Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, Volume 21, No. 1, ISSN 1726-9679
ISBN 978-3-901509-73-5, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2010
Make Harmony Between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals of DAAAM International

THE CREATING AND PREPARING 3D GRAPHICS MODELS TO USE THEM IN THE REAL-TIME APPLICATION

POKORNY, P[avel]

Abstract: *The important part of the modern 3D applications is the visual look. The bad graphics design can deprecate many users. This graphics design especially depends on the quality of the 3D graphics models and their surface - used materials and textures. But the high poly models and high quality textures are very heavy for the computer, whereon is this application running. This paper describes my experience with the possibilities of optimizations of 3D models, which can help to make real time applications with the high quality graphics design.*
Key words: *computers, graphics, modelling, visualization*

1. INTRODUCTION

The modern real-time applications are mostly complicated and large applications with the many functions and possibilities. The software developers also implements into their software comfortable user interface and suitable graphics design, which makes the best communication between the application and the user. The very important part of the graphics design is own graphics output. Many of application (like graphics programs or computer games) offer the three dimension visualization. But the best quality of visualization graphics brings much more system and hardware requirements of the computer, on which is this application running.
The most difficult situation arises in applications that visualize real world. One example could be a computer game. The actual story takes place in any environment. The virtual environment (world) can be formed by the ocean, mountains or natural landscapes. (Lander, 2003) Next, in this environment, there are exist hundreds or thousands objects (trees, stones, buildings, characters, etc.). And we want to visualize all of these objects and fluently animate in the real-time. The modern computers are very powerful computer, but they are not still capable to compute and render so much of data in the high quality in the real-time. Also we need to seek the ways how to optimize the 3D models and visualization algorithms.

2. 3D MODELS REPREZENTATION

The first problem is the 3D object representation. There are many types of representation, but currently the boundary representation is most used. The borders of objects are most represented by the analytic or by the polygons. (Zara, 1998)

The analytic boundary representation is based on theory of curves and surfaces. The parametric and implicit surfaces are very often used in the sphere of computer graphics. The greatest advantages of this representation are the accuracy and low memory economy. This representation is also often used in the CAD/CAM systems or architecture.

The polygonal boundary representation (fig. 1) is based on polygons. The base polygon is a triangle, but many APIs like OpenGL or Direct3D also supports quadrilaterals or general polygons. (Shreiner et al., 2006) (Walnum, 2006) This representation does not have the advantages like analytic representation, but the polygons are very suitable for visualization (render) algorithms like ray-tracing. The second advantage is the polygonal boundary representation. This is the standard supported by the hardware devices (graphics cards). This means, most of the operations with the polygons are accelerated. Also in the final phase, the analytic boundary representation is transferred into the polygonal boundary representation. (***, 1997)

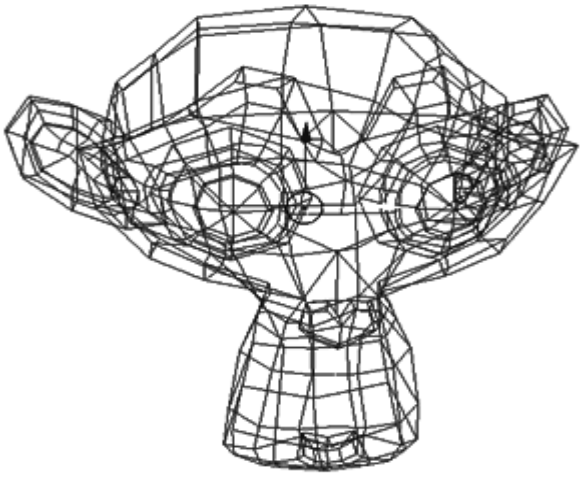

Fig. 1. The polygonal object representation (wire frame)

3. MODELLING

To create 3D models, we need to choose a suitable graphics program. There are many programs in this area, commercial and non-commercial, which have a greater or lesser number of tools for creating graphics models. The most sophisticated programs of non-commercial sector includes program Blender, with which I have much experiences.

Blender is the free open source 3D content creation suite, available for all major operating systems under the GNU General Public License. Blender has a large number of graphics tools, like modelling, shading, UV unwrapping, imaging and compositing and works with physics and particles, rendering, simple and complex animation. (***, 1998)

Blender contains a huge number of modelling tools for the analytic and polygonal representation. In the field of analytic representation here we can add and modify various 3D objects begins with simple curve or a circle and ending with a complex 3D surface. Manipulation of these objects is simple and it can easily connect individual curves or surfaces. Upon completion of this modelling is the representation usually converted into a polygonal representation, as stated in the previous paragraph.

For the models defined by a polygonal representation is available to an even greater number of modelling tools. In Blender, these objects are generally named as the mesh and being characteristic, that consists of vertices, edges and faces. In the simplest elements can freely manipulate and apply them

tools like Extrude, Subdivide, Smooth, Decimate, SubSurf, Spin or Modifiers. (fig. 2)

After creating 3D models, it is continued with the setting of the material properties, texturing (there is usually used UV Unwrapping for precise application of textures), and possibly animation. (Pokorny, 2009)

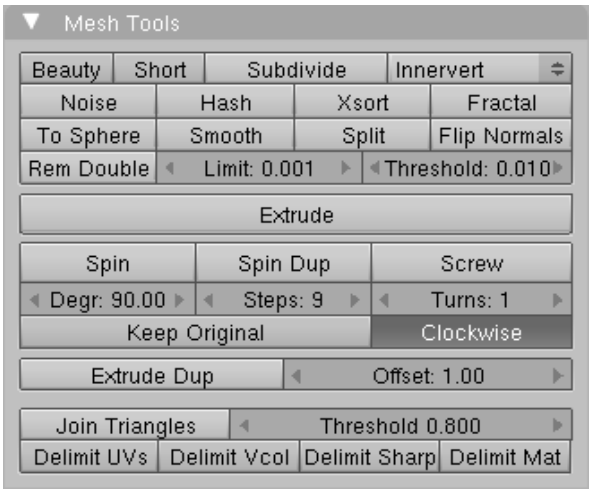

Fig. 2. Mesh modelling tools in Blender

Depending on the type produced by the model and its purpose is usually the entire process of modelling complex. If we create a model, that will be used in any application, it is not just important its visual impression, but also its optimization. This is primarily to minimize the number of vertices, edges and faces for faster rendering later, but at the cost to a minimum loss of model quality. Both requirements are contradictory, so they used all sorts of compromises to achieve as much as possible of both. This includes tools to minimize the number of vertices. This is normally addressed by creating two versions of the 3D model - low poly and high poly. High poly is the perfect model with the maximum number of vertices created so, that best suited to the object that is the subject of modelling. For him, then are textures unwrapped (specifically concerns the normal maps), which they are later applied to the low poly model. The low poly model is considerably simplified, but it was good applications can achieve a good texture quality very close to the high poly model. The application is later using of course low poly model with these textures. In Blender, this process is called Baking.

My personal experiences also show, that it is necessary to check the existing vertices (which often happens that the model contains redundant vertices, which later present computational process of rendering).

4. EXPORTING MODELS

After the 3D model is created, textured and possibly animated, we need him to pass and for use in our application. Perhaps all of the 3D graphics programs have its own format, which structure often is not publicly known or it is too complicated. Fortunately, there are existing 3D graphics formats that can be called universal and are certified for years in many applications. These formats include files with extensions .x (DirectX), .3ds (3D Studio), .bsp (Quake 3 levels) or .wrl (WRML). A big advantage of these formats is the available documentation and the number of libraries that are able to read or to write them well.

In Blender, all these formats can be found under the File-Export menu. In the ground state, it only exports alone mesh object. To export also the UV coordinates of textures, all of these textures must be mapped on a 3D model with UV mapping.

5. IMPLEMENTING INTO THE APPLICATIONS

Very important factor in the whole process of rendering complex 3D scenes in real-time is not only to optimize the 3D model itself and its texture, but also the appropriate organization of the entire program, especially the part, that deals with the visualization of the whole scene. The biggest impacts have the rendering algorithms and the administration of created 3D graphics models including textures. The entire organization and connecting between algorithms are very complicated, and everything depends on the skills and experience and the programmers, who created the particular application. In some cases, we can help out and finished the existing libraries and engines that may have already programmed most of the used algorithms.

An example of such an engine can be Irrlicht, with which I have very good experience. Irrlicht is a cross-platform high performance real-time 3D engine written in C++. It features a powerful high level API for creating complete 3D and 2D applications such as games or scientific visualizations. It comes with an excellent documentation and integrates all state-of-the-art features for visual representation such as dynamic shadows, particle systems, character animation and collision detection. All this is accessible through a well designed C++ interface, which is extremely easy to use. (***, 2003)

6. CONCLUSION

The modern real-time applications with the 3D graphics visualization can be more exacting for computers, on which they are running. By default, the frequent requirement of very good graphics needs more system and hardware computer resources. This is, why we are looking for the ways, how to get the best graphics quality in the real-time together with the minimal load of the computer. This paper describes my own experiences, possibilities, how to do it in two phases - the 3D models creating and exporting them into the own applications. My experience shows, that the software optimization is mostly limited by the possibilities of computer hardware.

7. ACKNOWLEDGEMENTS

This work was supported by the Ministry of Education of the Czech Republic in the range of research projects No. MSM 7088352102.

8. REFERENCES

Lander, J. (2003). *Graphics Programming Methods*, Charles River Media, ISBN 1-58450-299-1, Hingham, Massachusetts

Pokorny, P. (2009). *Blender – teach yourself 3D graphics* (in czech), BEN – technicka literatura, ISBN 80-7300-244-2, Prague

Shreiner, D.; Woo, D.; Neider, J. & Davis, T. (2006). *OpenGL programming guide* (in czech), Computer Press, ISBN 80-251-1275-6, Brno

Walnum, C. (2006). *Direct3D Programming Kick Start*, Sams, ISBN 978-0672324987, Indianapolis, Indiana

Zara, J.; Benes, B. & Felkel, P. (1998). *The Modern Computer Graphics* (in czech), Computer Press, ISBN 80-7226-049-9, Prague

*** (1997) http://www.opengl.org – OpenGL – The Industry standard for High Performance Graphics, *Accessed on: 2010-06-10*

*** (1998) http://www.blender.org – Blender, the free open source 3D content creation suite, *Accessed on: 2010-06-10*

*** (2003) http://irrlicht.sourceforge.net – Irrlicht Engine – A free open source 3d engine, *Accessed on: 2010-06-10*