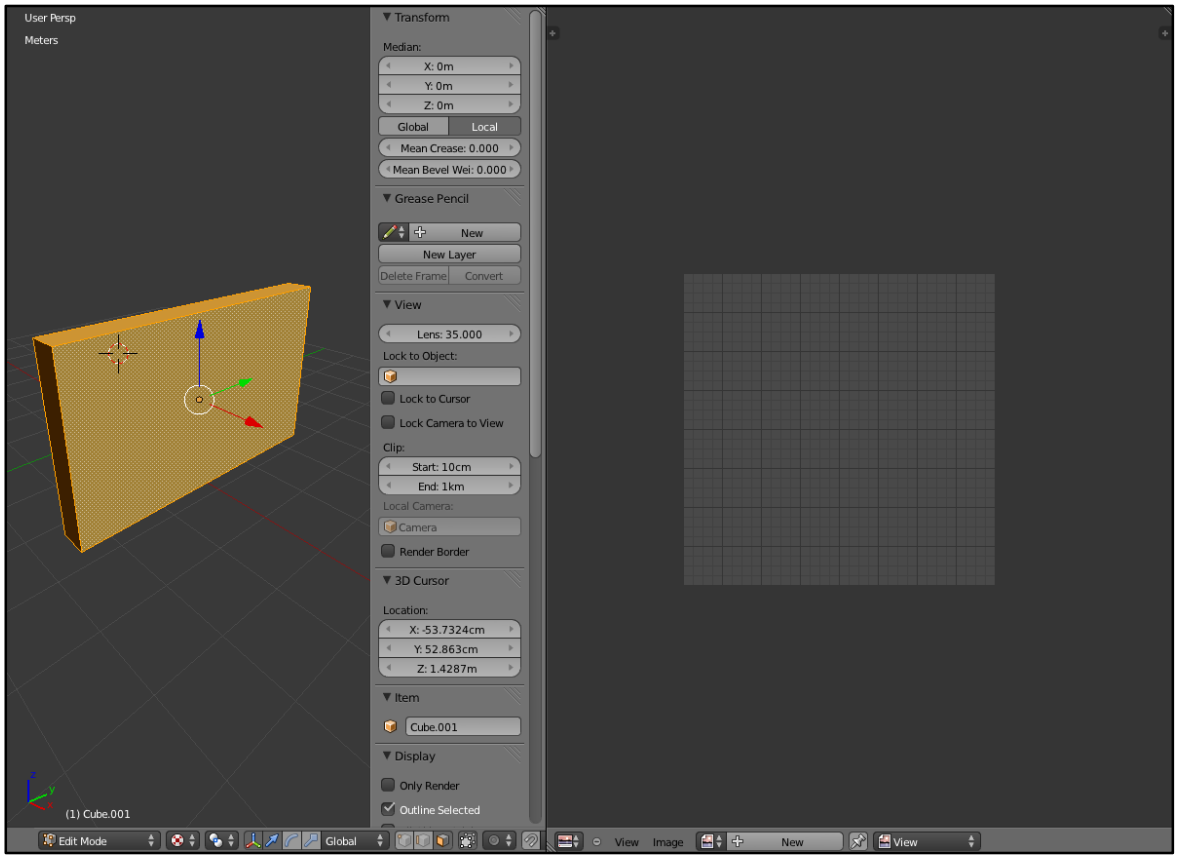


KUVA 27. Jaettu näkymä



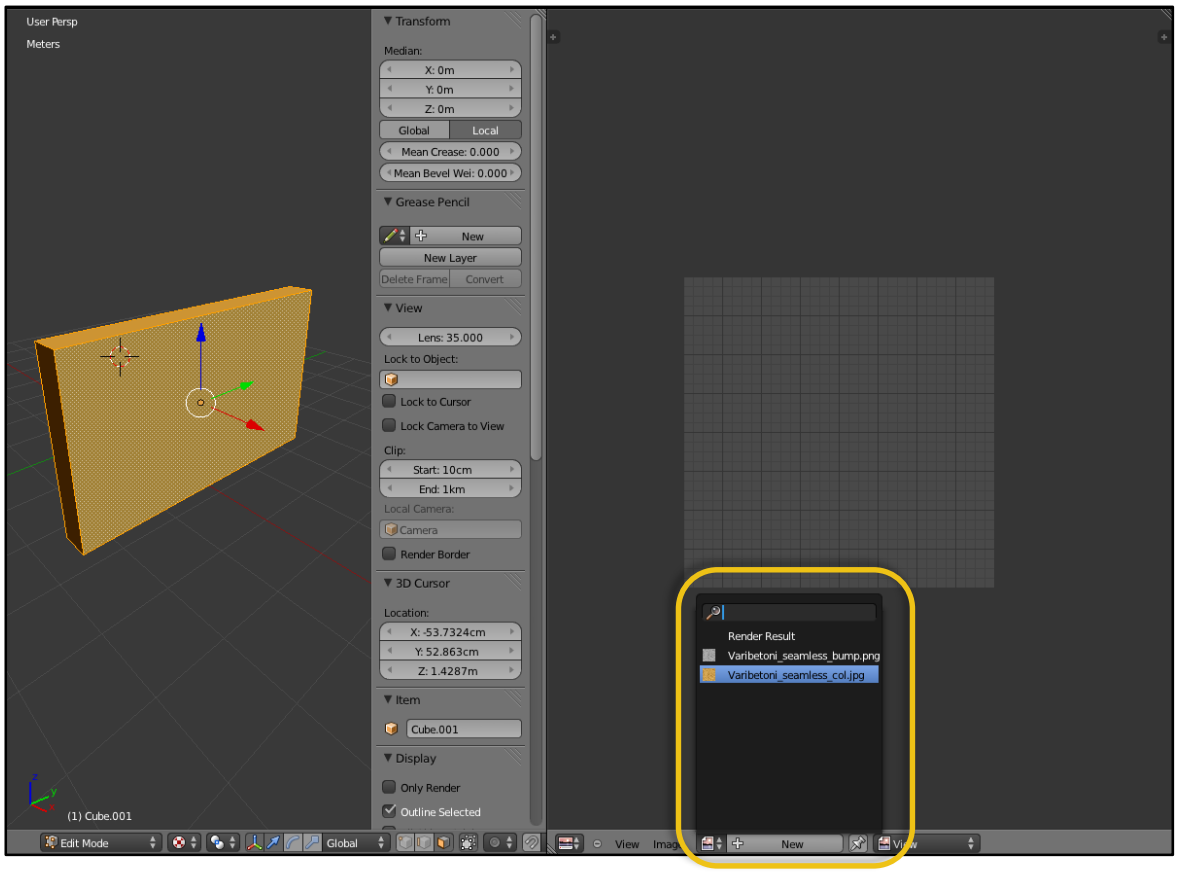
KUVA 28. Tyyppi-valikko

Tässä vaiheessa jaetusta näkymästä toinen puoli muutettiin *UV/Image editoriksi*. (kuva 28.) Muunnoksen jälkeen näkymästä tuli seuraavanlainen. (kuva 29.)



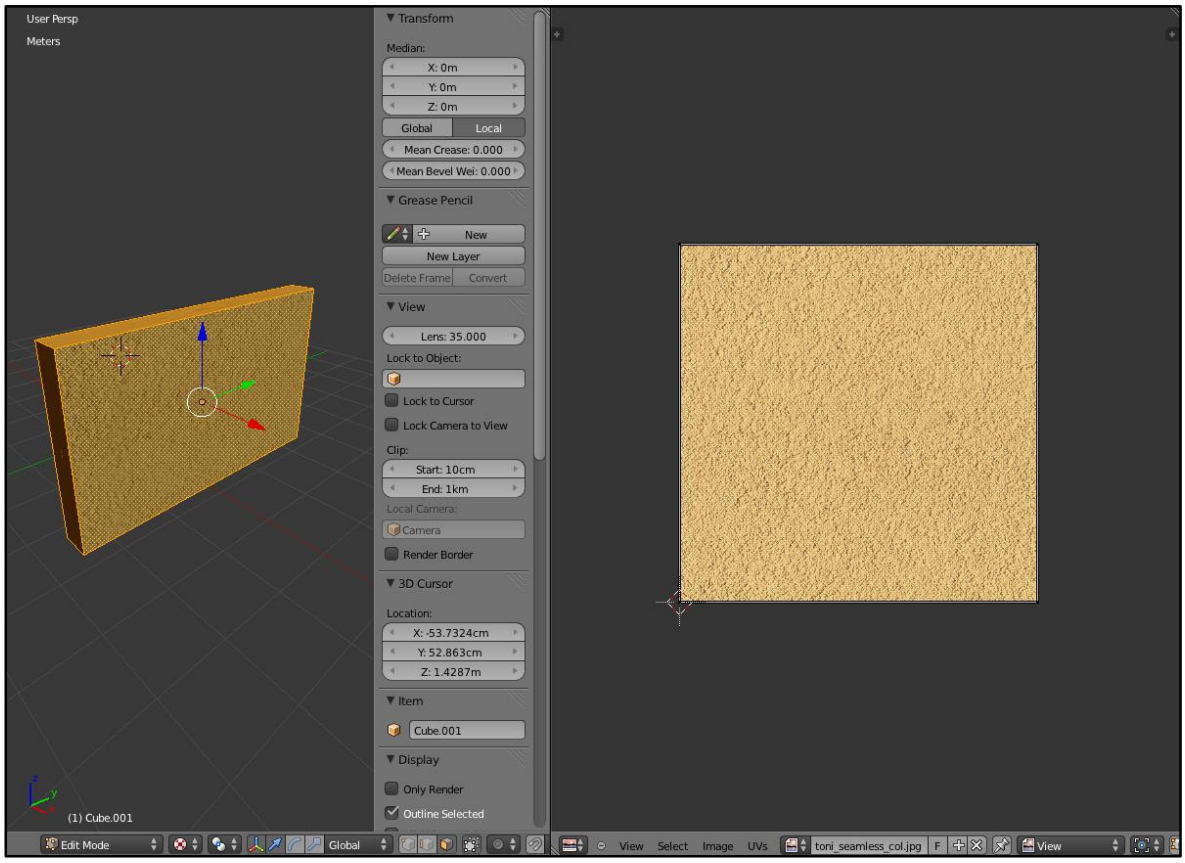
KUVA 29. 3Dview- ja UV/Image editor

Kaikki seinäpinnat valittuna, valittiin *UV/Image editorista* sama kuva jota käytettiin seinän tekstuurien teko vaiheessa, *Varibetoni_seamless_col*. (kuva 30.)



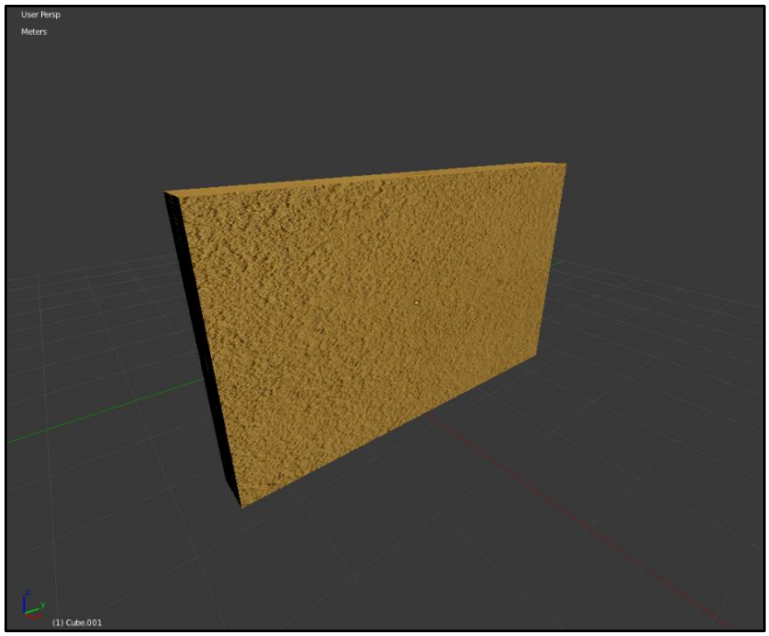
KUVA 30. UV mappingiin käytettävän kuvan valinta UV/Image editoriin

Kuvan valinnan jälkeen näkymä näytti seuraavalta. (kuva 31.) Jos kuva ei asetu tasaisesti valituille pinnoille voi käyttää komentoa nimeltä *Unwrap* (U-näppäin), jolloin Blender tekee kuvan pinnalle automaattisesti oikein.



KUVA 31. UV mappingissa käytettävä kuva valittuna UV/Image editoriin

Nyt oli saatu meshi *UV mapatuksi* ja editointitilasta päästiin takaisin objektitilaan *tab*-näppäimellä. Alla olevassa kuvassa valmis meshi. (kuva 32.)



KUVA 32. UV mappingilla toteutettu meshi

Tätä samaa teksturointikaavaa voitiin käyttää muidenkin meshien teksturointiin. *UV mappingia* ei silti tarvitse käyttää kaikkiin mesheihin, esimerkkinä ArchiCAD:stä tuodut meshit on jo valmiiksi *UV mapatuja* kun ne tuodaan Blenderiin. Silloin riittää vain se, että vaihtaa materiaalit ja teksturit Blenderissä kohdilleen. Jos Blenderissä joudutaan mallintamaan meshejä tai ArchiCAD-meshejä joudutaan raskaasti muokkaamaan Blenderissä, vaaditaan meshien *UV mappingia*.

Seinän teksturointi esimerkissä näytetään vain havainnollistamis tarkoituksessa, miten *bump mapping* tässä tapauksessa *height mapping* toteutetaan. Rocket clientin suora *bump mapping* -kuvien puutos aiheutti sen, että kaikki tekstuurit virtuaalimalliin tehtiin pelkkien värikuvien avulla.

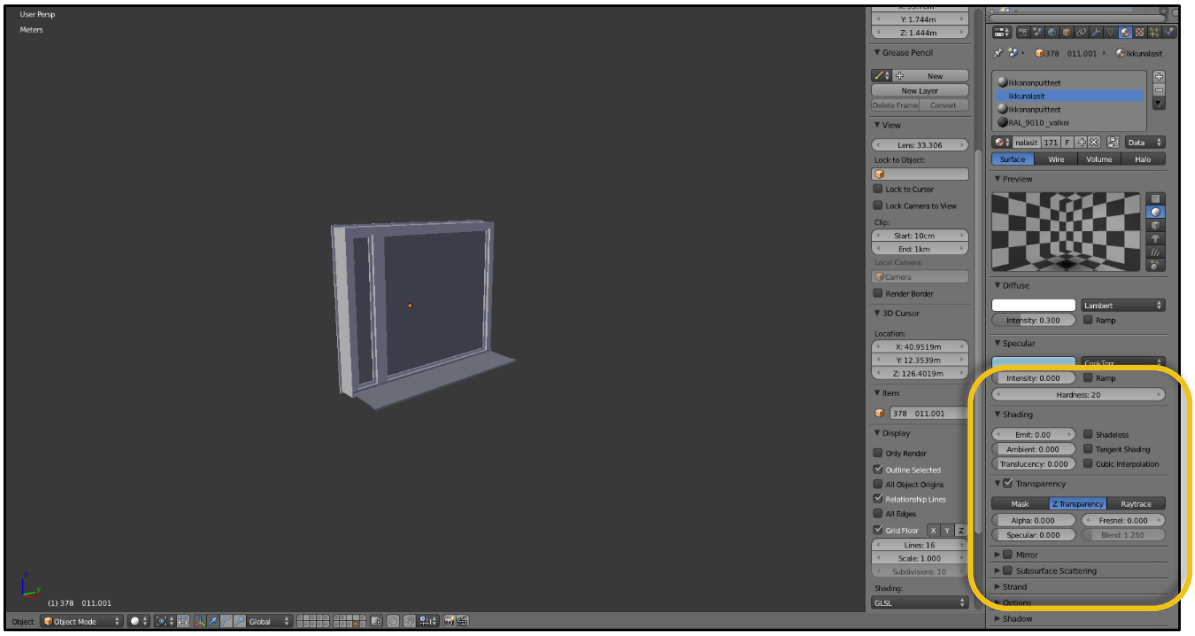
Alla olevassa kuvassa (kuva 33) esimerkki *UV mappingilla* toteutetusta rakennuksesta. Koska mallista ei haluttu liian raskasta oli helpoin keino havainnollistaa ympäristön muita rakennuksia luomalla yksinkertaisia massoja rakennuksista ja *UV mappaamalla* ne alueen rakennuksen kuvilla, jolloin saatiin vaikutelma oikeista rakennuksista sekä ympäristöstä aikaiseksi.

KUVA 33. UV mappingilla toteutettu rakennus

4.2.2 Materiaalien ja tekstuurien määritys – Erityisobjektit

Ikkunat

Ikkunoiden tuonti Blenderiin *3ds*-muodossa ei tehnyt ikkunan lasiosasta läpinäkyvää vaan se piti määritellä läpinäkyväksi Blenderin materiaalityökalulla. Lasin teossa riitti pelkkä materiaalin uudelleen määritys, koska ei ollut tarpeellista luoda tasaiselle lasipinnalle tekstuuria. Lasin sai läpinäkyväksi asettamalla ruksi kohtaan *Transparency* ja asettamalla *alphan*-arvo ykkösestä nollaan. Lasin materiaaliasetukset kuvassa 34.



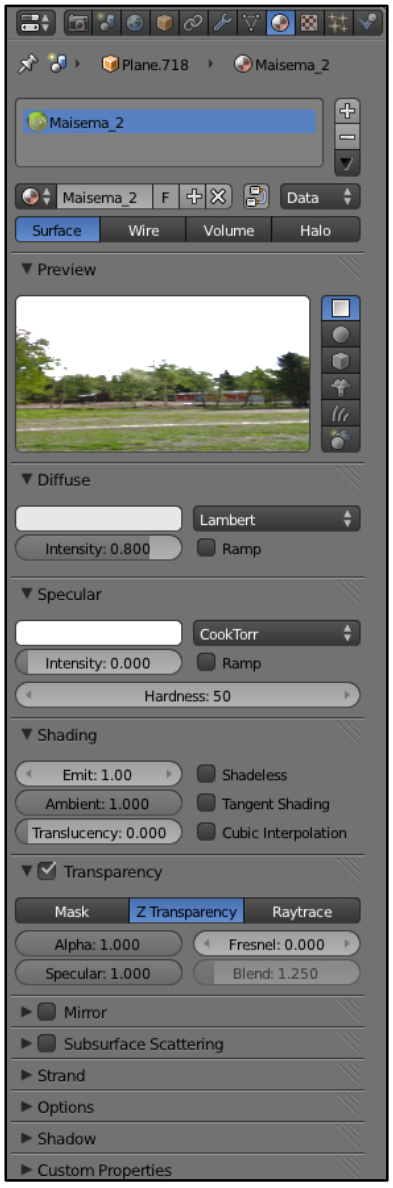
KUVA 34. Lasin materiaaliasetukset

Maisema-plane

Tässä virtuaalimallissa rakennuksen ympäröivää aluetta haluttiin havainnollistaan kuvien avulla. Kuville tehtiin *plane*, jolle kuva *UV mapatiin*. *Plane* on meshi, jolla on vain x-, ja y-koordinaatistoarvot mutta ei z-koordinaattiarvoa eli tässä tapauksessa syvyyttä. Ennen *UV mappingia* kuva jouduttiin käsittelemään kuvankäsittelyohjelmassa, jossa kuvalle määriteltiin *alpha*-kanava. *Alpha*-kanava on tyypillisesti läpinäkyvyyskanava, joka mahdollistaa sen, että kuvaan voidaan määritellä tietty alue läpinäkyväksi, tässä tapauksessa kuvista määriteltiin läpinäkyväksi alueeksi taivas. (kuva 35.)

KUVA 35. Alpha-kanavakuva

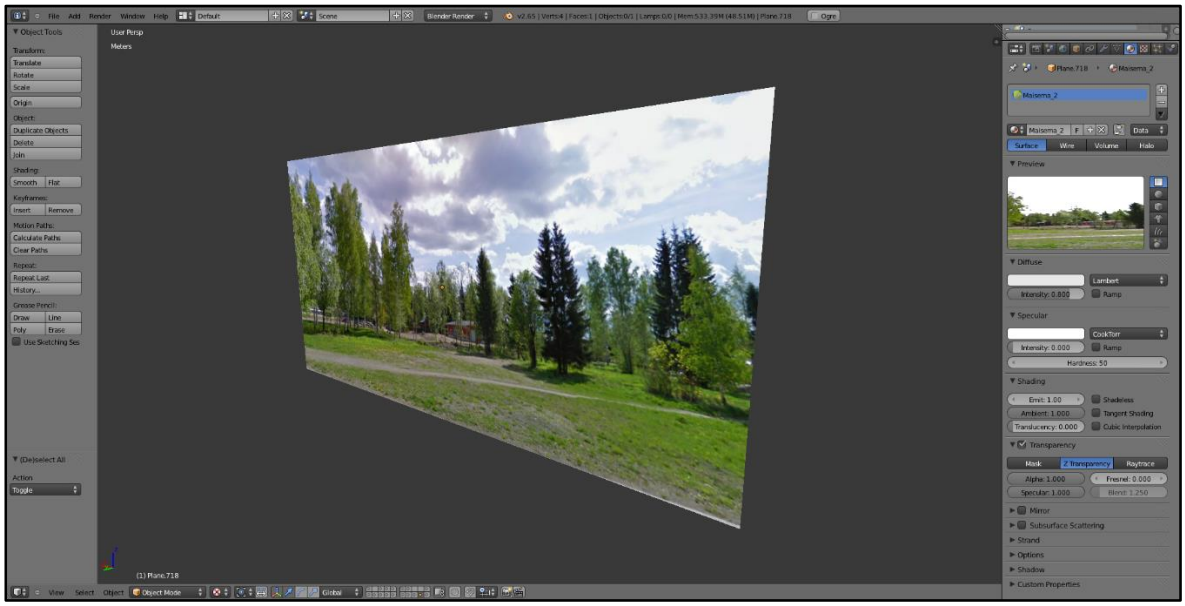
Alpha-kanavan lisäyksen jälkeen kuvat *UV mapatiin planelle* ja materiaali- ja tekstuuriasetuksiin käytettiin seuraavia asetuksia. (kuvat 36 ja 37.)



KUVA 36. Materiaaliasetukset

KUVA 37. Tekstuuriasetukset

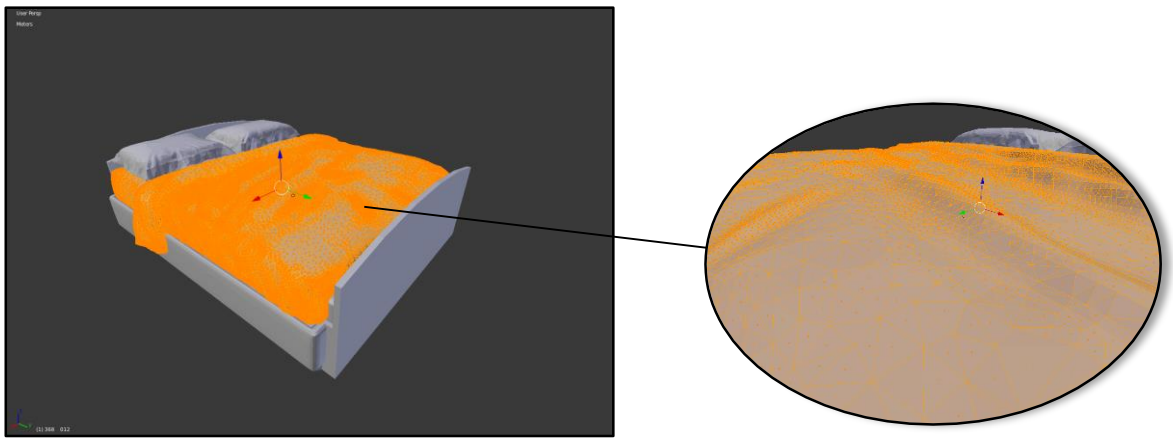
Blenderissä *alpha*-arvon vaikutusta ei näkynyt (kuva 38) mutta, kun meshi vietiin Rocket clientiin tuli asetettu *alpha*-kanava voimaan.



KUVA 38. Alpha-kanavakuva UV mapatuna planelle

Puut

Tässä tapauksessa virtuaalimallin puiden toteutustapaan oli kaksi tyyliä. Puut voitiin toteuttaa ns. *low poly* -mesheinä tai käyttämällä avuksi *planeja* ja kuvia sekä kuviin asetetun *alpha*-kanavan yhdistelmää. *Low poly* -käsite tarkoittaa 3D-meshiä, joka koostuu vähäisemmästä määrästä monikulmioita mitä *high poly* -meshi, meshit siis rakentuvat pienistä tai suurista määristä monikulmioita. Mitä vähemmän monikulmioita on, meshi on kevyempi mutta ulkonäöllisesti huonomman näköinen. Mitä enemmän monikulmioita on, meshi on raskaampi koneelle käsitellä mutta ulkonäkö parantuu. Esimerkkinä *high poly* -meshi, sängynpeitto on hyvin yksityiskohtaisesti mallinnettu ja sisältää 64 999 monikulmiota, joka tekee meshistä hyvin raskaan. (kuvio 4.) Blenderissä on myös olemassa käsky nimeltä *decimate*, jolla monikulmioiden määrää mesheissä saa karsittua, tästä kerrotaan luvussa ”tärkeät toiminnot”.

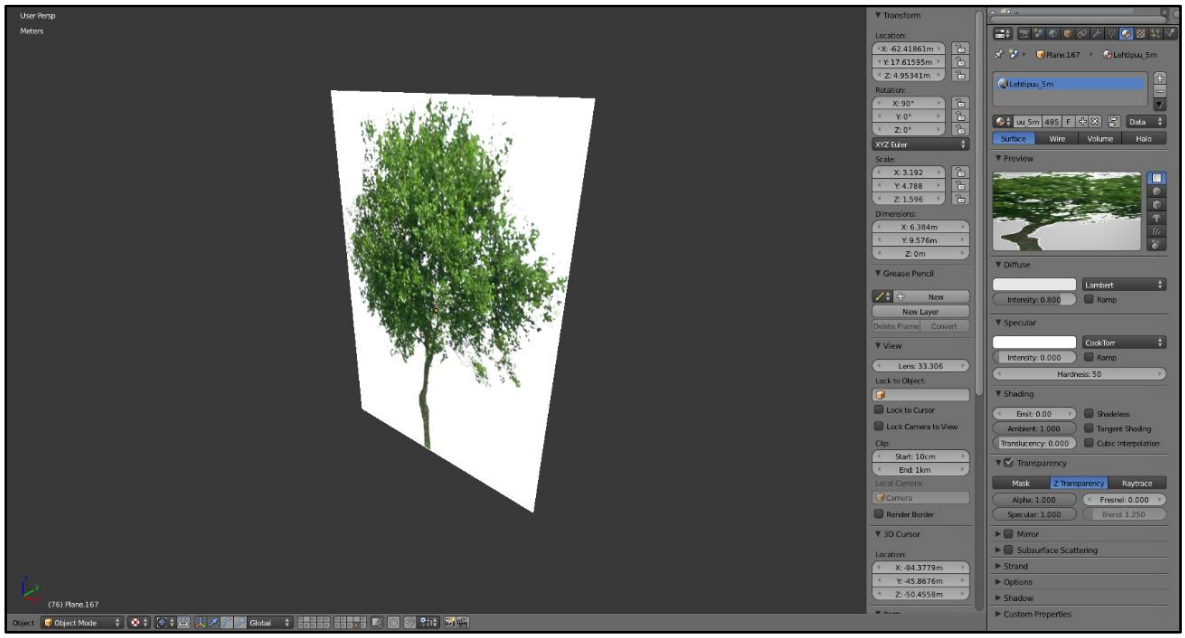


KUVIO 4. Esimerkki high poly -meshistä

Seuraavaksi tarkastellaan puiden tekoa virtuaalimalliin kuvien avulla, koska tässä työssä puut toteutettiin virtuaalimalliin kyseisellä tavalla.

Puiden teko aloitettiin valitsemalla sopivat kuvat mallinnettavista puista. (kuva 39.) Kuvien löydyttyä, kuvat *UV mapatiin planeille* samalla tavalla kuin maisemat *UV mapatiin*, kuvissa käytettiin *alpha*-kanavaa. Tulos näytti Blenderissä seuraavalta. (kuva 40.)

KUVA 39. Alpha-kanavakuva

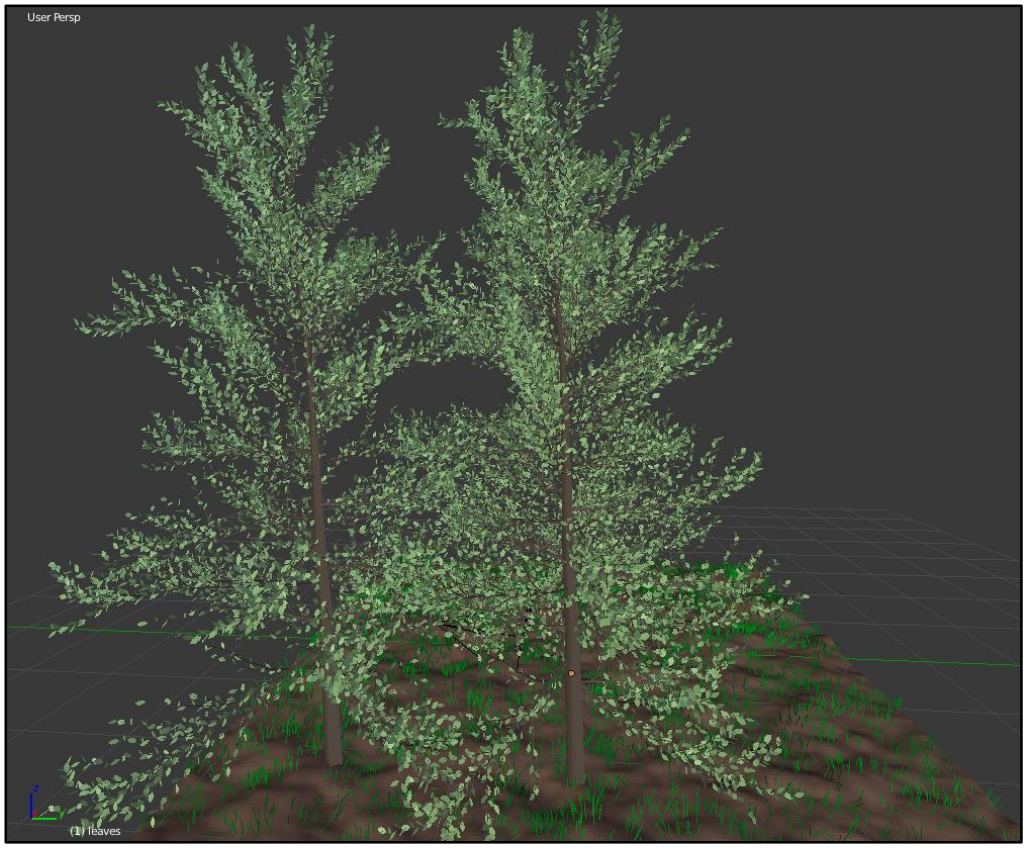


KUVA 40. Puunkuva UV mapatuna planelle

Jotta puusta saatiin enemmän realistisen tuntuinen tuli *planeja* kopioida z-akselin ympäri 45°:n välein näin puut näkyivät virtuaalimallissa joka kulmasta katseltuna oikein. Yhtä puuta kohti tuli silloin neljä *planea*. (kuva 41.)

KUVA 41. Virtuaalimallissa toimiva puu

Puiden toteutus kuvina vähensi huomattavasti virtuaalimallin lopullista kokoa ja virtuaalimallin käytettävyys parani, koska mallissa ei ollut raskaita 3D-meshejä. Alla olevassa kuvassa (kuva 42) esimerkki *high poly* -puista joissa yhdessä puussa (lehdet+runko) oli 52 000 monikulmiota. *High poly* -puiden käyttäminen mallissa lisäsi huomattavasti mallin visuaalista ilmettä, mutta samalla teki mallista raskaan.



KUVA 42. High poly -puita

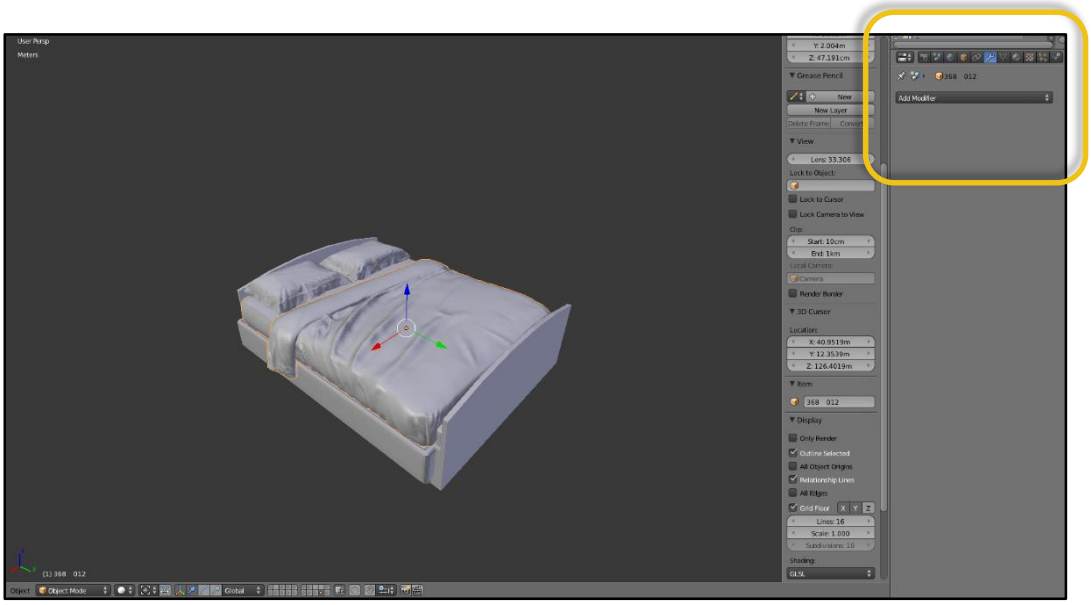
4.2.3 Tärkeät toiminnot

Tämä luku kertoo parista tärkeästä toiminnosta, joita virtuaalimallin tekovaiheessa tarvittiin.

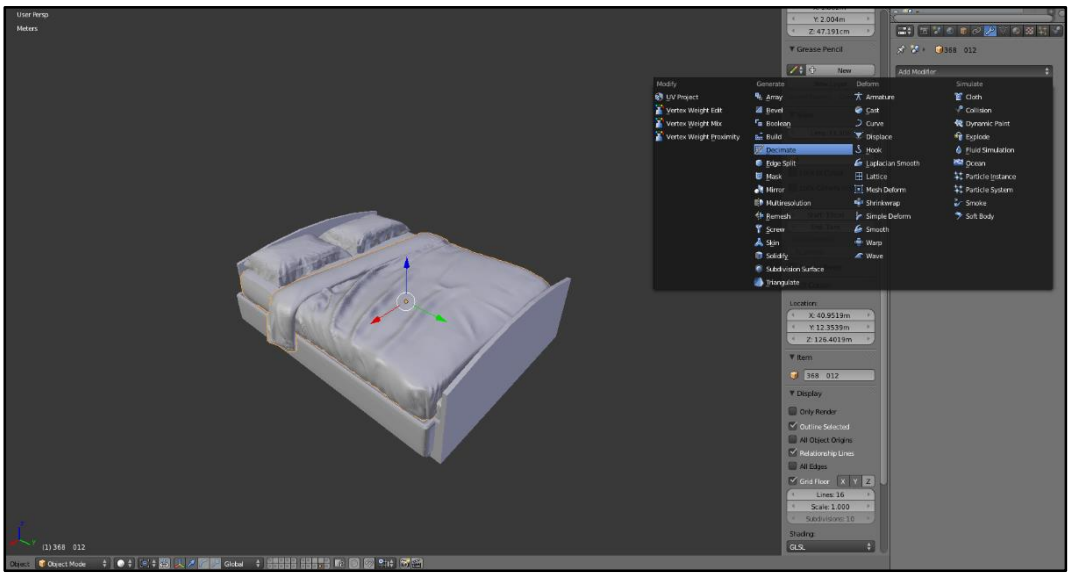
Decimate-käsky

Kuten aiemmin jo kerroin *low-* ja *high poly* -meshien taustaa, niin tässä luvussa kerrotaan miten mesheistä voidaan karsia monikulmioita *decimate*-käskyllä.

Ensimmäiseksi valittiin karsittava meshi, tässä tapauksessa sänky. Valittiin *Object modifier* välilehti (kuva 43) ja *Add modifier* -valikosta *Decimate*. (kuva 44.) Nyt oli lisätty meshiin *decimate*-käsky, jolla karsinta tapahtui. Kun *decimate*-käsky oli lisätty, tuli näkyviin montako monikulmiota meshissä on. *Ratio*-arvoa muuttamalla saatiin monikulmioita karsittua. Ykkönen vastaa 100 %:a jolloin kaikki monikulmiot on käytössä ja mitä pienempi luku käytössä sitä vähemmän monikulmioita on.

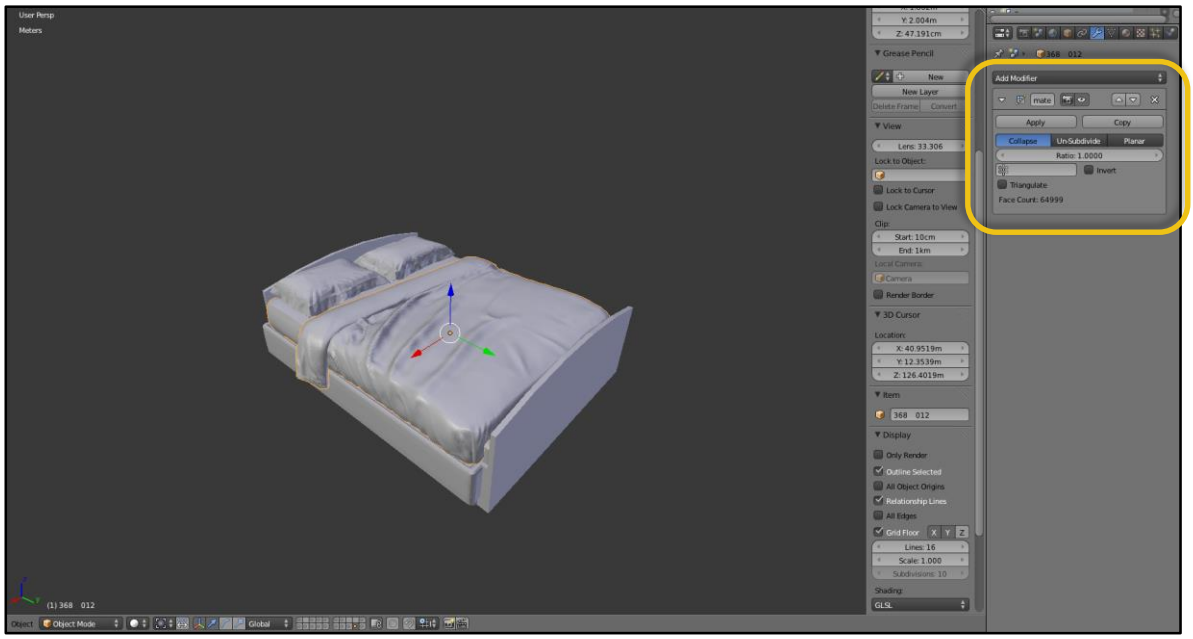


KUVA 43. Object modifier välilehti

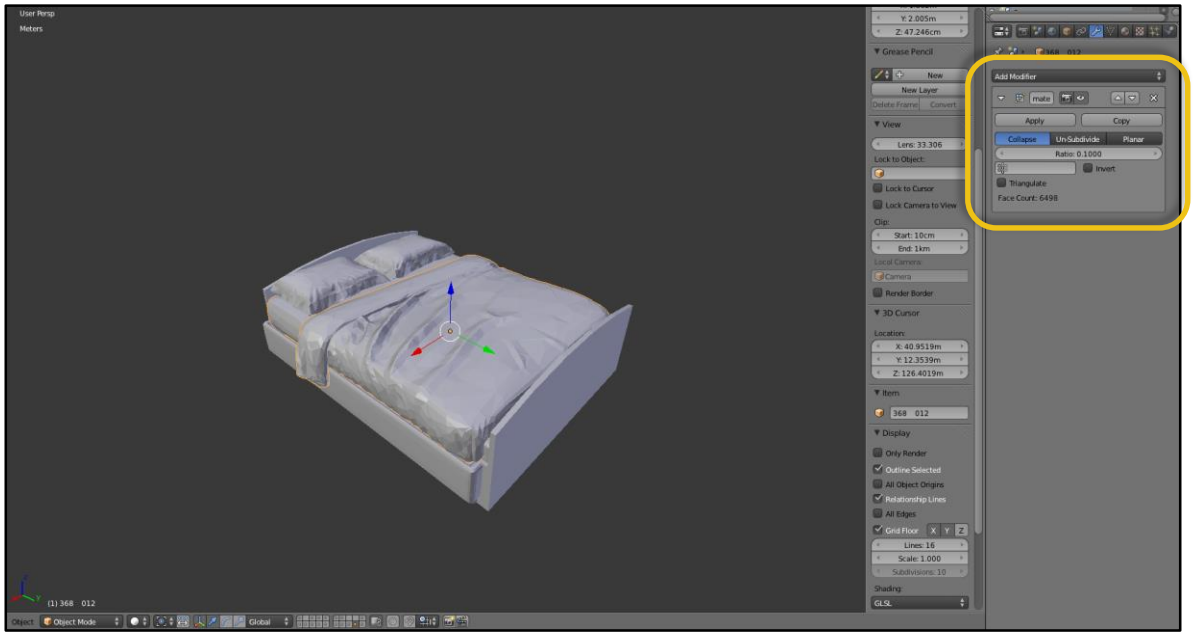


KUVA 44. Decimate-käsky

Alla olevissa kuvissa (kuvat 45 ja 46) vertailtu *decimate*-käskyn vaikutusta sängynpeittoon. Ylemmässä kuvassa *decimate*-käskyn *ratio*-arvo=1.0 jolloin monikulmioita on 64 999 ja alemmassa kuvassa *ratio*-arvo=0.1 jolloin monikulmioita 6498. *Ratio*-arvon alentaminen siis vähentää monikulmioiden määrää ja kasvattaa niiden kokoa, jolloin meshistä tulee kevyempi. Vaikutuksen voi huomata katsomalla sängynpeiton pinnanrakennetta kuvista.



KUVA 45. Monikulmiota 64 999

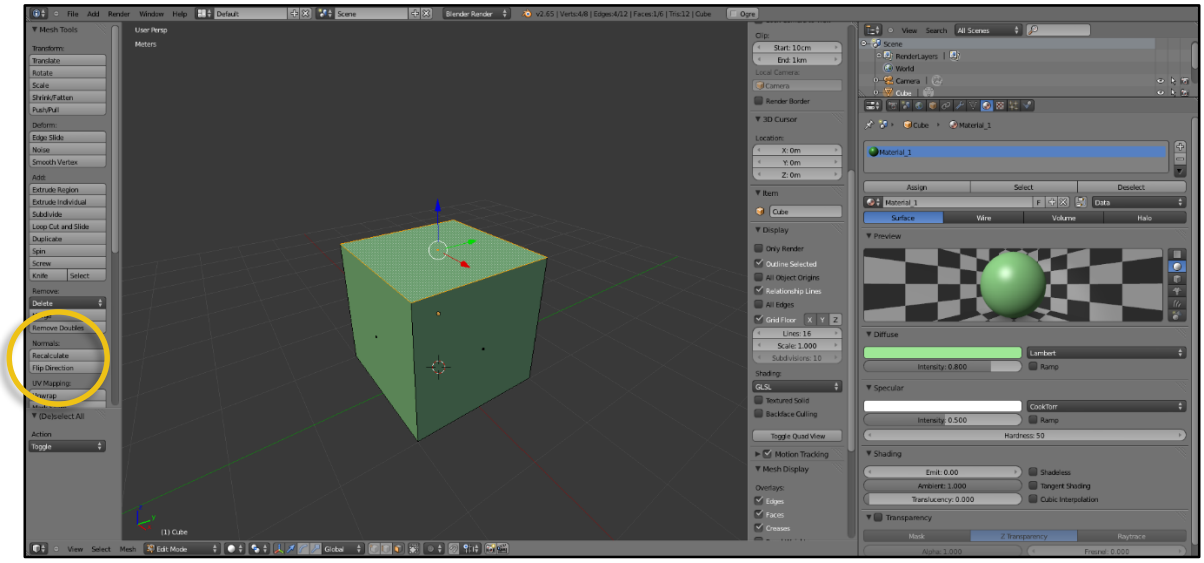


KUVA 46. Monikulmioita 6498

Flip-käsky

Flip-käskyn tarkoitus on kääntää teksturoitu pinta 180°. Tämä luku kertoo miten *flip*-käsky Blenderissä toimii. Tämän käskyn löytäminen Blenderistä oli hyvin kriittinen asia, koska joissain tilanteissa meshien pinnat (*facet*) tahtoo kääntyä eli peilautua itsestään, kun meshejä viedään Rocket clientiin. Itse virhettä ei voi havaita Blenderissä vaan vasta, kun meshi avataan Rocket clientissä. Peilautumisella tarkoitetaan sitä, että pinta jolle tekstuuri on asetettu peilautuu pinnan vastakkaiselle puolelle eikä tekstuuri siksi näy Rocket clientissä. Tekstuurin peilaantuminen johtuu *OGRE enginestä*, joka on tiedostettu ongelma.

Flip-käskyn käyttäminen onnistui seuraavasti, ensimmäiseksi valittiin meshi jossa pinta peilautui ja avattiin editointitila *tab*-näppäimestä. Asetettiin alanauhasta *Face select* päälle ja valittiin halutta pinta. Haluttu pinta valittuna klikattiin vasemmalla olevasta valikosta *Flip Direction* -käskyä, jolloin pinta kääntyi 180°. (kuva 47.)



KUVA 47. Flip-käsky

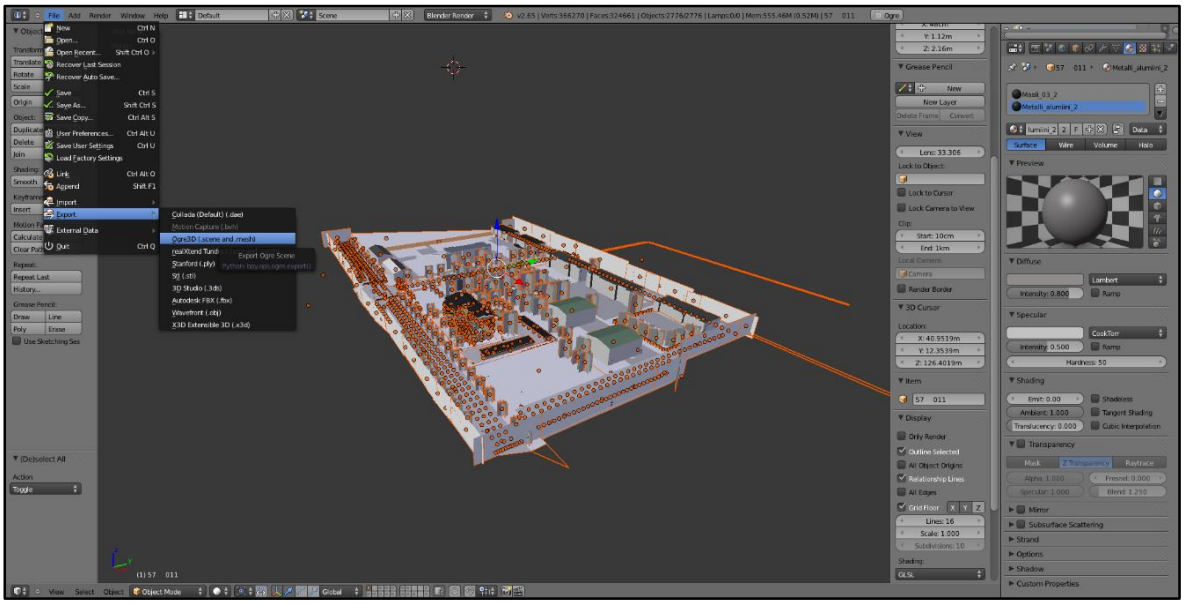
Blender-mallin nollakorkoon vienti

ArchiCAD-mallia työstettiin oikeassa korkomaailmassa ja *3ds*-muotoon tallennettaessa ei haluttu lähteä korkoja muuttamaan, jolloin malli piti laskea Blenderissä nollakorkoon. Mallin laskeminen nollakorkoon Blenderissä on suositeltavaa, koska Rocket clientissä tehdään virtuaalimalliin taivas- ja pilvikomponentti. Taivas- ja pilvikomponentin teossa tulee ongelma jos malli on esimerkiksi korossa +134.000, tällöin Rocket clientissä tehtävä taivas on virtuaalimallin ”sisällä” eli toisin sanoen malli nousee ”taivaisiin”. Mallia ei tarvitse Blenderissä tasan viedä nollakorkoon vaan sinne päin, hyvänä nollakoron merkkinä Blenderissä on *Grid Floor* -verkko, joka on nollakorossa. Koko malli valittiin ensin A-näppäimellä ja sen jälkeen painettiin G-näppäintä, joka mahdollisti mallin liikuttelun joko x-, y-, tai z- suunnissa. Mallia raahattiin z-suunnassa alaspäin sen verran, että mallin maasto oli lähellä *Grid Floor* -verkkoa, raahaus onnistui painamalla hiiren rulla pohjaan ja raahaamalla mallia haluttuun suuntaan.

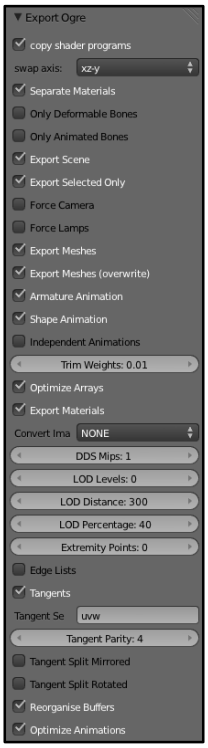
4.2.4 Mallin exportaus Rocket clientiin

Nollakorkoon viennin jälkeen malli voitiin exportata Rocket clientiin, tässä luvussa tarkastellaan exportaus- vaiheet Rocket clientiin.

Ensimmäiseksi valittiin koko malli A-näppäimellä. Sen jälkeen *File* välilehdestä valittiin *Export* ja valikosta valittiin *OGRE3D (.scene and .mesh)*. (kuva 48.) Avautui valikko jossa ohjelma pyysi tallentamaan tiedoston. Tiedostelle luotiin uusi kansio johon malli exportatiin. Itse exportaus voi kestää mallin koosta sekä käytettävän koneen tehoista parista sekunnista tuntiin. Tallennus vaiheessa kysyttiin mallin exportaus-asetuksia, kuvassa 49 toimivat asetukset.

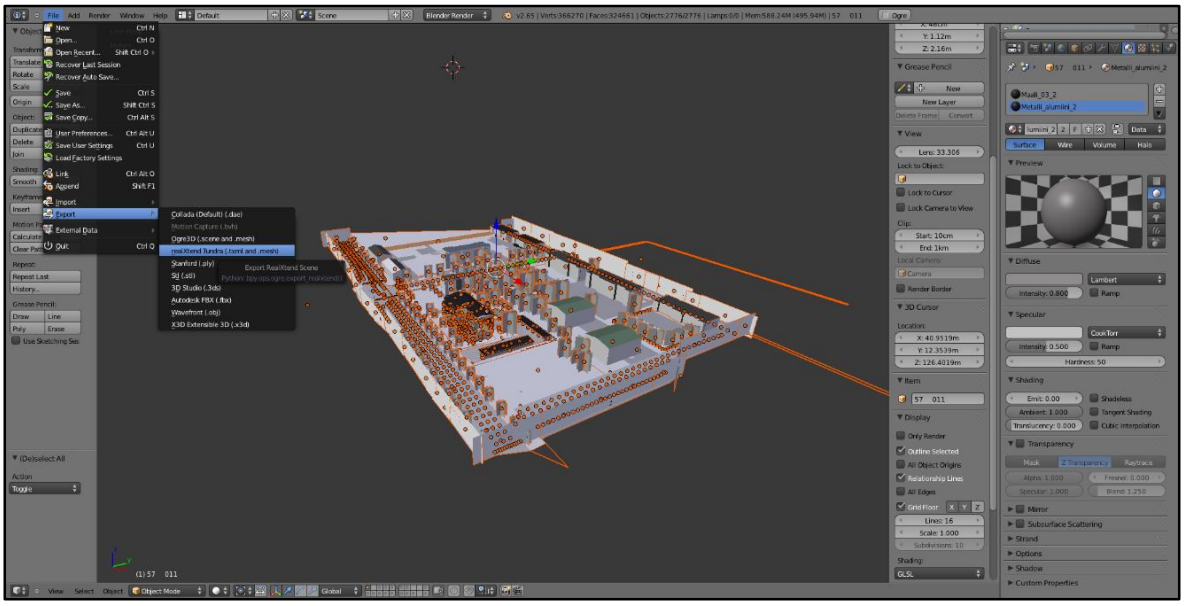


KUVA 48. OGRE3D-export

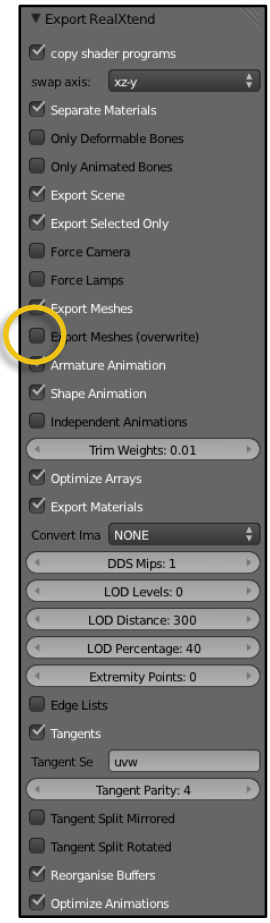


KUVA 49. Export-asetukset

Seuraavaksi exportatiin koko malli ulos samaan kansioon kuin edellinen tehtiin, mutta tällä kertaa exportatiin malli ulos *RealXtend Tundra (.txml and .mesh)* -muodossa. (kuva 50.) Tässä exportissa luotiin varsinainen virtuaalimalli-tiedosto (*txml*) Rocket clientiin, joka käyttää kansioon aikaisemmin tehtyjä muita tiedostoja hyväksi. Exportaus-asetukset kuvassa 51. Asetukset olivat muuten samat kuin edellisessä exportauksessa paitsi *Export Meshes (overite)* -ruksi poissa. Tämä exportaus ei kestä kuin enintään yhden minuutin.

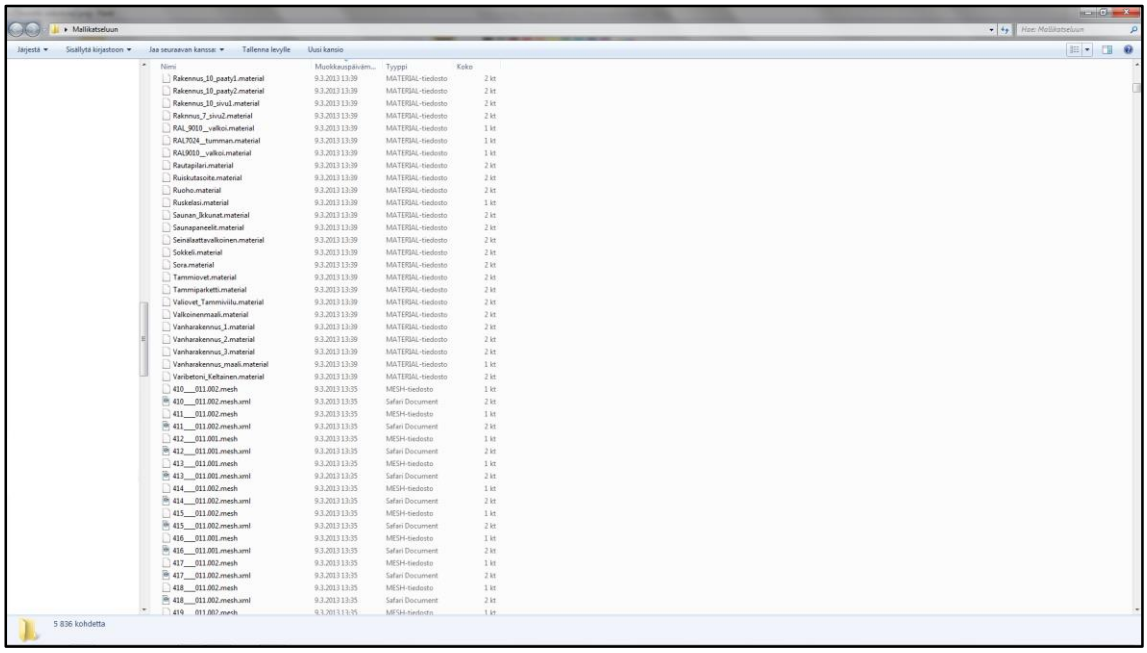


KUVA 50. RealXtend Tundra -export

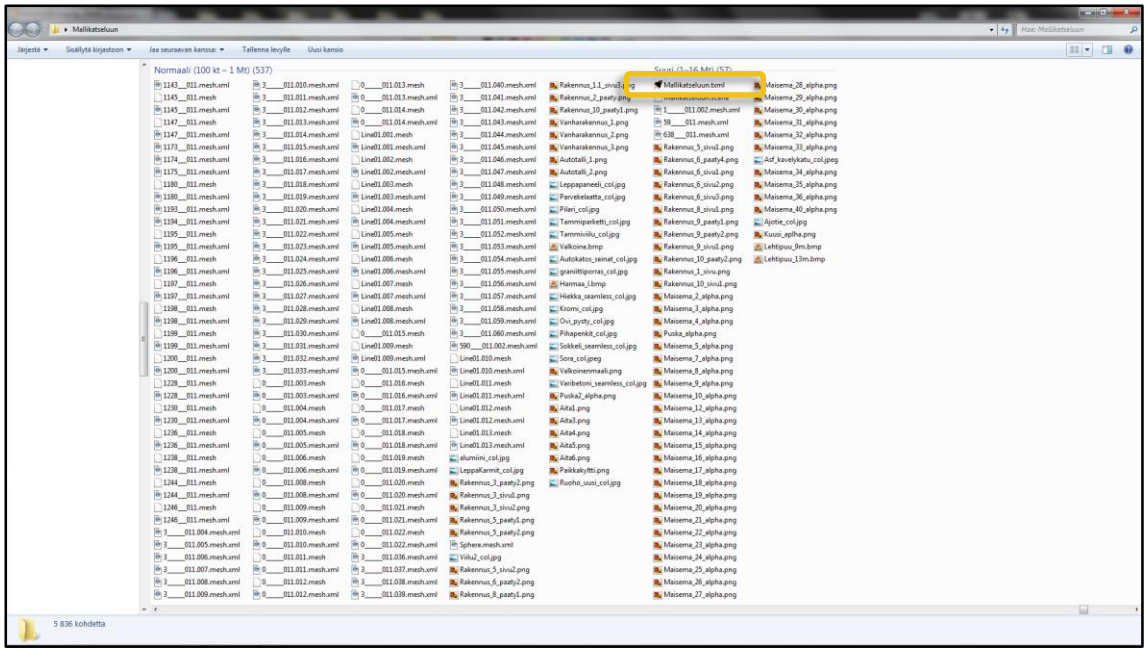


KUVA 51. Export-asetukset

Kun malli oli saatu exportatua, kansionrakenne näytti tältä. (kuvat 52 ja 53.) Kansioon muodostui kaikkineen 5836 eri tiedostoa.



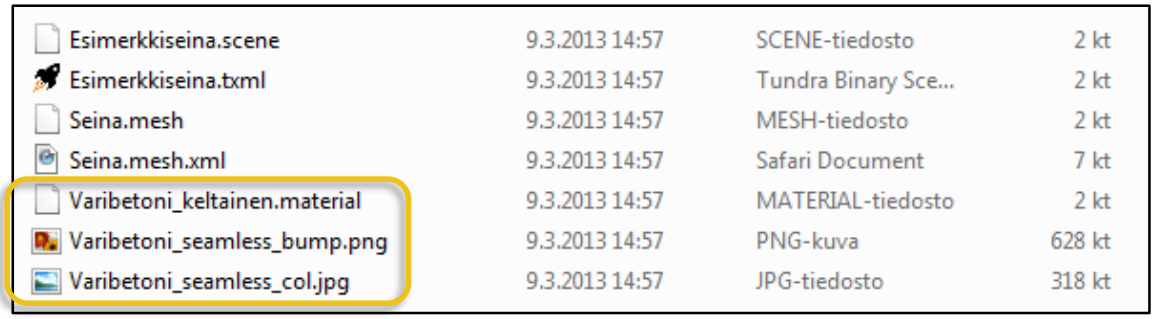
KUVA 52. Kansionrakenne



KUVA 53. Kansionrakenne

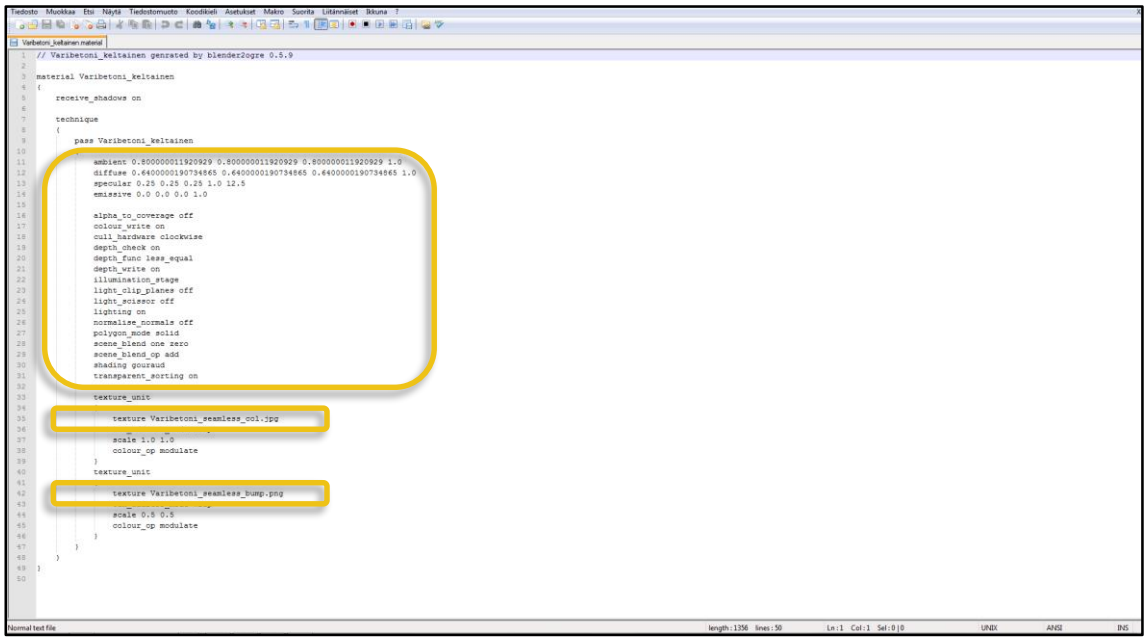
Kansioon muodostui ensimmäisen exportauksen myötä *material*-tiedostoja, *mesh*-tiedostoja, kuva-tiedostoja joita teksturointi vaiheessa käytettiin, meshin *xml*-tiedostoja ja mallin *scene*-tiedosto. Toisella exportauksella luotiin *txml*-tiedosto, joka on varsinainen virtuaalimallin-tiedosto. Tämä tiedosto saadaan tuplaklikkauksella lokaalisti katseluun omalle koneelle. (kuva 53.) Nyt oli saatu malli exportatuksi ulos Blenderistä Rocket clientiä tukevaan muotoon. Seuraavassa luvussa tarkastellaan mallin avaamista Rocket clientissä ja siellä tehtäviä asetuksia, sekä virtuaalimallin julkaisuun liittyviä verkko-asetuksia. Sitä ennen tarkastellaan exportaus vaiheessa luotua kansiorakennetta.

Esimerkkinä alussa luotu seinä, seinään käytettiin kuvia nimeltä *Varibetoni_seamless_col.jpg* ja *Varibetonini_seamless_bump.png*, kyseiset kuvat näkyvät kansiorakenteessa. Materiaali nimeksi annettiin *Varibetoni_keltainen*, joka näkyy *Varibetoni_keltainen.material* -tiedostona kansiossa. *Material*-tiedostot ovat tiedostoja joihin on koodattu mesheissä käytetyt materiaali- ja tekstuuriasetukset. Näiden lisäksi kansiossa on seinä meshin *Seina.mesh* -tiedosto ja meshin *xml*-tiedosto, *mesh*-tiedostot kuvaavat Blenderissä luotuja meshejä. Näiden tiedostojen lisäksi kansiossa on varsinainen virtuaalimallin-tiedosto joka avataan Rocket clientissä, *Esimerkkiseina.txml* ja sen *scene*-tiedosto. (kuva 54.) Esimerkkiseinä exportatiin ulos samalla tavalla kuin edellinen vaihe kertoo.



KUVA 54. Esimerkkiseinän kansiorakenne

Material-tiedostot voi avata Notepad ++:lla, jolloin tiedostorakenne avautuu. Alla olevassa kuvassa on avattu *Varibetoni_keltainen.material* -tiedosto Notepad ++:lla. (kuva 55.)

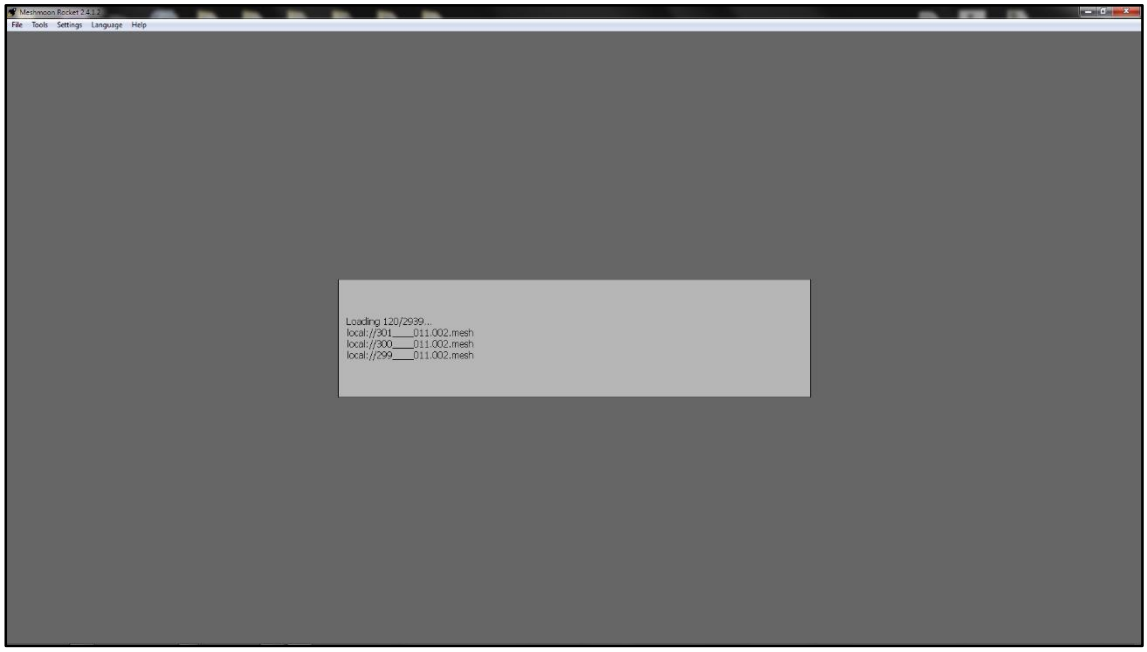


KUVA 55. Varibetoni_keltainen.material -tiedostorakenne

Kuvasta näkee sen, että meshin *material*-tiedosto pitää sisällään meshiin asetetut tekstuuriasetukset ja kertoo mitä kuvaa meshissä on käytetty, sekä listaa materiaaliasetukset mitkä on Blenderissä säädetty. Periaatteessa meshejä voi koodata jälkeenpäin avaamalla *material*-tiedoston ja kirjaamalla haluttaja arvoja, mutta tässä vaiheessa pitää tietää mitä tekee.

4.3 Kolmas vaihe – Rocket client ja Meshmoon

Exportausten jälkeen voitiin kokeilla toimiiko malli niin kuin pitää. Kansiosta avattiin äsken luotu *txml*-tiedosto ja malli avautui lokaalisti Rocket clientiin. (kuva 56.)



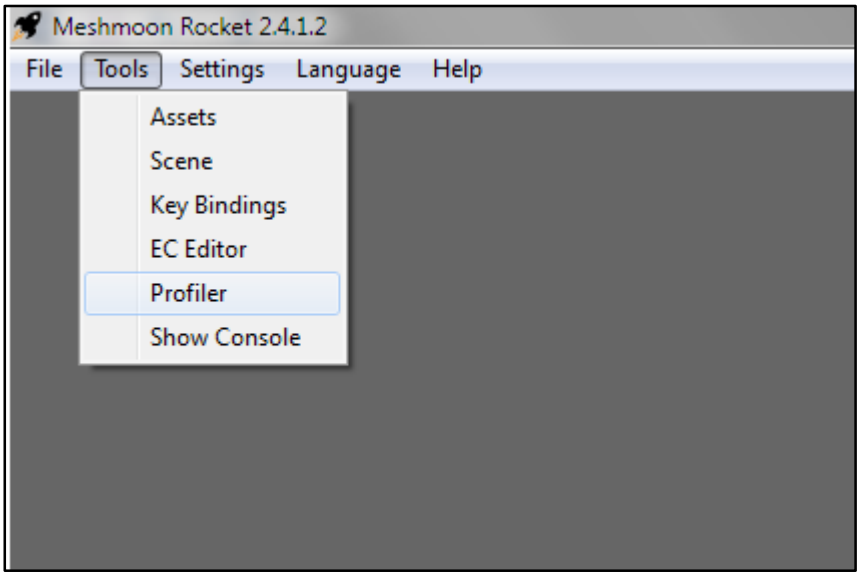
KUVA 56. Malli latautuu Rocket clientiin

Kaikki tekstuurit, meshit ja objektit latautuneena malli avautui seuraavan näköisenä. (kuva 57.)

KUVA 57. Malli latautuneen Rocket clientiin

Lokaali-mallissa tehtiin kaikki tarkastelut mm. löytyykö tekstuuri- tai materiaalivirheitä, jos löytyi tehtiin korjaukset malliin Blenderissä ja exportatiin malli uudelleen ja avattiin uudestaan lokaalisti tarkasteluun. Vasta kun malli oli saatu lokaalisti korjattua, voitiin se viedä Meshmooniin. Korjausten lisäksi lokaali-mallista tarkastettiin paljonko tekstuurit ja meshit vie muistia.

Muistin kulutuksen näki *Tools*-valikosta kohdasta *Profiler*. (kuva 58.) Profilerin *ogre* välilehdestä näki paljonko tekstuurit ja meshit vei muistia kohdasta *TextureManager* ja *MeshManager*.



KUVA 58. Profiler-valikko

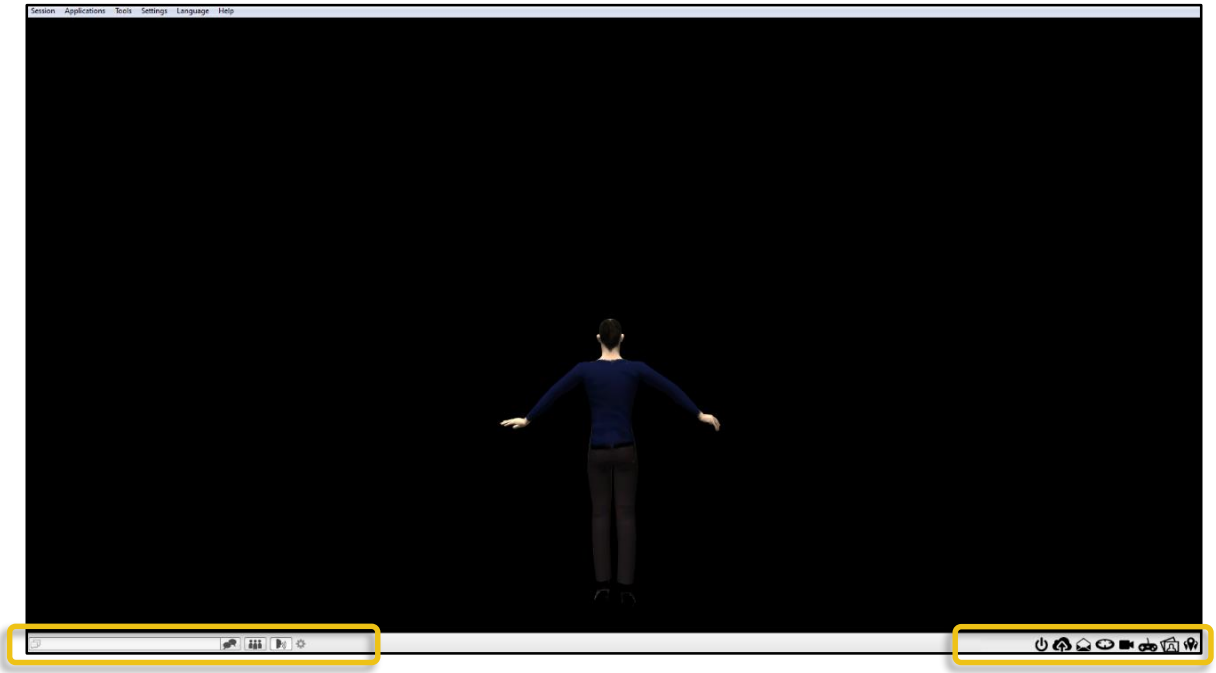
Tekstuurien MB -arvo saisi enintään olla 200 MB ja meshien arvo enintään 100 MB. Tekstuurien arvoon vaikuttaa suoraan käytettyjen kuvien koko (max. 2048x2048) sekä kuvienmuoto (jpg, png, bmp). Meshien MB -arvoon vaikuttaa se miten raskaita meshejä Blenderissä on käytetty esimerkiksi yksi *low poly* -puu voi viedä jo 25 MB. Mitä pienemmät arvot *TextureManagerissa* ja *MeshManagerissa* on sitä kevyempi malli. Kun nämä vaiheet oli tarkastettu läpi, voitiin aloittaa mallin siirtäminen Meshmooniin.

Meshmooniin siirto tapahtui seuraavassa järjestyksessä: Tehtiin Meshmoon.com:iin käyttäjätunnukset, joilla kirjaudutaan Rocket clientiin ja ladattiin clienti omalle koneelle (clientin latausvaihe tapahtunut jo aikaisemmassa vaiheessa). Kun käyttäjätunnukset oli tehty, voitiin mennä Mesmoon.com:in kautta tekemään *space* eli tila virtuaaliympäristölle. *Spacen* teko edellytti oman tilin avaamista. Tilin avattua käyttöön sai ilmaiseksi 100 MB tallennustilaa. Virtuaalimallin *spacea* tehtäessä voitiin asettaa erilaisia asetuksia *sceneen* muun muassa virtuaalimallin nimi, jolla mallia voitiin hake Rocket clientissä (*Meriluodon_Helmi_Ymparisto*), onko virtuaalimalli julkinen vai salainen, minkälaisia applikaatioita virtuaalimalliin luodaan (avatarit, kamera-ajot, 3D-tekstit, Facebook jako, VoIP, tekstimuotoinen keskustelu mahdollisuus).

Kun *space* oli saatu tehtyä, avattiin Rocket client työpöydältä. Avautui seuraavanlainen näkymä (kuva 59), jossa näkyi aikaisemmin Meshmoon.com:ssa tehty *space*, *Meriluodon_Helmi_Ympäristö*.

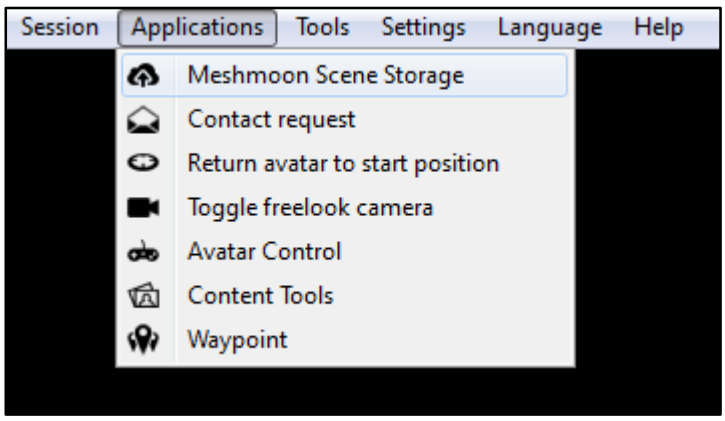
KUVA 59. Meshmoon.com:ssa tehty space

Tämän jälkeen klikattiin *spacea* listasta ja päästiin sisään *sceneen*, näkymä näytti tältä. (kuva 60.) Sisäänkirjautumisen jälkeen *scenessä* näkyi vain aikaisemmin Meshmoon.com:ssa luodut applikaatiot, tämän jälkeen piti vielä saada varsinainen malli *sceneen*.



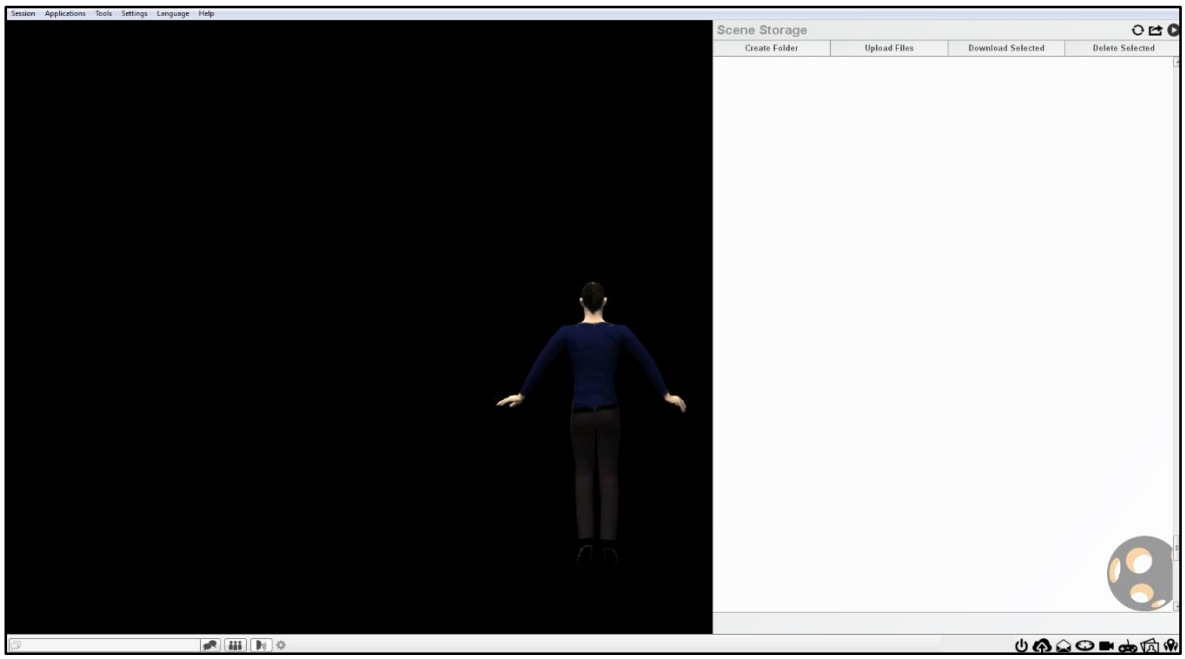
KUVA 60. Ensimmäinen sisäänkirjautuminen sceneen

Mallin siirtäminen *sceneen* onnistui seuraavasti, valittiin *Applications* välilehdestä *Meshmoon Scene Storage*, jolla hallinnoidaan varsinaista *scenen* sisältöä. (kuva 61.)



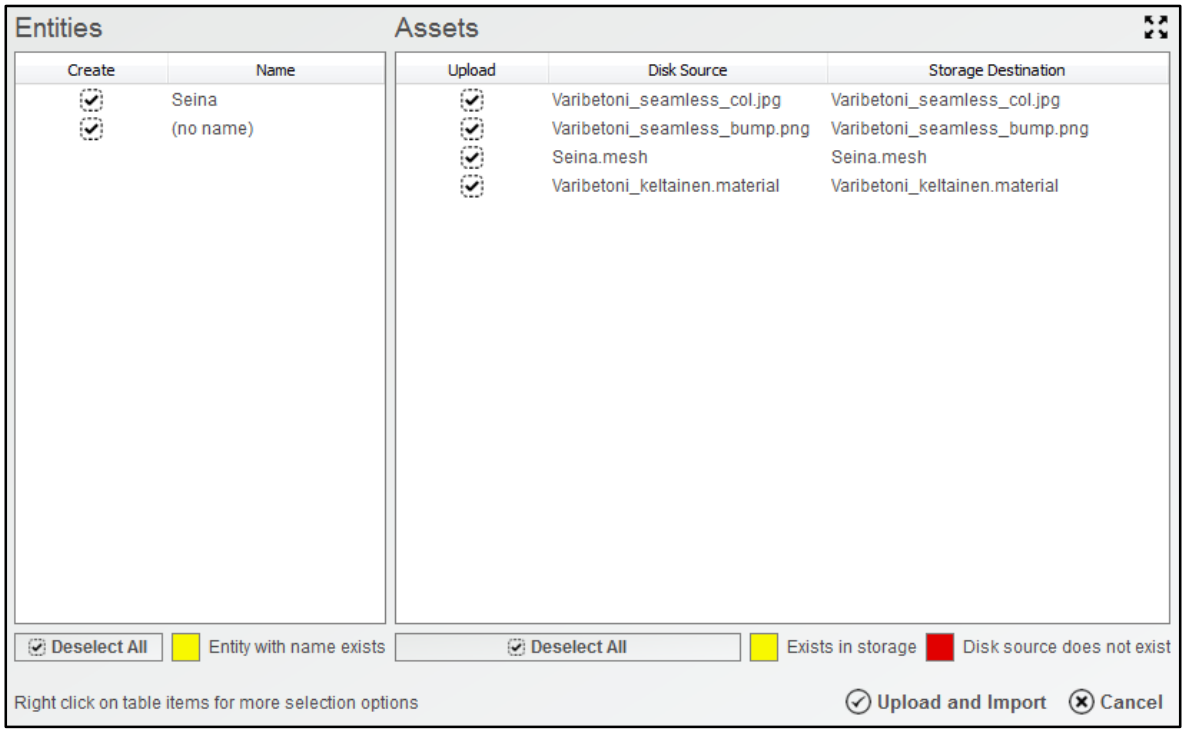
KUVA 61. Meshmoon scene storage

Tämän jälkeen ohjelma kysyi tunnuksia joilla vain *scenen*-admini pääsee muokkaamaan *scenen* sisältöä, tunnukset sai *space*-tilin avauksen yhteydessä Admino Technologiesilta. Tunnusten laiton jälkeen avautui *Scene Storage* -valikko ohjelmaan. (kuva 62.)

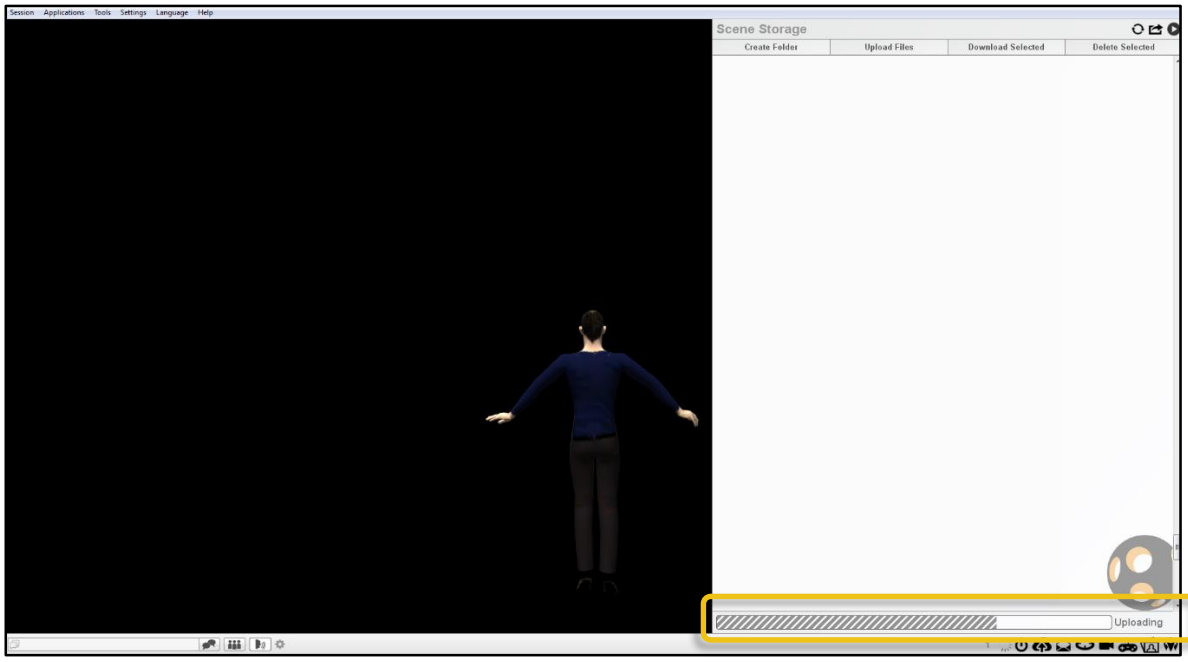


KUVA 62. Scene storage

Tässä vaiheessa *Scene Storage* -valikossa ei näkynyt vielä mitään, koska *sceneen* ei ollut vielä tuotu mitään. Seuraavaksi pitikin *sceneen* tuoda aikaisemmin tehty *txml*-tiedosto, joka oli siis varsinainen virtuaalimallin-tiedosto. Tiedoston tuonti tapahtui *drag and drop* -tyylillä eli aikaisemmin luotu ja lokaalisti tarkasteltu *txml*-tiedosto raahattiin *Scene Storage* -valikkoon ja ohjelmaan avautui seuraavanlainen näkymä. (kuva 63.) Kuvassa esimerkin vuoksi esimerkkiseinän tuonti sceneen, oikean mallin *txml*-tiedoston *assets*-lista huomattavasti pidempi. Ohjelma kysyi tässä vaiheessa mitä kaikkea haluat *sceneen* tuoda, kun oli katsottu että kaikki oli kunnossa painettiin *Upload and Import* jolloin malli alkoi latautumaan *sceneen*. (kuva 64.)



KUVA 63. Esimerkkiseinän tuonti sceneen



KUVA 64. Upload-vaihe

Mallin latautuminen *sceneen* kesti pari minuuttia, kun malli oli latautunut ilmestyi *Scene Storageen* tiedostot. (kuva 65.)

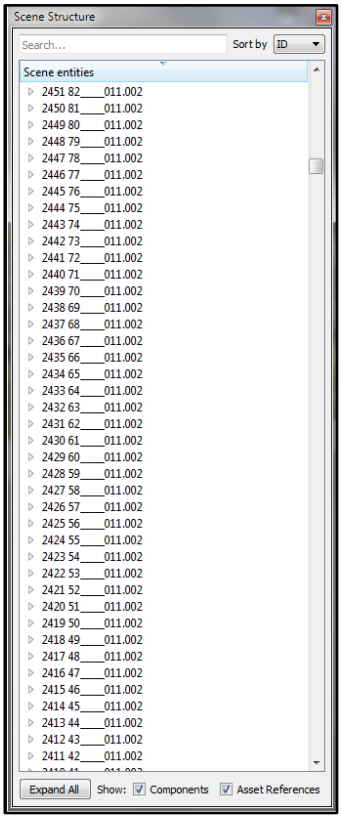
KUVA 65. Tiedostot latautuneena sceneen

Tiedostojen latautumisen jälkeen *sceneen* ilmestyi malli. (kuva 66.) Tässä kuvassa näkyvä taivas ja pilvet on lisätty jälkikäteen ja myös mesheistä on tehty ”kiinteitä”, jotta avatar ei tipu niistä läpi. Seuraava luku selvittää Rocket clientissä tehtäviä virtuaalimallin viimeistelyasetuksia.

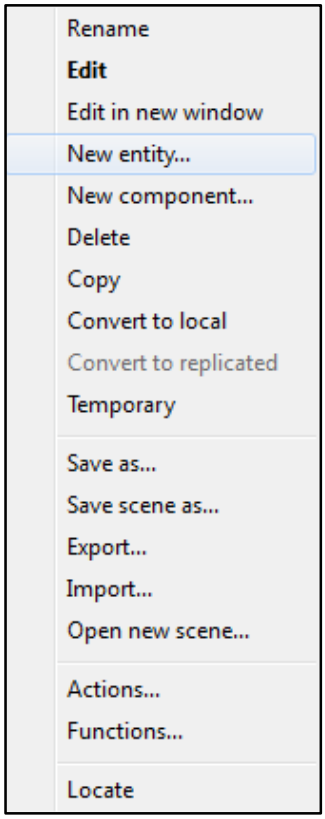
KUVA 66. Malli latautuneena sceneen

SkyX

Taivas- ja pilvikomponentin lisäys onnistui seuraavasti, avattiin *Tools* välilehdestä kohta *Scene*, avautui ikkuna nimeltä *Scene Structure*, joka listaa kaikki *scenessä* olevat asiat. (kuva 67.)

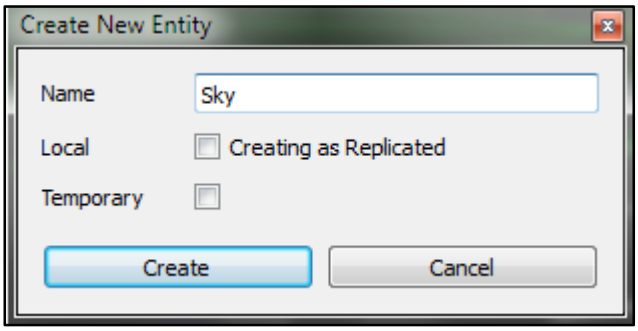


KUVA 67. Scenen rakenne



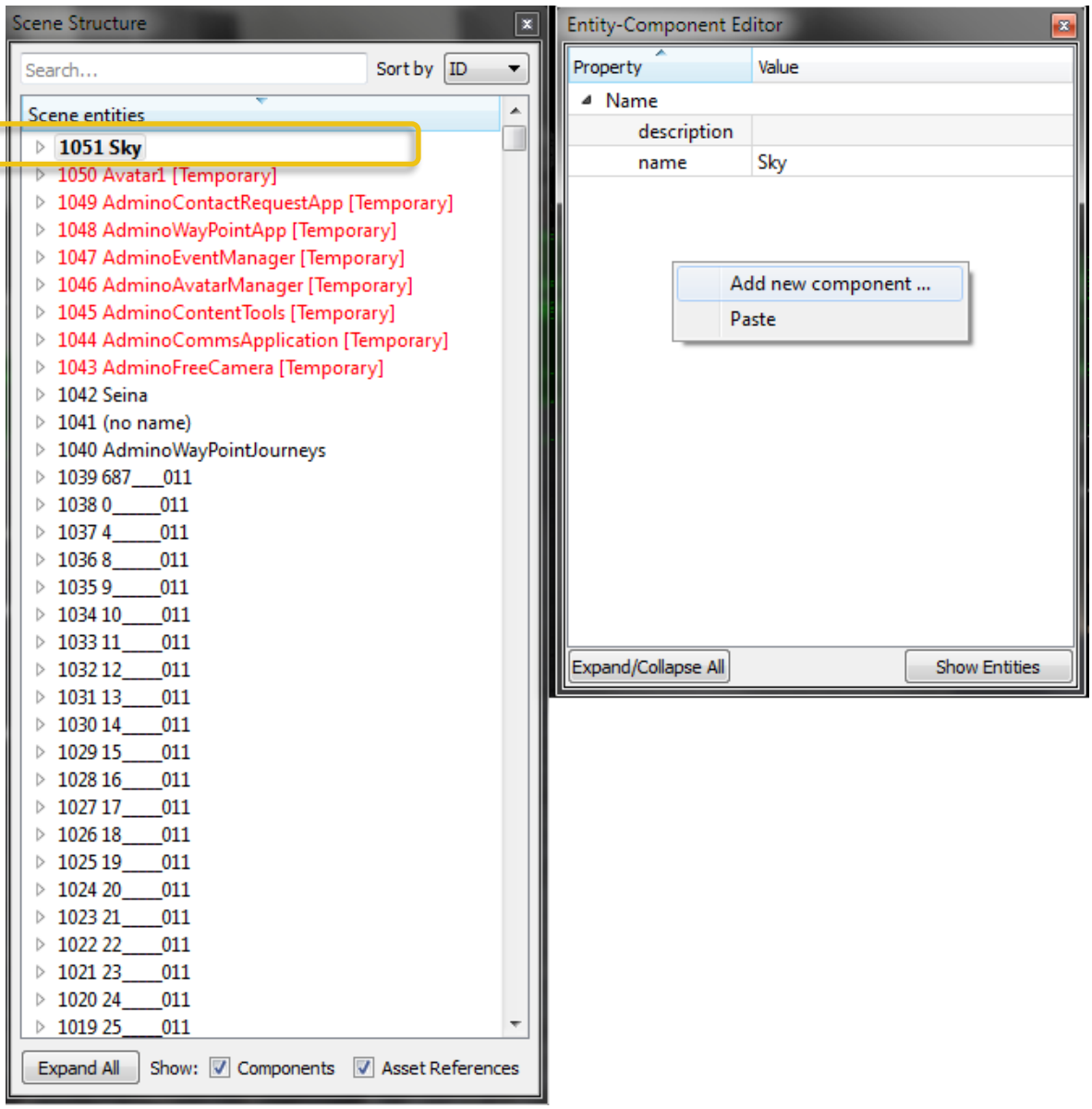
KUVA 68. New entity

Painettiin hiiren oikealla painikkeella valkoisen alueen päällä ja valitaan *New Entity*. (kuva 68.) Aukesi seuraavanlainen näkymä (kuva 69), annettiin haluttu nimi ja painetaan *Create*.



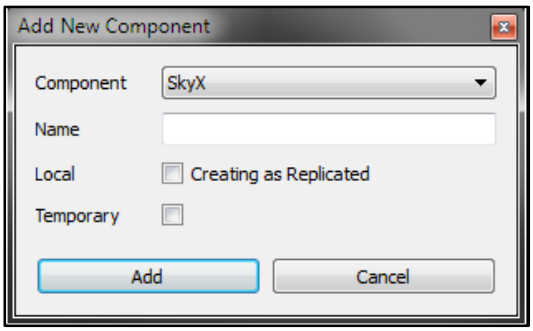
KUVA 69. Taivas- ja pilvikomponentin lisäys

Scene Structure -listaan ilmestyi äsken tehty *Sky* (kuva 70), sitä tuplaklikkaamalla päästiin *Entity Component Editoriin* johon lisättiin uusi komponentti, hiiren oikealla painikkeella painettiin valkoisen alueen päällä ja valittiin *Add new component*. (kuva 70.)



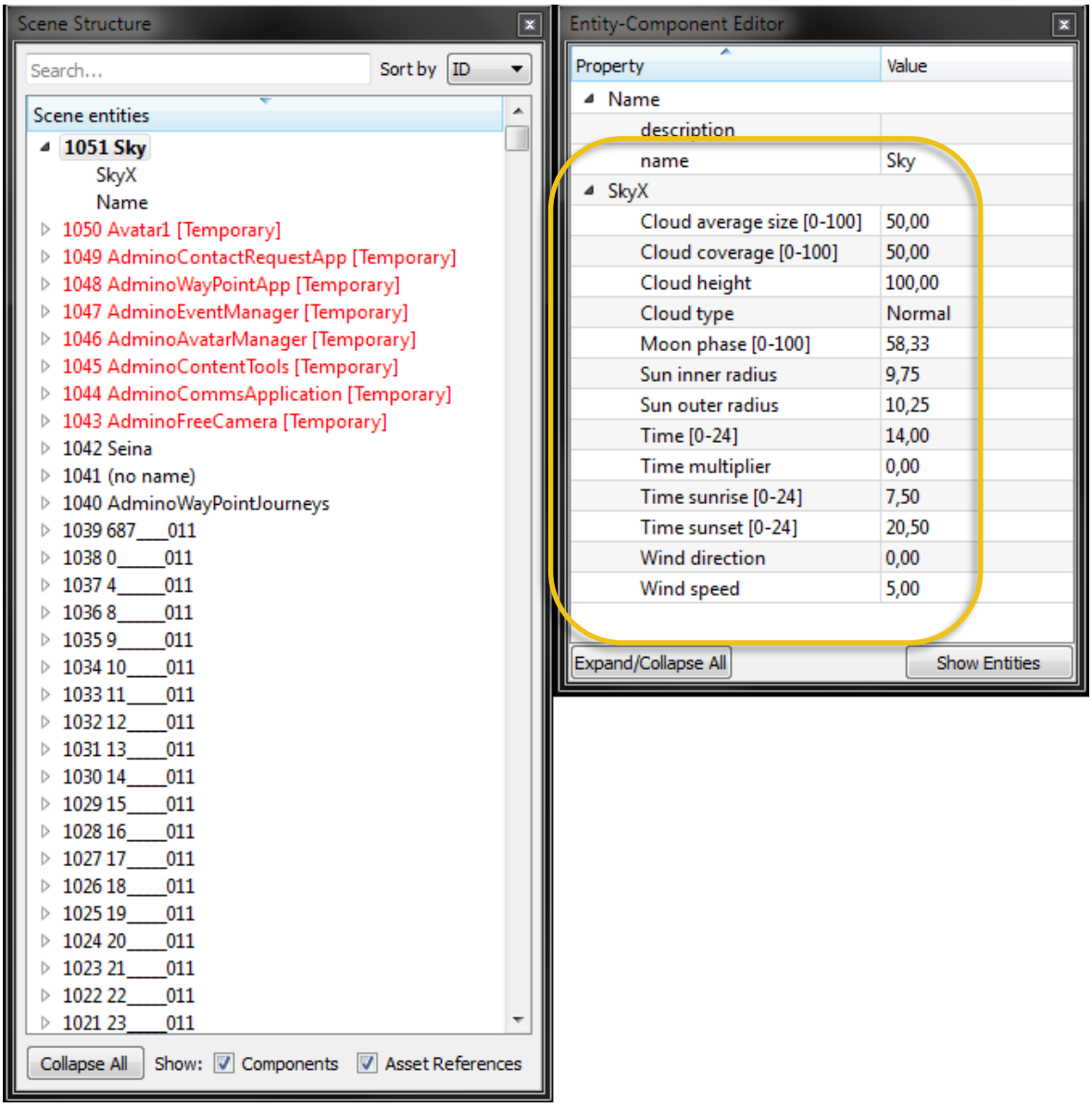
KUVA 70. Scene structure ja entity component editor

Avautui *Add new component* -ikkuna (kuva 71), jonka alasvetovalikosta valittiin *SkyX* ja lopuksi painettiin *Add*.



KUVA 71. Add new component -ikkuna

Taivas ja pilvet ilmestyi *sceneen* ja *Entity Component Editoriin* tuli erilaisia säätömahdollisuuksia taivaalle ja pilville muun muassa kellonajan määritys, vuorokaudenajan vaihtelunopeuden määritys, pilvien tyypin määritys sekä tuulen nopeuden määritys. (kuva 72.)

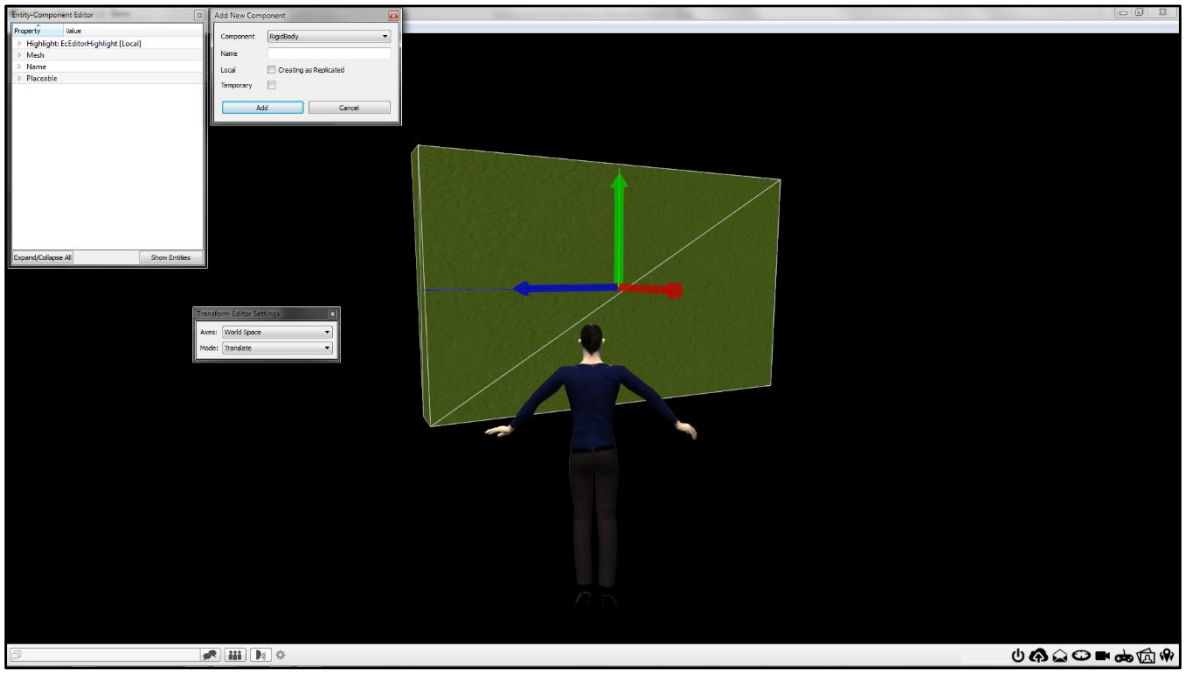


KUVA 72. SkyX:n säätömahdollisuudet

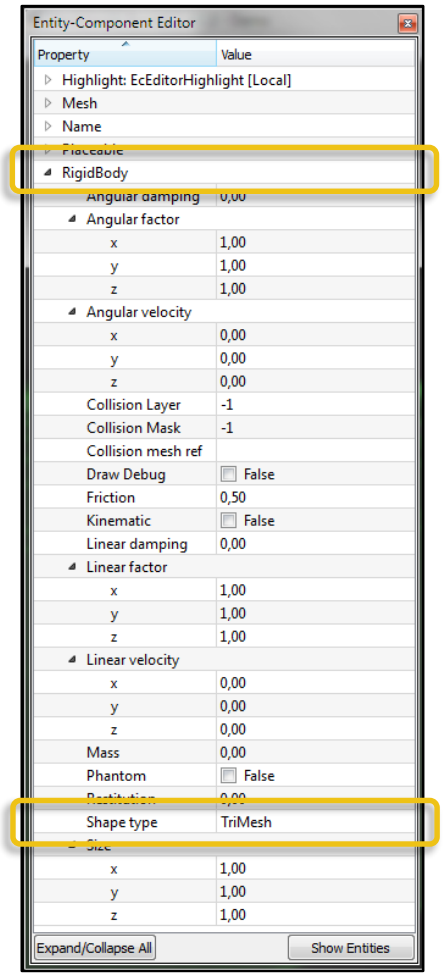
Seuraava Rocket clientissä tehtävä muokkaus oli tehdä tietyistä mesheistä kiinteitä (kadut, tiet, rakennuksen osat), tämä tehtiin sen takia että avatar ei tipu niistä läpi.

RigidBody

Tekeminen aloitettiin painamalla *scenessä Shift+E*, jolloin siirryttiin suoraan *Entity Component Editoriin*. Valittiin *scenestä* kiinteiksi halutut meshit, *CTRL*+hiiren oikea painike antoi valita monta kerrallaan. Kun kaikki kiinteiksi halutut meshit oli valittu, *Entity Component Editorin* valkoisen alueen päällä painettiin hiiren vasemmalla painikkeella *Add new component* ja alasvetovalikosta valittiin *RigidBody*, joka tekee meshistä kiinteän, lopuksi painettiin *Add*. (kuva 73.) Kun *RigidBody* oli lisätty, piti vielä muuttaa meshin *RigidBody*-asetuksista *Shape Typeksi: TriMesh*. (kuva 74.)



KUVA 73. Ridigbodyn asetus meshiin



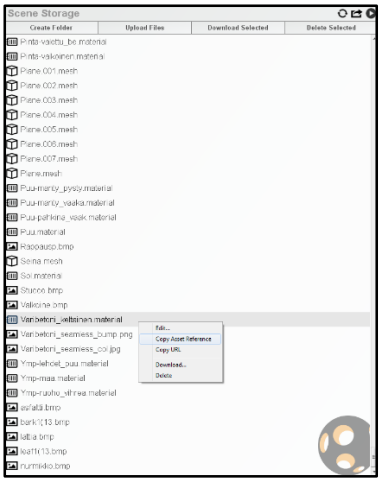
KUVA 74. Shape type

Meshien tuonti sceneen jälkikäteen

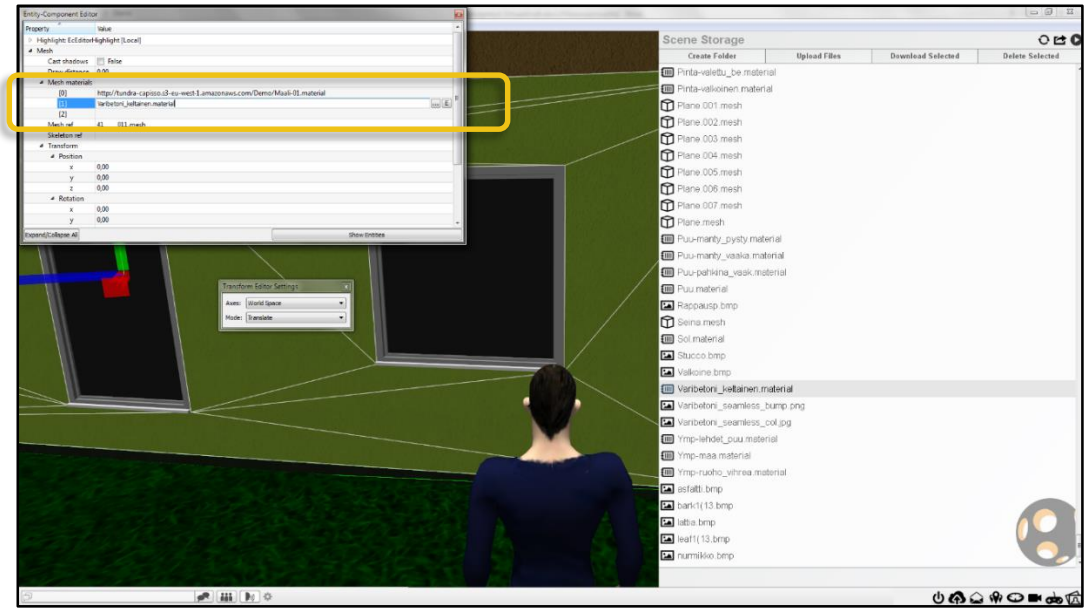
Jos mallista vielä puuttui meshejä, voitiin siihen jälkeen päin niitä lisätä. Luotiin Blenderissä halutut meshit ja exportatiin ne ulos samalla tavalla kuin varsinainen malli exportatiin. *Drag and drop* -tyyliä käyttäen tuotiin *Scene Storageen txml*-tiedosto ja meshit ilmestyivät *sceneen*.

Tekstuurien korvaus

Jos mallissa haluttiin korvata tekstuuri, onnistui se tuomalla *drag and drop* -tyyliä käyttäen *material*-tiedosto ja *material*-tiedostossa oleva kuvatiedosto *scene storageen*. Jotta tekstuurin sai korvattua valittiin *Scene Storagessa* hiiren vasemmalla painikkeella *Copy Asset Reference* sen *material*-tiedoston päällä, joka tuotiin *sceneen*. (kuva 75.) Seuraavaksi painettiin *Shift+E* ja valittiin meshi, jonka tekstuuri haluttiin korvata, valittiin *Mesh materials* -kohta ja *CTRL+V:llä* liitettiin kopioitu *material*-tiedosto lohkoon [0], [1] tai [2], lopuksi *Enter*. Nyt oli uusi korvaava tekstuuri asetettu meshin pintaan. (kuva 76.)



KUVA 75. Copy asset reference



KUVA 76. Meshin materials-valikko

Nyt oli saatu kaikki tärkeimmät asetukset tehty Rocket clientissä ja malli oli siinä pisteessä, että se voitiin julkaista yleiseen jakeluun. Alla kuvia valmiista virtuaalimallista. (kuvat 77–88.)

KUVA 77. Näkymä mallinnetusta alueesta, alueen koko 300 x 180 metriä

KUVA 78. Näkymä Keskuskadun varrelta

KUVA 79. Näkymä Huvilakadun- ja Savontien risteyksestä

KUVA 80. Näkymä Savontien- ja Huvilakadun risteyksestä

KUVA 81. Näkymä Keskuskadun- ja Huvilakadun risteyksestä

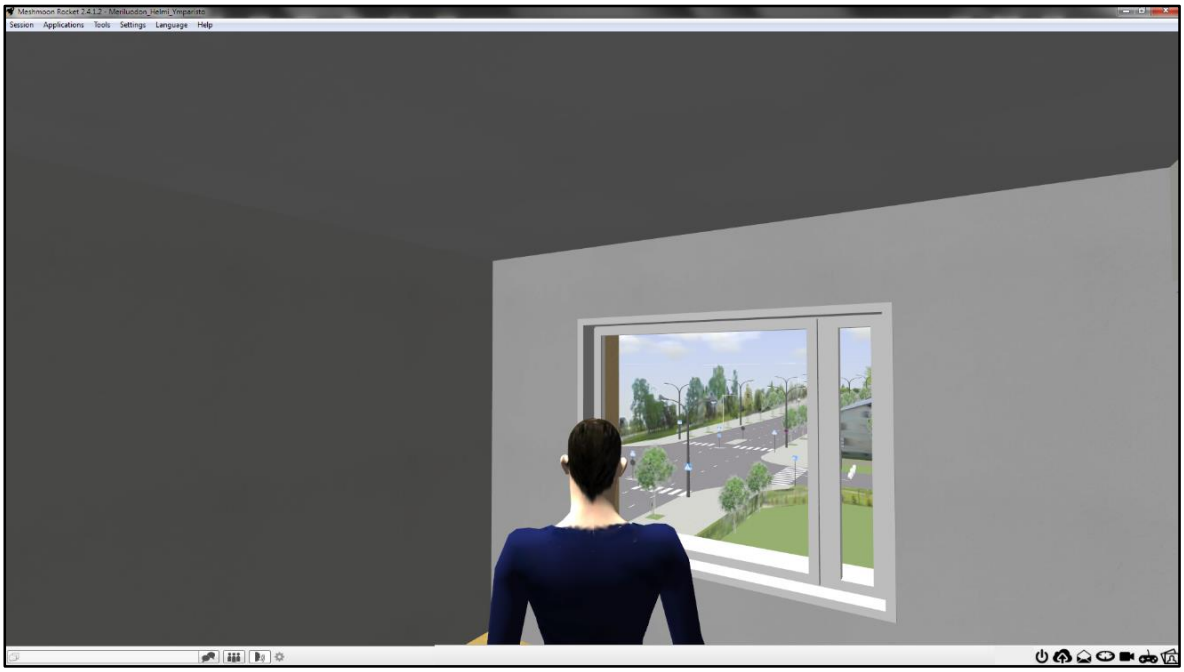
KUVA 82. Näkymä As Oy Meriluodon helmen sisäpihalta

KUVA 83. Näkymä As Oy Meriluodon helmen porrashuoneesta

KUVA 84. Näkymä 2. kerroksen parvekkeelta Keskuskadulle

KUVA 85. Näkymä 4. kerroksen parvekkeelta Keskuskadulle

KUVA 86. Näkymä 2. kerroksen parvekkeelta sisäpihalle

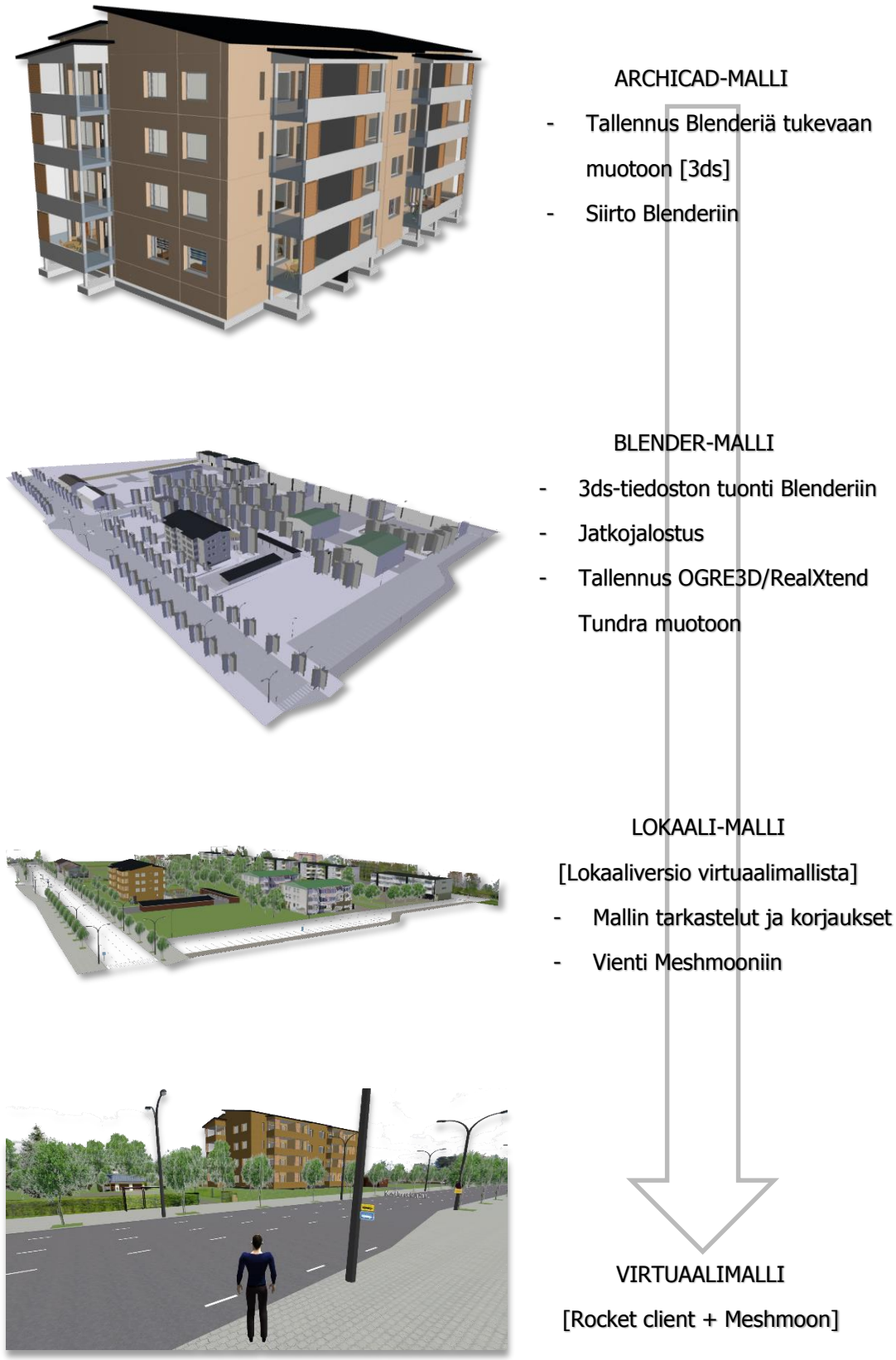


KUVA 87. Näkymä 4. kerroksen asunnosta Keskuskadun- ja Huvilakadun risteykseen

KUVA 88. Näkymä 3. kerroksen asunnosta sisäpihalle

5 TULOKSET

Tuloksena saatiin ympäristöön keskittyvä virtuaalimalli. Alla olevassa kuviossa (kuvio 5) on kuvattu prosessikaaviomuodossa Meriluodon helmen ympäristön -virtuaalimallin toteutuksen päävaiheet.



KUVIO 5. Virtuaalimallin prosessikaavio

6 JOHTOPÄÄTÖKSET JA POHDINTA

Virtuaalimallin toteuttaminen oli varsin kiinnostavaa ja haastavaa työtä. **Asetettuihin tavoitteisiin** opinnäytetyö vastaa omasta mielestäni hienosti, koska työsuunnitelmaa tehdessä ei osannut kuvitellakaan miten paljon työtä ja testailua vaati eri aloilta tuttujen ohjelmistojen yhdistäminen. Nämä lähtökohdat ja käytettyjen ohjelmistojen puutteet mielessä pitäen on saavutettu hyvä tulos.

Virtuaalimallin tekovaiheessa vastaan tulleista ongelmista keskeisimmäksi muodostui mallin raskaus. Piti huomioida, että malli pyörisi myös katselijalla sulavasti, joten siitä ei voitu tehdä liian raskasta. Tämän takia mallin kokoa jouduttiin rajoittamaan. Tässä mallissa aluetta oli mallinnettu 300 x 180 metrin kokoiselta alueelta sekä käytettyjä meshejä jouduttiin karsimaan suuren monikulmiomäärän takia.

Virtuaalimallin lopputulokseen eniten vaikutti kompromissi, että tarjotaanko katselijalle ”silmänkarkkia” eli tehdäänkö virtuaalimallista visualisesti hieno mutta samalla raskas vai pyritäänkö tekemään virtuaalimalli, joka pyörii koneella kuin koneella mutta on vähän rujompi. Loppujen lopuksi päädyttiin viimeiseen vaihtoehtoon. Lopputulokseen vaikutti myös se miten virtuaalimalli saadaan toteutettua mahdollisimman kustannustehokkaasti ja tämä asia karsi tietenkin pois visuaalisuuteen vaikuttavia tekijöitä. Jos mallia olisi alettu rakentaa sen perusteella, että siitä olisi tehty enemmän visualinen malli mitä se nyt on, olisi silloin siirrytty jo hyvin lähelle pelien tekoa, mikä taas tarkoittaa sitä että virtuaalimallista olisi tullut hyvin raskas käyttää.

Virtuaalimallin toteutusta tällä menetelmällä rajoitti se että virtuaalimallin koko pyrkii aina kasvamaan liian suureksi. Rocket clientin suora *bump mapping* -kuvien tuen puutos ja se että Rocket client ei pyöritä suuria monikulmiomeshejä oli myös rajoittava tekijä. Esimerkkinä yksi tarkasti mallinnettu puu voi olla jo liikaa ja se vie mesheille tarkoitetun 100 MB kapasiteetin helposti jo yksikseen eikä tämän takia voida tehdä tarpeeksi näyttäviä malleja vaan joudutaan tyytymään kuvilla toteutettuihin ratkaisuihin. Myös suuret kuvatiedostot ja kuvien käyttämä muoto on ongelma, koska ne vie tekstuureille tarkoitetun kapasiteetin helposti yli 200 MB:n.

Jotta virtuaalimalli koetaan tarpeeksi havainnolliseksi, tulee ympäristön olla mallinnettu tarpeeksi laajasti, ympäristöön pitää luoda yksityiskohtia (liikennemerkit, katuvalot, tiemerkinnät jne.), ympäristön rakennukset pitää toteuttaa laadukkailla kuvilla sekä kasvillisuutta pitää lisätä laajalle alueelle ympäristöön. Pääpainon keskittyessä virtuaalimallissa ympäristön mallinnukseen, rakennusalan ohjelmistoista tuodut rakennukset pitäisi pyrkiä mallintamaan niin että niissä esiintyy ainakin seuraavat asiat: ulkoseinät, vesikatot, ikkunat, ovet, muut ulkopuolen detaljiikat sekä sisäpuolelta välipohjat ja väliseinät.

Virtuaalimallin potentiaalia voidaan hyödyntää muun muassa osallistuvaan suunnitteluun, jossa mahdolliset asukkaat voivat vaikuttaa jo rakennushankkeen alussa kohteen ratkaisuihin. Lisäksi virtuaalimallin potentiaalia voidaan myös hyödyntää rakennusliikkeiden ja kiinteistönvälitysfirmojen myynnissä olevien rakennusten markkinoinnin tukena. **Virtuaalimalli tarjoaa myös mahdollisuuden** siihen että mallia voidaan käyttää myös ns. CAVE-tilassa (cave automatic virtual environment), tällöin virtuaalimalliin voidaan astua ikään kuin itse sisään näin tulevan kokeminen on entistä tehokkaampaa. Virtuaalimalleilla pystytään myös toteuttamaan ns. 3D-simulointitilanteita joissa voidaan esimerkiksi testata rakennuksen poistumisreittien toimivuutta avatar-hahmojen avulla.

Tässä opinnäytetyössä tarkoituksena oli toteuttaa rakennuksen ympäristöön keskittyvä virtuaalimalli ja samalla tutkia miten virtuaalimalli käytännössä toteutetaan. Seuraavia asioita pidän itse jatkotutkimuksen arvoisina: *virtuaalimalli myynnin tukena* eli ovatko virtuaalimallit pelkkää rakennusalan sisäistä ihannointia vai pystytäänkö niillä vaikuttamaan ostajiin, jos virtuaalimallia käytettäisiin myynnin tukena — asioitahan on turha tehdä, jos ei ole kysyntää. *Rocket clientin kehitys*, eli mitä muuta virtuaalimalliin voitaisiin tuoda kuin mallin kokeminen avatarilla.

LÄHTEET

3D Technologies R&D. 3D Tallinn [viitattu 6.3.2013]
Saatavissa: http://app.3d.tallinn.ee/et/

Driver-Inter. Virtual Europe [viitattu 6.3.2013]
Saatavissa: http://www.d-inter.ru/software/virtual-europe/

Sito. Virtuaalimallit [viitattu 6.3.2013]
Saatavissa: http://vrs3d.sito.fi/Tapiola.html

Vianova.fi. Novapoint Virtual Map [viitattu 6.3.2013]
Saatavissa: http://www.vianova.fi/Toimialat/Maankaeytoen-suunnittelu/Novapoint-Virtual-Map#.UTcXBjtfZ8F

Vianovat.dk. Nyheder [viitattu 6.3.2013]
Saatavissa: http://www.vianova.dk/Nyheder/Norway/Forbilledlig-3D-projektering-af-boligomraade#.UTcmYztfZ8E0

Wikipedia-Zephyris. File:Cube Representative UV Unwrapping.png [viitattu 6.3.2013]
Saatavissa: http://en.wikipedia.org/wiki/File:Cube_Representative_UV_Unwrapping.png

YouTube. Veturi Shopping Centre Kouvula [viitattu 6.3.2013]
Saatavissa: http://youtu.be/zS0oBCzR6Vw