Mécanique & Industries 5, 121–127 (2004)

DOI: 10.1051/meca:2004012

Mécanique & Industries

Virtual reality for aircraft engines maintainability

Aiert Amundarain[1,a], Diego Borro[1], Alex García-Alonso[2], Jorge Juan Gil[1], Luis Matey[3] and Joan Savall[1]

[1] CEIT (Centro de Estudios e Investigaciones Técnicas de Guipúzcoa), Lardizábal 15, 20018 San Sebastián, Spain
e-mail: {aamundarain, dborro, jjgil, jsavall}@ceit.es
[2] University of the Basque Country, Lardizábal 1, 20018 San Sebastián, Spain
e-mail: agalonso@si.ehu.es
[3] University of Navarra, Lardizábal 13, 20018 San Sebastián, Spain
e-mail: lmatey@tecnun.es

Received 30 June 2003, Accepted 20 October 2003

Abstract – REVIMA is a virtual reality system for maintainability simulation in Aeronautics. It comprises both hardware and software developments, plus system integration. REVIMA required the design of a new haptic system. It is used both to track hand movements and to return force feedback that provides the sensation of working with a physical mock-up. The main software modules are: image generation, collision detection and control. System integration is based on two LAN connected PCs that share the different tasks and data. The visualization module has been built using low-cost graphic systems, and we have thoroughly analysed this problem to achieve a drawing frame rate acceptable for simulation analysis. The models comprise more than two thousand different elements that require about two million polygons to describe their shapes. Different organization strategies have been tested in order to achieve the real simulation goal. We also present the different visualization algorithms we have used.

Key words: virtual reality / real time visualization / maintainability / haptic

1 Introduction

In the field of Aeronautics the term Maintainability is defined as "the ability of an element to keep in service or to be returned to adequate status in order to develop its function, after being maintained at conditions previously established, using the personnel, the means and adequate procedures" [1].

One of the most relevant aspects of maintainability concerns man and tool accessibility task analysis, which is undertaken in order to calculate paths and assembly-disassembly sequences and times. Design based on electronic mock-up is widely used in the creation of engine externals (piping, harnesses and installations) by the aeronautics industry. Pipes and harness are routed over these parts and accessories are installed by means of a workstation network. This allows a group of designers to work quasi-concurrently over an assembly, copying and automating the original process. This technology is known in the industry as DMU/DPA (Digital Mock Up/Digital Pre-Assembly).

DPA/DMU technology has overcome the need for a hard mock-up for design purposes, significantly decreasing time-to-market and thereby saving money. However, nowadays the use of a physical mock-up is mandatory in order to evaluate the maintainability of externals during the development stage. Although these mock-ups can be used for other applications, the ultimate purpose of the construction is to check the maintainability. The expenses of these mock-ups led ITP (Industria de TurboPropulsores) to research an alternative using haptics. ITP is the exclusive supplier of low-pressure turbines for Rolls-Royce engines of greater than 35 000 lbs of thrust – primarily the Trent engine family. It is also the Spanish participant in the EJ200 engine for the Typhoon Eurofighter, and earlier this year became a 13.6% shareholder in the TP400 engine programme for the A400M European military transport aircraft.

2 Description of the system

REVIMA is a system developed to check the maintainability of aircraft engines. The system has been created from scratch by CEIT Applied Mechanical Department.

[a] Corresponding author: aamundarain@ceit.es

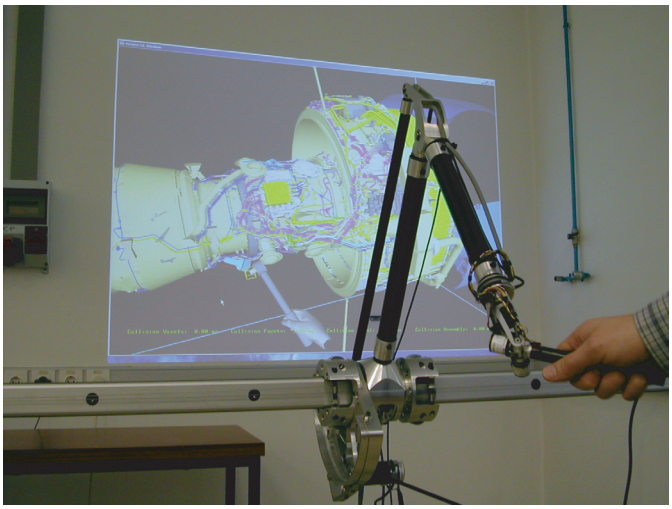

Fig. 1. Photo of the CEIT haptic in a virtual maintainability operation on a CAD aircraft engine model.

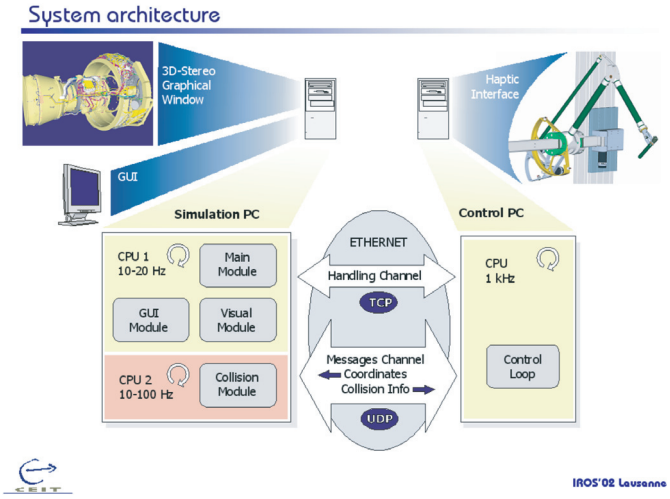

Fig. 2. System architecture of REVIMA.

This is a multidisciplinary development that includes, amongst others, the following disciplines: mechanical design, control theory, computer graphics, computational geometry and human-machine interaction.

The research involved in the project concerns two main areas: mechanical design and software development. Both of them deal with important challenges since system maintenance simulation needs to be very close to reality.

One of the main targets of the mechanical design was that the workspace of the haptic device should match that of an aircraft engine (see Fig. 1). At the same time, any haptic device had to have low inertia. Both requirements have been achieved by combining mechanical design with significant sensible use of a force sensor. The need of large workspace was established by ITP to perform ergonomic studies [2].

The haptic device designed and built for this project defines a cylindrical workspace whose dimensions are: internal radius, 232 mm; external radius, 772 mm; depth 1500 mm; angle 120°. These dimensions allow working in a 1:1 scale with the virtual engine aircrafts.

In turn, software development has involved the integration of a fast control loop that reflects force to the operator, the evaluation of collisions, and the visualization of the scene. The two last tasks are especially difficult because of the enormous size of the model (more than 2 million triangles). Section 3 describes this integration.

3 System architecture

REVIMA is a complex project that involves software (computer graphics, geometry computational, control) and hardware (mechanics) disciplines. The software has been developed using Microsoft Visual C++ and OpenGL. It runs on Microsoft Windows 2000. Its architecture is based in two PCs. The first one (control PC) executes the control module to command the haptic. The second one (simulation PC) runs the main, the Graphical User Interface (GUI), the collision solver and the graphics engine modules. Both PCs are connected through an Ethernet LAN network.

The control PC has a 233 MHz Pentium II. The simulation PC has two 866 MHz Pentium III Xeon processors with an Intense3D Wildcat 4210. To achieve the maximum frame rate in the visualization and collision modules, those tasks have been distributed between the two processors. Figure 2 shows the system architecture.

The simulation PC has two video outputs: one contains the GUI, the other one displays the 3D virtual scene. This second display can be connected to a monitor, a screen projector or both.

The communication between two PCs uses both UDP and TCP protocols. The UDP protocol is used in the normal and general case: with position messages and collision messages.

The positions messages send from control PC to simulation PC when the Control Module detects position or orientation changes. When the Collision Solver detects interferences, this module sends collision messages to Control Module. Even though, an UDP channel does not guarantee reliability, it is the most appropriate one for a real time application that makes use of a point-to-point connection. The system is not affected if it occasionally misses one of these messages.

When REVIMA needs a reliable communication channel, it uses the TCP protocol. Examples of this kind of messages are: starting the control module from the simulation PC, messages to grasp or release pieces from the assembly, notification from the control PC to the Simulation PC such as problems with the haptic interface, i.e., excessive heat of the motors, etc.

4 Visualization module

The visualization module generates images that replace the physical viewing of the mock-up. Our aim was to develop a program, where the virtual aircraft engine

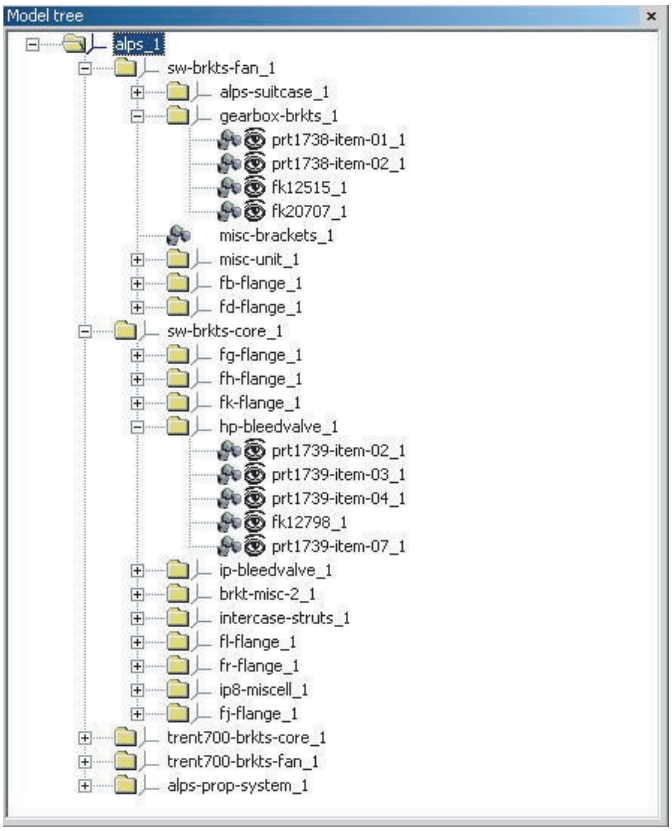

Fig. 3. Input hierarchical structure.

can be displayed in an interactive frame rate, more than 10 fps. The necessary steps to achieve an interactive frame rate have followed three ways:

1. find the most suitable structure for storing the scene;
2. analyze the most optimized graphics commands;
3. develop visualization techniques.

Before analyzing the visualization module itself, we shall present some experiences related to the source data. The system receives CAD data structures, without element limitation and based in nodes. This information is received loading a text file based in the "product structure" format. This file defines a structure where is stored the information that places spatially the model and defines the operational features of each element of the model.

The structure defined in the file is a hierarchical one (see Fig. 3). The structure has a root node, which can have an unlimited number of child nodes. Each node of the structure provides information for operational purposes, indicating which is its behavior in the simulation operations. A position matrix also is contained in the nodes that place spatially its descendents in the scene.

The leaf nodes structure contains the name of a graphic file that defines the actual geometry and topology of one part. These files can be VRML, GAF or STL format.

The aircraft engine digital models used in the virtual environments are very complex, with a great density of faces per element. An engine consists of around 2000 elements. The sizes of the elements are different. Some elements have few polygons as the bolts that consist of around a hundred polygons. On the other hand some elements are very big, more than a hundred thousand polygons are necessary to define them.

4.1 The most suitable structure

It is very important to find the most suitable structure for storing the information of the scene [3].

Due to the structure that is read from the files, in our first design we chose a sloppy n-ary space partitioning tree [4] to store all the geometrical information. This structure does not impose a fixed number of child nodes and the bounding volumes of tree nodes of the same tree level are not necessarily disjoint. These features make suitable this structure in relation to the one that is received.

After working with this structure, the frame rate achieved, less than 3 fps, was not as good as required. The problem was found in the input structure. This first structure was built following the operational structure, instead of geometrical properties. As a consequence, the visualization techniques that must be applied at the rendering process and which will be explained later, were not good enough.

The next step was to build a hierarchical structure where the scene was stored following spatial sorting. This structure was totally independent in relation to the input structure. Anyway both structures should be maintained, because the input structure is necessary for the simulation operations.

The chosen hierarchical structure was an Octree (Fig. 4). It has a valuable advantage: It supplies valuable spatial information about the scene. On the other hand, this structure has an important disadvantage. The objects of the scene must be split up to suit the structure. However, this action can be advantageous in big objects, because these objects usually have some parts of them hidden. If these objects are split up, we can identify the parts that are hidden.

However, we are still studying the problem in order to find automatically which is the optimal decomposition level for an arbitrary model. If we increase the number of levels of the structure, we can obtain more spatial information and we can implement the visibility techniques more efficiently. However, in that case, the objects are split up in more parts, which increase the memory requirements and decrease the graphic hardware performance.

4.2 Representation techniques

The rendering had to be done using the OpenGL API. So, it was important to find the fastest representation techniques that OpenGL could offer.

Two graphic cards have been used to render. One of the cards belongs to the family of the Wildcat GPU and

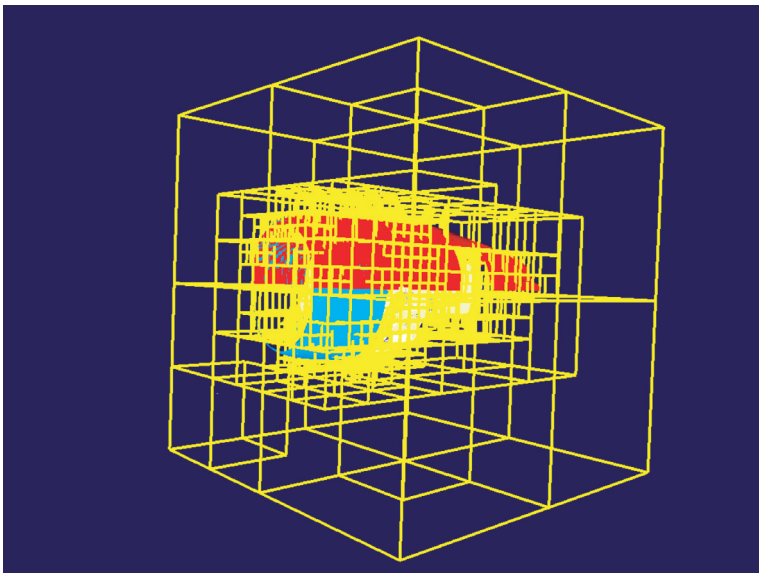

Fig. 4. An aircraft engine stored in an octree structure.

the other one was built up on the geForce3 chips of nVidia. The most interesting discovery is that in both cards the most effective representation techniques were different.

In the case of the Wildcat, the use of display lists was the most effective technique. The test we performed showed that using a display for each part was the most optimized method, reaching twice bigger fps than using immediate mode commands.

However this technique is not as fast as the use of the vertex arrays for the other cards. The reason of this difference is that the graphic cards based on the geForce3 chip, allow storing vertex information in video memory. This way the access to this information is fastest, and the rendering process is accelerated considerablely.

4.3 Visualization methods

The system has implemented several visualization methods. These methods when rendering the scene, try to use the minimum number of polygons.

The program has implemented simplification algorithms. We have implemented the quadric error metrics algorithm [5]. The input data specifies the degree of simplification for each element when the model is charged. The degree of simplification does not change during the simulation operations, because the entire scene has not enough depth to use the LOD technique [6]. The use of LODs also supposes to employ twice the necessary memory, which was not available considering the memory requirements of the collision module.

When the scene is displayed, all the elements must not be rendered. Some elements are not visible due to several reasons and other elements are not enough important because of their tiny contribution to the final image.

An element may not be visible for several reasons: it is outside of the viewing pyramid, it is hidden by other elements or the user has marked it as not visible. Also the faces whose normal points away from the viewpoint are not visible.

The elements whose projection is less than a threshold value are not rendered because it is considered that their contribution to the final image is not worthy. It is very important to identify these elements before calling the commands that render them.

The use of efficient hierarchical structures is very important to identify these elements. Bounding volumes of all the nodes of the structure are calculated. The use of these volumes makes easier the necessary calculus. The bounding volumes are spheres because they are the simplest and fastest for calculus purposes [7]. With a collision test between the bounding volume and the viewing pyramid can be known if the elements inside the bounding volume are or are not inside the viewing volume. If the volume is totally outside the viewing pyramid, all the elements are not visible; on the other hand if the volume is inside all the elements are visible. If both volumes collide, there is not enough information and the process follows in the next level of the hierarchical structure. In order to identify if an element contribution to the image is important, the same method is employed. In this case the bounding volume projection area is calculated.

If we use the octree structure, instead of the first one to store the scene, we can obtain important advantages in this stage. Due to the features of the octree structure and it is built following spatial information, the bounding volume of each node is smaller than in other structures and it contains more objects in it. This way the visibility computations are accelerated.

4.4 Occlusion methods

An algorithm that finds which objects are not visible has been also developed. Although we must stress that most of the algorithms used nowadays are not valid

within our application due to specific features of our virtual environment.

A lot of algorithms are based in an important preprocess step [8]. These algorithms usually precalculate the objects that are hidden from different possible camera positions. Therefore, at any moment the hidden objects are known just knowing the camera position. These methods are not valid for our program, because the user can vary the visibility state of the elements in run time; therefore the occlusion information obtained in the pre-process step loses all its value.

Either, the methods based in replacing part of the scene by images are not also valid [9]. The scene is not depth enough to replace some elements by an image without producing visual artifacts.

The working scene does not also fit to the methods that divide the scene in cells and portals [10]. These methods are used in architectural environments.

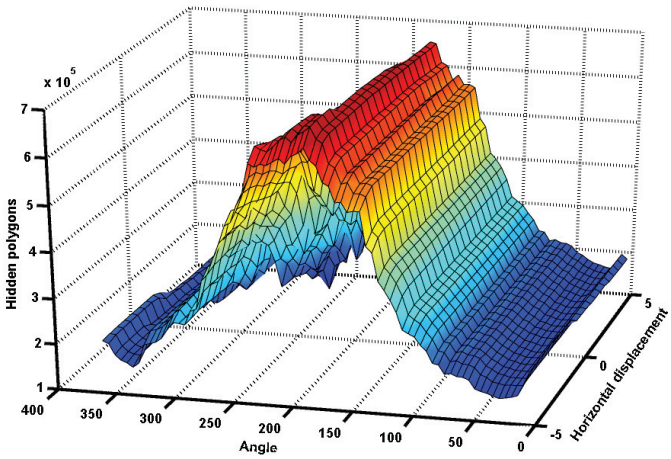

Fig. 5. Number of objects ($\times 10^5$) occluded for different camera positions.

4.5 Occlusion method implemented

Analyzing the main features of the working scene we concluded that the method that best fits is the one based in hierarchical occlusion maps [11].

This method calculates a set of possible occluders in a pre-process step. The set of occluders are different for different viewpoints. Good occluders for each point are those whose projection area is big.

When the set of occluders is projected to the screen, the area of its projection is made opaque. An occlusion map is a two-dimensional array. The information stored in an occlusion map indicates which part of the image has become opaque. Once this occlusion map is obtained, applying the average operator to rectangular blocks of pixels, we get occlusion map at different resolutions.

An estimation buffer is also built at every frame, which requires determining the depth value in every pixel where the occluders have been projected. This operation is very expensive, so, instead of calculating the exact value, depth estimations are calculated.

Once the information about the projected area and the depth is obtained the rendering step begins. The following two tests are performed for each element. The screen projection of the element is tested against the occlusion maps to see whether it is completely within the opaque area of the occlusion map. Also is compared against the depth information to determine if it is behind the occluders. If the element is overlapped and behind the occluders, then, it is occluded.

As we needed to improve the throughput of this technique we studied how to save computations, we realized that an important feature is that the aircraft engine virtual models have a cylindrical shape.

Another important property of our program is a special navigation system. To make easier the simulation operations the camera must be located on a cylindrical surface, looking at the center of the engine all the time. The movement of the tools drives the movement of the camera, this way the place where the operation is happening is always visible and the user have not to worry about controlling the camera.

The shape of the model and the navigation mode makes very useful the plane that is perpendicular to the viewpoint and crosses the center of the shape. Almost all the hidden objects are behind this plane. We avoid the step where the depth information is calculated, considering all the objects behind this plane as candidates to be hidden. From this group, the objects that are occluded by the occlusion map will be identified as hidden and will not be rendered

The occlusion map must be built only considering the parts of the occluders that are in front of the perpendicular plane. This is achieved replacing in the viewing volume the far plane by the perpendicular plane when we project the occluders.

In Figure 5 can be seen how many elements are considered as hidden for the different positions a camera can take.

5 Collision module

The collision module sends messages to the control module when a collision is detected among the objects. The control PC with this information calculates a force feedback that is applied by the haptic interface, inhibiting some user movements. This is a simplification of the problem. In Section 6 the problem of penetration and control is analyzed.

It should be noticed that the collision problem we are tackling with has two specific characteristics. The first one has a geometric nature. There are a reduced number of mobile objects (the hand and the grabbed tool or engine part) while the rest of the scene is a large set of static objects. Therefore, the mobile objects are only about a 0.5% of the total number of facets in the scene. The second one is the relative position between the static and mobile geometry: the facets of the mobile objects are usually close to a subset of static facets.

These two characteristics are common to other environments whose maintainability must be analyzed, and have a notable influence in the algorithms required when checking for collisions and penetrations. This environment is quite different from others that also deal with the collision problem: path finding, robotics, vehicles, virtual walks, even mechanism design.

The algorithm used is based on uniform spatial grid decomposition: voxels [12]. Each voxel can point to a list of facets that intersect with its volume.

Instead of building the voxel decomposition for each static object, we make only one structure, treating all the static objects as a unique solid [13, 14].

Here we present a similar method of [13]. We named it *MF* (Mobile with Facets). This method doesn't use voxels to describe the mobile objects; it uses only facets. The *MF* algorithm uses only two levels of accuracy: interferences among mobile facets and static voxels; and interferences among mobile and static facets.

In the first one, for each mobile facet it computes the subset of static voxels intersected by that mobile facet. If the subset of voxels has some facet, in a second level of accuracy, the algorithm takes the facets contained in those voxels and searches for interference among each pair of facets. We use an algorithm from Möller [15] to detect an intersection between two triangles.

Besides the detection problem, the collision solver also computes a collision response. This response is two values, the normal contact and the penetration between two 3D objects, and the collision module sends them to control module.

6 Control module

The control loop, located in the control PC, has a sampling period of 1 kHz. Most authors choose one sample frequency of 500 Hz or 1 kHz, according to the studies of Shimoga [16].

This module acquires the position and orientation of the tracking device and sends this information to the simulation PC.

Collision forces are calculated depending on the actual user position, the last collision information received from the simulation PC and the contact model in use. Figure 6 shows the data that make up the collision information: the contact normal $\mathbf{n}$, the penetration x and the tool position $\mathbf{p}$.

The contact model used in REVIMA is a simple spring [17]. The collision force is directly proportional to the penetration x of the virtual tool in the environment. It also has been implemented a friction model that follows the one described by Salisbury et al. [18].

Since the control loop runs faster than the collision module, different numerical strategies have been implemented to obtain a sufficient stiff and smooth touch. These strategies are based on the interpolation of the different values sent by the collision module [2].

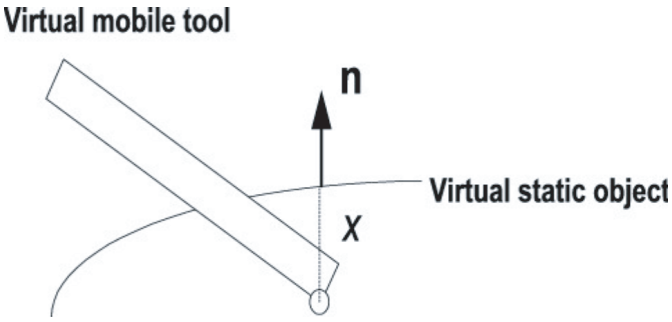

Fig. 6. Collision information.

7 Conclusions

We have wholly designed and built a new large haptic device. It is being used by the industry in 1:1 mock-ups like turbo engines. All the software that integrates the system has been developed at our laboratory.

The main conclusion of this work is that virtual reality systems with force feedback based on low cost hardware can be directly applied to the industry and that these systems may lead to important reduction in costs.

Another interesting characteristic of this system is that CAD models used in the design phase are also useful for simulation studies without manual or interactive preprocessing.

The used architecture and visualization policy has proven to be a good approach to treat the problem of managing large CAD models in real time. The specific characteristics of this problem allow us speeding up occluding computations.

There are two complete developed systems that have been validated and are currently in use by the industry.

Acknowledgements. The authors would like to thank to ITP and Sener for promoting and funding the application REVIMA. This application has been partially financed by the project of the Basque Government, number CI01TP03.

We would like to especially thank Iker Aguinaga, Emilio Sánchez, Angel Rubio and Jaime Rubí for their support and ideas in the development of this work.

References

[1] B. Blanchard, S.D. Verma, E.L. Peterson, Maintainability, John Wiley & Sons Inc., New York, USA, 1995

[2] J. Savall, D. Borro, J.J. Gil, L. Matey, Description of a haptic system for virtual maintainability in aeronautics, Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, EPFL, Lausanne, Switzerland, September, 2002, pp. 2887–2892

[3] M. Meißner, D. Bartz, T. Hüttner, G. Müller, J. Einighammer, Generation of subdivision hierarchies for efficient occlusion culling of large polygonal models, Technical Report WSI-99-13, Deparment of Computer Science, University of Tübingen, 1999

[4] D. Bartz, M. Meißner, T. Hüttner, OpenGL-assisted occlusion culling for large polygonal models, Computer and Graphics, Vol. 23(5), 1999

[5] M. Garland, P Heckbert, Surface simplification using quadric error metrics, Computer Graphics (SIGGRAPH 97 Proceedings), 1997, pp. 209–216

[6] D. Aliaga, J. Cohen, A. Wilson, E. Baker, H. Zhang, C. Erikson, K. Hoff, T. Hudson, W. Stuerzlinger, R. Bastos, M. Whitton, F. Brooks, D. Manocha, MMR: An interactive massive model rendering system using geometric and image-based acceleration, Proceedings of ACM Symposium on Interactive 3D Graphics (I3D), 1999, pp. 199–206

[7] D. Bartz, J.T. Klosowski, D. Staneker, k-DOPs as tighter bounding volumes for better occlusion performance, ACM SIGGRAPH, Conference Abstracts and Applications, 2001, 213

[8] C. Saona-Vazquez, I. Navazo, P. Brunet, The visibility octree. A data structure for 3D navigation, Computer & Graphics, 23, 1999, pp. 635–643

[9] D. Aliaga, A. Lastra, Automatic image placement to provide a guaranteed frame rate, Proceedings of ACM SIGGRAPH, August, 1999, pp. 307–316

[10] D. Luebke, C. Georges, Portals and Mirrors: Simple, fast evaluation of potentially visible sets, In ACM Interactive 3D Graphics Conference, Monterrey, CA, 1995, pp. 105–106

[11] H. Zhang, D. Manocha, T. Hudson, K. Hoff. Visibility culling using hierarchical oclusion maps, in Proc. of ACM Siggraph, 1997, pp. 77–88

[12] A. García-Alonso, N. Serrano, J. Flaquer, Solving the collision detection problem, IEEE Computer Graphics and Applications, 13(3), May 1994, pp. 36–43

[13] M. Held, J.T. Klosowski, J.S.B. Mitchell, Evaluation of collision detection methods for virtual reality fly-throughs, Proceedings of the Seventh Canadian Conference on Computer Geometry, Vol. 3, Québec City, Québec, Canada, August 1995, pp. 205–210

[14] W.A. McNeely, K.D. Puterbaugh, J.J. Troy, Six degree-of-freedom haptic rendering using voxel sampling, Proceedings of the ACM Siggraphm, Los Angeles, California, USA, August 1999, pp. 401–408

[15] T. Möller, E. Haines, Real-Time Rendering, A. K. Peters Ltd., 1999

[16] K. Shimoga, Finger force and touch feedback issues in dextrous telemanipulation: A Survey. Proc. of NASA-CIRSSE International Conference on Intelligent Robotic Systems for Space Exploration, NASA, Greenbelt, September 1992, pp. 159–178

[17] G.C. Burdea, Force and Touch Feedback for Virtual Reality, John Wiley & Sons, Inc., New York, 1996

[18] K. Salisbury, D. Brock, T. Massie, N. Swarup, C. Zilles, Haptic rendering: programming touch interaction with virtual objects, Proceedings 1995 Symposium on Interactive 3D Graphics, Monterey, California, 1995, pp. 123–130