Sami Nykänen

Blender in architectural modeling

Case: Modeling of Kalevalatalo

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Media Engineering

Thesis

4.5.2017

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Sami Nykänen Blender in Architecture modeling – Modeling Kalevalatalo 41 pages + 5 appendices 4.5.2017
Degree	Bachelor of Engineering
Degree Programme	Media Engineering
Instructor	Toni Spännäri

The goal of this Bachelor's thesis was to study the open-sourced 3D-modeling software called Blender, and how it is suited for architectural modeling. Blender will be compared to other well-known CAD-software, and the clearest differences between them will be discussed. Originally Blender was created for more visual and photorealistic rendering than the default CAD-software used in architecture, although Blender does have some architectural add-ons.

This thesis also goes through the basic principles of using Blender. Different approaches for modeling such as "low-poly" and "high-poly" are discussed, and how different materials and textures can affect the model. Also, the possibility to create fake geometry with materials and textures is covered. Blender also has, in addition to the pre-installed rendering engines, many third-party rendering engines. The properties and differences of the rendering engines is covered and use purposes for each engine is given.

For the project, a 3D-model of Kalevalatalo was made. The model was created based on blueprints drawn by Eliel Saarinen in 1921. Copies of the blueprints were provided by the Museum of Finnish Architecture and they were the client that requested the modeling of the building.

Three models of the building were made, each with varying levels of details. As the modeling is done completely with real geometry, instead of textures, the amount of details greatly influences the number of polygons needed. The number of polygons affects the file size and the amount of time needed for the modeling process itself. The idea is also to determine what is a good balance between amount of details and modeling time. In the end it will be at the customer's discretion of what they want to emphasize on depending on the project.

Keywords	architectural modeling, 3D model

Tekijä Otsikko Sivumäärä Aika	Sami Nykänen Blender arkkitehtuurimallinnuksessa – Kalevalatalon mallinnus 41 sivua + 5 liitettä 4.5.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Media Engineering
Ohjaaja	Lehtori Toni Spännäri

Insinöörityön tarkoituksena oli tutustua Blender-nimiseen ilmaiseen 3D-mallinnusohjelmaan ja tutkia, kuinka hyvin se soveltuu arkkitehtuuriseen mallintamiseen. Blenderiä verrattiin yleisiin CAD-mallinnusohjelmiin käymällä läpi muutamia selkeimpiä eroja, joita ohjelmien toiminnoissa on. Koska Blender on tarkoitettu alun perin visuaalisempaan ja fotorealistisempaan mallintamiseen kuin perinteiset arkkitehtuurissa käytettävät CAD-mallinnusohjelmat, insinöörityössä perehdyttiin myös arkkitehtuurilaajennuksiin, joita Blenderiin voidaan liittää.

3D-mallintamiseen voidaan liittää eri mallinnuslähtökohtia kuten ”low-poly”, jolla luodaan yksinkertaisia malleja, joissa on vähän yksityiskohtia. Sen vastakohta on ”high-poly”, jossa polygonien määrä on niin suuri, että sillä voidaan luoda hyvin yksityiskohtaisia malleja. Materiaaleilla ja tekstuureilla voidaan myös vaikuttaa paljon mallin lopulliseen ulkoasuun. Blenderiin on mahdollista lisätä valmiina olevien renderöintimoottoreiden lisäksi myös kolmannen osapuolen tarjoamia renderöintimoottoreita. Ne saa joko suoraan lisättyä sisälle Blenderiin tai niitä voidaan käyttää ohjelman ulkopuolella. Jokaisella renderöintimoottorilla on omat hyvät ja huonot puolet, ja ne soveltuvat parhaiten kukin juuri tietynlaiseen mallintamiseen.

Insinöörityön osana tehtiin myös Kalevalatalon mallintaminen. Se mallinnettiin Eliel Saarisen vuonna 1921 piirtämien pohjapiirustusten pohjalta. Pohjapiirustusten kopiot tarjosi Suomen Arkkitehtuurimuseo, joka tilasi 3D-mallin Kalevalatalosta.

Tarkoituksena oli luoda kolme eri mallia, joissa kaikissa olisi eri määrä yksityiskohtia. Mallit luotiin täysin oikealla geometrialla arkkitehtuurimaiseen tapaan eikä tekstuureilla, joten yksityiskohtien määrä vaikutti huomattavasti mallin tiedostokokoon ja sen mallintamiseen tarvittavaan aikaan. Tarkoituksena oli myös selvittää, mikä olisi sopiva tasapaino yksityiskohtien määrän ja mallinnusajan välillä. Verrattaessa malleja saatiin selville, että mallinnusaika lähes kaksinkertaistui aina siirryttäessä enemmän yksityiskohtaisempaan malliin. Asiakkaan harkittavaksi jäi tulevaisuudessa se, halutaanko painottaa enemmän yksityiskohtien määrään vai mallinnuksen nopeuteen riippuen kulloisestakin projektista.

Avainsanat	arkkitehtuurimallinnus, 3D-mallinnus

Helsinki Metropolia University of Applied Sciences

Contents

List of Abbreviations

1 Introduction 1

2 Architectural modeling 2

2.1 CAD – Computer-aided design 2

2.2 Photorealistic modeling 3

3 Working with Blender 7

3.1 Details 7
3.1.1 High-poly 7
3.1.2 Low-poly 8
3.1.3 Materials 8
3.1.4 Textures 10

3.2 Creating the model 14
3.2.1 Set-up 14
3.2.2 Add-ons 17

3.3 Render engines 18
3.3.1 Cycles 21
3.3.2 V-Ray 22
3.3.3 Renderman 23
3.3.4 Luxrender 24
3.3.5 Mitsuba 25

4 CASE: Modeling the Kalevala building 27

4.1 Planning 27

4.2 Modeling 29
4.2.1 Low-poly 29
4.2.2 Med-poly 32
4.2.3 High-poly 33

4.3 Comparison 36

5 Conclusion 39

References 40

Appendices 42

Appendices

Appendix 1. Kalevalatalo reference images

List of Abbreviations

2D	Two dimensional
3D	Three dimensional
AR	Augmented reality
BSDF	Bidirectional scattering distribution function
CAD	Computer-aided design
DFX	Drawing Exchange Format
OBJ	Object file
VR	Virtual Reality

1 Introduction

The goal of my thesis is to create a realistic 3D model of Kalevalatalo with Blender and to see which kind of tools Blender 3D modelling software offers for architectural modeling. This thesis covers the differences between high and low polygon count modeling, how textures and materials work and effect the end result, and what different kinds of rendering engines there are available for Blender. Setting up Blender to be ready for use and what add-ons aid in architectural modelling are also some of the topics.

There are other CAD-software that are built for architectural modeling, but these can have quite expensive licenses. This high price tag is the reason many hobbyists find an open-sourced and free software like Blender to be a more appealing choice. As a very community reliant software, most of its tools and strengths come from the community itself. Blender foundation releases a new version every now and then, but new tools are being made by the community all the time and they can be easily added as add-ons to the software. With these add-ons anyone can streamline their version of Blender to be the most suitable for their use cases.

2 Architectural modeling

2.1 CAD – Computer-aided design

CAD has during the last couple of decades all but replaced the semi-transparent paper, pen and a ruler as the main tool of the trade. At first architects continued to draw their plans in 2D but nowadays 3D has replaced it almost altogether. Creating the plans in 3D makes the whole job a lot easier and allows for clearer presentation of the object. 2D representation limits the perspective a lot compared to 3D where you can see the object from any angle and it is even possible to enter a building to see it form the inside. (Laakko 1998; Virolainen 2006.)

CAD-models are not limited to being just a 3D representation of the object. Architects can include detailed information of the object when creating the models. Things like sizes and volumes of certain parts of the object are calculated and included to the model easily. It is also possible to assign materials to the objects so the strengths of each part can be calculated. In a building-model for example every wall would be its own object that has its own size, volume, mass, and other relevant details. When all these details are combined for the whole building, for example a quite accurate estimation for the material expenses can be calculated. (Gangsö 2006.)

CAD-models can be also animated to create demonstrations of the object for marketing purposes. Models can also be placed in locations where they are planned to be located in the real world and these then can be viewed with devices that support augmented reality (AR) such as mobile phones. With AR anyone can walk around the location to see how a planned building would look like. AR technology has advanced greatly and is starting to show up but it is still more common to see an aerial photo where the model of the building has later been placed. (Zellner 1999.)

CAD is mostly meant for creating accurate plans for buildings or other objects to be constructed in the real world. CAD-software allows for some visualization of materials or animating the model, but it does fall behind on these aspects when compared to software that are meant for this kind of work.

One key difference with CAD software and other modeling software like Blender, is that CAD deals mostly with solid modeling instead of surface modeling. Simply put, in CAD

the objects actually have volume and there is no empty space inside. Whereas, in Blender for example, the objects are more or less skins representing a solid surface. When cutting a cube open, in CAD it would look like a solid object, but in Blender, it will look like a hollowed out shell of a cube as seen in image 1.

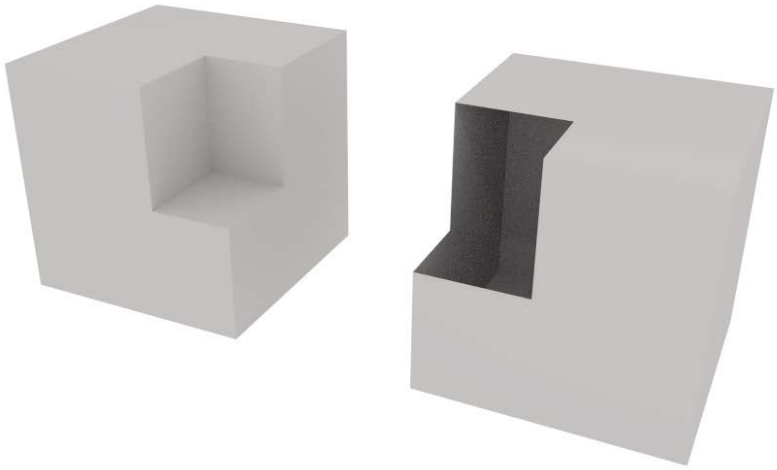


Image 1. Illustration of cubes cut in CAD and Blender. Nykänen 2016.

2.2 Photorealistic modeling

The software used for photorealistic modeling are equipped with many similar tools to the ones in CAD software used by architects, but the focus is more on the visualization side of the modeling.

Many of these software like Blender, which will be the software to be used in this thesis project, support the DXF file format that can be imported from the CAD software. This allows for cases where the object or building is first modeled in CAD and later made visually more pleasing for example for marketing purposes. Blender also supports other formats from different software like 3DS-files from 3DS Max by Autodesk or OBJ-files from Maya and many more. (Brito 2008.)

When making photorealistic models it is good to remember that exact dimensions are not that important. Proportions on the other hand are still important to be kept in mind or the model will not look realistic. This comes more apparent when adding other elements

to the scene such as furniture or people. When doing this kind of modeling it is not required to stick with 'real-life' measurements as no-one will be able to tell from a rendered video or image that is a wall 15 meters wide or 5 meters wide. (Brito 2008.)

Common problem that new users face is that part of the object is clipped out when rendering (see image 2). This can be sorted out with either having the scale the model smaller in size or by going to the camera settings and changing the end value in the clipping parameter. In the example image below both of the models share the same proportions but the size of the model on the right side has been scaled down. Model on the left side is scaled to the real world measurements which causes the model to be partially clipped from the camera's view. This can also be done by increasing the end clipping value. By default, the value is 100 which seems like an arbitrary number as Blender does not give any info on what that value means, but by increasing it the clipping boundary can be pushed further. (Brito 2008; Blender Foundation 2016)

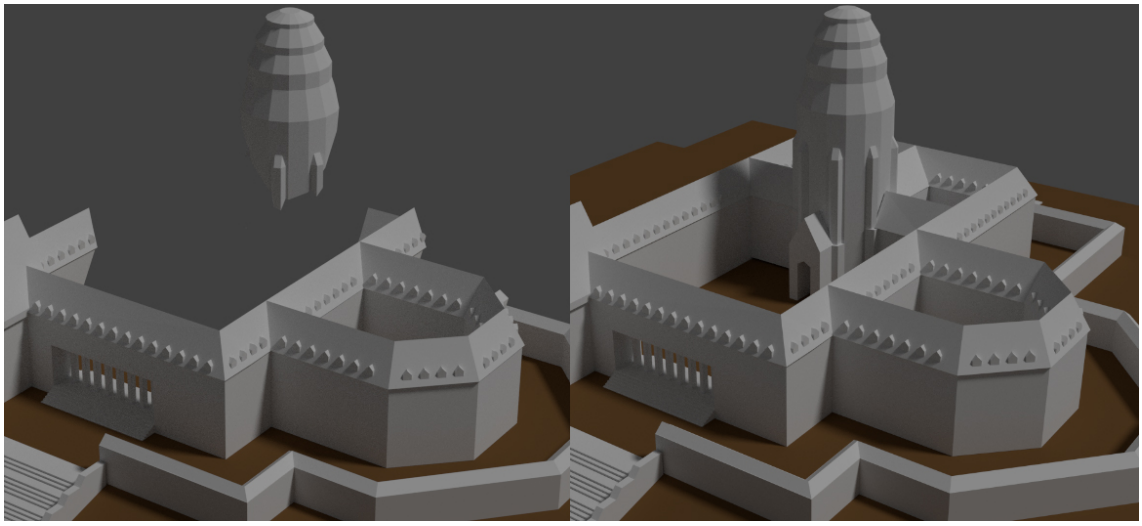


Image 2. Example of camera clipping. Nykänen 2016.

What the camera clipping does is that it allows the user to manipulate the depth of the area that the camera sees when it is rendering the image. It is also possible to change the start value in the camera clipping settings. By changing this value, it is possible to make the renderer ignore objects that are closest to the camera. With this it is possible to make a render of a building so that the camera sees through the wall and making the inside of the house visible without actually removing the wall from the model. After the modeling is completed and it is time to add it to a scene, the model can be scaled up or down to fit the scale of the scene. (Blender Foundation 2016)

When making a photorealistic model or scene it is advised to set the cameras to their final locations and angles before going too far with the modeling. This way it is easier to see which parts of the scene require more detailed work and which parts are obstructed and cannot be seen by the camera. It is pointless to sink too much time and effort to areas that will not be visible in the final render anyway. When the end-result will just be a static image you can fake many things to make an illusion of a complex solid model. (Brito 2008)

The images below demonstrate this method in use. Image 3 portrays a tree and creates the illusion of a larger tree that is left outside the borders of the image, but in the second example (image 4) it can be seen that the elements in the image 3 are just composed from separate copied objects that are placed so that it creates an illusion of a whole tree.

Image 3. Green cherry blossom tree. Nykänen 2015

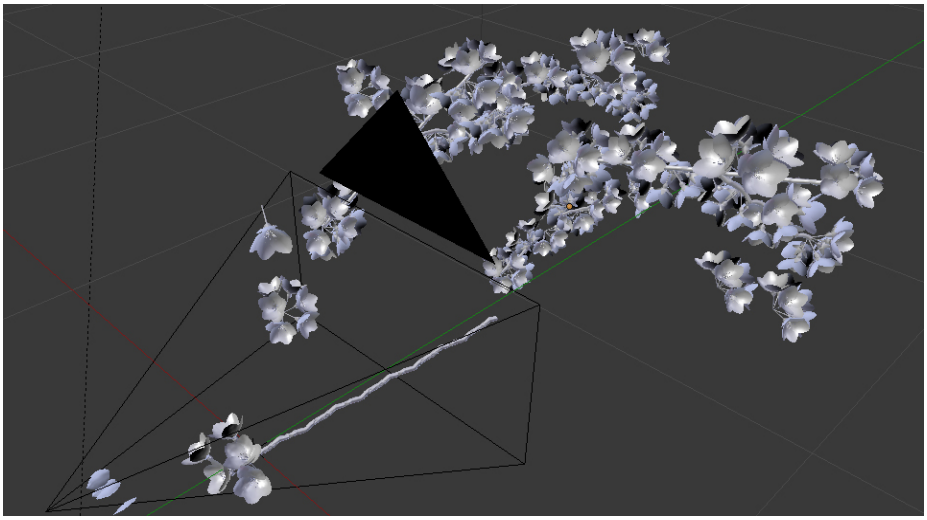


Image 4. Green cherry blossom tree scene. Nykänen 2015.

Nowadays, Virtual Reality (VR) is getting more and more popular and the artists cannot rely on these kinds of tricks in those projects. In VR all the objects have to be made fully and with high detail or otherwise the user will immediately notice it and the immersion will be broken. In VR the user is able to view almost everything from any angle he or she wishes and the artist needs to take this into consideration when creating the world or the object. A larger scene of course can include elements that may be out of reach for the user; so these could technically be made with lower detail, in order to save time and resources without sacrificing immersion.

3 Working with Blender

3.1 Details

The level of detail that is wanted from the model is good to be thought through before starting the modeling process. This can affect greatly the way it will be modeled. There are many ways to start building a model. The model can be made very highly detailed with a large amount of polygons; this is called high-poly modeling. The other way of modeling is called low-poly and the name comes from the idea of creating the model with as few polygons as possible. When modeling for architecture it is advised to consider the placement of each object when deciding on the level of details needed. It is pointless to create a highly detailed model if it will be placed so far from the camera, that almost none of the details will be visible anymore in the final render. (Brito 2008)

3.1.1 High-poly

Models done with high-poly count can utilize the sculpting tool that Blender offers where the user can model in a similar way as sculpting with a piece of clay. It is very easy to add and remove geometry from the model with this tool and achieve the desired look for the model. This tool creates a very high amount of polygons and usually these meshes need to be retopogolized before it can be used in any animation or other project. Re-topologizing is important because the sculpting tool does not consider the edge flows of the mesh which are very important when adding materials or animating the model. If it is not done the mesh can distort the materials in unwanted ways when the models are being animated and already the UV-mapping that is done before adding materials will be a great deal harder if not impossible. More about UV mapping will be explained in the Mapping section of the thesis. (Williamson 2012)

One downside in meshes with very high polygon count is that they can be very heavy and extremely demanding from the computer as there is such a large amount of data the computer needs to take into consideration when rendering the scene. This is where re-topology comes into the picture to create another version of the mesh with a much lower amount of polygons and with the same details, which then can be animated and textured properly.

3.1.2 Low-poly

Retopologizing a high-poly mesh is not the only way low-poly models are used, although it is a very popular workflow used by game artists and such as it allows the creation of detailed lightweight models. Low-poly modeling is its own art form also and is used nowadays a lot especially by many indie studios. Quite often when seeing low-poly models they have a much more simplified look than typical high-poly models. Materials used are often more cartoonish than realistic as seen in image 5.

Image 5. Example of low-poly models. Nykänen 2015.

3.1.3 Materials

Materials are used to determine how the surface of the object interacts with light. The main concept is that materials set up how the surface reflects or if the light passes through it. These settings combined with textures and other details define the look of the object in the final render. Most of the material editing takes place in the node-editor, which allows the creation of very complex materials by building node trees. (Brito 2008)

Blender is equipped with many different shaders natively, but users are able to also create their own or download them from the internet. A few of the most common shaders used in architectural modeling are listed below

Diffuse BSDF - receives light and diffuses it without any visible reflections. It is used for non-reflective surfaces like walls or paper. It is also used for mixing with other shaders.

Glossy BSDF - reflects lights and the environment. It is used to add reflections to any object. It is a very common shader to be used in combination with the diffuse shader to create common materials such as plastic or metal.

Glass BSDF - acts like real glass by bending and reflecting light that hits its surface according to its IOR (Index of Refraction). Used when creating glass for windows or any object that bends light. Can also be used for water by changing the IOR to match that of water.

Subsurface Scattering - simulates the light scattering beneath the surface of an object. Used to create realistic skin, wax or marble.

Mix Shader - Takes the output of two shaders and mixes them together. In reality, almost no material has the qualities of just one shader, but rather a mix of different properties. Most surfaces can be created with just a diffuse shader to give colour and then glossy shader to add a little bit of reflection to the surface so it looks more realistic as can be seen in image 6. (Blender Foundation 2016)

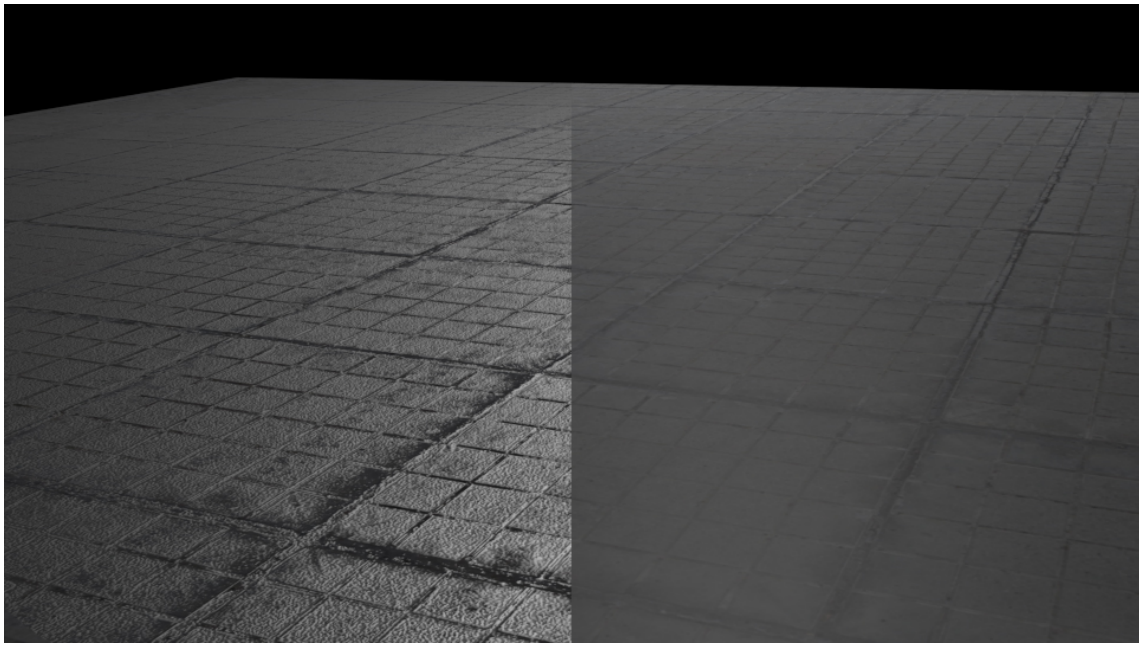


Image 6. Material example. Nykänen 2016

The example image above demonstrates clearly how much more realistic a material can be made with just a little bit of tinkering. On the right side the plane has just an image attached to it with a diffuse shader and also a glossy shader to give some reflection to the surface. This creates a very flat and not realistic surface. The left part of the picture has the same diffuse and glossy shaders plus a bump and specular map to create more details. The end result has a great deal more details without having to add any more geometry to the model.

3.1.4 Textures

When adding textures, it comes quickly apparent that without UV mapping it will not look good. In the earlier days of 3D people used to fill the surface area by tiling a smaller texture image on it. This creates a very easily distinguishable artificial pattern that does not look realistic. The way to create a good-looking texture is to UV map the mesh and then position a higher resolution image on top of the UV map. This tells the software the exact locations of the texture on the models surface by using the U and V coordinates. (Brito 2008)

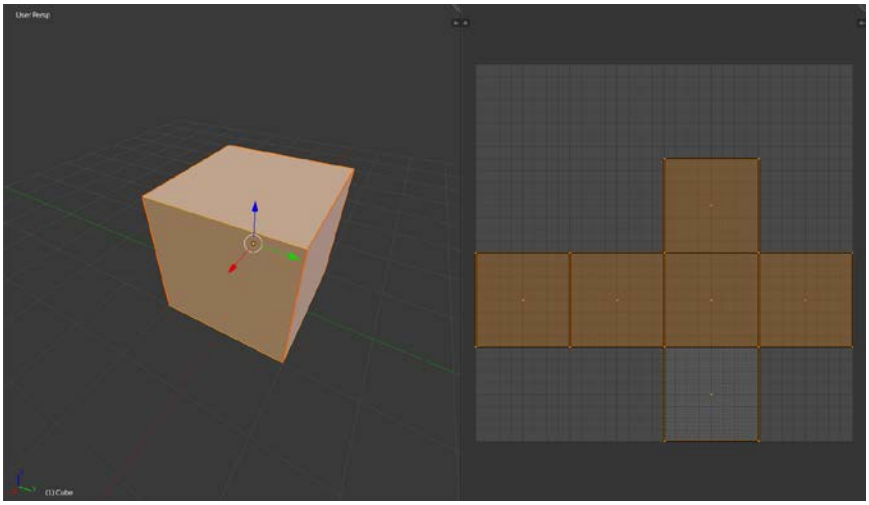


Image 7. Example of UV unwrapped cube.

The Process of creating the UV map is very simple, but can be very tedious task with more difficult meshes. Image 7 demonstrates how the UV-map is done by marking a seam along the edges of a mesh so it can be cut into several parts, and then unfolded along those seams. The seams should be marked at the key edges or otherwise the mesh will not unfold properly for the map. A good rule is to leave all the faces that will have the same texture together. This way it is easy to add a texture to all of them. (Brito 2008)

Another way to achieve more realism from the textures is to use normal maps. Normal is a vector that is perpendicular to a given object or polygon in this case. How light affects the polygon is calculated from the angle of the normal to the light source. With normal mapping it is possible to change the orientation of the normal for each pixel in the final render to create an illusion of depth and details in the object. The map is usually created from the texture image that will be used in the object. It can also be created from a high poly version of the mesh to give the low poly version the almost the same amount of details without increasing the amount of polygons. Normal map works by giving each pixel a RGB colour value corresponding its orientation in the sample image. The colours are determined with the following table. (Wikibooks 2016)

	-1	0	1
X (RED)	0	128	255
Y (GREEN)	0	128	255
Z (BLUE)		128	255

The most common colour in the map is light blue, which is given to the pixels that have the value (128,128,255). The normals of these pixels are pointing straight towards the camera.

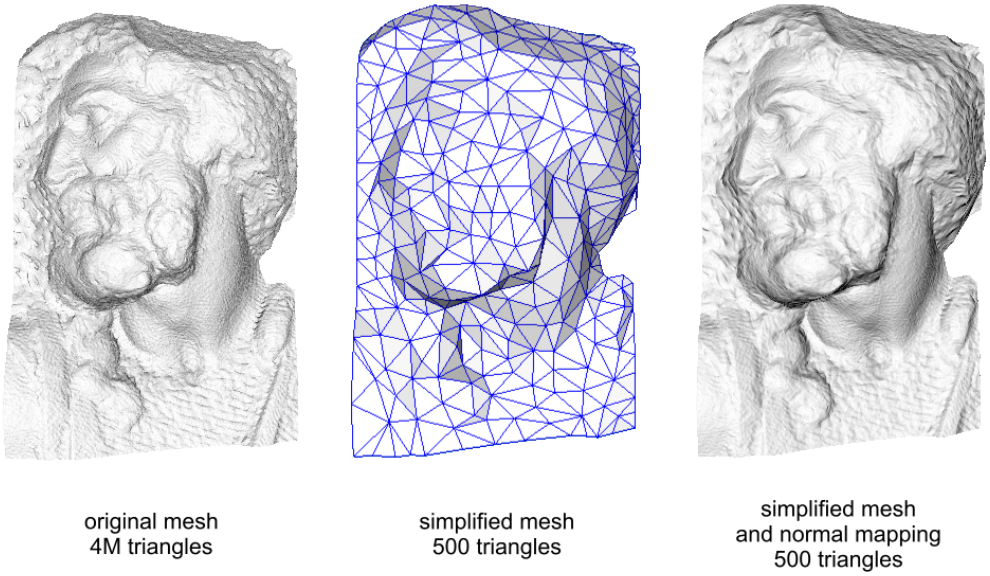


Image 8. Normal map example. Cignoni 2016.

Normal maps can also sometimes be called bump maps as it was earlier used to fake bumps and dents in the texture. Bump maps are represented as grayscale images where the intensity of each pixel represents the height value to be faked in the final render. The grayscale images can also be used as displacement maps where the actual geometry of the mesh is altered based on the intensity values. For this use, the polygon count needs to be much higher or otherwise the result will be poor as there is not enough geometry to create the details. (Brito 2008)

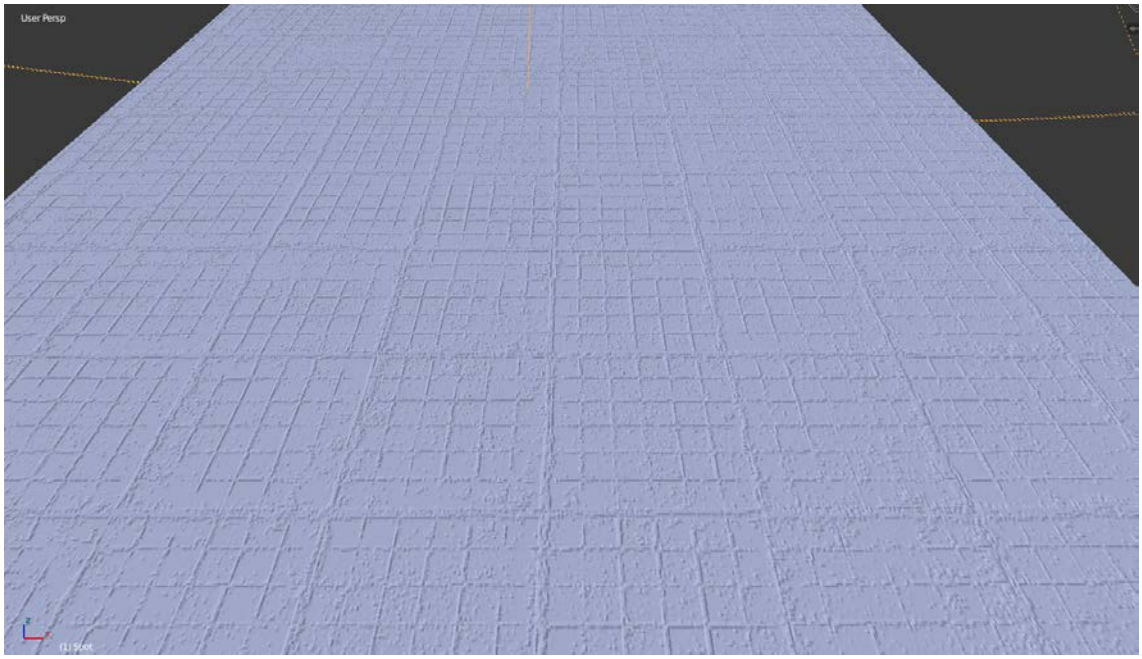


Image 9. Displacement map result. Nykänen 2016.

In the example above we have the same plane that was used for the comparison of a material with and without bump mapping. In this version the details have been created with a displacement map. As mentioned earlier this can only be achieved by adding a lot more geometry to the model. The original model had only a single polygon, whereas this version has 654,481 polygons. This of course is absurdly high for a simple floor, but it serves as a good example how much difference in the polygon count the use of displacement maps makes. Displacement map is a very helpful tool to create more detailed geometry easily without having to spend hours creating it by hand. If this kind of object that had so many polygons would be wanted to use in an animation, it would have to be baked. This makes it so that all the lighting, shadows, textures and displacements of the material would not have to be counted every time a frame is rendered. Downside of baking the material is that it cannot be edited anymore without having to bake it again.

Baking can make a significant difference in the rendering times. As an example the image below takes 3 minutes and 58 seconds to render without baking and after baking the time is reduced to just 15 seconds. If I would use this scene in a ten-second-long animation, which would mean 250 frames, without baking the rendering of that animation would take 16 hours and 31 minutes, whereas with baking it would only take 1 hour and 2 minutes.

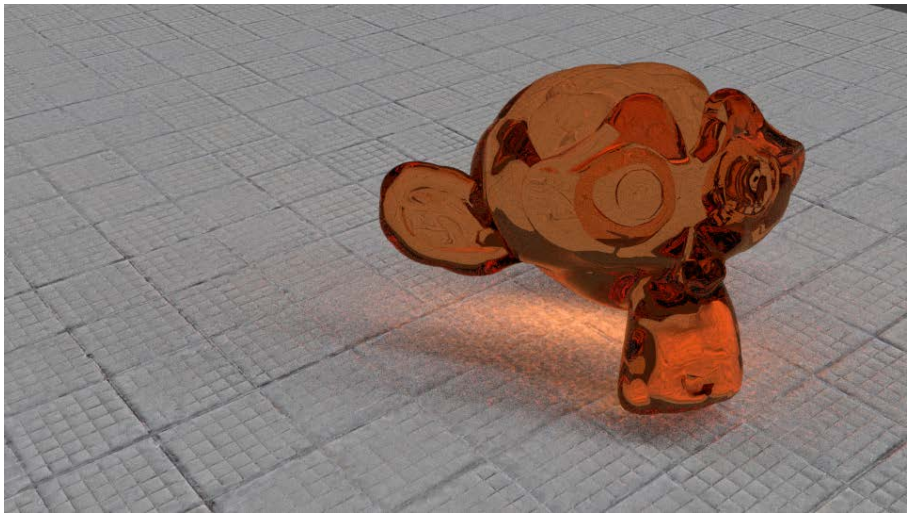


Image 10. Example of baking. Nykänen 2016.

3.2 Creating the model

3.2.1 Set-up

When starting any modeling process it is good to make sure all the settings are optimal as it eliminates a lot of headache that might otherwise occur along the way. Blender is notorious in having sub-optimal settings on a fresh install. The first thing anyone starting Blender for the first time should do is to open the settings menu and under the ‘interface’ tab find the option for ‘Prompt quit’ (see image 11). By default, there is no warning when quitting Blender while the work is unsaved. This would mean the loss of all work done after previous save unless by luck the work can be recovered from auto-save files. With this setting enabled Blender will ask the user if they are sure that they want to quit if there are any unsaved changes in the scene.

Image 11. Prompt quit option.

Another important thing to change is the amount of steps in the undo function. The default number of steps is only 32 which in some cases is quite little amount. Especially if doing some small changes, the step amount is used up fast. In the current Blender v2.76 the maximum amount of steps is 256. It is also possible to limit this just by how many megabytes of memory will be allocated to this task (see image 12). Higher amount of steps of course means higher amount of memory required but it should not be a problem with modern computers.

It is also important to note how the undo function works when changing between object and edit modes in Blender. While staying inside edit mode clicking undo will undo just the previous change done to the mesh, but if the user first goes back to object mode all the changes done while in edit mode will be undone.

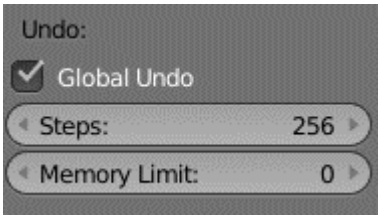


Image 12. Undo settings. Blender 2016.

A minor thing that confuse many users at first is how objects are selected in the 3D view. Most other 3D software use the left mouse button to for selecting and right mouse button to rotate the viewport. Viewport is the region in the 3D modeling software that shows the actual model. All the modeling will be done in this window.

In Blender selection is done with right mouse button and the left button places the 3D cursor. Viewport rotation is done with the middle mouse button. These also can be changes quickly from the settings menu under the ‘input’ tab. The way the camera orbits around the scene when rotating the viewport can be changed between default ‘turntable’ or ‘trackball’ style that is used by default in Autodesk’s software.

More detailed information about the object that is being edited can be enabled from the properties menu that can be found on the right side of the 3D viewport, if it is not visible it can be opened and closed with the shortcut key N (see image 13). While in edit mode the properties menu will have a section called ‘Mesh display’ that has checkboxes for edge info and face info. This additional info might not be that important or useful in all modeling projects but for architectural modeling this makes it much easier to keep the proportions correct.

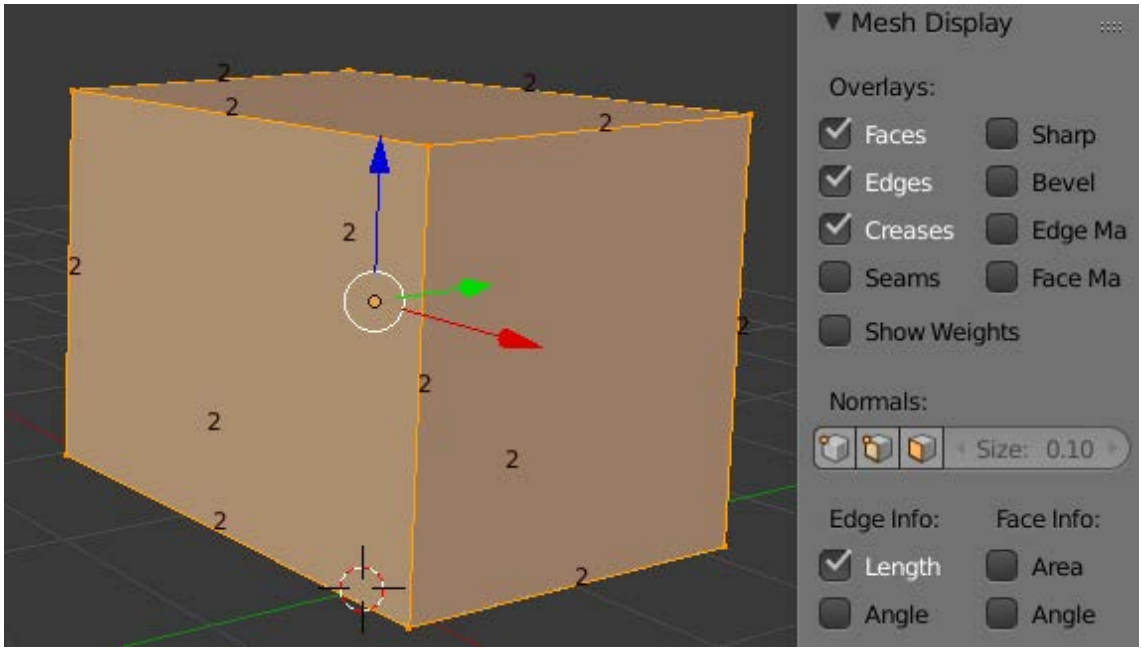


Image 13. Mesh Display options enabling edge lengths. Blender 2016

Reference images can be added as background images and they can be made to be visible only when viewed from specific axis. This way it is possible to have several reference images without them cluttering the whole scene. They also can be set to be visible from any axis or just when viewed through the camera in the scene. Background image is not limited to just a still image but can also be a movie clip. It can be a selected form the file many same as normal image backgrounds (see image 14), or if some movie clip is also being used in the scene for example for camera tracking it can be selected by checking the “camera clip” -checkbox in the menu.

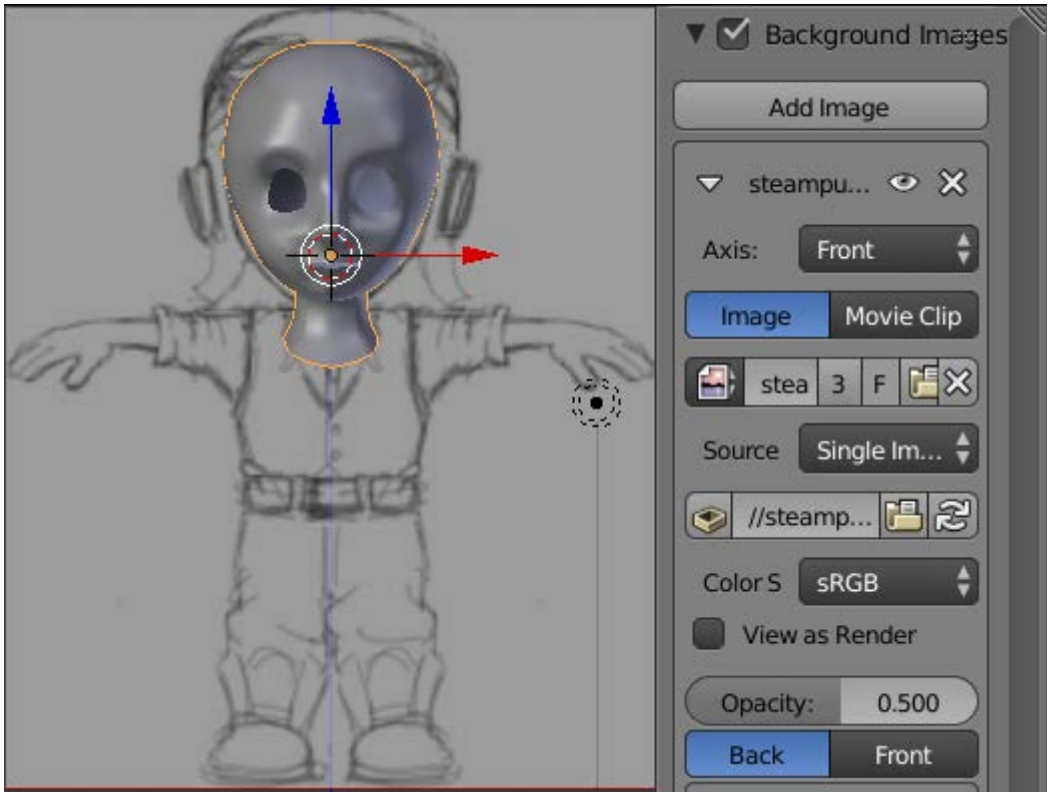


Image 14. Background image to help character modeling process. Nykänen 2015

3.2.2 Add-ons

As Blender is an open-sourced software its true strength comes from the vast and active community. Numerous great add-ons enable new tools or make certain tasks faster and easier to complete.

For architectural visualization one almost indispensable add-on is Archimesh which helps streamlining the creation process a lot. Most of the assets are easily modeled from scratch also but when the project is large it saves time by doing it with some automated tools. Archimesh allows for the creation of a room for example in just minutes by inputting the lengths of each individual wall. It is also possible to change the angle of the corner and curved walls can be created effortlessly.

The layout of a house might be something that the artist wants to create from scratch but when it comes to adding some assets to the scene to make it visually more pleasing it is recommended to use these kinds of tools like Archimesh. It is pointless to waste too much time and effort in creating these assets that are there only for show. Things like

lamps and collections of books can be added with a push of a button. Of course an artist that has been doing it for a while most likely will have a library of assets at his or her disposals rendering these kind of add-ons redundant.

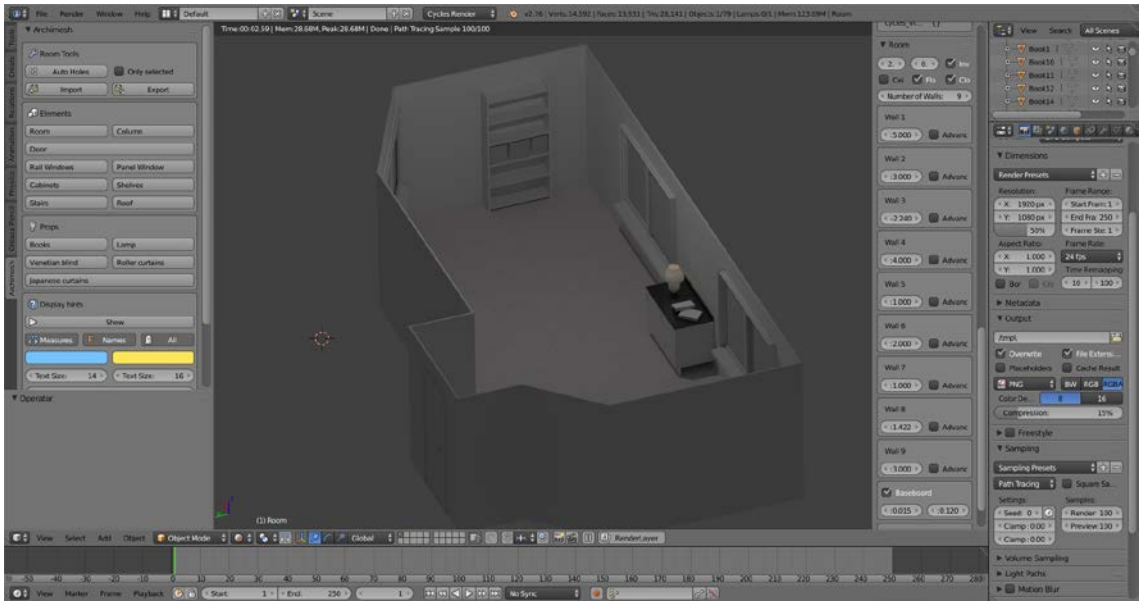


Image 15. Archimesh menu and model. Nykänen 2016.

While Archimesh does come with several different asset creation tools, there are also some other add-ons that are dedicated for specific tasks. Window Generator 2 for example allows for the easy creation of almost any kind of window. From the tools menu it is easy to change any parameter of the window to change the size or shape from typical rectangular to arched window. Materials are also automatically created and linked to the correct meshes. Another similar add-on is Floor Board Generator, which does exactly what the names implies. Fast way to create floorboards with actual geometry instead of an image mapped on a plane. Here good thing to remember is that, as mentioned previously in the details section of this thesis, this added geometry will have an effect on the weight of the whole project and will slow down the rendering process. (Baker 2016)

3.3 Render engines

Blender has its own internal rendering engine which it uses by default. Blender Render has been there since the beginning of Blender. Although it has been updated a lot during the years, many in the community feel that it has become outdated and should soon be completely replaced by Cycles Renderer that was released in the 2.61 version in 2013,

as the default renderer. Many still argue that it has its place in Blender and performs some tasks better than Cycles.

Blender Render is a CPU based renderer. This means that instead of using your graphics card to do the calculations it will use your processor. This brings some advantages and some disadvantages. CPU usually has more memory to be allocated for the rendering task than a GPU would have. This means that heavier scenes can be rendered. On the other hand, CPU has less cores to do the calculations than the GPU has, which can lead to a lot longer rendering times especially when making more realistic renders. Using CPU also limits you from using your computer to do other tasks while rendering in the background since all your CPU cores are being used for the rendering. It is still possible to use some of the cores for other tasks, but this will increase the rendering time significantly. (Blender Foundation 2016)

With Blender Render it is much more difficult and in some cases impossible to reach a photorealistic render of a scene. This is because it lacks some features like caustics, and some features are not as mature as they are in other renderers like Cycles for an example. Blender Render can be a good choice when creating non-photorealistic and more cartoon style renders or real-time rendered models in games. (Baldi 2016)

Since Blender is an open-source software it is freely customizable by its users. Many of the Blender Renderer's shortcomings can be overcome or worked around by some python coding which can be done in the python console that can be accessed with Shift+F4 (see image 16). This of course is not something your typical user is capable of.

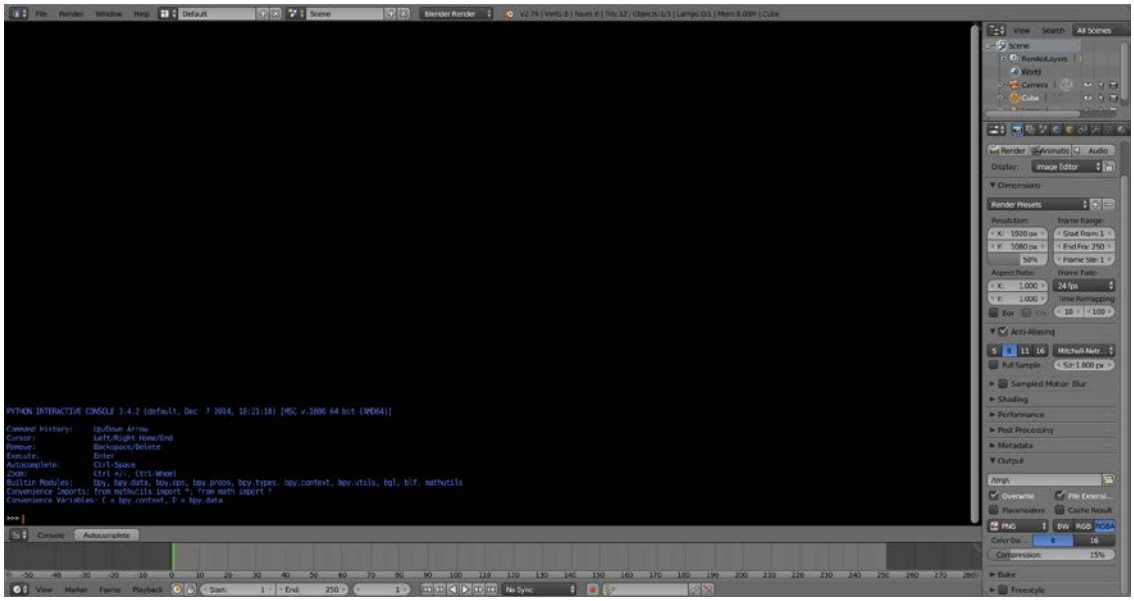


Image 16. Python console in Blender. Blender 2017

Current renderers can be divided into biased and unbiased. Biased renderers tend to be faster but also by achieving this they give up on realism (see image 17). They are vastly more tweakable than unbiased renderers and to get the most out of it the users need to get familiar with all the settings to find the optimal values to reach the best quality render. Realism can be achieved but it requires more work in the rendering settings. (Cameron 2016)

Image 17. Render engine comparison. Baldi 2016.

Before the rendering process is started some calculations are already premade under which rules the renderer then does its job. When working with unbiased renderers the emphasis of render quality lies more in the modelling itself as well as the materials and textures. Unbiased renderers have very few settings and will render a physically correct image of whatever the camera sees. The rendering process will not finish on its own and the user has to stop it when the quality of the render is at the desired level unlike in biased renderers where the process will eventually stop according to the parameters set before starting. (Cameron 2016)

3.3.1 Cycles

Cycles is the most popular rendering engine used in Blender as it is available to users immediately when installing Blender given they select it as the active engine. It was added to Blender with the release of Blender version 2.61 back in 2011. Since then it has seen very rapid development and received a plethora of new features. (Cycles official documentation 2016)

One of its biggest advantages over its predecessor was and is that it is using a path tracing algorithm to illuminate the world. This algorithm allows Blender to calculate how light travels and bounces around different surfaces and objects in the scene very accurately which allowed for a Global Illumination system to be added to Blender. This enables it to render more realistic images than Blender Render before it. In Blender Render to reach similar Global Illumination effects it had to be essentially faked by different methods. With this Global Illumination system, the lighting in the scene can be made to look very realistic by making each object's colour influence the colour of the objects around it and vice versa. (Baldi 2016)

Cycles also allowed the creation of far more advanced materials by its new node-based shader system. This allows users to combine and tweak many different materials together to create very complex node-trees in order to achieve realistic materials for their models. Basically this adds much more control over materials and how they are rendered in which situation. Editing these materials was made easy by Cycles' fast real-time rendered view where the user could see very quickly the final result as they are making changes to the materials. (Baldi 2016, Iraci 2016)

Another big change that came with Cycles that it is meant to be powered by the computer's GPU instead of the CPU. CPU rendering option is still available but it is a lot slower compared to CUDA option for NVIDIA's GPUs and OpenGL which was later on added for AMD's GPUs. As mentioned before GPU based rendering allows for far faster rendering since they are designed to crunch more numbers than CPUs. Only drawback being that due to lower memory capacity rendering more complex scenes can cause problems. When Cycles was first released the GPUs most users had had only around 2 gigabytes of GDDR memory. As technology advances and user are upgrading their graphics cards to newer models which have closer to 6 gigabytes of GDDR memory problem is not as notable anymore. (Cycles official documentation 2016)

With all these new features that Cycles brought helped Blender to become a more notable in the 3D modelling industry. This increased the user base of blender vastly since now it was possible to create 3D scenes comparable to ones made by the bigger 3D software in the industry like 3Ds Max.

3.3.2 V-Ray

V-Ray is a 3D rendering plugin that can be used with several different 3D modelling software. As a more widely used and very robust and advanced rendering engine it stands to reason that unlike to Blender's core software and its official add-ons the V-Ray renderer is not free. This is a more professional rendering engine that costs around 250€ for the standalone renderer which then can be used with all supported 3D software. (V-Ray official documentation 2016)

In Blender it cannot be used as a rendering add-on the same way as Cycles. In order to render with V-Ray in Blender the user needs to install a separate build of Blender which is provided by Chaosgroup the company behind V-Ray. This Blender build has V-Ray additions pre-installed in it. Blender still cannot render on its own using V-Ray. These additions are only for the tools needed to create the materials for V-Ray renderer. A standalone V-Ray rendering software still needs to be installed to the computer. The additions in this Blender build make using this renderer faster than it would be normally. Normally user would need to export the scene first from Blender in a format that V-Ray can read. After that it needs to be imported to V-Ray and rendered there. With this separate Blender build all this is automated. Models are created in Blender and when the user wants to render it the scene is automatically rendered in V-Ray and the end result

can be shown either in blender or in a separate window. (V-Ray official documentation 2016)

This separate Blender build makes working with V-Ray easy and fast, but it also means that users of this build will not get the new and updated features that come with new official Blender releases until Chaosgroup releases their version of the newest build. (V-Ray official documentation 2016)

V-Ray is the most popular renderer for a reason. It is all-in-all the fastest renderer out there and produces high quality renders. It is the most adjustable and robust rendering engine currently available for Blender. Unfortunately, the amount of complexity that comes from so hugely adjustable features and the very lacking official documentation it causes the learning curve to be very steep. This makes it difficult for new users to get into using V-Ray, but the community provides many tutorials which eases it noticeably. (Iraci 2016, V-Ray official documentation 2016)

3.3.3 Renderman

In 2015 Pixar released the rendering software they have been using in their animations as a free to use addition to Blender. Renderman works the same way as V-Ray in the sense that the standalone rendering software is needed in addition to the add-on inside Blender. The difference is that Renderman does not require a separate Blender build but works fine with the official build. Renderman is free to use for anyone but for commercial use a 450€ license needs to be purchased. (Baldi 2016, Stepheson 2016)

Renderman could be considered to be in a separate group from the other rendering software since previously it has been unavailable for normal users. It is the go to renderer for nearly all animation houses in Hollywood. It is a very powerful renderer that can handle complex scenes with decent rendering times. (Stepheson 2016)

Even though Renderman is an external renderer for Blender it still provides real-time rendering through the rendering window called “it”. “it” is also equipped with powerful image manipulation and compositing tools which allow the users to do all the post-production right there and then without having to rely on other software. Blender itself also has internal compositing tools that go a long way but “it” provides a wider array of tools. (Stepheson 2016)

3.3.4 Luxrender

Luxrender was initially released in June 2008. It was developed to be an unbiased renderer based on the book called Physical Based Rendering by Matt Pharr and Greg Humphreys. Currently Luxrender has both unbiased and biased rendering modes available for the user. Luxrender is considered being a very accurate and powerful renderer with many different rendering algorithms to choose from, but it is also one of the slowest renderers out there even in biased mode. It also shines in post-processing where it has many options from lens effects and noise reduction for the user to manipulate the end result. (Baldi 2016, Luxrender official documentation 2016)

Luxrender is not used much by visual artists as it does not support all the features typically used by them such as hair modifiers. It shines in especially in scenes which have complex lighting. Luxrender has features that really set it apart from other renderers in this report. It aims to produce extremely natural and realistic results and of the features that allows Luxrender to do that is its full spectral rendering. This means that instead of the typical usage of RGB colour bands that is common in these renderers Luxrender simulates colours using the whole visible spectrum of colours for the internal calculations. (Luxrender official documentation 2016)

Luxrender also supports geometry instancing. Instances allows the rendering software to use the object's data to be reused multiple times in the rendering process for different identical objects. This allows the renderer save a large amount of system memory since recalculating the object information such as materials, shaders, displacements or animations is not needed. This is done by first creating one object with the desired parameters and then it is copied to for example a particle system where each particle creates one copy of the object. It is also possible to add variance to the particles so they will not all look exactly like each other while still not needing to recalculate the whole object data. (Luxrender official documentation 2016)

One of the most interesting features Luxrender offers is the multilight rendering option where it allows the user to adjust the intensity and colour of any light source in the scene even after the rendering has been finished. This is a very powerful post-processing tool, and this kind of effect in other renderers that don't offer this multilight feature can only be faked and amount of adjustments are very limited compared to this. (Luxrender official documentation 2016)

Another handy feature that is missing in many other renderers is the possibility to pause and continue rendering processes. This comes especially important and useful feature when rendering high detailed realistic scenes where in order to achieve good noiseless results the rendering times need to be somewhere 10 to 20 hours and beyond. Normally in these kind of situations the user has to leave their computer rendering for extremely long periods of time without being able to use it for anything else. Another option would be to use a rendering farm where you can split the rendering job to be calculated by several computers in order to make the rendering times be considerably lower. With this feature Luxrender saves the rendering process to a fleximage-file and in this file the current state of the rendering process is saved. This can then be later opened and read by any other system and continue the rendering process from where it was previously paused. This allows normal users that do not have the resources to have very powerful rendering hardware to create high-detailed renders that can be rendered in many sittings for example over several nights when the user is not using the computer for anything else rather than rendering it in one continuous session that might take several days. (Baldi 2016, Luxrender official documentation 2016)

3.3.5 Mitsuba

Mitsuba falls also into a section that is not really meant for artistic 3D production. Mitsuba's user base focuses more in research. It is equipped with an array of different rendering algorithms that might not yet be available in the mainstream of 3D software. Mitsuba is also an open-source software which explains the popularity amongst researchers since it offers a good playground for them to create and try different rendering algorithms. In contrast to many other renderers Mitsuba is focusing in CPU based rendering instead of GPU based. (Iraci 2016)

Mitsuba is a highly modular renderer with a library of well over a hundred different plugins available and more being created all the time. While it main focus is users who want to play around with the source code it also has integration plugins created which allows Mitsuba to be used as an add-on in Blender, Rhinoceros or in Maya. From these the Blender version is the most stable and the others are still in a very experimental stage of development. These plug-ins allow the scenes to be created in their respective software without having to deal with Mitsuba's native XML scene description format. This makes

it possible to make more complex scenes for Mitsuba. (Mitsuba official documentation 2016)

The true strength of the renderer comes from its scalability in the network rendering. As a CPU based renderer it can be really slow when trying to render complex images with only one computer. Nowadays processors come usually equipped with 4 or 8 cores. This means that if all 8 cores are utilized for the rendering process the rendering engine can calculate 8 tiles simultaneously. Mitsuba is able to be connected to other systems so that more CPU cores can be utilized for the rendering process. Mitsuba has been successfully scaled into a large cluster where more than 1000 cores were working together to render a single image. This kind of scaling in network rendering enables researches to create immensely detailed large images that are rendered still in a reasonable amount of time. (Mitsuba official documentation 2016)

To round it up, I would recommend to use either Cycles or LuxRender when doing architectural projects with Blender. They are both free and still powerful rendering engines. In this thesis project I will stay with Cycles as it comes straight with blender and I don't need to create very realistic lighting or materials for the model. Also for new users it might be recommendable to first stay with Cycles and later when they are familiar enough with the program they can branch out and try different rendering engines. LuxRender might outperform Cycles in rendering quality and rendering options, but it comes with the price of much slower rendering times. Also as it is an external renderer, to get it work properly might take some tweaking in the options, which might be a challenge to many users.

The other rendering engines mentioned in this chapter are meant more towards artistic rendering and depending on the case they might not be the best option for your project. They also come with a price tag that might be high for the average user especially when Blender itself is free and more commonly used by hobbyists.

4 CASE: Modeling the Kalevala building

The Kalevala building is an enormous building designed in 1921 by Eliel Saarinen who was an internationally known Finnish architect. He is known for designing the Helsinki Central Railway Station as well as the railway station of Vyborg that was later on destroyed in 1913. He also made a career in USA by designing the Cranbrook Educational Community.

The Kalevala Association ordered the Kalevala building, as it has been its part of its plans since the formation of the association in 1911. It was supposed to be a centre for Finnish culture, which would have included a culture research centre, exhibition halls and workstations for artists. The building would have included a crypt where notable Finns were planned to be buried. Kalevala building was planned to be located in Munkkiniemi.

The plans for the building revealed that the project would be extremely expensive, so soon the project was dropped and the plans to build a building was dropped from the Kalevala Association's agenda in early 2000.

4.1 Planning

As the building never progressed further from the initial plans and drafts, there is no finalized design for it. The Museum of Finnish Architecture has several almost one-hundred-year-old drawings that were used as reference images when building the 3D models. Each of the drawings have their differences, which made it harder to make an accurate representation of the design. In the end the modeling was done by combining multiple drawings (see images 18,19).

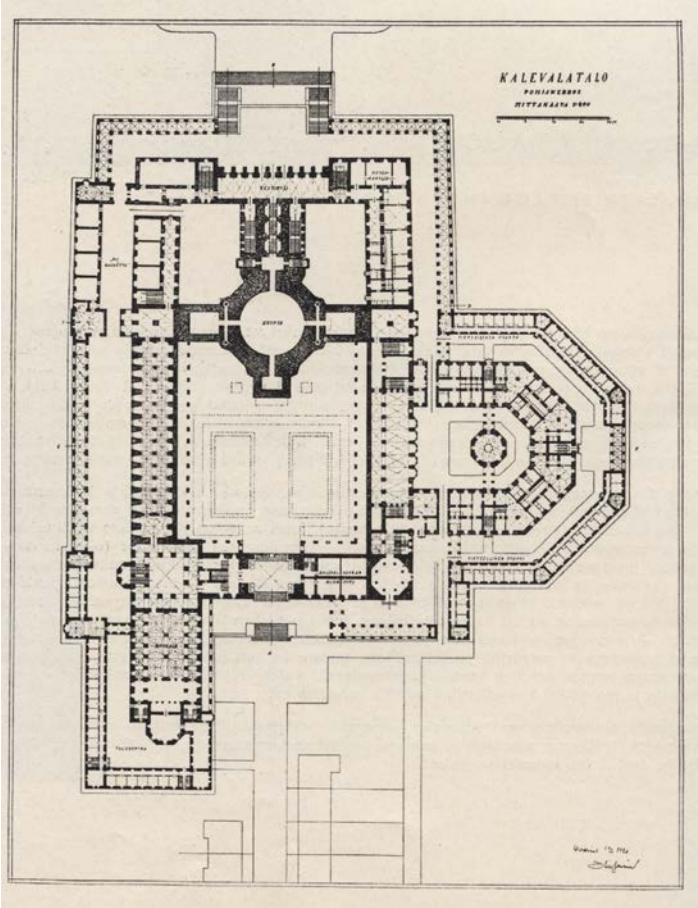


Image 18. Blueprint of the building. Saarinen 1921

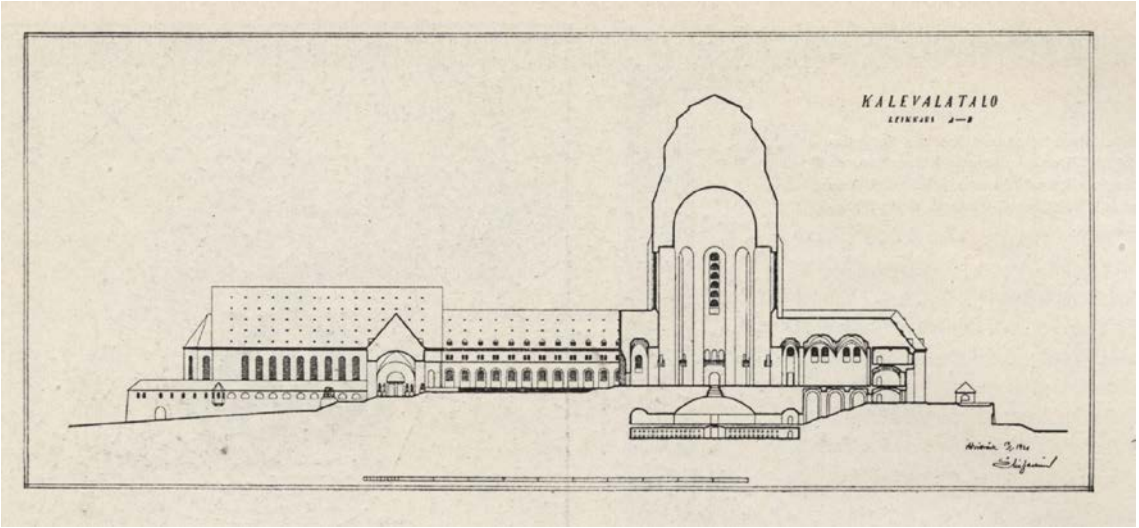


Image 19. Side-view drawing. Saarinen 1921.

The client allowed for some artistic freedom for the modeling and interpretation of the drawings, as it is a design that was never realized. This thesis project works as a test for them to see if creating these kinds of models from other designs in their archives would be plausible.

4.2 Modeling

I started the modeling process by adding the blueprint of the ground floor (see image 20) as a background image in Blender visible when viewing from the top axis. This allowed me to start creating the walls one by one on top of the image. Other reference images were not added to the scene as none of them had a proper side-view perspective that would have aligned well with the top-view. All other images were viewed separately on the side as needed throughout the modeling process. Then plan was to first create the low poly version of the building and then modify and add more details to achieve the higher poly-count versions planned for the project.

4.2.1 Low-poly

As mentioned earlier the low-poly model was started by creating first the main walls of the building on top of the blueprint (see image 20). The walls went through many iterations as I was trying to figure out the best and most efficient way to create them. In the first iteration the main part of the building was made from one single cube as it is a rectangular shape. This method would have provided the least amount of polygons; but I scrapped this idea since it made modifying the shape later on a bit more tedious.

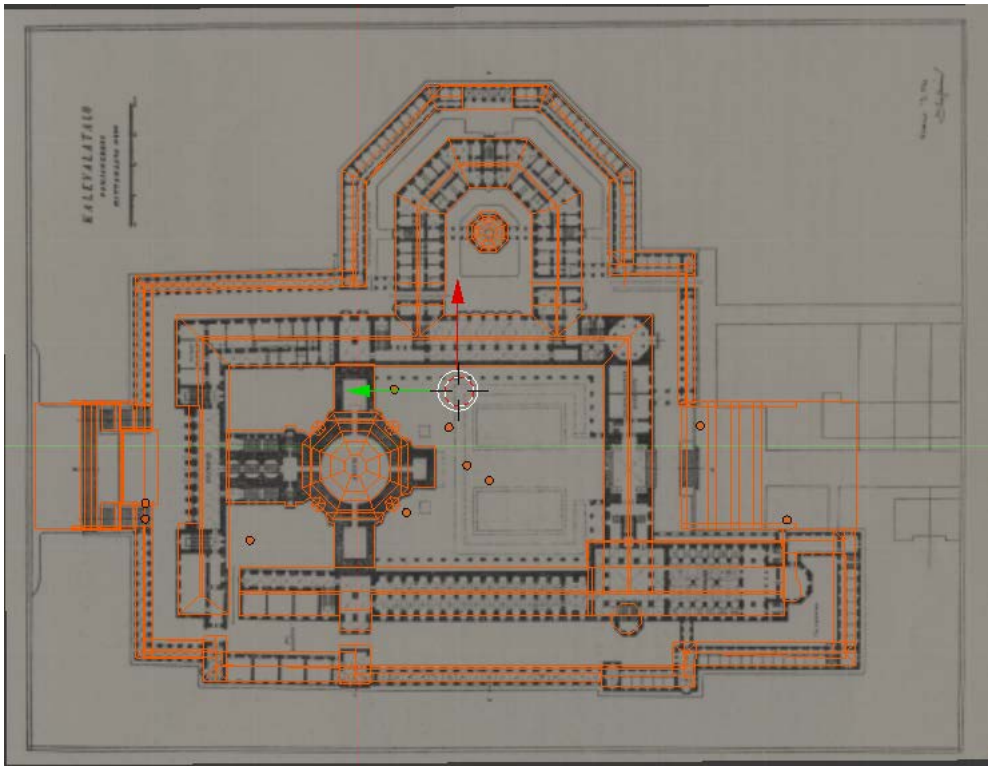


Image 20. Wireframe over blueprint. Nykänen 2016.

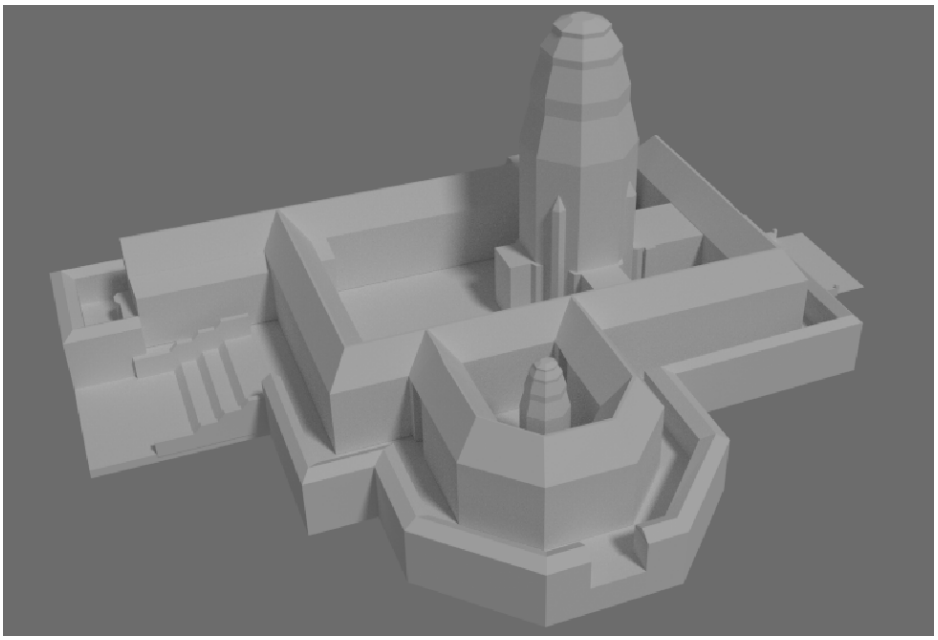


Image 21. Low-poly model. Nykänen 2016

Next iteration had all the walls as their own objects. Editing the walls was simpler than in the first iteration and the polygon count was not affected too much. This could be a perfectly valid method of creating the building; except in this method I had to place the walls so that they were overlapping each other in some places and this caused occasionally some problems in the form of unwanted artefacts when rendering the camera view. Objects had to be overlapping or the gap between them would be visible be it how small.

In the end I ended up creating the walls as a single object by extruding the next wall from the previous one form the corner (see image 20). This method resulted in a small rise in the polygon count, but not great enough to have any drastic effect. Having all the walls as a single object makes it much easier to manage in my personal opinion. Later, if the model would be textured, it will be easier to map properly and there will be no weird texture clipping in the corners of the walls.

Image 21 shows that the smaller inner yard had first a smaller tower in the middle, but in a discussion with the client it became apparent that it is not supposed to be a tower but a small building that would not be visible over the walls. This was changed later.

This same single object method was used throughout the modeling process, as the low-poly version is supposed to be a simple version without too much detail, it was the most efficient and fastest method of modeling.

After sending the first version to the client for viewing it came apparent that the surrounding walls seemed a little bit too massive for the model, even though they were done according to the designs. Here we have a good example of problems that might rise from not having matching blueprints for reference in multiple angles. Not having a proper reference for side-view makes it difficult to estimate heights. The walls seemed a little bit too high so the height was adjusted. After the adjustment the space between the building and the surrounding walls still seemed too narrow so the width of the walls was also reduced until it was more visually pleasing.

This is a good example on how to differentiate CAD modeling and visualization modeling. As this project is just about creating a visualization for the building, it is not as important to have the measurements as precise as in typical CAD project. Projects that are done with CAD most likely will also have more precise references and other detailed information of the building or other subject being modeled.

4.2.2 Med-poly

After the low-poly model started to have its final shape; and I was happy with it, I started working on the second model that would have a slightly higher amount of details. The first thing I changed was to give the stairs on both sides of the building some more detail. In the low-poly version I used just simple straight planes to represent the stairs, but they had no steps. In this version I added the steps and also created the large passageways to the building that lead into the inner yard.

The pillars in these passageways are the first objects that were created by using the array modifier (see image 22). Array modifier works so that first one object is created fully and then duplications of it are made with a modifier.

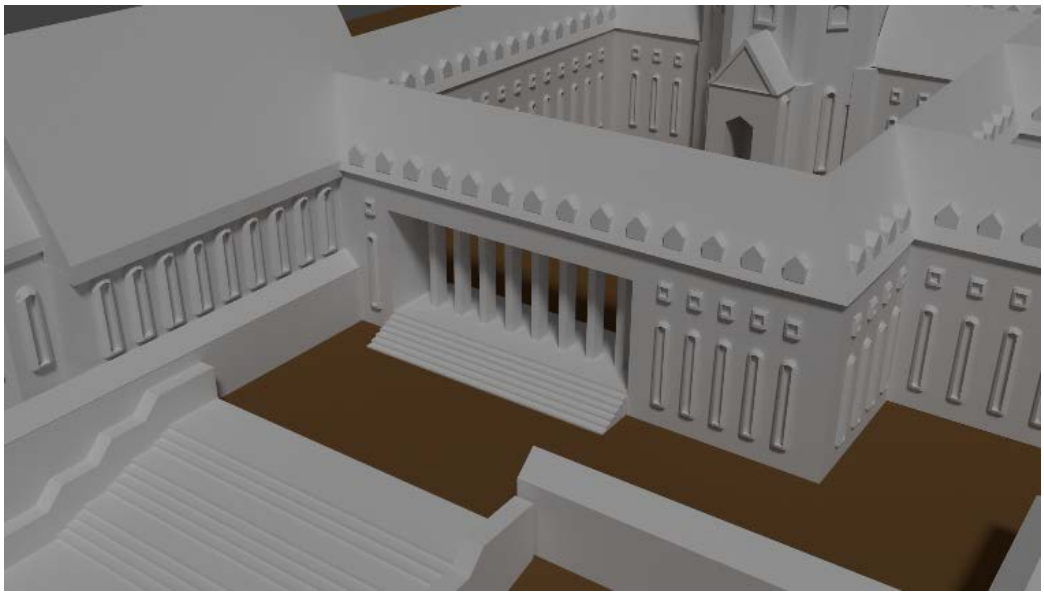


Image 22. Close up of med-poly model. Nykänen 2017.

From the modifier's settings I could choose how far apart each instance is from others and on which axis the separation occurs (see image 23). The arrays will act as a single object as the other instances will follow the parent object if it is moved or rotated. All modifications done to the parent mesh will be copied automatically to the array instances. However, if I would click apply on the array modifier settings all the instances would be separated in their own meshes inside the same object and editing them all at once would

not be possible anymore. Therefore, applying the modifier in this project is not necessary or advised, unless I would want to edit one specific instance of the object later on. (Blender official documentation 2016)

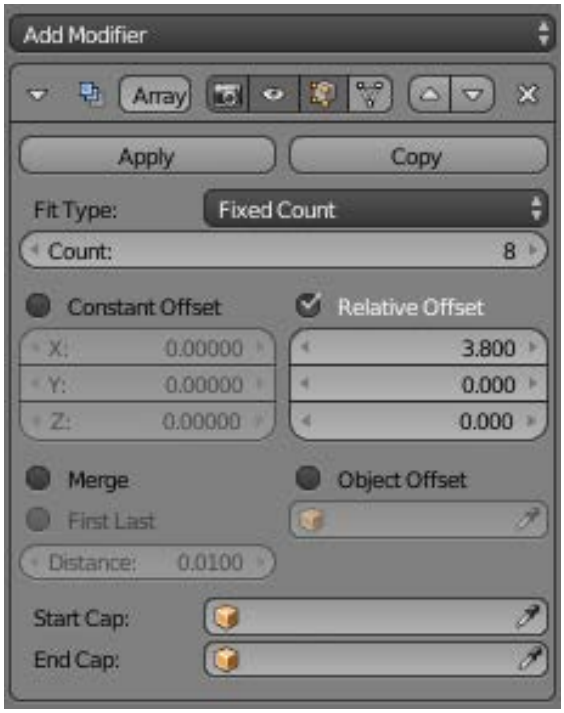


Image 23. Example of the array modifier menu. Nykänen 2016.

4.2.3 High-poly

The last version of the Kalevala building is the one with the most amount of details. This version includes heavy use of the array modifier that was explained in the previous chapter. This time also I started building the new version on top of the previous version as I did with the medium version also; as this saves a lot of time and ensures that the proportions of the building will stay the same.

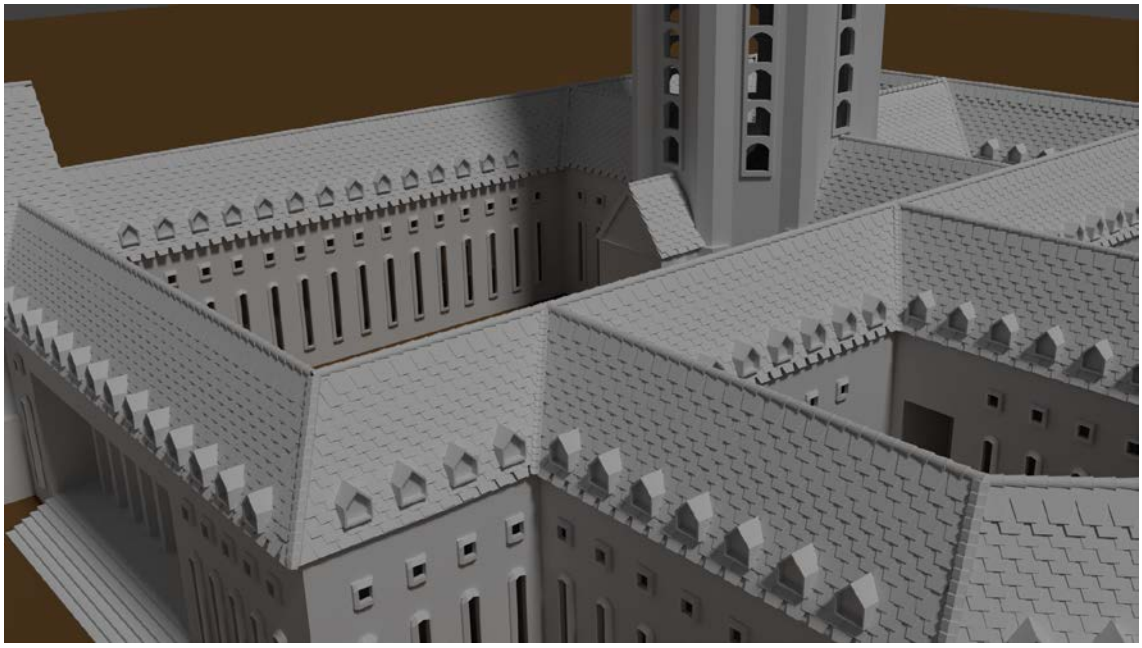


Image 24. Close up of high-poly model. Nykänen 2017

The largest part of work done in this model went into creating the tiling for the roof (see image 24). This is something that could be done easily with UV-maps by projecting a suitable tile texture to the roof. This would save in polygon amounts and make the model not as heavy, but in this project I decided to do everything by hand as it was discussed with the client that the models would be “raw” as in without any textures.

The tiling was done by first creating a “tileable” mesh of the roof tiles. Tileable mesh means that when copying the mesh for example with the array modifier, the copied instances can be placed next to each other so that there is no noticeable gap between then and they blend in well together to create a solid looking larger array (see image 25).

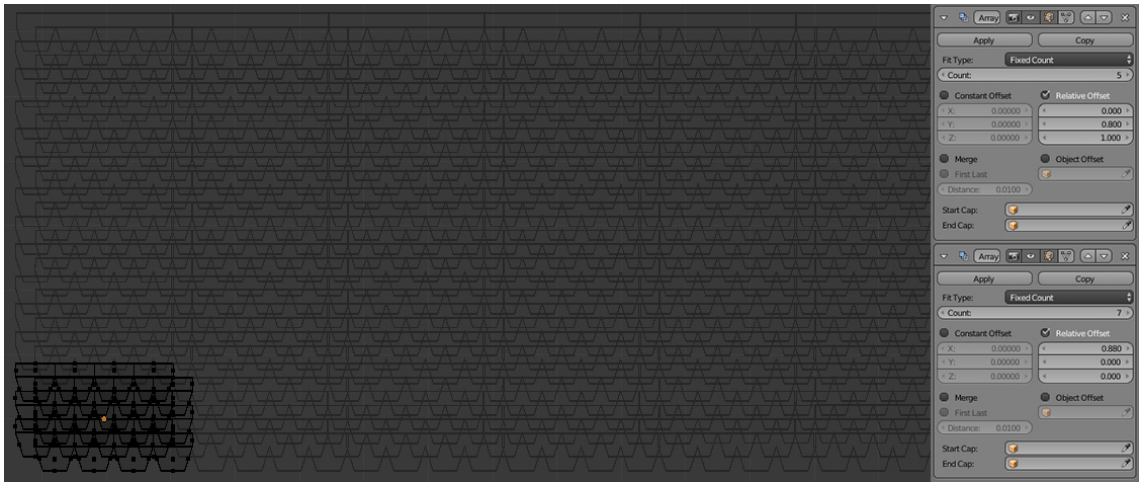


Image 25. Tileable roof. Nykänen 2016

The image above demonstrates the way how a tileable mesh was used to create one part of the roof in the model. In the lower left corner is the original mesh that was created so that if a copy of it is placed directly above it or next to it they will blend in together well. On the right side of the image can be seen the settings I used in the array modifier. For this part I first created five instances directly above the original mesh. After that I used another array to create seven instances of the original mesh on the right side of it. As this new array modifier is below the first modifier, it applies also to the original array allowing me to create this whole roof section easily. All the instances created with the array modifiers are shown in grey in the image.

As most of the different sections are not shaped as even squares, it required a lot more of tinkering than just copy-pasting tile arrays all around the roof. I did it by adjusting the arrays to each specific section and then applied both array modifiers in order to be able to edit it to fit that section properly. The arrays create a square roof but most sections actually have a diagonal edge on either side. My first solution was to delete and move the vertices in the edges by hand so that they would not protrude from the original roof. This was very time consuming and tedious work as well as sometimes the vertices would move slightly on the wrong axis making the end result look bad. Later I remembered that blender has a Boolean modifier that allowed me to easily cut parts of meshes and it will automatically create new vertices to the new edge (see image 26).

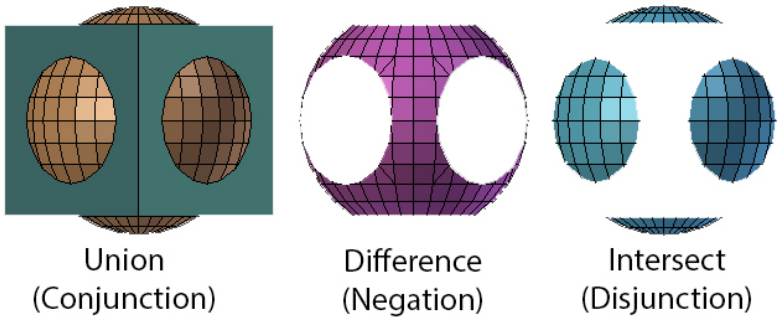


Image 26. Boolean modifier. Blender Foundation 2016.

The Boolean modifier is used to perform complex operations easier than editing the meshes manually. One of three different Boolean operators can be used to create a single mesh out of two separate meshes. First the mesh to be modified is selected and the Boolean modifier chosen from the modifier list. After that the target mesh, on which the Boolean operations will be based on, is selected. In the example image above a sphere is the modified object and a cube is selected as the target.

The union operation creates a mesh where the target mesh (cube) is added to the modified mesh (sphere). The difference operation subtracts the modified mesh from the target mesh; as in this case the parts of the sphere that are outside the cube are subtracted. And lastly the intersect operation subtracts the target mesh from the modified mesh. In this project I used solely the intersect operation since it allowed me to align a cube so that the unwanted part of the object being modified was inside the cube and would be deleted; and thus creating a new straight edge right where I wanted without the need to move vertices individually.

4.3 Comparison

Each version required approximately double the amount used in the previous model. All in all, the low-poly version most likely took me the longest time to finish, because I made several different iterations of it before I was satisfied with it to use it as a base for the other two models.

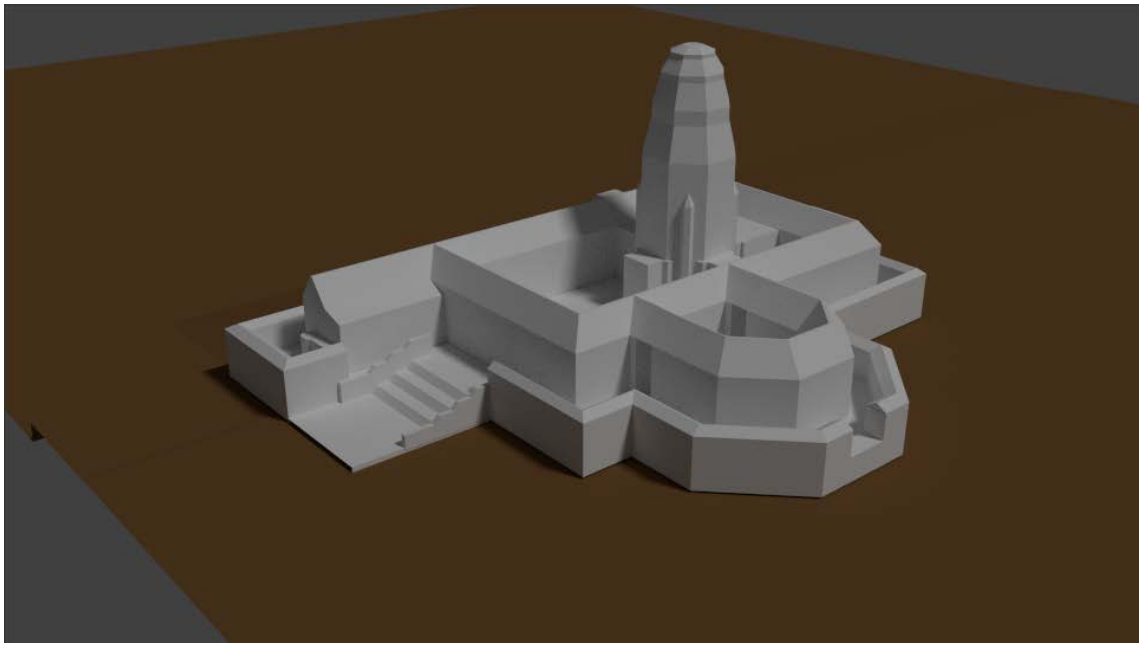


Image 27. Low-poly finished. Nykänen 2017

Image 28. Medium-poly finished. Nykänen 2017.

Image 29. High-poly finished. Nykänen 2017

5 Conclusion

Blender has not been viewed as a professional tool for creating 3D but rather as a small tool that hobbyists can play around with. The features that Blender is equipped with are more than enough for professional use. It is being updated actively by its users who are creating more and more content powerful tools that might have not yet even been featured in the bigger commercial 3D modelling software.

The biggest reason for these opinions have been the quality of the rendering engine in Blender. Cycles is now getting to the level of quality where it could be used for professional use. The Blender Foundation has initiated already several short movies that are done by Blender and funded with crowdfunding. These movies have brought Blender more to the surface in this industry and through each movie's production process Blender has seen huge advances in its features and capabilities. Also more and more of these external renderers like V-Ray have been getting stable add-ons for them to be used with Blender. These more professional level renderers are to tools that can help Blender to grow into a more professionally acknowledged 3D production tool.

Most of Blender's users will most likely still continue using Cycles as it is available for them from the start and it is getting better with every new build it gets. Cycles is a good all-around renderer but is prone to have a bit noisy results. With the release of Pixar's Renderman for Blender, most of the animation production done with Blender will most likely move towards using that since it is a very powerful unbiased renderer and nowadays, animation studios seem to be shifting more towards unbiased renderers.

References

1. Mullen, T. Mastering Blender. Hoboken, USA; 2010.

2. Brito, A. Blender 3D: Architecture, Buildings, and Scenery. Olton, Birmingham, GBR; 2008

3. Iraci, B. Blender Cycles: Lighting and Rendering Cookbook. Olton, Birmingham, GBR; 2013.

4. Stephenson, I. (2002). Essential Renderman. Poole, UK; 2002.

5. Baldi, D. (2015). Render Engine Comparison: Cycles vs the rest. [online] URL: http://www.blenderguru.com/articles/render-engine-comparison-cycles-vx-giants/. Accessed 16. January 2017.

6. V-Ray For Blender. V-Ray official documentation. [online] URL: http://docs.chaosgroup.com/display/VFBlender/V-Ray+for+Blender. Accessed 16. January 2017.

7. Mitsuba Renderer. Mitsuba official documentation. [online] URL: http://www.mitsuba-renderer.org/docs.html. Accessed 29. February 2016.

8. LuxRender official documentation. [online] URL: http://www.luxrender.net/wiki/Main_Page. Accessed 29. February 2016.

9. Cycles Renderer. Cycles official documentation. [online] URL: https://www.blender.org/manual/render/cycles/index.html. Accessed 29. February 2016.

10. Blender official documentation. [online] URL: https://www.blender.org/manual/getting_started/index.html. Accessed 29. February .2016.

11. Camera clipping. Blender official documentation. [online] URL: https://www.blender.org/manual/render/blender_render/camera/object_data.html#camera-clipping. Accessed 15. December 2016.

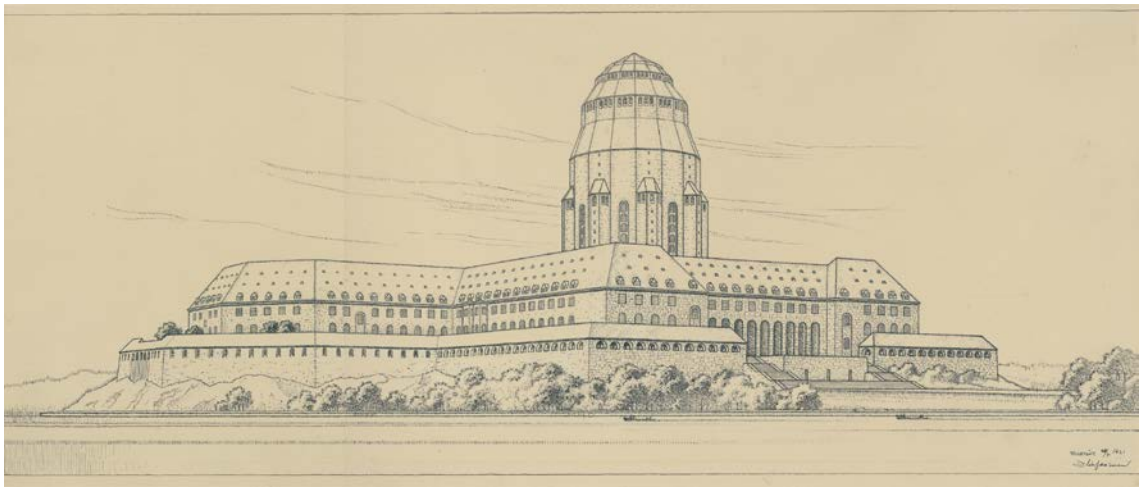
12. Blender: 3D content creation noob to pro. (2004). [PDF] URL: https://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro. Accessed 29. September 2016.

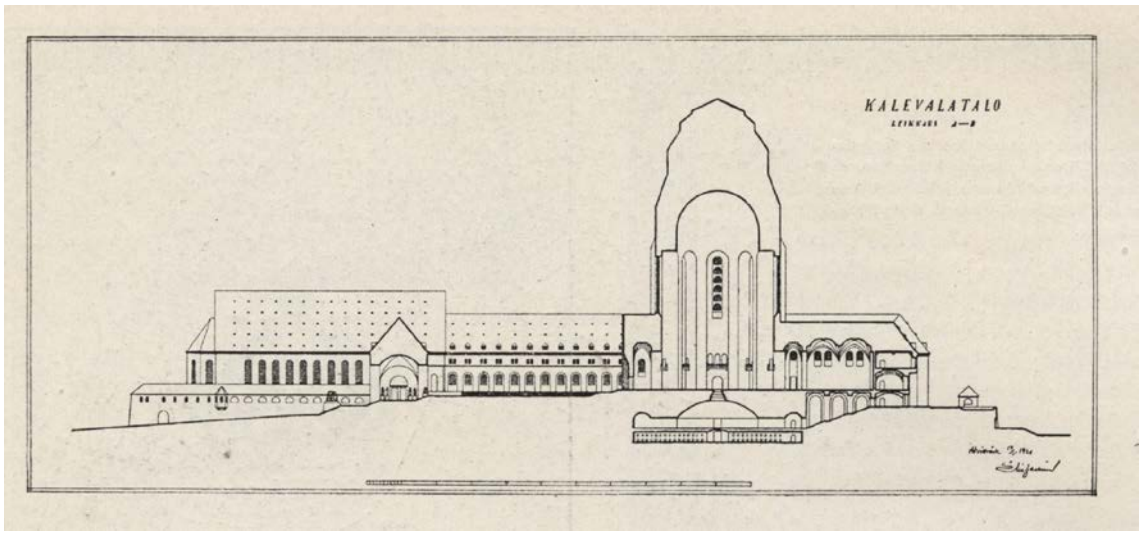
13. Baker, E. (2014). 5 Blender add-ons that streamline architecture workflow. [online] URL: Accessed 15. October 2016.

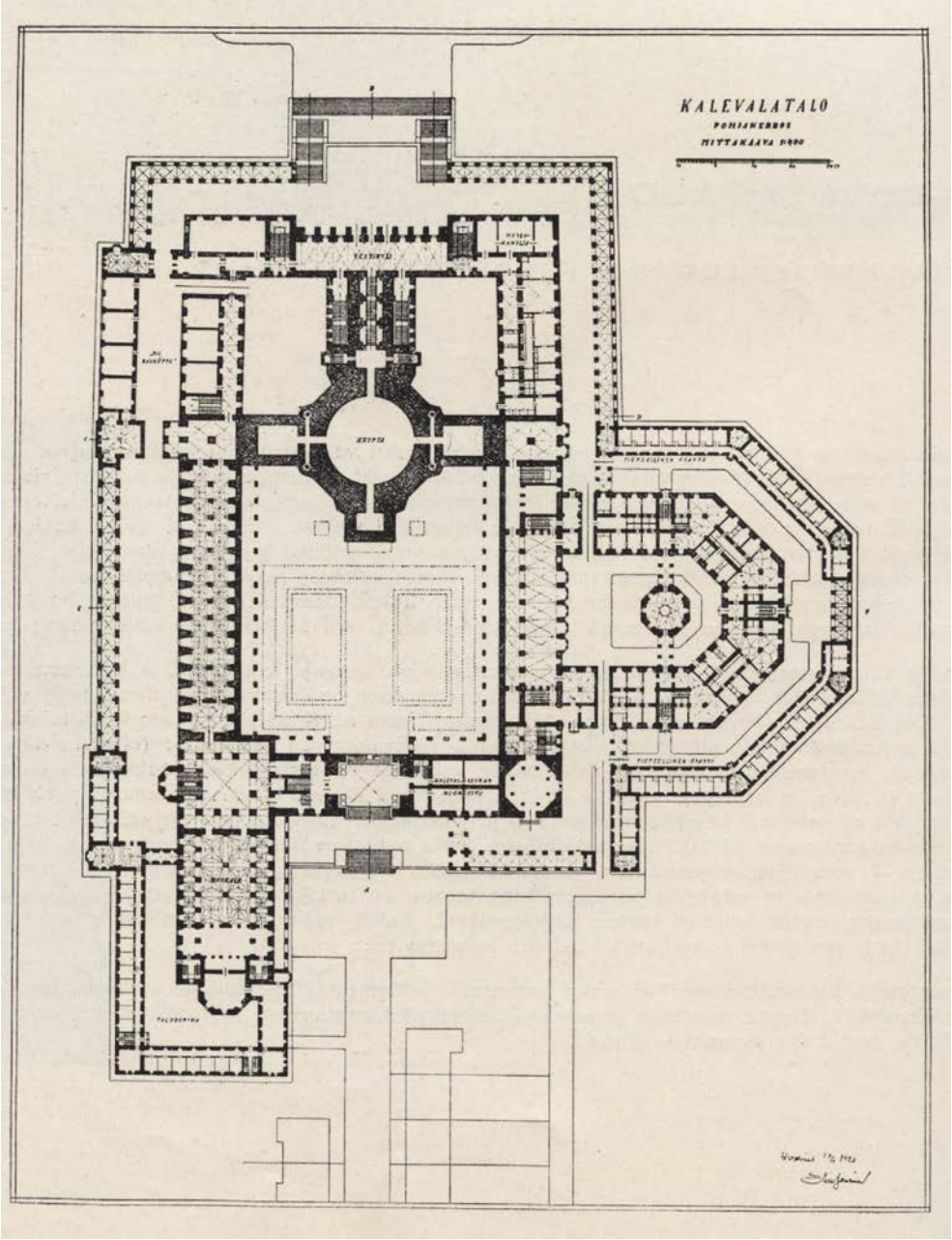
14. Cameron, V. (2012). What's the difference between biased and unbiased render engine? [online] URL: http://basic3dtraining.com/whats-the-difference-between-biased-and-unbiased-render-engine/. Accessed 29. February 2016.

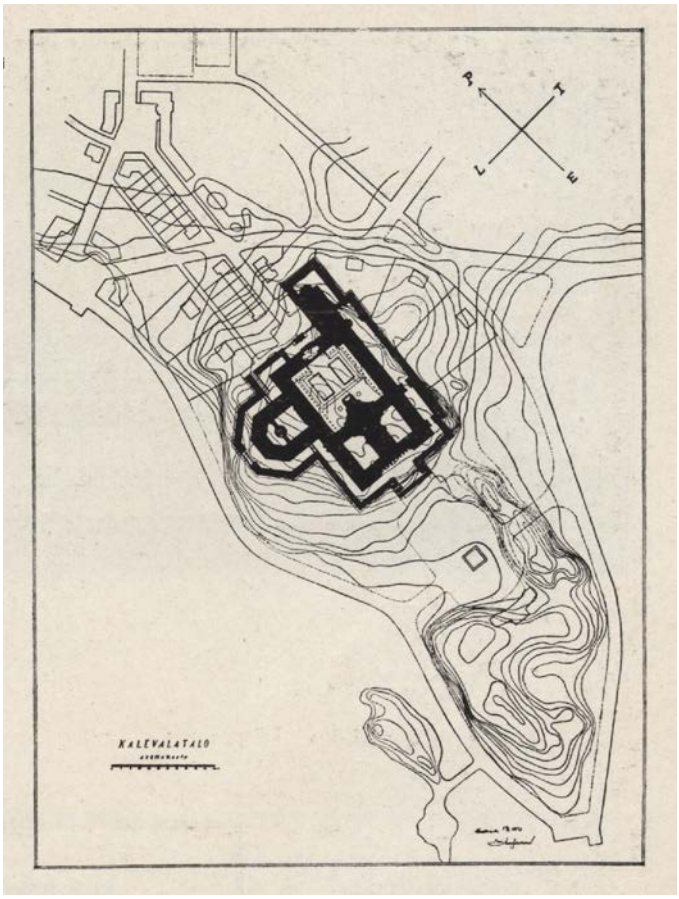
Appendices

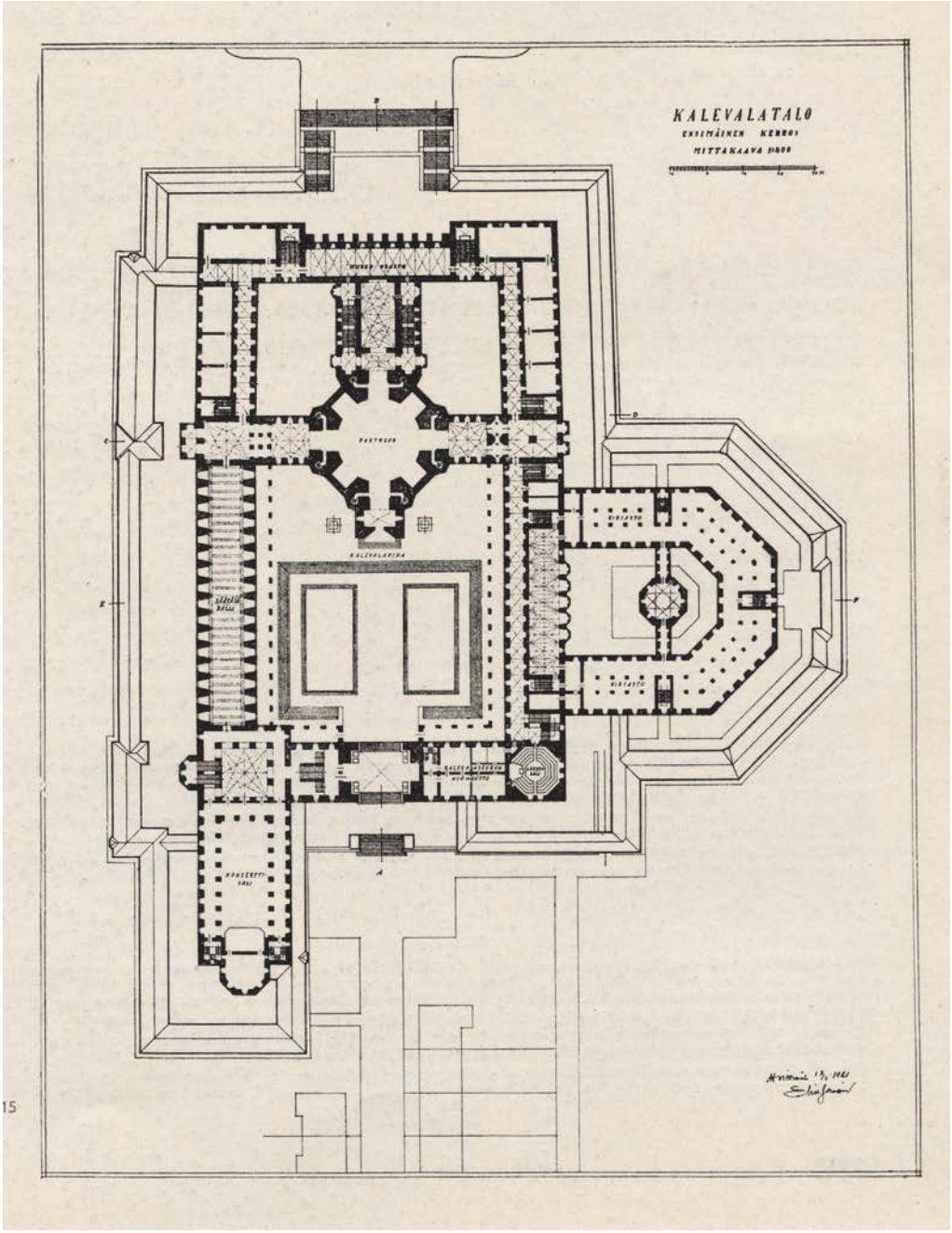
Kalevalatalo reference images

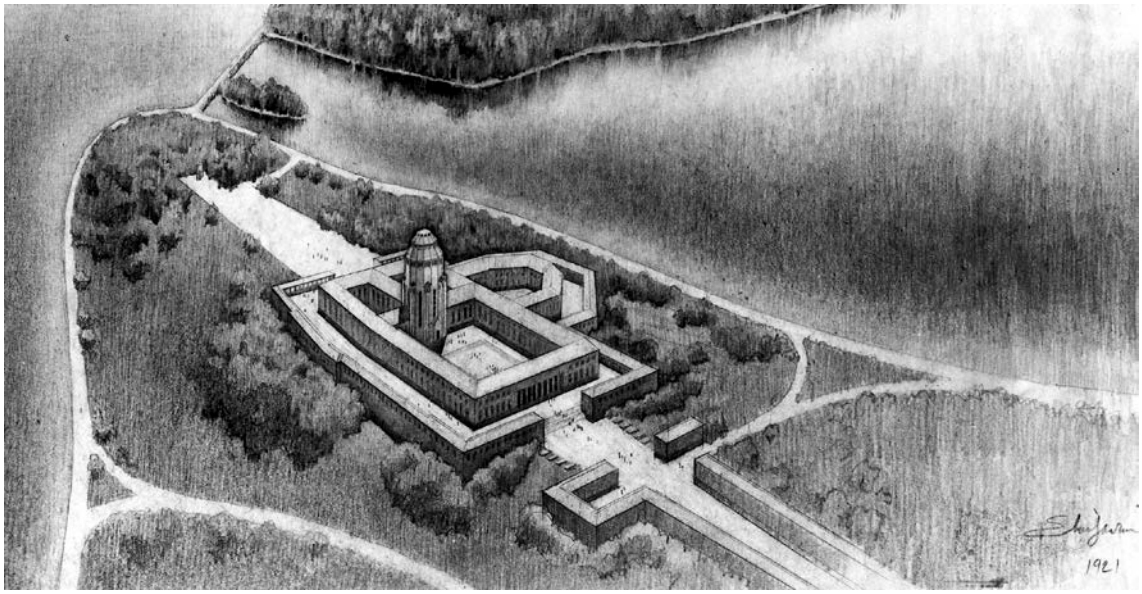







KALEVALATALO


KALEVALATALO
MITTAKAAVA 1:100


1921