A PRESENTATION TOOL FOR IKEA-STORES USING VIRTUAL REALITY BASED ON OPEN SOURCE

A Master's Thesis

by

Nils Hellstrand

Department of Design Sciences
Lund Institute of Technology,
Lund University, 2005

Preface

This master's project is part of a cooperation between RE-FLEX, the flexible reality centre at Lund Institute of Technology (LTH), and Inter IKEA in Helsingborg, Sweden.

The thesis is part of the requirements for a degree from a 4.5 year physical engineering program. As it's my personal conviction that good engineering not only requires knowledge about the technology, but also the people and the environment involved, I'm happy to present the key words in this thesis being; computer graphics, open source programming and usability, rather than simply physical engineering.
The work was mostly carried out at RE-FLEX, at Ingvar Kamprad DesignCentrum, Lund, with occasional visits to Inter IKEA in Helsingborg.

Supervisor: Dr Roy Davies
Commission assigner at Inter IKEA: Cenneth Bäckman.

I would like to thank Cenneth Bäckman and Mads Lundberg at Inter IKEA in Helsingborg for making it possible to study the non-academic, real world aspects of virtual reality, and for the interest and help given. I also want to thank Roy Davies and Joakim Eriksson at RE-FLEX for all the academic and technical support.

Lund, January 2005

Nils Hellstrand

Contents

Abstract

This project began with the question:
“Is virtual reality of interest when designing and presenting new IKEA-stores?”

The question was first formulated by the management at Inter IKEA (Helsingborg, Sweden) which is the division of the IKEA corporation that designs and presents new IKEA-stores to contractors all over the world. As a result, a cooperative scheme was introduced between Inter IKEA and RE-FLEX at Lund Institute of Technology (LTH). This master's project is the first product of this cooperation.

One objective in this master's project was to create an application for virtual reality that featured the entrance of an IKEA-store, another to establish (or at least try out) a method to design virtual environments representing IKEA stores. The method applied is to convert already existing CAD models of IKEA-stores using **open source** software to obtain the virtual environment; as a result much of this thesis concerns **OpenSceneGraph**, which is an open source software for virtual reality.

Inter IKEA provided the CAD model and 3D Studio Max was used to adjust and export the model into the OpenSceneGraph format. The result is a virtual reality software application featuring both a predefined model of the entrance to an IKEA-store and the ability to load any other future model. Interactivity is provided by the ability to explore the scene (using the keyboards arrow keys like in a 3D computer game) and to select and move specified objects in the scene. A model that has been changed using the supported interactivity can be saved. An animation path can be saved and will appear as a movie when loaded together with the model. The application has a user interface written in Visual Basic and can be run on both ordinary desktop computers or more advanced virtual reality hardware systems supporting stereovision.

This thesis discusses the challenges that appeared during the development of the model and application, and gives a small introduction to virtual reality, 3D computer graphics and usability. The final analysis concerns the pros and cons with using open source.

Sammanfattning på svenska

Det här examensarbetet har sitt ursprung i frågeställningen:
”Är virtual reality av något intresse när man ritar och pressenterar nya IKEA-varuhus?”

Frågan ställdes av ledningen på Inter IKEA i Helsingborg, vilket är den del av IKEA-koncernen som ritar och säljer nya IKEA-varuhus till olika agenturhållare världen över. Som svar på frågan inleddes ett samarbete med Reflex Reality Center vid Lunds Tekniska Högskola (LTH). Det här examensarbetet är det första resultatet av samarbetet.

Ett mål med det här examensarbetet var att göra en virtual reallity modell av entrén till ett IKEA-varuhus, ett annat var att ta fram en metod (eller i alla fall att testa en metod) för att åstadkomma virtual reality modeller av IKEA-varuhus. Metoden som använts i det här projektet går ut på att omvandla en redan befintlig CAD-modell av ett IKEA-varuhus till en virtual reality-modell, samt att använda **open source**-mjukvara för att programmera applikationen. Som ett led av detta har en stor del av den här rapporten ägnats åt **OpenSceneGraph** (vilket är just ett virtual reality-program baserat på open source).

Till det här projektet tillhandahöll Inter-IKEA en CAD-modell som kunde anpassas i 3DStudioMAX, och sedan exporteras till OpenSceneGraphs eget filformat. Resultatet är en virtual reality-applikation som innehåller en fördefinierad modell av entrén till ett IKEA-varuhus, men även ger möjligheten att visualisera eventuella framtida modeller. Applikationen är interaktiv så till vida att man kan använda tangentbordets piltangenter för att förflytta sig i modellen (på samma sätt som man vanligen förflyttar sig i 3D-datorspel), samt även markera och flytta på fördefinierade objekt. Flyttar man på objekt i modellen kan resultatet sparas, det går även att spara en ”animation path”, vilken upplevs som en film om den exekveras tillsammans med modellen. Applikationen har ett användargränssnitt skrivet i Visual Basic och kan köras på både vanliga persondatorer och på mer avancerad virtual reality-hårdvara som klarar stereoseende.

Rapporten tar upp utmaningar och problem som uppstått under utvecklandet av modellen och applikationen, samt behandlar kortfattat virtual reality, 3D-datorgrafik och användarvänlighet. Den slutliga diskussionen behandlar för- och nackdelar med open source.

1 Introduction

This chapter gives a short background to the project.

1.1 Presentation of the problem

Being a worldwide furniture corporation, IKEA needs to spread its concept and developed service image to many partners on a regular basis. Among temporary partners are architects and contractors designing and building new stores. The usual way to communicate with this group of professionals is through ordinary drawings, printings and texts. The need for a better way to communicate building concepts has been pronounced from the IKEA headquarter. This project aims to present a possible solution using virtual reality as an information/impression transmitter to these new partners.

In recent years there has been an explosive development in inexpensive technologies for computer graphics and display devices, and new application areas for these technologies steadily develop. From having been found only in research laboratories or occasional show-off investments by big companies, 3D accelerated graphic cards and projectors have become standard office equipment in many companies. But having access to this new hardware is not enough to integrate it in everyday work; one needs to know both how and when to use it, which includes knowledge about both the technology and the goal that it should help to achieve. The aim of this thesis has been not only to present a software solution to the particular problem but also to study, through this case, how best to transfer know-how about Virtual Environments (VEs). Another important aspect is the use of open source software and the pros and cons of this.

1.2 Presentation of “one room” and the given task

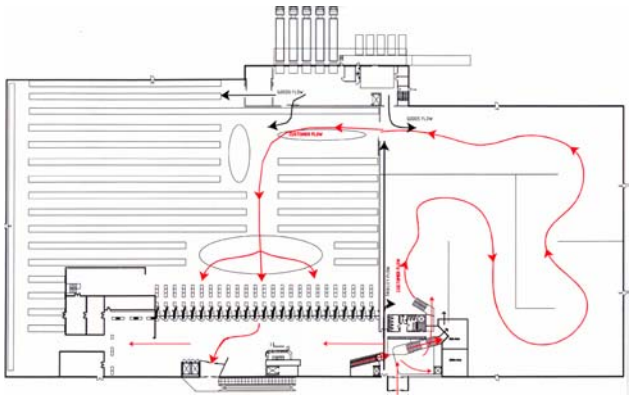
The IKEA project “one store” is an ambition to develop and visualize the “ultimate” IKEA store as a guideline for new buildings. For practical reasons, such as the specific plot, scale and national laws, it is not possible to build exactly identical stores all over the world, but the aim to make the customer recognize the IKEA concept independently of setting is important to strengthen the trademark. There are also economical big scale advantages when ordering building materials for say ten new stores instead of one at a time.

The IKEA project “one room” is part of the bigger “one store” project and aims to describe the “ultimate entrance/exit” to the “ultimate IKEA-store”. The entrance/exit is of great importance as it is the customer’s first, and last, contact with the whole store. The entrance/exit is also the connection between the store and its surroundings.

The current existing presentation forms for the “one room” project were rendered computer images, ordinary drawings, PowerPoint presentations and descriptions of

which impressions the environment should make (see Fig. 1, 2 and 3). Unfortunately the level of transmitted information and "overall feeling" had proved not to be satisfactory.

The given task was to explore the possibilities of using Virtual Environments as a way to communicate the ideas of the one room project.



Figures 1, 2 and 3: Examples from the original printed documentation of the One Room project.

1.3 Limitations

As VE is a relatively new and fast growing field (due to the explosion in hardware development), commercial application is a relatively unexplored area with many possibilities. As this was a one-term university thesis several limitations had to be applied, one limit being the number of user scenarios considered.

VE can be a fantastic tool when designing new objects and environments due to the immediate feedback of changes. However, the first defined problem for the project was to find a better way to communicate already designed environments. As a consequence the limit of interaction with the one room model was set to the ability to move the interior and exchange it, but not to reshape it within the program. The building itself was considered static.

Another limitation was the hardware. Even if expensive high frequency projectors, motion detectors and graphic cards are available in many laboratories all over the world, they are not likely to be found in the average IKEA conference room (Fig. 4). A restriction to an average laptop computer and LCD/DLP projectors was made. Due to the capacity of modern graphic cards and processors the last restriction was not as tough as one might think.

Figure 4: The thought of user environment was set to an average conference room, in contrast to a dedicated virtual reality show room. Picture taken from http://globalecology.stanford.edu.

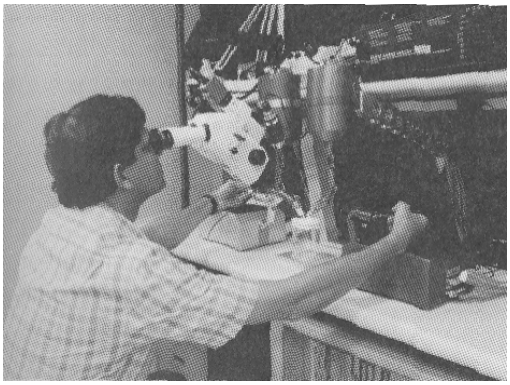
2 Background theory

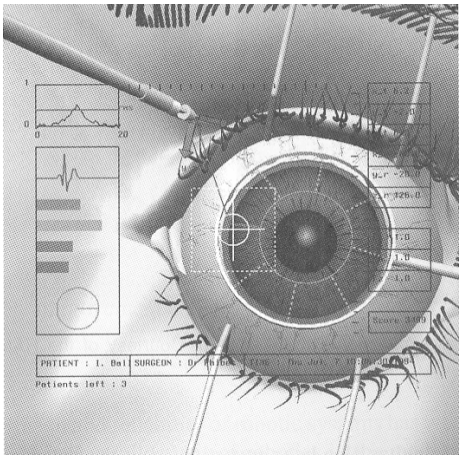
This chapter is intended to present a theoretical framework for the project. First comes a short introduction to virtual environments, a more technical introduction to computer graphics in general is given in the section "3.1 introduction to 3D modelling and computer graphics" in the next chapter. The rest of this chapter describes theories and methods that were suitable for the project.

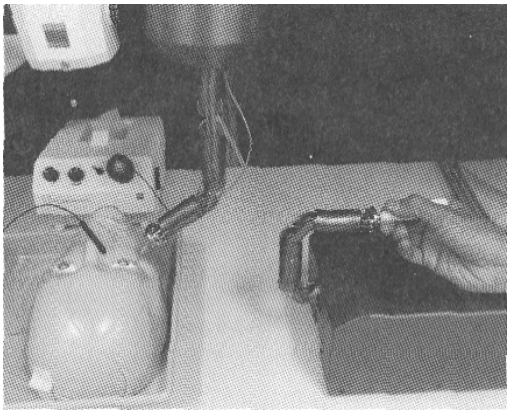
2.1 Introduction to virtual environments

Virtual reality (VR) aims to give a multimodal feeling of being in another world, place or situation, and to be able to interact with its environment, a virtual environment (VE). For the interested reader "Handbook of Virtual Environments" (Stanney, 2002) can be recommended.

Sometimes there are advantages with using a virtual environment instead of the real environment. Such cases can be training specific tasks in a dangerous environment or simulation of a not yet existing environment. VE is also useful in rehabilitation and treatment of psychological problems, such as fear of flying, acrophobia (fear of heights), fear of public speaking etc (North *et al*, 2002). A great advantage of VE is that it can be independent of the user's location and thus connect people (or people and machines) at different locations in a shared, or personal, multimodal environment. Teleportation using a haptic device (see Fig. 5, 6 and 7), can serve as an example (Kheddar *et al*, 2002).







Figures 5, 6 and 7: The surgeon receives force feedback when he moves the handle representing the scalpel, this is the essence of a haptic device. In the binoculars he can see a real time updated virtual image of the eye.

Pictures from Kheddar *et al,* (2002), page 981-982.

There are numerous different kinds of VR solutions, spanning over a broad range of hardware costs. Essentially it is a question of how many, or which, of the human senses one wants to support and how well this should be done; haptic devices are used to give force feedback to the user, stereo vision can be created for example with a head mounted display, perhaps connected to a tracking system, and 3D sound can be achieved using a multi-channel surround sound system. One of the more cost effective variants is "desktop VR" where an ordinary desktop computer is used and the VE is simulated with 3D computer graphics.

2.2 Introduction to usability

This project originated from a specific task and included an attempt to a technical solution in the form of a product; this meant that the end user had to be considered. The word usability is often used when describing the support a system, or product, gives a user as he or she tries to accomplish a task.
One definition of usability is "a measure of the ease with which a system can be learned or used, its safety, effectiveness and efficiency, and the attitude of its users towards it" (Preece *et al*, 1994). Dix *et al* (1997) however refers to the international standard ISO 9241, entitled *Ergonomic requirements for office work with visual display terminals (VDT)s*, which includes one part that applies equally to both hardware and software design. In the beginning of that document the following definition of usability is given:

Usability: The effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments.

Effectiveness: The accuracy and completeness with which specified users can achieve specified goals in particular environments.

Efficiency: The resources expended in relation to the accuracy and completeness of goals achieved.

Satisfaction: The comfort and acceptability of the work system to its users and other people affected by its use.

In order to support usability when *designing* an interactive system Dix *et al* (1997) propose the attention to three general principles:

Learnability: the ease with which users can begin effective interaction and achieve maximal performance.

Flexibility: the multiplicity of ways the user and system exchange information.

Robustness: the level of support provided to the user in determining successful achievement and assessment of goals.

Dix *et al*. then break down each of these principles into further principles that affect them:

(Tables exactly referred)

Principles affecting **learnability:**

Principle	Definition
Predictability	Support for the user to determine the effect of future actions based on past interaction history
Synthesizability	Support for the user to assess the effect of past operations on the current state
Familiarity	The extent to which a user's knowledge and experience in other real-world or computer-based domains can be applied when interacting with a new system
Generalizability	Support for the user to extend knowledge of specific interaction within and across applications to other similar situations
Consistency	Likeness in input-output behavior arising from similar situations or similar task objectives

Table 1.

Principles affecting **flexibility:**

Principle	Definition
Dialog initiative	Allowing the user freedom from artificial constraints on the input dialog imposed by the system
Multi-threading	Ability of the system to support user interaction pertaining to more than one task at a time
Task migratability	The ability to pass control for the execution of a given task so that it becomes either internalized by user or system or shared between them
Substitutivity	Allowing equivalent values of input and output to be arbitrarily substituted for each other
Customizability	Modifiability of the user interface by the user or the system

Table 2.

Principles affecting **robustness:**

Principle	Definition
Observability	Ability of the user to evaluate the internal state of the system from its perceivable representation
Recoverability	Ability of the user to take corrective action once an error has been recognized
Responsiveness	How the user perceives the rate of communication with the system
Task conformance	The degree to which the system service support all of the tasks the user wishes to perform and in the way that the user understands them

Table 3.

2.3 Introduction to user-centred design

A method called user-centred design is often utilised to support usability when designing a system or product. "User-centred design is a development methodology that focuses on people, their work and environment, and how available technology can be best applied to support them in this context" (Preece, 1994). Thus, it is the user and his or her needs that are in focus rather than the actual product. Naturally, human cognitive factors (such as perception, memory, learning, problem solving, etc.) play an important role in the development process. The main objective of user-centred design is to produce systems that are easy to learn and use, but also systems that are

effective in terms of performance and maintenance. To require information about the user and their needs, different methods can be applied. Apart from just asking or observing the user, methods such as brainstorming, workshops and advanced role-plays can be used to extract the problems the user might face in the future system (Dix *et al*, 1997).

2.4 Iterative design and prototyping

A problem when designing interactive systems is that the requirements cannot be completely specified from the beginning of the design process. The only way to be sure about some features of the potential design is to build them and test them out on real users (Dix *et al*, 1997). If the design is modified to correct any false assumptions that were revealed in the testing, hopefully a better design will be achieved. "This is the essence of *iterative design*, a purposeful design process which tries to overcome the inherent problems of incomplete requirements specification by cycling through several designs, incrementally improving upon the final product with each pass" (Dix *et al*, 1997).
These problems in a design process, which can lead to an iterative philosophy, are not unique to the usability features of an intended system. The problem holds for requirements specification in general, and so it is a general software engineering problem, together with technical and managerial issues.
On the technical side, iterative design is described by use of *prototypes*, artefacts that simulate or animate some but not all features of the intended system. Dix *et al* (1997) categorize three main approaches to prototyping:

Throwaway: The prototype is built and tested. The design knowledge gained from this exercise is used to build the final product, but the actual prototype is discarded.

Incremental: The final product is built as separate components, one at a time. There is no overall design for the final system, but it is partitioned into independent and smaller components. The final product is then released as a series of products, each subsequent release including one more component.

Evolutionary: Here the prototype is not discarded and serves as the basis for the next iteration of design. In this case, the actual product is seen as evolving from a very limited initial version to its final release.

Even if the end product in this project was most likely going to be more of a prototype itself, rather than a release product, the use of some sort of prototype probably had to be considered. The theories described above in this chapter form a basis for the methods and procedures in this thesis.

3 Background for the implementation process

This chapter aims to give the reader the necessary vocabulary and technical background required to understand the next chapter, concerning the actual implementation of the model. For the interested reader it's free to initially skip the first section (3.1 Introduction to 3D modelling and computer graphics) and only use it as a glossary when needed.

3.1 Introduction to 3D modelling and computer graphics

3.1.1 Polygons

Two points in a coordinate system can represent the endpoints of a straight line segment. Two line segments can have one point in common and thus be connected and form a polyline (see Fig. 8). Computer graphics work this way to define lines; a curved line is usually represented with a polyline where the points are located very close to each other to make the line look smoothly curved and not angular.

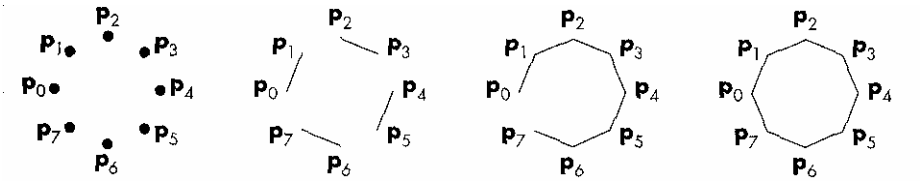


Figure 8, taken from Angel, 2003 page 49.

Line segments and polylines can model the edges of objects, but closed objects also may have interiors (see Fig. 9). Usually the name polygon is reserved to refer to an object that has a border that can be described by a line loop, but has an interior. Polygons play a special role in computer graphics because they can be displayed rapidly and be used to approximate curved surfaces. The performance of graphics systems is measured by the number of polygons per second that can be displayed. (Angel, 2003)

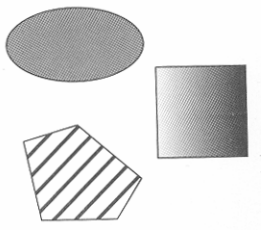
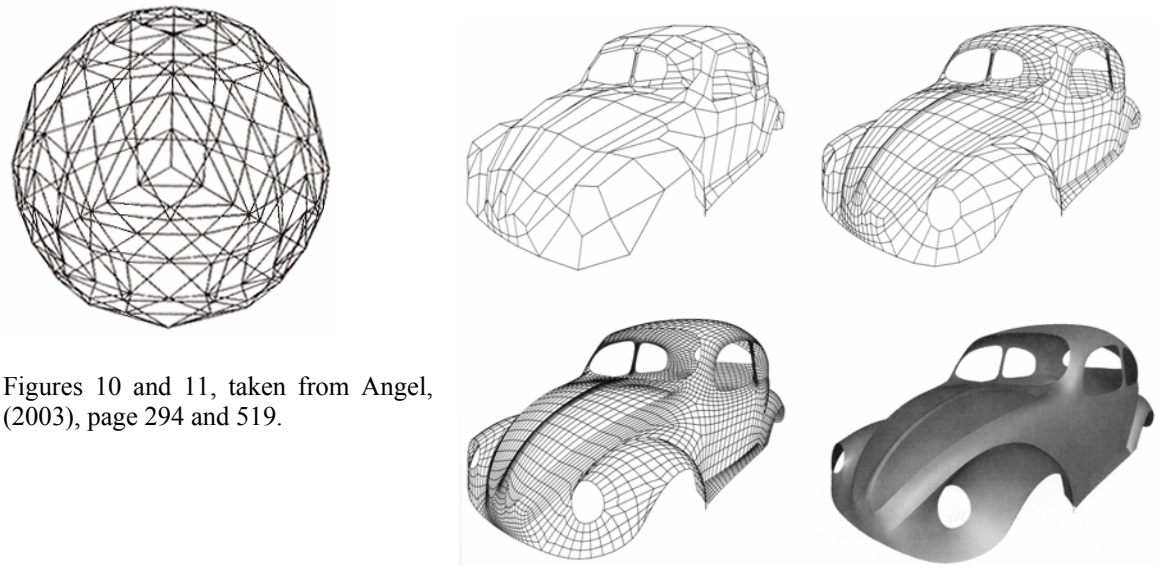


Figure 9, taken from Angel, 2003 page 50.

Usually in 3D modelling all objects are subdivided into triangles or rectangles, which form surfaces that build up the object (see Fig. 10 and 11).



Figures 10 and 11, taken from Angel, (2003), page 294 and 519.

3.1.2 Texturing

As all 3D objects in computer graphics are built up of flat surfaces, usually triangular shaped, it's quite easy to attach an image to the object. The Image can be artificial or originate from a real photograph. Different methods to decide exactly how each pixel on the display device shall be rendered can be used, but they all relate to some sort of map that describes the original image (see Fig. 12). Texturing can be a powerful way to induce references to the real world.

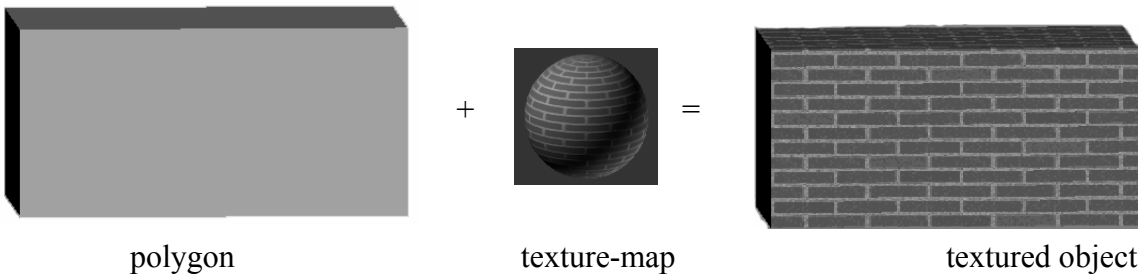


Figure 12.

3.1.3 Level of detail

Level of detail (LOD) is pretty much what it sounds like, it's a measurement of how detailed the scene is represented in the graphics, compared to reality. An important aspect of LOD is Level-of-Detail Management:
"Level-of-detail management, or distancing, is a widely used technique, whereby if a virtual object is more than a specified distance from the viewpoint of the participant it is replaced by a simpler model or else disappears altogether, thus reducing the amount of processing time required to render that object" (e.g., Rafferty *et al,* 1998).

LOD can refer both to geometry aspects, such as to replace a complex object with a box, and to texturing aspects such as using a plain colour instead of the texture when viewed from far away.

3.1.4 Skybox

Skyboxes are used to induce a sense of space and skylight. Normally a skybox consists of a half sphere that is textured on the inside portraying the sky (see Fig. 13), the half sphere is made big enough to cover the whole model as a lid. The image that is used for the texture has to be modified in a way that it fits the inside of the sphere without any visible artefacts.

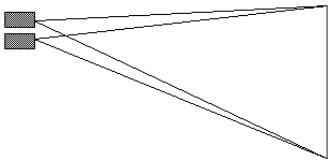
Figure 13: Skybox based on photographs taken from the roof of AF-borgen in Lund, Ludvig Ljungqvist 2003.

3.1.5 Stereovision

Stereovision is not something that necessarily has anything to do with 3D modelling or computer graphics; normally we use it in our daily life without even thinking of it. The basic idea is that if your brain gets visual input from two eyes, and knows the distance between them, it can compare the two images and calculate information about the depth in the viewed scene. However if the viewed scene is two dimensional, as when projected on a screen, the images will be identical. To benefit by stereovision, when viewing a 2D display device, one has to produce two different images and separate them to each eye.

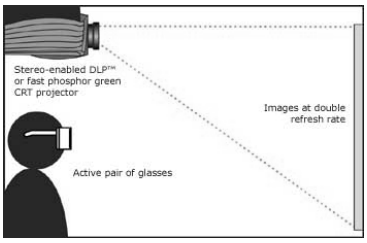
Computers can do the computation of the two different images, portraying a 3D scene from two viewing positions separated by the distance between your eyes. The separation of the two images to your eyes, when viewing a computer screen or a projected image, is usually done with glasses. Three methods are mainly used:

- Two projectors can be used to produce the two images, but with different polarization of the light. If the glasses contain polarization filters, one for each eye, it is possible to decide which projected image each eye should see (see Fig. 14 and 15). This method is suitable when the system is intended for many viewers, as the glasses are of relatively low cost. The method is often referred to as *passive stereo*.



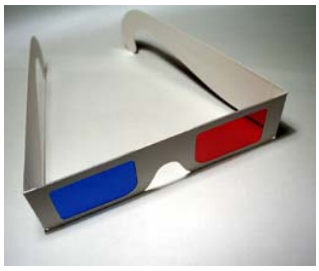
Figures 14 and 15. Pictures from www.visbox.com and http://astronomy.swin.edu.au, 2005.

- Another way to separate the images is to blacken out each eye at a time; using semi transparent LCD shutters in the glasses (see Fig. 17). This method is more suitable when few viewers are intended, as the glasses are considerably more expensive. On the other hand one projector alone can be used to project the two images, synced and with the same frequency that the glasses switches between the eyes. As one frame of the original material is projected two times using this method, the projector needs to accomplish double the frame rate relative to the frame rate of the original material. This method is called *active stereo*.



Figures 16 and 17. Pictures from www.barco.com, 2005.

- The last, kind of sneaky way to view stereo material, is to separate the images using colour filters, often red and bluegreen (Fig. 18). The two images then have to be rendered in either red or bluegreen tones (Fig. 19), which of course reduces the quality significantly. Nevertheless this is a cheap and easy way to achieve a stereo effect. *This method is called anaglyphic stereo.*



Figures 18 and 19. Pictures from www.crystalmaker.co.uk and www.bigbug.com, 2005.

3.2 Presentation of software suitable for the task

3.2.1 OpenGL

The interface between an application program and a graphics system can be specified through a set of functions that resides in a graphics library. These specifications are called the **application programmer's interface (API)** (Angel, 2003). OpenGL is such a graphics software system, or API, and has become a widely accepted standard for developing graphics applications. OpenGL is very fundamental and close to the hardware, a more user-friendly, high-end interface, on top of OpenGL is desirable for developing more detailed 3D applications. OpenGL is not object-oriented but more of a "state" machine, which means that an object cannot be accessed directly. Instead of telling the system to raise one hand of a rendered robot, one has to tell the system to erase the hand and draw it again in the new position.

3.2.2 OpenSceneGraph (OSG)

"The OpenSceneGraph is a portable, high level graphics toolkit for the development of high performance graphics applications such as flight simulators, games, virtual reality or scientific visualization. Providing an object-oriented framework on top of OpenGL, it frees the developer from implementing and optimizing low-level graphics calls, and provides many additional utilities for rapid development of graphics applications." (OpenSceneGraph, 2004).

In short OSG helps you keep track of your viewing position, and different 3D objects, in the world of a real-time application like a computer game. OSG can also compute the right images for stereovision when the hardware is supported.

One of the most important aspects of OSG is that it is open source, which means that you don't have to pay for a licence and that you are free to modify it any way you want. The OSG project was once started as a hobby by Don Burns in 1998, aimed to be used when programming hang gliding simulators, but has since gained ground in both the commercial and scientific world.

OSG is written and handled in C++, which means that normally no graphical interface is used. This requires that the user possesses some fundamental C++

programming skills. The downside is that this limits the number of possible users, and the learning process of OSG is quite substantial, the up side is that once you master the OSG fundamentals you can quickly accomplish pretty much what ever you want and have total control over it.

3.2.3 What is a Scene Graph?

"It's a tree! Quite simply one of the best and most reusable data structures invented. Typically drawn schematically with the root at the top, leaves at the bottom. It all starts with a top-most root node which encompasses your whole virtual world, be it 2D or 3D. The world is then broken down into a hierarchy of nodes representing either spatial groupings of objects, settings of the position of objects, animations of objects, or definitions of logical relationships between objects such as those to manage the various states of a traffic light. The leaves of the graph represent the physical objects themselves, the drawable geometry and their material properties. " (OpenSceneGraph, 2004). Also see Fig. 19.

The modules/libraries that OSG is composed of are:

- osg - *core scene graph*
- osgUtil - *utility library for useful operations and traversers*
- osgDB - *plugin support library for managing the dynamic plugins - both loaders and NodeKits*
- osgText - *NodeKit which adds support for TrueType text rendering*
- osgParticle - *NodeKit which adds support for particle systems*
- osgPlugins - *28 plugins for reading and writing images and 3D databases*
- osgGA - *GUI adapter library - to assist development of viewers*
- osgGLUT - *GLUT viewer base class*

Important nodes in OSG are the osg::Drawables which contains the actual 3D object and osg::Transform which contains a matrix that tells where in the world (relative to the origin) the object is located.

A typical structure of the scene graph in memory can be:

- osg::Group at the top containing the whole graph
- LOD's, Transform, Switches in the middle
- osg::Geode/Billboard Nodes are the leaf nodes which contain...
- osg::Drawables which are leaves that contain the geometry and can be drawn.
- osg::StateSets attached to Nodes and Drawables, state inherits from parents only.

To create the objects that should be stored in the scene graph one can use a 3D modelling software program, like 3DstudioMAX, that has an exporter to the OSG format.

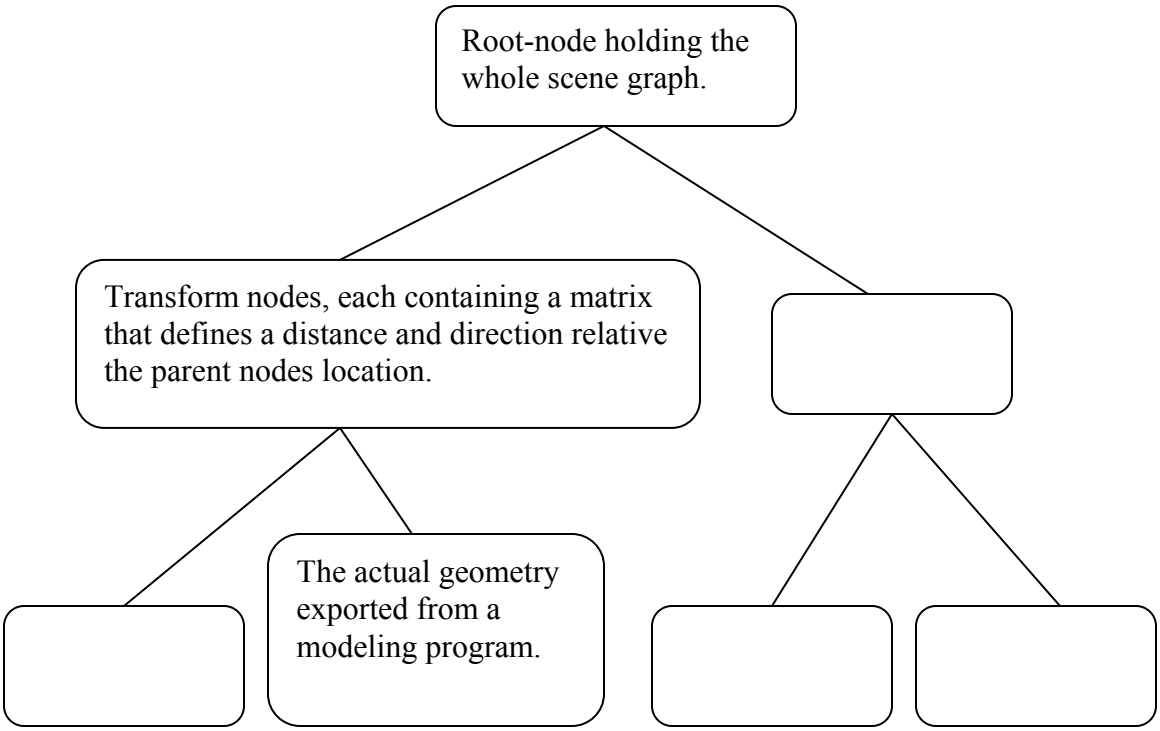


Figure 19 displaying the three main levels of a scene graph tree.

"A scene graph isn't a complete game or simulation engine, although it may be one of the main components of such an engine; it's primary focus is representation of your 3d worlds, and efficient rendering thereof. Physics models, collision detection and audio are left to other development libraries that a user will integrate with. The fact that scene graphs don't typically integrate all these features is actually a really good thing: it aids interoperability with clients' own applications and tools and allows it to serve many varied markets from games, visual simulation, virtual reality, scientific and commercial visualization, training through to modeling programs." (OpenSceneGraph, 2004)

OSG consists of a number of node kits, and libraries of functions that can be used on these nodes. It's up to the user to write an executable program using these libraries. In the additional node kit osgProducer a standard **OSG-viewer** module is provided. This viewer can be used to view the scene by traversing the scene graph. The standard viewer holds some special features such as saving the current scene graph tree to a file called **saved_model.osg**, and to save an animation path through the scene to a file called **saved_animation.path**. If this animation path is loaded back into the viewer together with the right model (scene graph) it will appear as a movie.

3.2.4 3DStudioMAX

3DStudioMAX (3DSMAX) by Discreet is probably the most used and well-known 3D modelling software (along with Maya by Alias). 3DSMAX provides good features for creating, twisting, bending and overall modifying of 3D objects. It also has a good material editor for texturing objects. 3DSMAX can also be used to arrange light sources in the scene and render pictures or movies. It's important though to distinguish this kind of time consuming rendering, where the appropriate shadows are computed and a 2D image is saved for future display, from the real-time rendering that takes place in VR or a computer game.

3DSMAX uses the 3DS format but models of various different other formats, including CAD and DWG, can be loaded into the program. In the process the file is converted into the 3DS format, which sometimes causes some information loss.

3.2.5 AutoCAD

AutoCAD by AutoDesk is the most well-known and used CAD program on the market. The application is mostly used for construction development and drawing of construction details. AutoCAD was initially intended for 2D drawings, even if it nowadays supports 3D modelling. IKEA uses a tailor-made version of a CAD program called AUTO DECO.

3.2.6 WorldUp

WorldUp by Sense8 is just like OSG a VR visual programming platform. WorldUp is a licensed product and possesses a graphical user interface; it also supports stereovision and importing models of the 3DS format.

When this project was started WorldUp was the mainly used VR-software at the research centre.

3.2.7 EON

EON is another commercially available software solution for 3D VR application, but was not available at the research centre when this project started.

4 Methods and Procedure

This chapter describes the actual implementation process of the model, first comes a more general description of the applied methods and theory. Then comes the results section, which is based on some key points in the process of developing the model.

4.1 Applied methods and theory

In order to develop any product, two major activities have to be undertaken: the designer must understand the requirements of the product, and must develop the product. Understanding requirements involves looking at similar products, discussing the needs of the people who will use the product, and analyzing any existing systems to discover the problems with current designs. Development may include producing a variety of representations until a suitable artefact is produced (Preece, 1994).

The aim was to apply user centred design in order to produce a VE that supported the specific information needed to describe the store entrance in a satisfying way. The information about the building itself was pretty much predefined as the original material was a CAD model and rendered drawings. The question was how this information best could be presented in a VE, and how the experience of the material presented this way compared to the usual way through rendered drawings. An obvious difference is that a VE can support different levels of interaction, such as to move objects in the scene, but the ability to choose your own viewing angle can in itself be very important for the perception of space. As these issues strongly connect to the specific model, and the situation when the application is used, user centred design was a desirable method to use.

Another aspect of applying user-centred design was the attempt to establish a method, or working scenario, for accomplishing VEs out of future building models. Even if this specific project was about one room and the entrance, it can also be seen as a pilot case about incorporating VR in the process of developing new IKEA-stores. Of course this meant that the ordinary working scenario had to be known.

The main methods used to gather information about the user in this project has been brainstorming and storyboarding. Storyboarding in this case means putting up a scenario in the context of the user's everyday work where the thought of system could have a role. As time is a big issue when the incorporated user is a fulltime working person with other obligations, these methods were more suitable than attending an actual situation where the system is supposed to be used. Moreover an official business meeting is a fragile situation, which it is not very appropriate to disturb as an outsider. To get the most out of every meeting with the assigners, questions and subjects to discuss were written down before the meeting.

The evolutionary process has been based around prototypes and meetings discussing these. The whole process has been very iterative and even the thought of user scenario for the final application has been altered throughout the process. This has complicated the attempt to support usability due to the change of users and goals, but

as VR was a fairly new area to the commission assigners the possible situations of when to use it had to develop over time.

Concerning the prototypes the applied method can be described as evolutionary (see **2.3 Introduction to user-centred design**). The main reason was that implementation of a VE is time consuming and there simply wasn't enough time to start all over again for every iteration cycle. The disadvantage of this method is that bad design decisions in the beginning of the process may steer the whole process in the wrong direction. It is often useful to use simple paper mockups as throwaway prototypes in the beginning of a designing process, they are cheap and the test users tend to be more honest if they don't think too much effort lies behind the prototype. However, as mentioned above, VE was a relatively new, unknown and exotic area to the assigners so a simple paper mockup was not thought to be able to represent the appearance of the end product well enough. Instead, the first prototype was designed to show as many possibilities with VE as possible rather than suggest a final result.

In the next section the result, in the form of some key points in the developing process, are listed. They are not only intended to describe the iterative process, but also concern the choice of using open source software.

4.2 Results

This section presents the results of the project. As the subject is an iterative process which not only concerns a final result, the section is based on some key points in the development process.

4.2.1 The first meeting

The first project meeting with IKEA was mostly spent defining the task. Some initial time had to be spent introducing different fields of knowledge and brainstorming the subject. A problem with new technologies is that the people who master them are often not the same people who know the situation were the technologies best could be used. Different user scenarios were discussed and essential demands on the solutions were discussed. Specific technical solutions were avoided at this early stage as this might inhibit the discussion of the problem.

4.2.2 The second meeting

At the second meeting the hardware at the reality center was introduced and demonstrated. Even if the envisaged solution should be applicable to the "desktop VR" kind, it was good to broaden the mind and look at previous projects with interactive worlds. The main reason for the meeting was to demonstrate what could be done with the available hardware and to brainstorm different user scenarios where

VR could have a role within the process of designing and presenting new IKEA-stores.

4.2.3 Results from the second meeting

The conclusion after the second meeting was that some sort of VE of the entrance described in the one room project should be achieved. A 3D-model of the building was available in the 3DS-format, converted from a CAD-model, but the specific interaction the system should support was not specified.

4.2.4 Choosing software

At this point a decision of witch VE-software to use was crucial. World Up (see Section **3.2.6**) was the first choice as it was implemented and well known in the reality lab, it also has a user-friendly graphic interface.

The other choice was OpenSceneGraph (see Section **3.2.2**). The most important advantage of OSG is that it is an open source program and can be adjusted in a way that is suitable for the specific task. This also means that it is possible to make an executable file that doesn't require a specific player or viewer. Other advantages of OSG are that it is free and has an exporter from 3DStudioMAX.

4.2.5 Decision on software

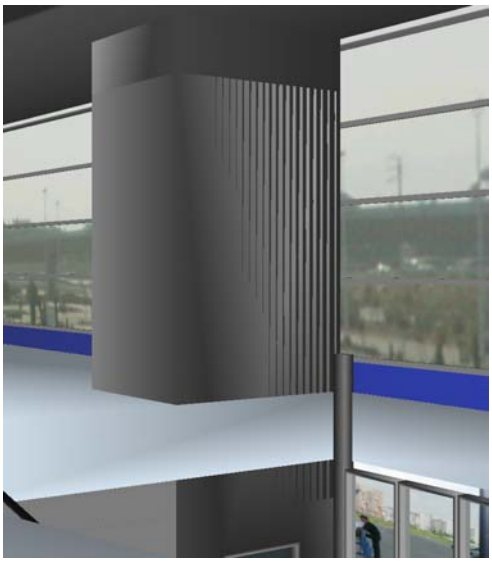
The choice fell on OSG as it seemed to be more flexible. The disadvantage of using a code-based freeware is that it can be badly documented and have a long initial learning time. The latter proved soon to be very true as no person in the reality lab, or even the building, had any experience of OSG. The only way to learn how to put together an OSG application was thus to search the web. However the possibility to create a standalone application for IKEA was considered more important than the graphical user interface of World Up.

4.2.6 The first prototype

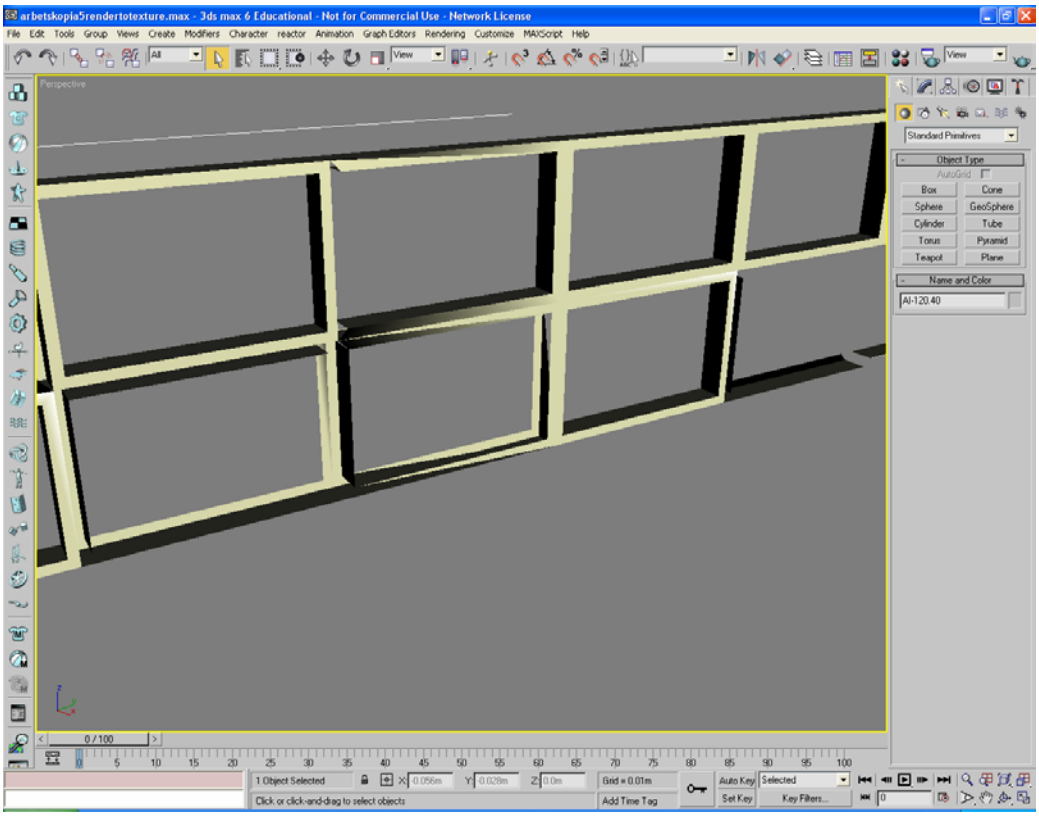
Once the decision on software base was taken the next step was to start programming. A major question that had to be answered in the beginning was how to navigate the VE. The standard OSG-application, based on the OSG-viewer module (see Section **3.2.2** about OSG), has three ways to navigate a scene; trackball, flying and driving mode. In the driving mode the system keeps you to the ground due to gravity, uses collision detection and was obviously the mode that was best suited to simulate a walk in a store building. The problem was that all motion is controlled by the mouse and all input resulted in a continuous acceleration in a chosen direction. This meant

that it would be hard to use the mouse to interact and move an object in the environment; moreover it was very difficult to move with an appropriate speed and to stop.

Apart from the programming that had to be done, the application needed a model to load and visualize. A big problem with the existing one room model was that it was not originally in the 3DS format, but rather a CAD model converted into a number of 3DS objects. As a result of the fact that an infinitely thin surface has two sides in the CAD world, but only one in the 3DS world, the model had been deformed in the conversion process. Moreover, the initial model had a level of detail that showed the internal building structure and was unnecessarily heavy to be used as a visualization model for volumes (se Fig. 20). The intention with this initial prototype was more to present options and possibilities with VE than producing a final model.



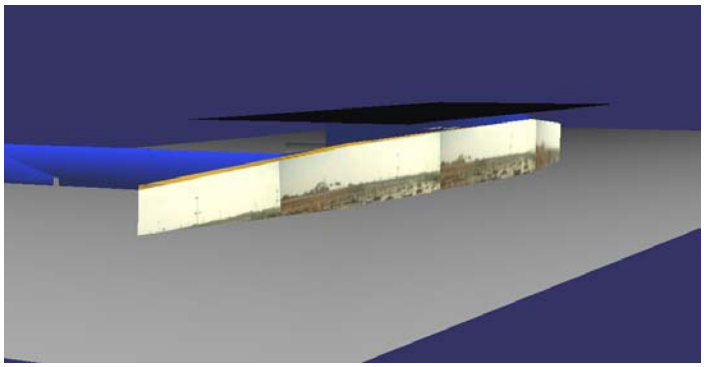
Above right Figure 20: Rendered picture from the model displaying visual artefacts due to double surfaces in the internal structure.
Below Figure 21: View from 3DStudioMAX displaying wrong pointing surfaces, no polygon is actually missing but the surfaces are invisible if viewed from behind.

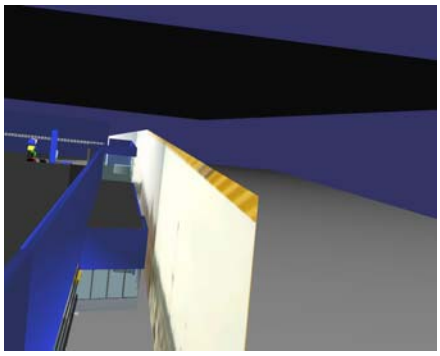


4.2.7 The resulting first prototype

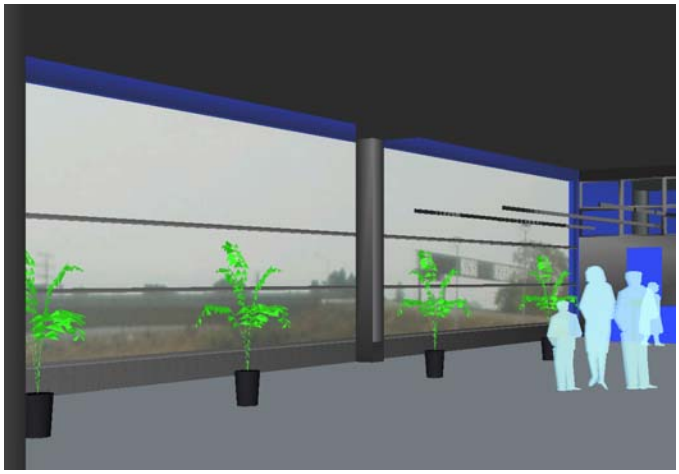
The navigation problem was solved by modifying the driving mode provided by OSG-viewer so the user could navigate with the arrow keys on the keyboard, and so the motion speed was constant while the key remained pressed and went to zero as soon as the key was released. This way it was easier to navigate the scene and the mouse could be used for other purposes.

A time-consuming analysis of the model had to be done, flipping wrong-faced surfaces and removing unnecessary internal polygons (see Fig. 20 and 21). Some materials (or colours) from the original CAD model were kept in the conversion process to the 3DS format, but additional colours and textured materials were also added to the scene. The use of a skybox (see Section **3.1.4)** was presented and the possibility to create a sense of ongoing spaciousness using a photo as a side-scene (see Fig. 22-24).





Figures 22, 23 and 24: Computer renderings displaying a photo used as a side-scene, the picture below shows the effect when viewed from the inside.



4.2.8 The third meeting

The exact specifications and limitations of the task were not decided until a third meeting had occurred. At this point the commission assigners could for the first time see a prototype of the model and navigate in it. It was now easier to specify the commission.

4.2.9 Results from the third meeting

It was decided that the system should support interaction with the interior of the building but not the building itself. The interaction should contain moving or exchanging objects and maybe the possibility to display stored information about the object (se the **1.3 limitations** section). Furthermore it was decided that the architects and designers already had suitable tools for the construction of new objects, but lacked a good way to see how they would all work together in a store environment. A conclusion of the third meeting was thus that the choice of LOD, lighting circumstances and the overall impression were very important; a reference to computer games was made. The importance of LOD was unanimous among the commission assigners, but what the LOD should be was more arguable. It was obvious that the LOD on most of the textures in the prototype was way to high and that it distracted the user from what was important. On the other hand, it was also considered very important that some company-specific trademarks in the scene should have a high LOD.

Another important result of the third meeting was that the IKEA tailor-made CAD version AUTO DECO, had to be explored and incorporated to get a meaningful user scenario for the application.

4.2.10 The SOVRI workshop

As mentioned above the available information about OSG was quite limited in the beginning. The consequences of this were shown when the VR lab in Lund hosted a Scandinavian conference (SOVRI, see references) about VR and open source. Among the participants were representatives from Umeå University, which has a longer history with OSG.

4.2.11 Results of the SOVRI workshop

After having consulted the new expertise it was obvious that parts of the programming job had to be redone, making the application more compatible with future upgrades of the OSG library. A basic example of a “walk-manipulator” was also provided, as well as usable basic information about how to program an OSG

application. In the first prototype the standard drive mode of OSG-viewer had been manipulated to simulate walk. The right thing to do would instead have been to make a new camera manipulator that could be loaded into the viewer. This way the same camera manipulator could be used when a new OSG-viewer is released. Afterwards it is easy to be smart but OSG is open source and the difference between the bricks in the wall and the wall itself can sometimes be difficult to discern, especially due to the bad documentation.

4.2.12 The second prototype

Even if the first prototype was supposed to be of the evolutionary type, most of the code was thrown away due to the new knowledge achieved at the VR conference. However when making the second prototype the user interface was pretty much retained and parts of the model could be used again, even though the One Room project had been developed further by the commission assigners. The second prototype became of the true evolutionary type and was just complemented and iterated through the development process until the end. One challenge when programming the second prototype was how to implement the interactivity.

4.2.13 The resulting second prototype

For the user, the most notable difference from the first prototype was the addition of interactivity, i.e. the possibility to select and move objects in the scene (see Fig. 25-27). To do this the transform matrix (see Section **3.2.2** about OSG) that belongs to a selected object has to be detected. The transform matrix defines where in the world the object will be located. OSG supports an easy way to select an object in the scene using the mouse pointer. This is done by "shooting a ray" from the mouse pointer into infinity in the viewing direction; the first object that the ray meets is detected. The objects are represented as leaves on the scene graph tree (see Fig. 19) and will all have a parent node that contains the transform matrix. If the matrix is manipulated the object can be translated or rotated relative to the origin of the transform matrix.

The user interface used the mouse pointer and a dedicated key on the keyboard to select an object, which made it possible to move it with the navigation keys. Due to possible hardware limitations the collision detection during navigation was skipped in order to increase the frame rate.

Another new feature was the possibility of "mouse look", where the mouse is used to change the viewing direction in all degrees *i.e.* makes it possible to look around for instance at the ceiling. Also the possibility to double speed by pressing the shift bar was added.

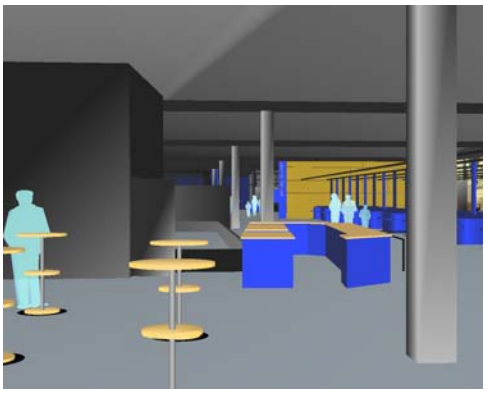
4.2.14 The fourth meeting

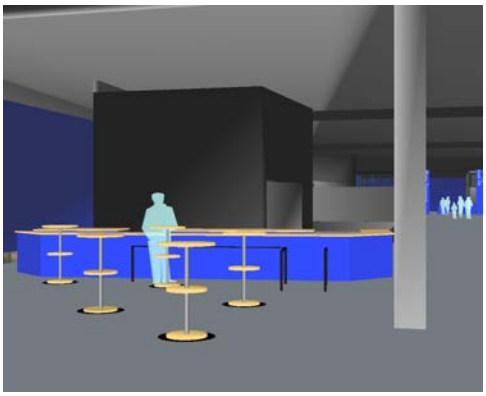
The intention of the fourth meeting was to achieve an exact vision about where in the development process of a new IKEA-store the application could play a role. This was necessary in order to provide exactly the right interaction and features of the program. Even if it seems good to have as many opportunities as possible, too many features can affect the usability and intuitive understanding of the program.

4.2.15 Results from the fourth meeting

As mentioned before, the intention was not to create a new modelling or construction program, but rather to enhance the impressions of a predefined model describing a future building in reality. It was decided that the application should be seen as "a way of walking around in one of the currently used printed pictures featuring renderings of models created in AUTO DECO". This means that the application should be used in a symbiosis with AUTO DECO via 3DSMAX. It should be mentioned that at the time of writing, a new exporter to OSG from Auto CAD is being developed which might eliminate the use of 3DSMAX in the future.

Even if interaction with the model is not necessary for visualising how the model might appear in reality, it supports the working process in AUTO DECO if one immediately can see the consequences of, say, moving an info counter to the other end of the room. The importance of keeping some parts of the model unmoveable (such as building constructions) was also emphasised. However the possibility to export any chosen model from 3DSMAX to the application is also appreciable, which raised the question of how to define the unmoveable parts of the model.





Figures 25, 26 and 27: The second prototype included interactivity *i.e.* the possibility to select and move objects in the scene, in this case the desk.

OSG provides an easy way to save an animation path in the model, which will appear as a movie when being replayed. This can also be performed in 3DSMAX but will be much more time-consuming as this is done by pre rendering every frame in the movie. The ability to predefine exactly which parts of the model the user will see can be a powerful tool in a presentation, and it was decided that some sort of communication clearly should inform the user how to create such an animation path in the loaded model.

Also it was concluded that "mouse look" should not be selected by default as some users tend to lose track of directions using this mode.

4.2.16 The third prototype

One way to identify moveable and non-moveable objects would be by name, or to "tag" all the objects in the scene. If the process of dividing the model into moveable/non-moveable parts is done in the code it will make it hard to integrate a new model in the application, and if it is done in 3DSMAX it will be highly dependent on the current 3DSMAX version and its exporter to OSG.
An appealing way to identify the moveable parts would be to tag them directly in the loading process to the application; this way all moveable parts could be loaded in one model and all the unmoveable parts in another. The solution is dependent on the ability to integrate two model parts, and make it appear as one model without any scaling or displacement problems.

4.2.17 The resulting third prototype

Fortunately OSG allowed the integration of differently loaded parts of a model in the same scene very well, using the OSG modules for reading arguments (osgDB). From the third prototype and on, the application reads the model as arguments to the executable file. The first argument is added as the first child of a super root-node in the scene graph (see Section **3.2.3** about scene graph), and the second and third arguments are added as the second and third children of the super root-node. When the user selects an object, the application traverses up in the scene graph hierarchy tree to the super root-node, and detects which child branch the object belongs to. This way the application can detect the model loaded in the first argument as unmoveable and the rest of the arguments as moveable objects.

Also the user interface was improved so moveable objects were selected and moved by drag and drop, using only the mouse and no keyboard.

4.2.18 The fifth meeting

The fifth meeting most concerned details regarding the user controls and the visual appearance, and not so much defining a user scenario. The prototype was fully working and the project assigners could navigate the scene and move selected objects. This made it possible to more exactly define parameters such as with what speed one should move in the model, which options should be selected by default, and what sort of help information should be available.

4.2.19 Results from the fifth meeting

It was concluded that the walking speed applied when moving around in the model had to be increased. Also the gravity system should be improved providing a more continuous movement when walking in staircases. The model should be even more abstract without so many distracting textures. Better models of customers would be appreciable, especially children.

4.2.20 The fourth prototype

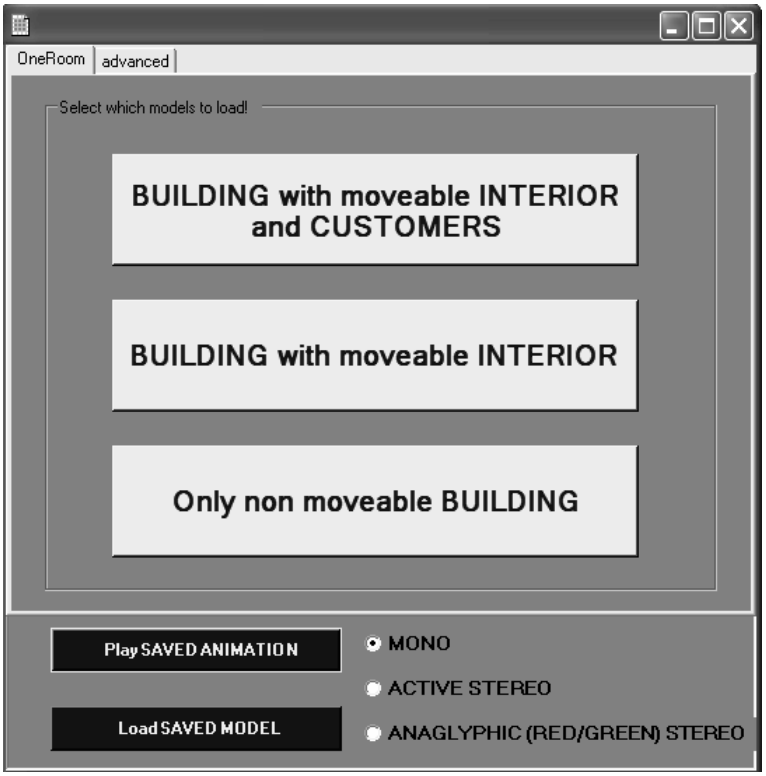
So far, writing the executable file name and the arguments in the command prompt had been the only way to start the application. The two remaining big parts of the project were now to create a user interface for loading the right model, and writing a user's manual.

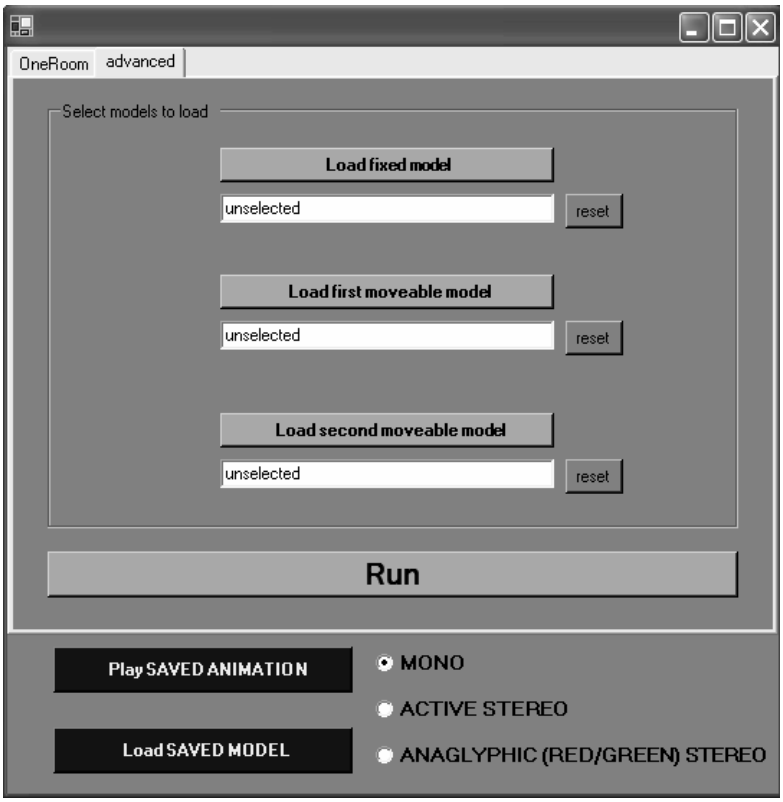
4.2.21 The resulting fourth prototype

After the fifth meeting the prototype was adjusted according to what had been said. Also an optional collision detection was implemented and the ability to manually change the eye height (which made it possible to explore the loaded model the way a child would see it).

A user interface was written in Visual Basic and provided both shortcuts to load the One Room model (Fig. 29) and an advanced part (Fig. 30) that could be used to load any other model. Also buttons for loading the **saved_model.osg** file and to play the **saved_animation.path** were added (see Section **3.2.2** about OSG).

A user's manual was written (see appendix A) which covered both handling the application and how to use it together with 3DSMAX.





Figures 29 and 30: A user interface was written in Visual Basic and provided both shortcuts to load the One Room model, and an advanced part that could be used to load any other model.

4.3 Summary of the model development process

The process of developing the model, or application, started off in pretty undefined ideas of what should be accomplished. However the reason for the project was clear; "it's much cheaper to change the design of a building before it's built than afterwards", and maybe VR can be a way to discover undesired results, or to transmit the ideas, of the current design. The evolutionary way of deciding the user scenario for the application was probably necessary due to the mutual exchange of knowledge about VR, and the knowledge about building IKEA-stores, with the commission assigners. The method of using iterative prototypes when approaching the final design was also a result of the need to establish a common platform to discuss around. The time schedule was violated and in a future project it may be easier to decide upon a final design sooner. The violation of the time schedule also concerns the choice of software; open source has a big advantage in the ability to support individual requirements, but the advantage is more obvious in the long run when one wants to expand systems or incorporate them with each other. For separate small projects the initial learning phase can become quite steep due to the insufficient documentation; however this of course changes if one has already mastered the software.

Even if the project originated in the one room project, the final prototype makes it pretty easy to load any other model. The result of the project can be seen as a method of converting a CAD model into a VE. This change of focus is a result of the evolutionary way of defining the task. During the process it has become quite obvious that the commission assigners and architects at Inter IKEA are the ones who are best suited to design the visual aspects of the model, and for an engineer the task is to help displaying and exploring the model.

5 Presentation of the final application

This chapter presents the final application; maybe "final application" should be exchanged for "latest prototype based on the last iteration process", meaning the commission assigners to this project have most likely only just started their quest for the right VR application for them. But this is how far the project is taken at this time. At the end of the chapter is a short section for the interested reader called "5.3 Theory connections to the final result" where references to the initial theory and an evaluation of the prototype are made.

5.1 Supported features

The final application has an interface written in Visual Basic and consists of two parts, one concerning the predefined One Room model and one advanced part, which supports the loading of any other model. The One Room part has three choices; to load the whole building with moveable interior and moveable customers, to load the building and the moveable interior without customers, or to only load the non moveable building. The advanced part supports in the same way three possibilities to load parts of a model, the first loaded part will be fixed and the optional two other parts will be moveable. To choose parts of the model an explore window is used, opened by default to the MODEL catalogue of the application. The features for saving or loading an animation path to **saved_animation.path**, or the displayed model to **saved_model.osg**, are present in both parts.

The Visual Basic interface is merely a light cover for the real OSG application, and in a future version it will most likely only consist of the advanced part in the case of the application used as a developing tool together with a 3D-modeling program. Alternatively a new interface like the One Room part can be written for any specific model and the whole application can be saved on a cd-rom and distributed as a VR brochure of the featured model.

When an option in the Visual Basic interface is selected the OSG application reads the chosen model as arguments and execute. The program initially always puts the user in the middle of the world, facing north; in the One Room model this is set to the entrance of the building. The up, down, left, right keys are used to move around; pressing the shift bar doubles the speed. To display help and key settings the user can press "h" or "H". Drag and drop with the mouse is used to select and move the parts of the model that has been chosen to be moveable. A moved object always keeps the same vertical position, but can be moved around or rotated in the horizontal plane.

The user has the option to toggle different settings such as collision detection, mouse look (to use the mouse to move around your virtual head), a simulated "trotting" walking style and different rendering options.

Other features are instant return to the start location, or save a new start location, and to change the eye height over ground. The original osg-viewer ways to navigate the scene (trackball, flying mode and driving mode) are still available.

5.2 Technical description

The OSG application consists of two classes, WalkManipulator which extends osgGA::MatrixManipulator and PickHandlerInteractive which extends osgGA::GUIEventHandler. A main program (IKEA-Viewer.exe) starts an osg-viewer instance and adds an instance of Walkmanipulator which has an instance of PickHandlerInteractive.

Walkmanipulator handles a matrix representing the viewer (a model-view matrix) and uses input from the keyboard and mouse in its "WalkManipulator::handle" method to calculate the movements in the scene. As WalkManipulator extends osgGA::MatrixManipulator, which in turn extends osgGA::GUIEventHandler, it can handle events from osgGA::GUIEventAdapter such as osgGA::GUIEventAdapter::FRAME, in FRAME the calculated movements from the "handle" function is used to update the scene.

WalkManipulator also includes the WalkManipulator::setPositionWithHeight function which uses a osgUtil::IntersectVisitor (can be described as an invisible beam that can detect if it crosses an object in the model) for adding gravity to the system and keeping the viewer to the ground. An instance of IntersectVisitor is also used when the collision detection is activated; in this case the "beam" shoots horizontally instead of vertically. If the "trotting" walking style is activated a sine function is used to modify the position over ground in setPositionWithHeight.

PickHandlerInteractive handles the interaction in the scene and uses an IntersectVisitor to detect the object that the user selects with the mouse pointer. Once the object is selected PickHandlerInteractive can traverse the scene graph upwards and detect if the object belongs to a branch that is supposed to be moveable. If the object is moveable the transform matrix, which defines where in the world the object is located, is detected so WalkManipulator can use the input from the user to move it.

IKEA-Viewer simply reads the up to three arguments containing the model, and puts them in up to three different branches in the scene graph tree. When PickHandlerInteractive traverses the tree it knows that the first branch is supposed to contain fixed objects and the rest moveable objects. IKEA-Viewer also continually calls the activity defined in WalkManipulators FRAME part so the window gets updated correctly.

5.3 Theory connections to the final result

In order to talk about usability one has to specify a user, goal and environment. In this specific project the user and the goal were initially very loosely defined. The environment was specified due to the hardware available in a conference room, i.e. a laptop and a projector. However the only specification of the user at the start of the project was that he or she could not be assumed to have any specific computer skills, but if the user was involved in the designing, construction or marketing business was unspecified. These three categories of users were the most likely but they had very different goals in their use of the system. No more specific description of the assumed profession of the user was actually decided, but the tasks the system should support in general was better specified later in the project.

Concerning usability aspects there are however general guidelines, or principles, that can be attended to in order to support usability when designing an interactive system (see Section **2.2 introduction to usability**).
Next come some examples of how the final application relates to these general guidelines that were presented in Tables 1-3.

5.3.1 Learnability:

The levels of supported **Predictability** and **Synthesizability** are in this case heavily depending on the computer hardware that the application is running on. If the graphic card is not up to the task and the system runs slowly, the user will lose the connection between his/her actions and the effect they have. Otherwise **Familarity**, **Generalizability** and **Consistency** are supposed to be well supported due to similarities in computer games most of all. One weakness might be the way objects are selected and moved around. In a windows based, 2 dimensional, computer environment the “drag and drop” system never has to support rotating an object, this makes it difficult to use the reference in a 3 dimensional environment.

5.3.2 Flexibility:

Dialog initiative and **Multi-threading** are thought to be automatic in an environment like this where the main feature is to visualize a 3D-model. **Task migratability** can be considered supported by the provided option of both ”mouse look” and normal viewing; in the mouse look mode the viewing angle can be chosen arbitrary, and when going back to normal mode the system simply sets the viewing direction to the move forward direction horizontally. One example of **substitutivity** can be that to unmark an object, either the right mouse button, or simply trying to select something else with the left mouse button, will work. **Customizability** is not generally thought of but an initial choice of keyboard bindings like the ones present in many computer games might be an idea. The options to change drawing mode, to for example wire

frame, or to show frame rate, is however provided by OSG-viewer from the beginning.

5.3.3 Robustness:

Observability and **responsiveness** are the reasons why a notification, telling the user that the system is ”loading”, was added when a model in the start-up interface is chosen (se Fig. 31). The loading time can be quite long and test users wondered why the screen suddenly turned black. **Recoverability** might be the correct motivation for the added option to return to the start position, but a way to exactly restore moved objects without reloading the model would be desirable. This would however require some sort of saved initial data, or a memory of the actions, which might cost a lot of computer power.
Task conformance is the major point that this thesis aimed to evaluate, over time the tasks themselves have changed but hopefully this project has been a step forward for the commission assigners.

Figure 31 showing the command prompt displayed when a model is loaded into the application.

5.3.4 Overall evaluation:

To do an overall evaluation of the final result from this project, the most desirable thing would be to construct some sort of user test. However, the resulting application was never intended to be a release product. To really evaluate the use of virtual reality in the process of designing and presenting new IKEA-stores a release product would be necessary as well as time to incorporate it in a new store construction project. There are however optional evaluation methods that don’t require user tests, such as expert analyses and cognitive walkthroughs. Expert analyses have been performed both with the author of this thesis (who has specialized in human computer interaction) as an expert, and with the commission assigners as experts in how to perform the task of designing and selling IKEA-stores. A cognitive walkthrough is a mental test performance of how the system would work in a real situation, this has also been done continuously as one objective was to evaluate a method, or working scenario, for producing and using VR applications of IKEA-stores. Finally some sort of unofficial user tests have been performed when the commission assigners have tried and commented on the prototypes.

The resulting overall evaluation, done under the premises described above, is that the resulting prototype probably has the right features. The initial intention to display predefined data connected to the objects in the model was never implemented, but the

possible advantage with this is obvious as one major problem has been to define the user and the ideal level of detail the model should obtain. In this way users would be able to obtain different amounts of information about a selected object in the scene. The only debatable feature of the final prototype is the interactivity added for moving objects in the scene; it is not necessary that this feature is essential for the user.

The conclusion to concentrate on the possibility to support future models was probably right; the importance of fast cycles was concluded by the commission assigners when incorporating virtual reality in there everyday work.

The usability aspects of the selected features also seemed to be satisfying; the commission assigners were able to run the system without too much introduction and training. This holds for the assumption that the system worked the way it was supposed to do, and no delay due to hardware incompatibility occurred. However OpenSceneGraph seems to be poorly compatible with certain graphic cards, which is a major and lately found disadvantage.

Finally it should be said that the commission assigners at the end of the project still thinks that the right project and question at issue was chosen at the beginning, and that the assumption that virtual reality can have a role when designing, building and selling IKEA store was correct.

6 Conclusions

This project started with an intention to orient the staff at Inter IKEA into the world of virtual reality. At this stage it is possible to conclude that OpenSceneGraph is probably not the right software for this type of applications, as the initiation process is way too long and the documentation and support is bad. The situation might however be another if the task was to build a bigger separate system, say a special show room, with a group of people specially dedicated to support and maintenance. One solution for IKEA might be to do as in the case with AUTO DECO, which is an in-house interface for CAD programming. The same type of interface could be ordered for OSG, and in this way the advantages with open source can be combined with the support of a commercial software.
It is obvious that IKEA can gain advantages by coordinating the used CAD standards so a quick conversion into a VR application can be achieved. This might require a material library where the standard wall, floor, furniture etc are defined. A major conclusion from this project has been that converting CAD models can be very time consuming, but this can be avoided if the original CAD model is intended to be used both as a 3D-object in a VR application, and rendered into a 2D drawing.

Finally it's my conviction that a fast-cycle, simple interface standardised method for converting CAD models into VR-models will be of great importance when both designing, selling and building new IKEA-stores in the future.

References

Angel, Edward, (2003), *INTERACTIVE COMPUTER GRAPHICS A Top-Down Approach with OpenGL (third edition)*.
Addison Wesley

Dix, Alan; Finlay ,Janet; Abowd, Gregory; Beale, Russell , (1997), *Human-computer interaction (second edition)*.
Prentice Hall

Kheddar, Abderrahmane; Chellali, Ryad; Coiffet, Philippe, (2002), Virtual Environment-Assisted Teleoperation, chapter 48 in *Handbook of Virtual Environments* (2002) edited by Kay M. Stanney.
Lawrence Erlbaum Associates, Pubblishers.

Ljungqvist, Ludvig, (2003), *Establishing Methods of 3D City Modeling based on Multiple Data Sources*. Master Thesis, Department of Design Sciences, lund Institute of Technology, Lund, Sweden.

North ,Max M.; North, Sarah M.; Coble, Joseph R., (2002), Virtual Reality Therapy: An Effective Treatment for Psychological Disorders, chapter 51 in *Handbook of Virtual Environments* (2002) edited by Kay M. Stanney.
Lawrence Erlbaum Associates, Pubblishers.

Preece, Jenny; Rogers, Y.; Sharp, H.; Benyon, D.; Holland, S.; Carey, T. ,(1994), *Human-Computer Interaction*.
Addison-Wesley

Rafferty, M.M., Aliaga, D.G., Popescu, V., & Lastra, A.A.,(1998), Images for accelerating architectural walkthroughs. *IEEE Computer Graphics and Applications.*

Internet references

Openscenegraph (2004)
http://www.openscenegraph.org/ [2005-01-17]

SOVRI
http://www.reflex.lth.se/sovri

The final prototype and a digital version of this thesis can be found under this web link:
http://www.reflex.lth.se/idc/

Appendix

The user's manual for IKEA Viewer…

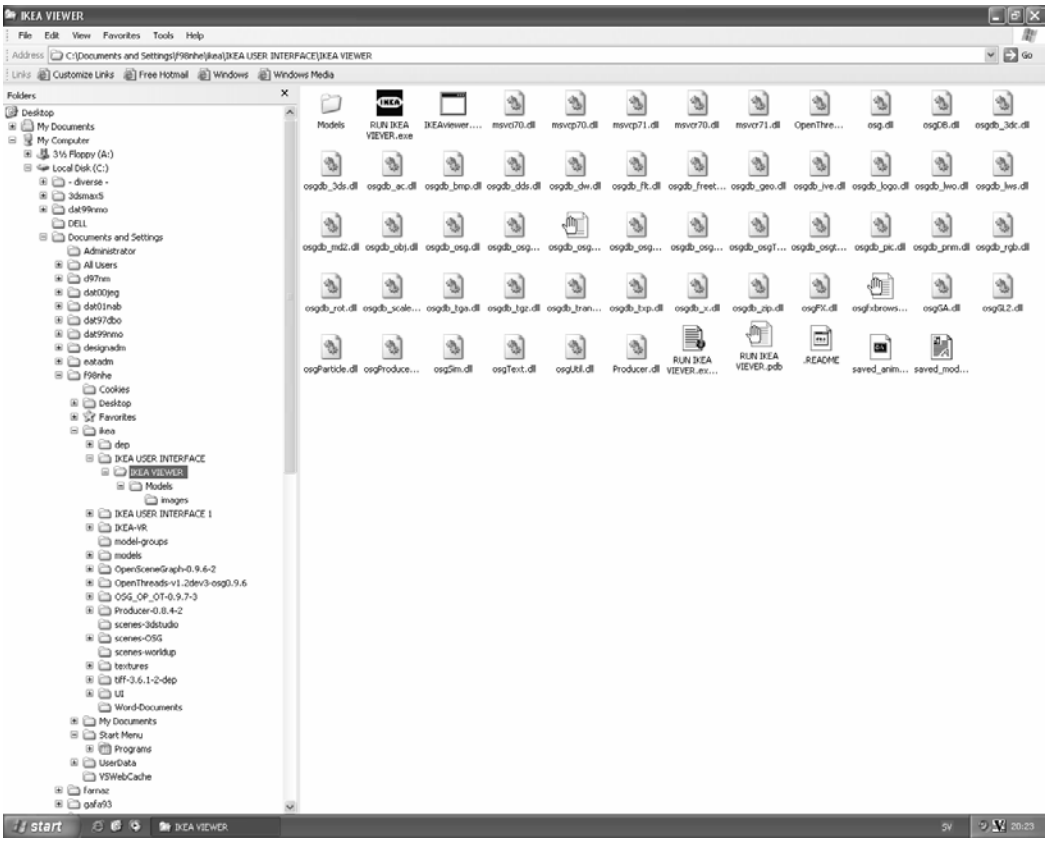
Manual for IKEA-Viewer

*IKEA-Viewer is an application that can load and visualize any 3D-model of the OpenSceneGraph (OSG) format. IKEA-Viewer also provides some interactive functions, such as to move around objects in the scene, and the ability to create or play a movie (animation path) in the loaded model. Any 3D-model that can be opened in **3D Studio MAX** can easily be exported to the OSG format using the open source program **OSGexp** provided at: http://osgexp.vr-c.dk*
This means that also models stored in various CAD formats can be exported to the OSG format. Learn more about the open source program OpenSceneGraph at: http://openscenegraph.sourceforge.net.

To open up and run IKEA-Viewer

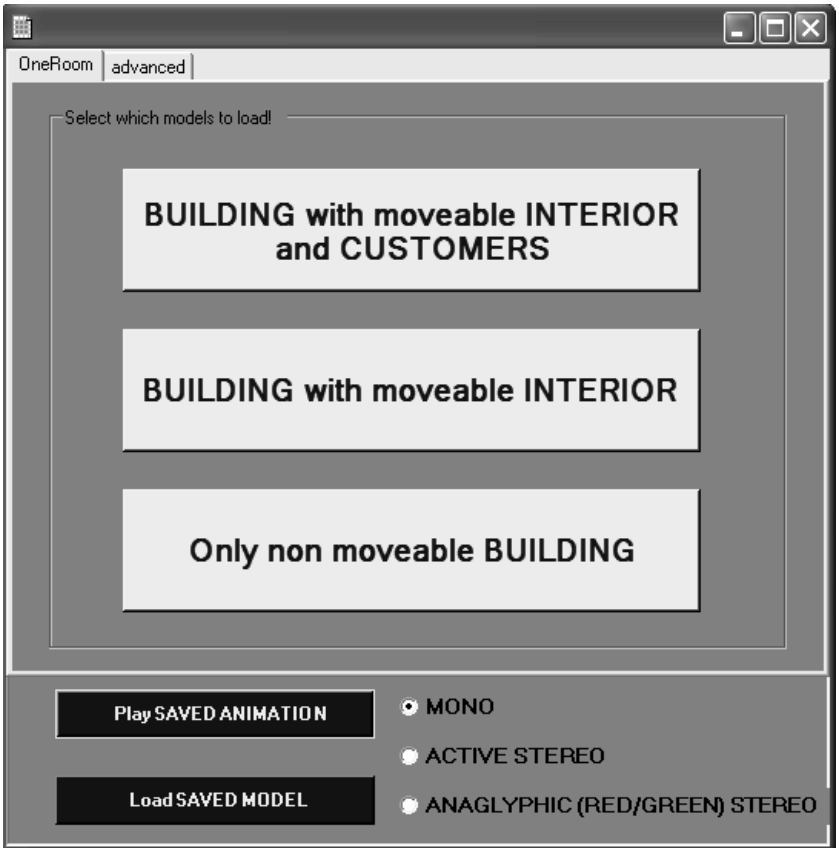
Download and unzip the entire "IKEA VIEWER" zip file where you want it on your hard drive.

The unzipped file includes, in addition to a couple of dll files needed by the IKEA-Viewer.exe file, a **RUN IKEA VIEWER.exe** file which should have an **IKEA logo**.



A folder named Models is also included containing the One Room OSG models. This is also the location to where you should export any of your own models.

Double click the RUN IKEA VIEWER.exe file to start the interface that controls the IKEA-viewer program. A window like the one below should pop up.



In the **One Room** part of the interface three versions of the One Room model are provided. Simply make your choice by pressing the appropriate button. The application may need some time to load the model but you should soon find yourself at the start of the One Room model.

Use the up, down, right, left -keys to move around. *By pressing* **"h"** *you can always view* **help**.

To play a pre-stored animation path
Press the "**Play SAVED ANIMATION**" button to run the saved animation path. The program will follow the animation path stored in the file **saved_animation.path** and load the model **saved_model.osg** (note that these files are located directly in the IKEA-Viewer folder and not in the Models folder). To view the model stored in **saved_model.osg**, press the "**Load SAVED MODEL**" button.

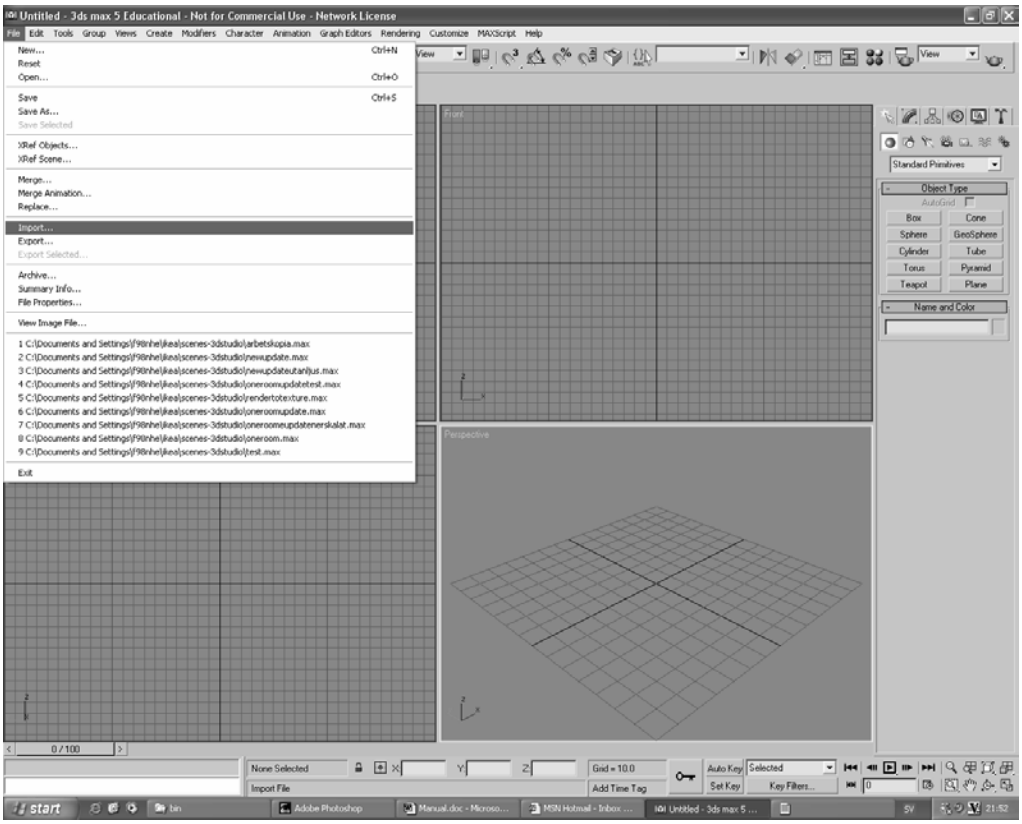
To store an animation path
When the program is running, press "**z**" to start recording and "**Z**" to stop recording. The animation path will automatically be stored in the **saved_animation.path** file and will start looping from the beginning as soon you pressed "Z". If you want to *save the currently viewed model* in the **saved_model.osg** file you should press "**o**". This is necessary if you want the currently loaded model to be used when you press the "Play SAVED ANIMATION" button.

To load your own model

When you have installed the OSGexp program to your 3ds MAX version (se page ?) you can export selected parts of a 3ds MAX scene using the "export selected" function in 3ds MAX. You can also export the entire scene by simply using "export" instead of "export selected".

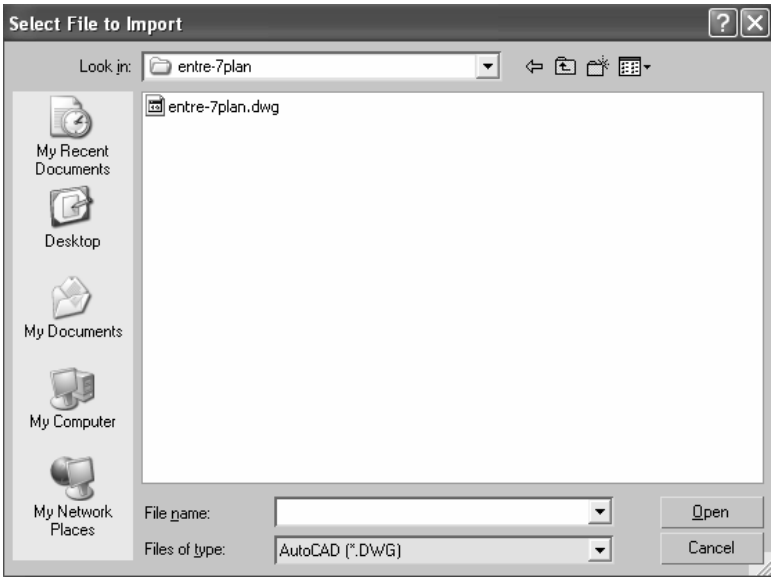
If your model was not created in 3ds MAX you have to start with importing it to 3ds MAX. Below is a DWG file imported to 3ds MAX and then exported to IKEA-Viewer as an osg-file.

Select "Import" in the "File" menu in 3ds MAX.

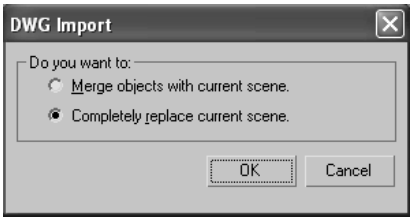


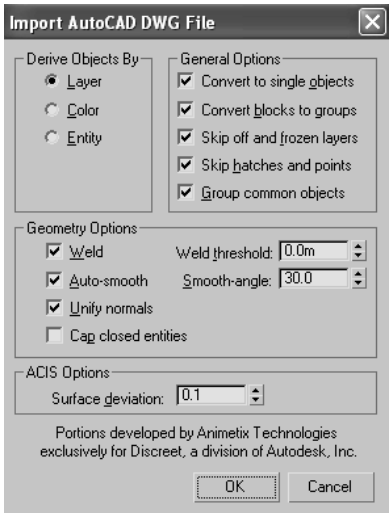
A "Select File to Import" window pops up where you can navigate to the model you want to import. Select the format for the file in the "Files of type:" menu, then select the model.

.

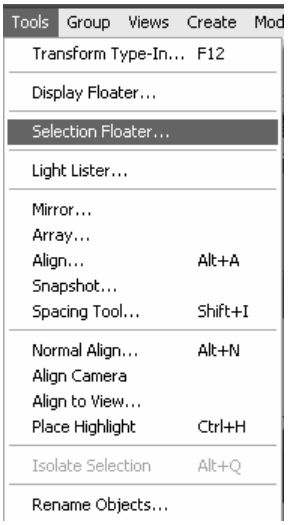


You might get several option windows when you import your model. The two windows presented below pop up when you import a DWG file into 3ds MAX 5 and can be selected as shown.

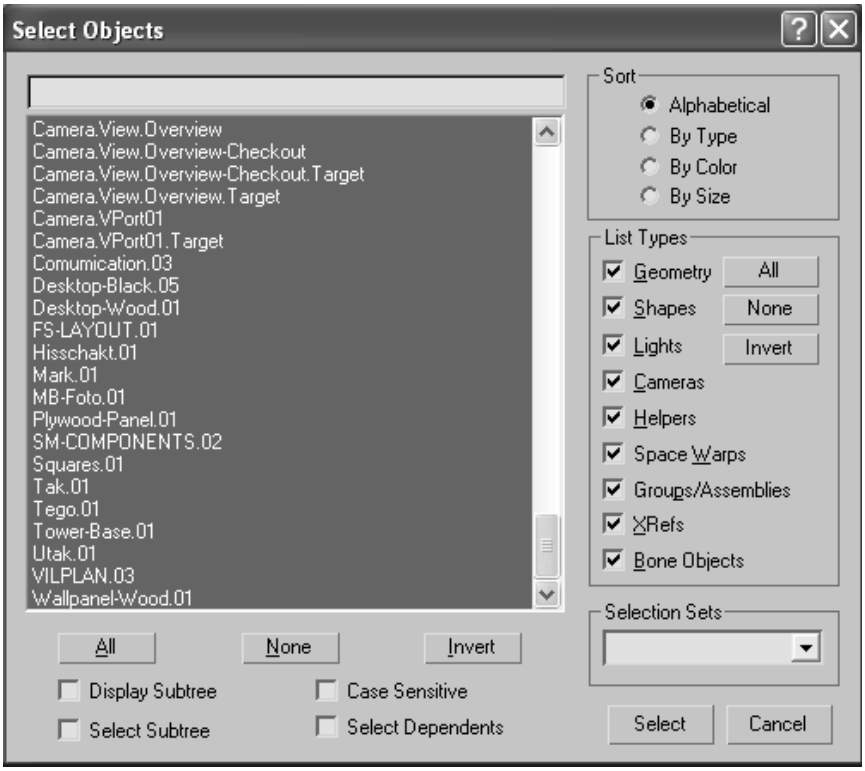




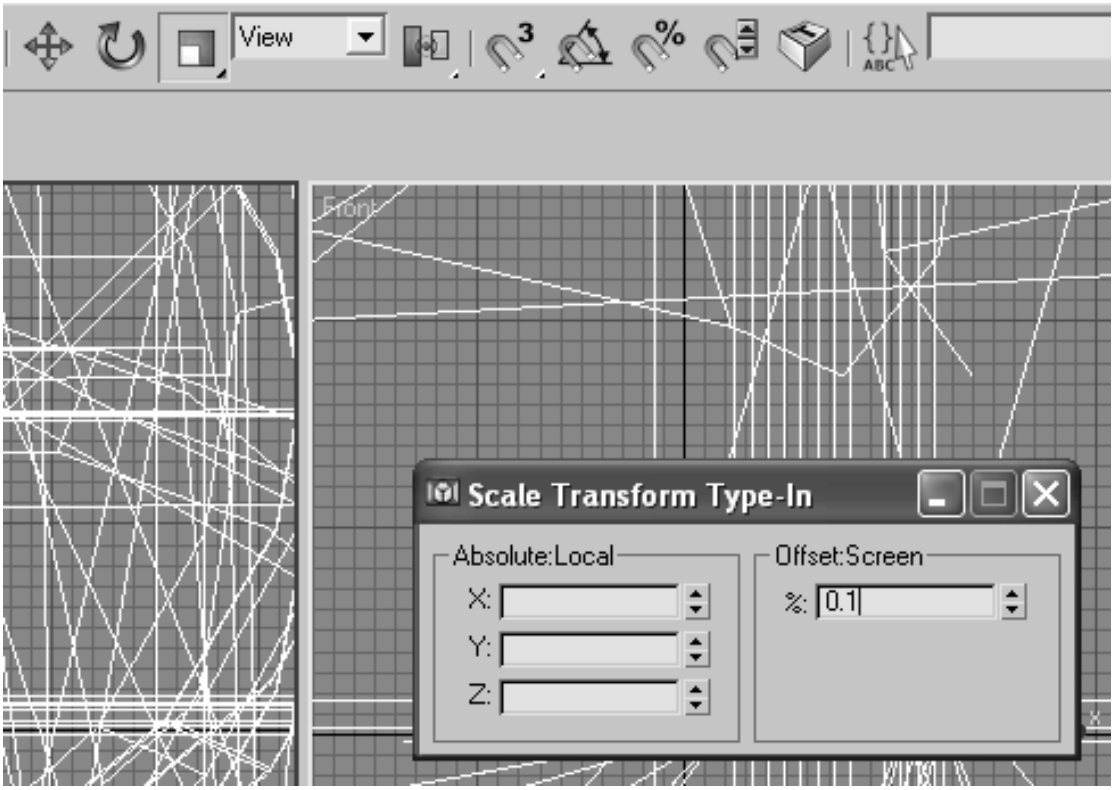
You may have to scale your model so it doesn't appear to big or small in IKEA-Viewer.
To get the "selection floater" in 3ds MAX press "h" or get it manually from the "Tools" menu.



In the "selection floater" mark all with the "ALL" button, then press "Select".

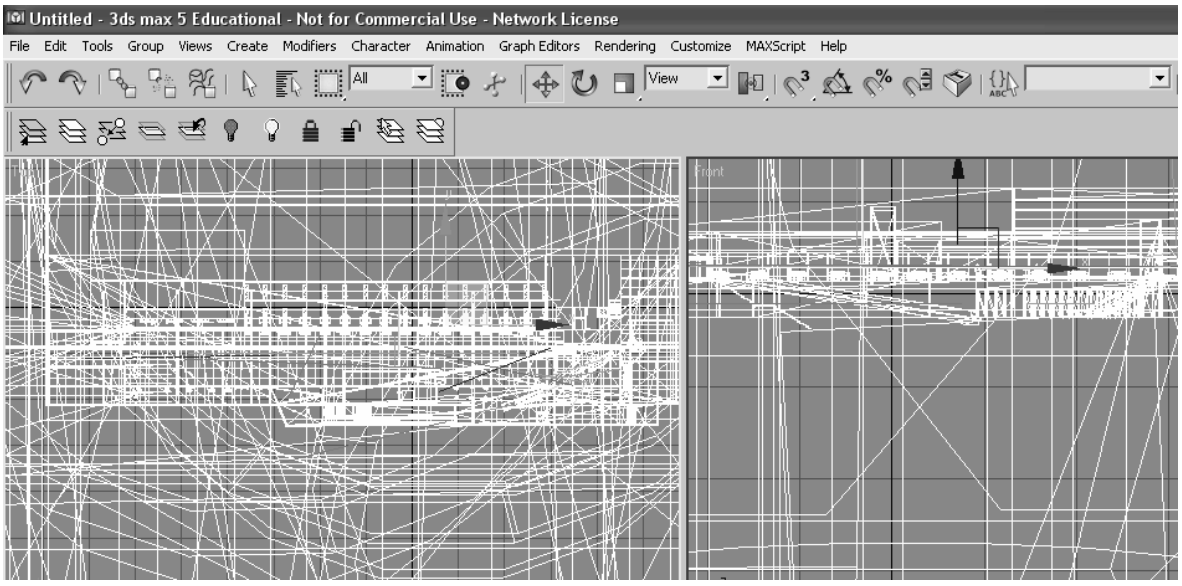


Right click the scale button in the top toolbar of 3ds MAX to get the scaling window. ***The proper scale for IKEA-Viewer is 1:1000****.*

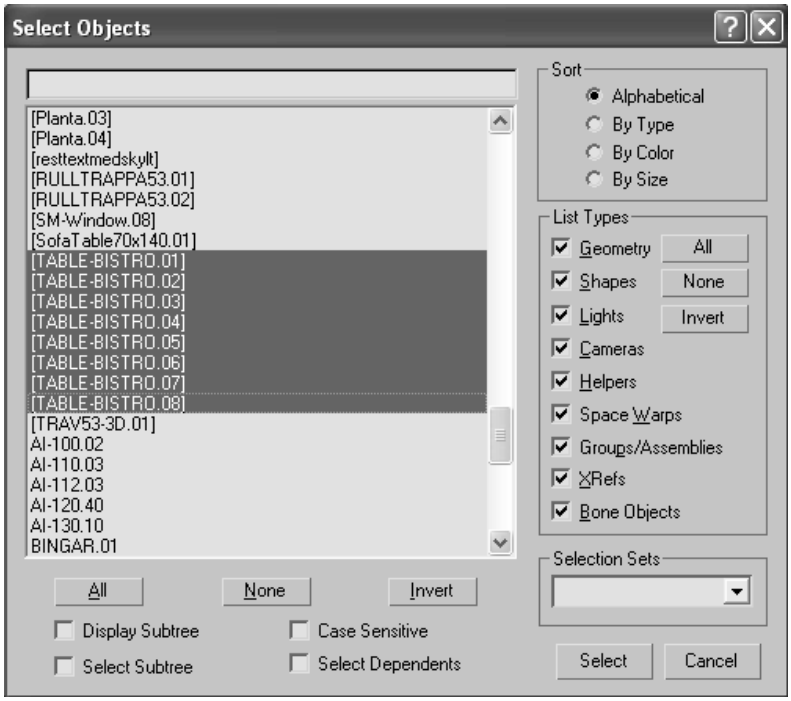


You also have to decide where in the scene model the start position for IKEA-Viewer should be. ***IKEA-Viewer will always initially place you in origo*** *(x = 0, y = 0, z = 0) of the 3ds MAX world, facing north. This means that you should select the whole scene in the "selection floater" and translate it so that origo is located where you want to start in IKEA-Viewer.*

Use the translation tool by pressing the translation icon:

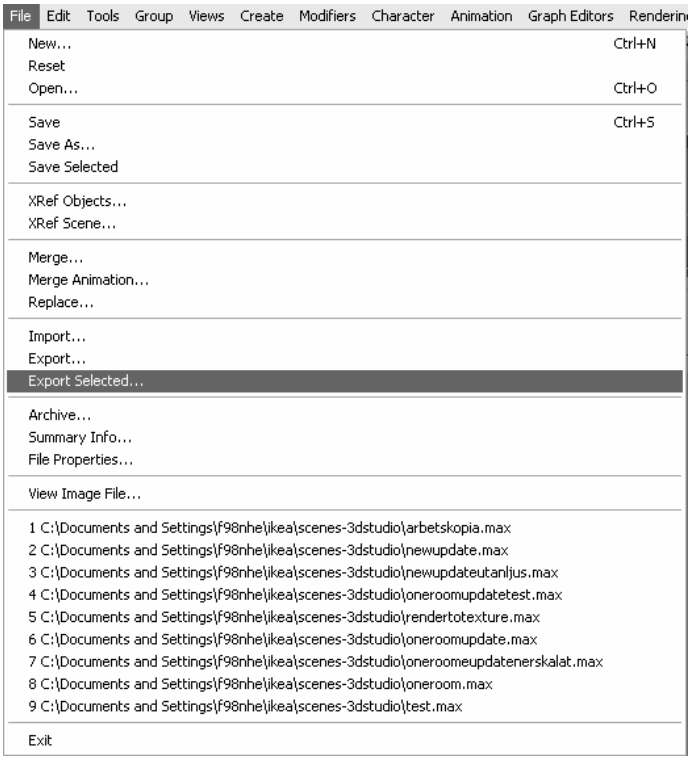


You can now select which parts of the model you want to export to the Models folder in IKEA-Viewer, converted to the osg-format. Use the selection floater:

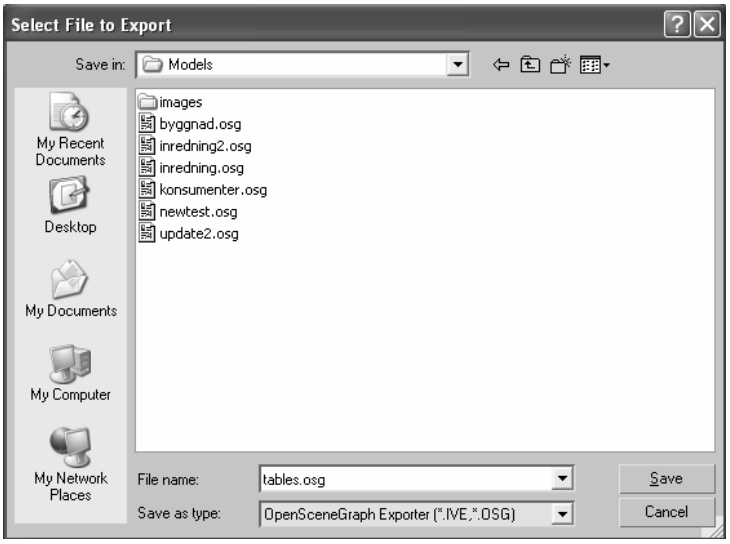


Bluemark the objects in the scene you want to export and press "Select". Use the CONTROL button on the keyboard to select more than one object.

Then choose "Export Selected" in the "File" menu.



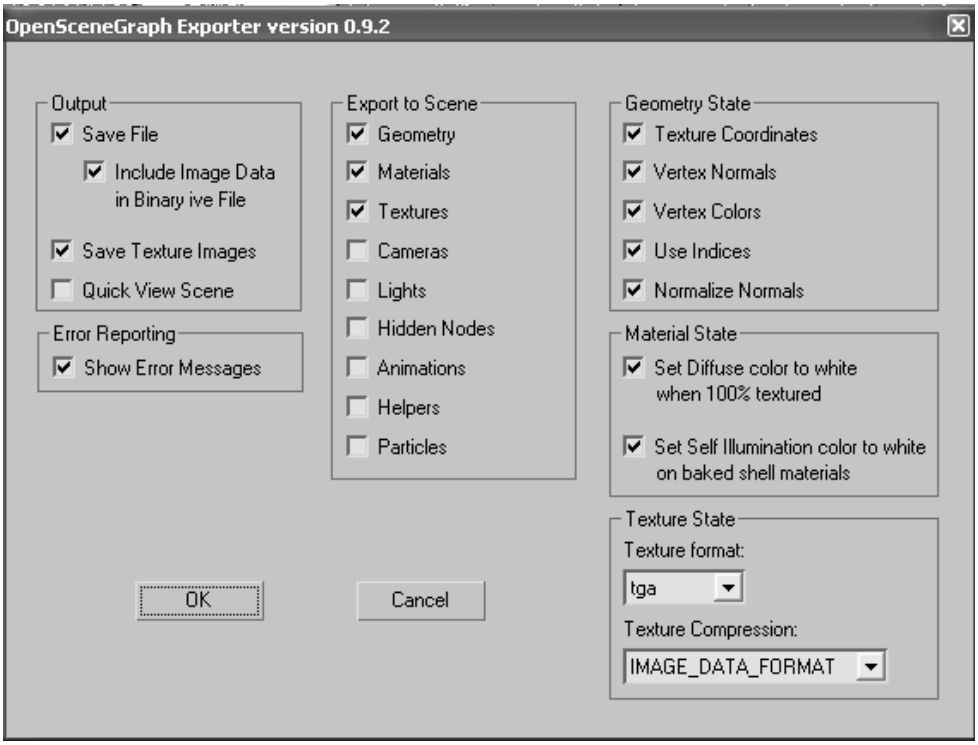
You then get a "Select File to Export" window looking like this.



Navigate to the Models folder in IKEA-viewer. Name the file and select "OpenSceneGraph Exporter (.IVE, *.OSG)" in the "Save as type:" menu.*

If you name the file as an **xx.osg** *file it will be stored as an OSG file, if no suffix is set after the name it will be an IVE file by default.*

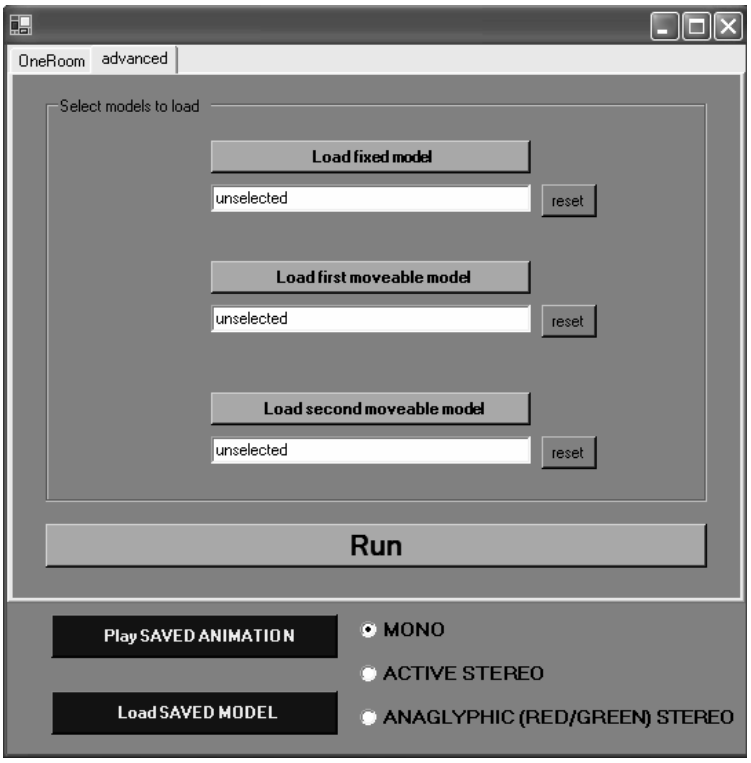
Press the "Save" button and the OSGexp interface pops up:



You can select various options using OSGexp but the above selected options can stand as a standard example.

Press "OK" to export the selected objects to the Model folder in IKEA-VIEWER.

In the IKEA-Viewer interface you can now under the "**advanced**" part load up to three parts of a model



Press the "Load …" buttons to get the Open window where you can select the objects in the Models folder.

Press "RUN" to start the program. The three model-parts will load over each other and appear as one model, but they will be loaded in three different branches of the open scene graph tree. This makes it possible for IKEA-Viewer to decide whether it should be possible to select and move an object or not.

The selection of objects loaded in the first part will always be fixed, while up to two other selections of objects will be moveable. If only one selection of objects is loaded (as when using "export" instead of "export selected") the whole scene will be non-moveable.

In the One Room example the building is loaded in the "Load fixed model" branch and the interior is loaded in the "Load first moveable model" and the customers in the "Load second moveable model" branches.

MONO or STEREO?
OSG and IKEA-Viewer supports in addition to normal mono viewing, also both active stereo and anaglyphic (red/green) stereo.

Mono is the normal viewing mode and should be selected when no stereo glasses are present.

Active stereo means that you need active stereo glasses that can switch from right to left eye synchronized to the appropriate image display by the screen or projector .

Anaglyphic stereo means that you can use cheap red/green glasses that can be found in any hobby-store.

Tips when exporting your own models

As understood from the above you should use 3ds MAX to make your selections of which parts of your model should be moveable. Note that if a couple of objects in 3ds MAX is grouped together, say a couple of chairs, all chairs will move if you move one of them in IKEA-Viewer. You'll have to ungroup them in 3ds MAX before you export them in order to move just one chair in IKEA-Viewer.

IKEA-Viewer will always initially place you in origo (x = 0, y = 0, z = 0) of the 3ds MAX world, facing north. This means that you should select the whole scene in 3ds MAX and translate it so that origo is located where you want to start in IKEA-Viewer.

If the model seems to be too big or too small when you open it in IKEA-Viewer you should rescale it in 3ds MAX. The ideal scale for IKEA-Viewer is **1:1000**.

OSG is a real-time player which means that exported lights **will not cast any shadows**. When you walk out of the region of a light everything will simply suddenly look darker. An easy way to make things look normal is simply to not export any lights and let OSG use its own default light, or to light up the whole scene by using four "direct lights" from above and one direct light from down under the model. This way you can set the color of the light to be a little bit yellow to get a warmer feeling in the scene.

By using textures and experimenting with the material editor in 3ds MAX you can also induce a specific atmosphere in the scene. Try out the controls for specular level, glossiness, soften, self illumination and opacity in the 3ds MAX material editor.

Try to keep the model as simple as possible; the frame rate when you run IKEA-Viewer will be highly dependent on the number of polygons in the scene. Moreover an infinitely thin object in 3ds MAX has only one surface, whereas it has two in the CAD format. This means that models converted from CAD to 3ds sometimes get double, or wrongly directed, surfaces. There are commercially available converter programs that are supposed to be exceptionally good at converting CAD models to the 3ds format, but in most cases opening up the CAD model in 3ds MAX will work sufficiently well.

Keyboard bindings when running IKEA-Viewer:

Basic keys

Up, Down, Right, Left:	Move around, if an object is selected-rotate it.
Shift bar	Double speed.
Ctrl	Hold down and use left, right to move sideways.
Left Mouse button	Select a moveable object.
Left Mouse button and Drag:	Select and move a moveable object.
Right Mouse button	Unselect
m	Toggle mouse-look (use the mouse to look around).
Space bar	Return to start position.
q	Save new start position.
c	Toggle collision detection.
p	Toggle walking style.
PgUp, PgDn:	Change eye height.
h	View help.
Escape button	Quit

Animations and graphics

z	Start recording camera path
Z	If recording camera path - stop recording camera path and save to saved_animation.path.
o	Write scene graph to saved_model.osg.
f	Toggle full screen.
l	Toggle lightning.
s	Toggle instrumentation.
b	Toggle backface culling.
t	Toggle texturing.
v	Toggle block and vsync.
w	Toggle polygon fill mode.

Camera manipulators

1	Select ‘trackball’ manipulator.
2	Select ‘flight’ manipulator.
Flight: a	No yaw when banked.
Flight: q	Automatically yaw when banked (default).
3	Select ‘drive’ manipulator.
Drive: a	Use mouse left, right mouse button for speed.
Drive: q	Use mouse y for controlling speed.
4	Select ‘terrain’ manipulator.
5	Select ‘walk’ manipulator (default).