Towards the Use of CAD Models in VR Applications

Alberto Raposo* Eduardo T. L. Corseuil* Gustavo N. Wagner* Ismael H. F. dos Santos*,†
Marcelo Gattass*
*TeCGraf – Computer Graphics Technology Group / Department of Computer Science
PUC-Rio – Pontifical Catholic University of Rio de Janeiro, Brazil
†Petrobras Research Centre - CENPES, Rio de Janeiro, Brazil

Abstract

One of the main objectives of the engineering departments of large industries is the construction of integrated information systems to control their projects, offering resources for the 3D visualization of their models with enough realism to be used for virtual prototyping, design review, change management systems, and training, among other activities. This work analyzes the main problems related to the production of Virtual Reality (VR) models derived from CAD (Computer-Aided Design) models, which allows the user to interact with them in real-time and have an immersive sensation. The paper presents ENVIRON (ENvironment for VIRtual Objects Navigation), an application that was developed motivated by the necessity of using VR in large industrial engineering models coming from CAD tools.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality; J.6 [Computer-Aided Engineering]: Computer-aided design (CAD)—;

Keywords: Virtual Reality, Real-time Visualization; Computer-Aided Engineering; Design Review

1 Introduction

Large industries, such as automobile, aerospace, and oil & gas, are investing in the construction of integrated information systems to control their projects. They look for computational systems that, besides accessing the databases with project information, offer resources for the 3D visualization of their models with realism.

The difficulties encountered in this process are basically due to the fact that the engineering models are not constructed to be visualized in real time. In some cases, the models are visually simplified representations, serving only as schematic representations of the characteristics to be analyzed. In other cases, which are the ones we are interested in, the models are too large and complex to be visualized in real time. For example, the generation of a CAD model demands that the objects forms be highly detailed, since it is aimed at the execution process, and does not consider the effects of this in the real time 3D visualization.

In the conversion process from CAD models to VR models, we may detect several problems regarding the visualization of the models:

- **Low performance for complex models.** The achieved frame rates are unsatisfactory when very complex models are loaded, especially in regions with large object concentrations. This is worsened by the fact that the CAD to VR conversion normally generates an unneeded model complexity, for example, by using improper algorithms, tessellating invisible details, making too fine LODs (Level of Details) or missing them [Stanney 2002].
- **Lack of realism.** CAD models generally don't have material and texture information associated to objects, although many CAD systems offer this possibility. This happens because this information is not essential to the building process, which is the aim of CAD models. However, this information is important for a realistic VR visualization. The application of textures in individual objects in VR models is normally not feasible, due to the complexity of the models.
- **Inadequate treatment of geometry.** During the conversion from the CAD to the VR model, there is normally a loss of geometry or precision [Berta 1999]. It is common to generate VR models with insufficient quality, with errors such as wrong side normals, cracks etc. Moreover, complex surfaces, such as NURBS (NonUniform Rational B-Splines) that may appear in CAD models, generate polygonal meshes in the VR model that are generally inefficient for visualization, demanding a different treatment.
- **Loss of semantics.** CAD models include project information linked to each object. Some systems loose this link in the CAD to VR conversion.
- **One-way conversion.** Generally, modifications realized during the VR visualization cannot be automatically transmitted back to the CAD model. For example, the results of a VR-based design review must be registered somehow and reproduced manually in the CAD system.

Due to those difficulties, it is necessary a chain of adaptation steps (usually manual) to adequately convert CAD models to VR ones [Berta 1999], [Paillot et al. 2003]. To the best of our knowledge, there is not yet a fully-functional integrated system where one can migrate from a CAD to a VR model (and possibly vice-versa), allowing easy interaction to make the necessary adjustments. The conversion processes currently available are direct and generally imprecise translations from CAD to VR, which are strongly dependent on the translated CAD format. In the absence of a standard for CAD to VR conversion, VR tools currently available use partial solutions, limited to specific CAD formats and quite dependent on user adjustments in the generated VR model.

In this context, this paper introduces ENVIRON (ENvironment for VIRtual Objects Navigation), a tool developed to facilitate the use of CAD models for VR applications. In this paper, we use the term "CAD model" referring to the engineering model, and "VR model" referring to the 3D model for real time visualization and interaction.

In the following section we present some related efforts in the combination of CAD and VR. In section 3 we present ENVIRON, fo-

*{abraposo, thadeu, gustavow, ismael, mgattass}@tecgraf.puc-rio.br

VRCIA 2006, Hong Kong, 14–17 June 2006.
© 2006 ACM 1-59593-324-7/06/0006 $5.00

cusing mainly on its visualization module. Section 4 concludes the paper.

2 Related Work

The integration of VR and CAD are investigated in two distinct lines of research. At one side, VR is viewed as an advanced form of human-computer interaction and used in common CAD tasks, such as picking, drawing, etc. This kind of tool, called VRAD (Virtual Reality Aided Design) tool, uses VR in the process of constructing the CAD models [Fiorentino et al. 2004]. In the present work, we are interested in the other kind of VR-CAD integration, which is that of facing VR as an advanced form of visualizing the CAD model in real-time and interacting with it in common CAD models usages, such as design review and training [Berta 1999].

The problems encountered in the latter VR-CAD integration form exist due to the diversity of purposes between CAD and VR tools. CAD tools are used to create detailed models, aimed at the execution process. VR tools are used to support activities with a high visualization demand, trying to provide the best possible immersion in the physical setting by means of the virtual model. In order to bridge the gap between both domains, very distinct VR-CAD integration solutions have been proposed. A possible way to analyze the solutions is regarding the coupling between the CAD and VR software [Vahl and Lukas 2003]. In this analysis dimension, we may distinguish four approaches:

1. Systems connected by means of gateways to ease the conversion process from the CAD model to the VR model. This is the most common approach, suitable for the majority of CAD and VR systems. In this process, CAD models are converted to a format suitable for VR, such as VRML. Normally this format is exported from the CAD system. The drawback is that this approach does not offer any solution to the problems mentioned above, such as inadequate treatment of geometry, loss of semantics, etc.

2. Definition of a common file format for both CAD and VR models. An example of such approach is the XMpLant, a "neutral" CAD format based on XML to describe process plants [AVEVA Group 2005]. However, XMpLant is more focused on CAD to CAD formats conversion. In the CAD-VR integration context, XMpLant's interoperability potential is clearly useful, but VR tools that will use it as input still need to process it in order to solve some of the mentioned problems of CAD to VR conversion.

3. Systems connected by means of an API. An example is the use of OMG CAD Services interface [OMG 2005], which is an interface standard, based on CORBA technology, to enable the interoperability of CAD, CAM (Computer Aided Manufacturing) and CAE (Computer Aided Engineering) tools. A prototype solution integrating this interface into a VR system is presented in [Vahl and Lukas 2003]. The current limitation of CAD Services interface is that is not widely adopted by CAD and VR systems, demanding a high implementation cost on both sides to integrate the CAD and the VR system with that interface.

4. Integration in one process. In this approach, the VR system is integrated into the core of the CAD system. This approach is certainly the one more capable of solving the problems related to the CAD-VR integration, since the VR system is capable of covering all of the CAD model functions. However, it is a vendor specific solution, lacking interoperability aspects present in the previous approaches. An example of this approach is presented in [Berta 1999].

ENVIRON follows a hybrid approach between categories 1 and 3 above. At the end, the information extracted from the CAD system is stored in a file format developed specifically for this system, similar to category 1 above. However, it is not simply a format exported from the CAD file. The format has been defined to include some relevant semantic information, such as the links with associated databases. Moreover, the information is extracted from the CAD system by means of APIs native to the CAD systems, which enables the extraction of the relevant semantic information (similar to category 3).

Another way to see the CAD-VR integration is by analyzing the coupling point between CAD and VR systems in their graphics pipelines. Since both systems generally have a complete graphics pipeline,"the level at which we cut one pipeline and hand-over the data to the other system determines the integration concept" [Vahl and Lukas 2003]. The considered graphics pipeline has four levels: Feature modeler, modeler, tessellator, and renderer. Based on this pipeline, there are five viable connection points:

a. From CAD renderer to VR image buffer. In this case, the VR system is used only as a way to relate VR-specific input and output devices to the visualization of the CAD system. The limitation is that the CAD system must have an efficient rendering performance to enable real-time visualization, which is not always possible because as the data is extracted from the CAD as it is needed, there is not much time to prepare the data for a more efficient visualization.

b. From CAD tessellator to VR renderer. This approach demands few computing costs from the VR system, since it is used only for visualization of the CAD objects, whose meshes are transmitted from CAD to VR. Although the VR renderer already provides some optimizations to enable real time rendering, this approach does not allow optimization in the tessellation, hindering, for example, an adequate treatment of complex surfaces for the VR visualization.

c. From CAD modeler to VR tessellator. In this case, the CAD system transmits graphical entities to the VR system, which is responsible to the tessellation. Although the VR system is capable of better preparing the surfaces for real time visualization, this approach still does not transmit important features of the CAD model, such as the topology of parts, and the relation with external databases.

d. From CAD feature modeler to VR modeler. This approach transmits parts and assemblies of the CAD model to the VR system, which should be capable of interpreting and making use of the semantic information associated to the geometry of the parts. The assemblies are transmitted through their geometric models.

e. From CAD project semantics to VR feature modeler. In this case, a compact description of the CAD features is transmitted to the VR system, which should be able to understand such features and create the appropriate geometric and semantic models.

From the above analysis, it is observed that moving from *a* to *e*, we offer more possibilities to the VR system to treat the problems of CAD-VR integration. However, the latter categories require the extraction of more information from the CAD models, what is not always possible, because some of them are closed formats, or because parts of this information are lost in the format translation. Moreover, the latter categories require more sophisticated VR sys-

tems. For example, a VR system in category *e* must be able to understand the complete CAD semantics.

Following this line of analysis, we can not say that ENVIRON fits in a single category, since we try to get the best of each CAD format and respective elements. For example, when working with Autodesk models, ENVIRON connects the CAD tessellator to the VR renderer (category *b*), since the latter receives the tessellated meshes. On the other hand, when working with PDS (Plant Design System) [Intergraph 2005] formats, ENVIRON fits partially in different categories. For some kind of geometries, the meshes must be received already tessellated (the same case of Autodesk models), while others are transmitted parametrically, and tessellated by ENVIRON. In the latter case, ENVIRON connects the CAD modeler to the VR tesselator (category *c*). The exported file format also includes some information regarding the objects' topology and more general semantic information, such as database links (categories *d* and *e*).

There are a number of VR tools being developed to visualize CAD models. However, to the best of the authors' knowledge, none of them provides a complete solution to the CAD-VR integration problem. Nevertheless, some tools provide good solutions to specific problems.

Among the academic solutions for the real time visualization of very large models, GigaWalk [Baxter et al. 2002] is one of the most important initiatives. This system uses scene graph reorganization techniques, such as hierarchical LOD and occlusion culling to achieve good visualization rates in CAD models with dozens of millions polygons. The limitation of GigaWalk is that, because of the necessary scene reorganization, it strongly depends on the model pre-processing, which may take several hours in large models. It is important to clarify that efficient pre-processing techniques may duplicate (or more) visualization performance, becoming quite important when the models are very large.

Another system with similar objectives is REVIEW (Real-time Virtual Environment Walkthrough) [Shou et al. 2001], which combines "conventional" visualization techniques–such as frustum culling–with large databases I/O optimization techniques.

Both GigaWalk and REVIEW have been developed focusing on the visualization of very large models. Commercial systems tend to have more generic objectives, and in general provide more resources to the VR visualization of CAD models.

Among the available products, we may cite Division Reality [PTC 2005] and Walkinside [VRcontext 2005]. Although they have different focuses, both aim at generating a VR experience from the CAD model. The Walkinside has a simple interface, based on games, with few commands and intuitive use of the keyboard and interaction devices. However, it is restricted to a couple of CAD formats. The Division Reality, on the other hand, accepts more CAD formats and has a direct manipulation of objects. However, it is not accessible by desktop computers, being used only in high performance graphic stations. Both have problems with complex surfaces, such as NURBS.

3 ENVIRON

ENVIRON is a system composed of a 3D environment for real time model visualization, and exportation plugins, which translate model data from other applications into a format that can be understood by ENVIRON.

This allows ENVIRON to view and interact with different kinds of 3D data, as long as they are in a format which ENVIRON may import or can be exported into one. Currently there is a DGN [Bentley 1995] and a 3ds Max [Autodesk 2005] exporter developed, respectively, as MicroStation and 3ds Max plugins. The goal of MicroStation plugin is not only to convert DGN files into a graphic format that enables the real time interaction and navigation in the CAD model, but also to recover and export the semantic information associated to the CAD's objects. In parallel, the 3dsmax plugin enables the use of more photo-realistic models, not necessarily generated by a CAD tool.

The initial step of the CAD to RV conversion process is the understanding of the CAD format and its internal structure. The DGN format was chosen as the first one to be exported to ENVIRON because it is the standard in important Brazilian industries, whose projects are generated in PDS–Plant Design System [Intergraph 2005]. We use MDL (MicroStation Development Language) [Bentley 1999] to develop the plugin that accesses the internal structure of DGN models and the data capture. The DGN internal structure is then exported by the plugin to another format, which we called TDGN. This format makes the link between the CAD tool and ENVIRON visualization module, since DGN full internal structure is only accessible by MicroStation tools.

DGN objects are exported in three distinct types: parametric objects, NURBS and triangle meshes. The use of these types for visualization is discussed in section 3.1.1. Some important information of DGN is included in TDGN. An example is the layer identification of each object, which is used to separate objects of distinct contexts in the CAD model. This information in TDGN associates the visualization to the project semantics and may be used to restrict the visualization to appropriate parts of the model.

Following the idea of bringing the design information to the VR visualization, it was also necessary to understand the relations between the information in the DGN file (graphic objects) and the information coming from the project database system. These links are also included in the TDGN format.

Although ENVIRON is committed to the visualization of CAD models, it was necessary to make it compatible with a tool capable of generating more realistic models. This necessity arises in non-technical and marketing presentations, where appearance is more important than accuracy. Therefore, we developed another plugin for exporting 3ds Max models.

3.1 Visualization Module

The visualization module of ENVIRON imports the formats that are exported by the Microstation and 3ds Max plugins and executes the real time visualization of the models.

To enable a robust and efficient scene structuring and to enhance the application performance, we use the OpenSceneGraph (OSG) [Osfield and Burns 2005]. It implements important optimizations for the real time visualization of models, such as frustum culling and techniques to reduce OpenGL state changes (lazy state change). Moreover, OSG is open source and has been showing extremely extensible. Some implementation aspects of the visualization module are detailed in the following subsections.

3.1.1 Handling objects for visualization

Objects are loaded from TDGN as parametric ones, triangle meshes, or NURBS. Due to the characteristics of process plants,

production units, such as platforms and refineries, there are objects such as pipe and pipe joints that may appear in large number in these models. These common objects are stored in TDGN using the parametric form. This way we reduce the file size and the memory footprint during the visualization, and enable the dynamic creation of meshes. The parametric objects support was integrated to OSG, being available for simple primitives, such as cylinders, spheres, lines, and curves, and their extrusion and revolutions.

For parametric objects, the information read is used to generate, when necessary, different visualization meshes, according to their distance to the observer. These objects enable the use of a LOD algorithm, implemented in the ENVIRON visualization module, that uses the parametric information of the object to generate, according to its position, the triangle mesh needed for the visualization. Therefore, in this visualization phase, there is a reduction in the required memory because no mesh is generated for objects that are far away from the observer. This process offers the advantages of an automatic LOD algorithm, without the necessity of a preprocessing phase.

The simpler parametric objects, like cylinders and spheres, are rendered using GPU vertex and fragment shaders. The shaders allow us to use the parametric parameters of each object to determine its shape, directly on the graphics card. The result is a perfectly smooth shape, without requiring an excessively tessellated mesh. In addition to improved rendering quality, these shapes also give us performance advantages over the CPU-based approach, which creates many different meshes for each object. As we have a fixed mesh, which does not need to be varied over time for each object, we are able to put them on Display-Lists and to group nearby parametric objects of similar types in clusters. Grouping small geometry in clusters is key to improving rendering performance on large scenes, once we avoid the repetitive reloading of vertex and fragment shaders. Additionally, the amount of processing used by fragment shader for each object is proportional to the number of fragments rendered, which is proportional to the space occupied by the object in the screen. This creates a kind of "intrinsic LOD", reducing the computational cost for rendering small objects.

In some cases, due to the difficulty of recognizing an object as a simple parametric form, such as a sphere or a cylinder, it is stored in the intermediary file in a mesh format and loaded directly by ENVIRON visualization module. Once these meshes are generated directly by internal functions of MicroStation (or 3ds Max), we have no control on their efficiency. In these cases, ENVIRON uses OSG resources directly, such as frustum culling and small features culling.

The third form that an object may be imported in ENVIRON is in NURBS form, which needs a special treatment. As previously stated, complex surfaces in the CAD model are a bottleneck in the generation of the VR model. CAD tools normally have limitations to generate models based on this kinds of surface. Generally, surfaces generated by these tools are not optimized for visualization, since they have an excessive number of definition parameters.

High precision surfaces are important in CAD models as their accuracy has direct impact on the quality of any information extracted from the model. For real-time visualization, on the other hand, most of this extra precision may be discarded.

Although CAD tools may offer functions for surface simplification and triangulation, they generally do not have the desired result, since these processes are not aimed at real time visualization. To reduce the general problem of displaying surfaces in real-time, we created an algorithm to make the pre-processing of the imported parametric surfaces and generate an adequate visualization mesh, with respect to performance and visual accuracy. The goal is to obtain a mesh which geometric representation is analogous to the exported surface, but with a simplified discretization compared to the original model.

Morevover, there is another problem besides generating a simplified mesh. In the real CAD models we worked with, some objects were represented as a combination of many unconnected NURBS surfaces, which when triangulated independently would create vertices which would not match each other, on the borders of adjacent surfaces. This creates an artifact called T-Vertex (Figure 1a), very noticeable in the generated images.

To eliminate these T-Vertices, we developed a special NURBS tessellation algorithm, which uses information from adjacent surfaces to match the vertices on their borders.

T-Vertices are an effect of the NURBS surfaces triangulation. By working during the tessellation process rather than by trying to fix an already existing mesh, we can take advantage of the information present on NURBS formulation, which is much more accurate and reliable than the discrete mesh. The algorithm works in 3 steps:

1. Evaluate a string of discrete points along the border of the NURBS surface.
2. For each surface, walk along the string of points that define its border defining which points will be used on its tessellation:
   - **a** If a vertex of the triangulation of the border of another surface is found in that position, force it to be used on this surface's border. Usually, only two surfaces share a single border: This step ensures that the vertices selected for the second surface matches the ones of the first.
   - **b** If no vertex from another surface is found, use a any traditional criteria for deciding whether this point should be used, like edge maximum length, maximum surface curvature or others.
3. Tessellate the surfaces using two methods:
   - **a** The inner portion of the surface is formed by a uniform NxM grid (areas formed by squared polygons in Figure 1b).
   - **b** The border of the surface is formed by a single strip of triangles, containing all the vertices selected on the border.

The resulting mesh has no T-Vertices, which makes it suitable for 3D visualization. For this implementation, a library for NURBS visualization, NURBS++ [Lavoie 1999], was adapted to the visualization module.

3.1.2 Object manipulation

An important feature for a VR system, when working with CAD generated models, is the ability to move, rotate and scale objects during the visualization, and also to measure distances. This is interesting for various purposes, like joining different models in a scene, testing the placement of equipment on a plant, or permitting the visualization of hidden portions of the model.

Currently, the main input device used for manipulating scene objects is the mouse, due to its universal availability. However, it has the inconvenient of being a 2D device, while the manipulations takes place in a 3D world. The 3D motion can be simulated by allowing the user to restrict its motion to a certain axis or 2D plane on the scene. This is achieved intuitively by using a manipulation gizmo, as shown in Figure 2. By choosing one of the three axes of

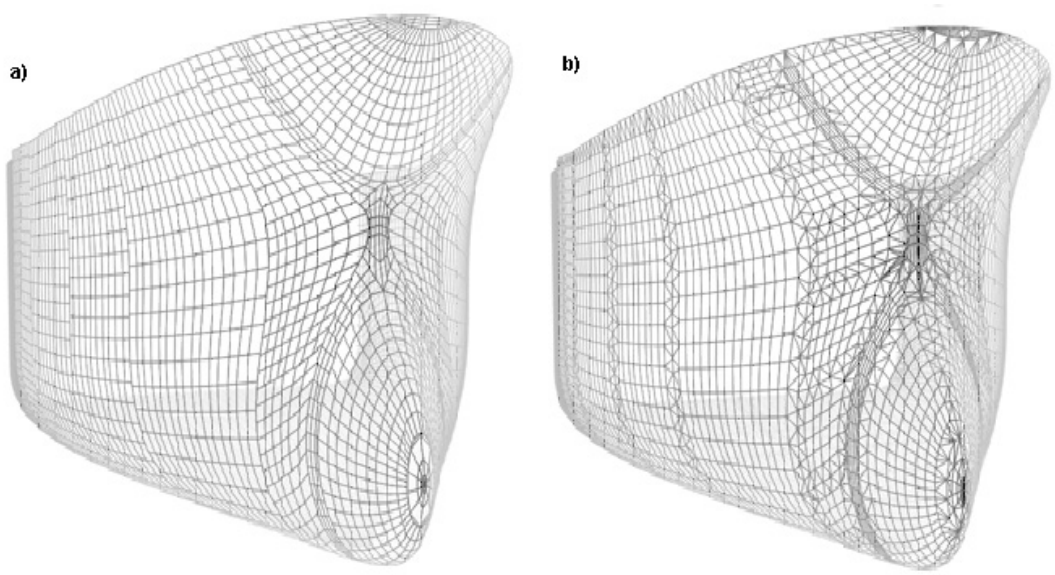

Figure 1: Shell of a real ship, composed of 90 NURBS surfaces. a) Wireframe view of the simplified surface with T-Vertices; b) T-Vertices eliminated after the application of the algorithm.

the gizmo, the user restricts its motion in the direction of that axis. If a pair of axes is chosen, motion is restricted to the plane formed by those two axes. Rotation and scaling works in analogous ways.

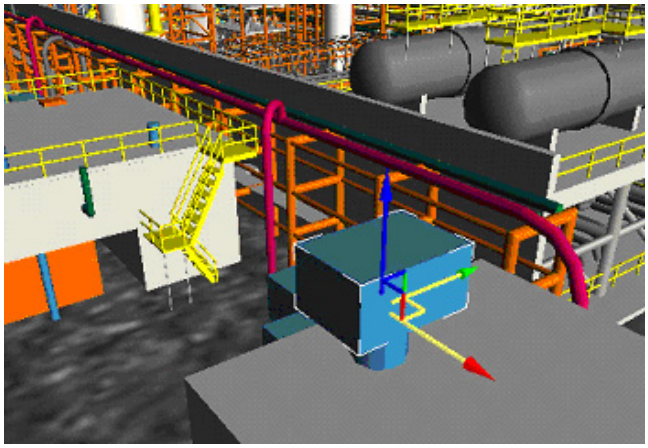

Figure 2: Moving an object in ENVIRON.

Other 3D input devices, such as 6DoF trackers, may be used to manipulate the objects on the scene when using ENVIRON attached to ViRAL, a VR framework for device abstraction (Section 3.1.4).

3.1.3 Ambient elements

In addition to a good representation of the CAD model, another important aspect is its insertion in a realistic environment. Regarding this aspect, ENVIRON integrates dynamic representations of the sky, terrains, and the ocean. Since most of the models we work with are offshore oil platforms, special attention was given to the ocean simulation.

The ocean simulation used is based on a simple yet effective implementation, working in two steps: First it uses terrain rendering techniques to generate a base mesh which, on the second step, is animated with a Gerstner Wave [Tessendorf 2002] generation model.

The most basic idea behind any terrain rendering engine is to assign more triangles to areas which are nearer to the camera than to areas which are far away from it, which is exactly what we require for a realistic ocean simulation. Additionally, terrain engines usually try to ensure that regions with sharper variations are assigned most triangles than regions with smoother variations. This is not important for our ocean simulation, and can be ignored for our algorithm. As in a terrain rendering algorithm, our ocean mesh is generated recursively, by successive subdivisions of a base squared surface. Regions nearer to the camera are subdivided more deeply, and the borders joining adjacent regions with different resolutions receive a special treatment, to avoid the formation of T-Vertices.

Additionally, resolutions transitions are blended smoothly into each other, by using geomorphing, which avoids that vertices which won't be present on a lower resolution region simply disappear abruptly. Glitches are avoided by moving these vertices progressively to a position on the surface of the face that will occupy its place, before the resolution changes.

The wave simulation is obtained by the use of Gerstner Waves, which consists of a simple summation of sinusoidal waves. Each vertex on the surface of the ocean is displaced according to the equations 1 and 2 below:

$$\vec{x} = \vec{x_0} - \sum_{i=1\ldots N} \left(\vec{k_i}/k_i\right) A_i \sin\left(\vec{k_i}\cdot\vec{x_0} - \omega_i t\right) \quad (1)$$

$$z = \sum_{i=1\ldots N} A_i \cos\left(\vec{k_i}\cdot\vec{x_0} - \omega_i t\right) \quad (2)$$

$$k = 2\pi/\lambda \quad (3)$$

$$kA > \lambda \quad (4)$$

In equations 1 and 2, $\vec{x}$ and $\vec{x_0}$ are 2D vectors corresponding respectively to the displaced and original position of the vertex on the ocean's 2D plane. Variable z represents the vertical displacement for that vertex. Vector $\vec{k}$, named wavevector, represents the wave's direction. Constant k is calculated from the wavelength (equation

3). The frequency of this wave is represented by λ and its amplitude is given by A. In order to avoid inconsistent waves, we must always respect inequality 4. Figure 3 shows the redered ocean.

Figure 3: The ocean rendered by ENVIRON, and its configuration dialog.

Experiments indicates that the impact of the ocean renderization on overall performance is relatively small, specially in large models. For example, in a Pentium 4, 3GB RAM, nVidia Quadro FX1000, using a VR model with 2.3 million polygons, the rendering rate was 12.9 fps without the ocean and 11.5 fps with the ocean.

3.1.4 Other visualization features

One of the key problems in the CAD to VR conversion is the loss of semantics. Regarding this aspect, ENVIRON is capable of accessing the information stored in the project database associated to the CAD model. This is possible if the key to the objects identification is present in the original CAD model. This capacity offers better parameters for the user to evaluate the model and to analyze possible alterations (Figure4). For example, it is possible to identify the composition of a process plant sub process based on the identification of their objects that are defined in the database. It is also possible to generate exploded views of the model [Niederauer et al. 2003].

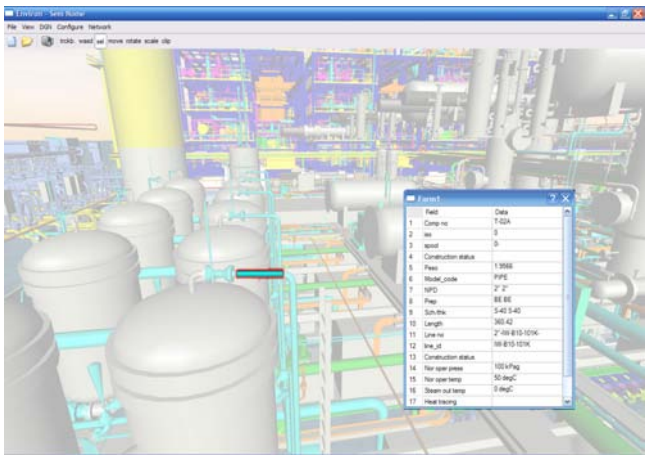

Figure 4: Visualization of data information associated to the objects.

To enable the use of ENVIRON in VR environments with different kinds of devices, we developed an independent tool, called ViRAL (Virtual Reality Abstraction Layer), which may also be used with other VR applications. ViRAL is a tool designed to facilitate the development of VR applications. Applications developed with ViRAL are device independent, because ViRAL abstracts the context in which the application will be executed. The application doesn't have to know, for example, in how many windows, with how many users, or with which devices it is going to function. All these incognites are configurations defined by the application operator, making it possible to execute the same application in different VR systems without the necessity to modify the application. ViRAL has a graphical user interface used to configure all devices, displays, projections and scenes.

In essence, ViRAL has the same objectives of any other abstraction device tool, such as VRJuggler [Bierbaum 2000] or Diverse [Kelso et al. 2002]. However, ViRAL has at least two major differences with other VR toolkits. The first one is that it can be run as the VR application. The developer creates plugins and uses ViRAL to load and run them. The second important difference is that ViRAL can also be used embedded in a host application, as is the case of ENVIRON. The configuration panels of ViRAL can be shown from within a third party application. This way ViRAL is not the running application, but all its VR abstraction features can be called from the host application. The use of ENVIRON combined to ViRAL enables the presentation in fully immersive multi-display VR settings, as well as the use of 6 DoF input devices.

In order to give the user a better visualization control, some parameters may be redefined in real time, during the interaction. For example, the near and far cutting planes and the FOV may be redefined at any time (Figure 5). The same applies to the configuration of the stereo visualization.

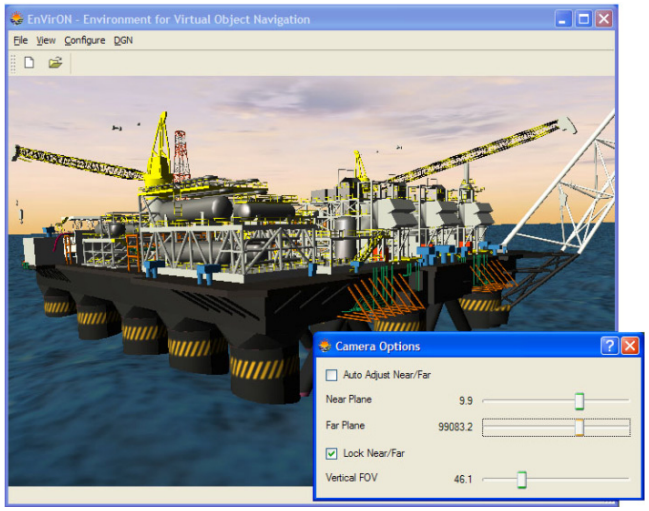

Figure 5: ENVIRON screenshot.

Since the navigation in a 3D environment is not an easy task, specially in places with a high density of objects, as is the case of industrial plants, ENVIRON provides the possibility to store navigation paths. These paths may also be used to automatically generate stereo AVI files, that can be viewed independently of ENVIRON.

An important step toward expanding the use of VR in industry is to integrate other systems or environments to the VR model converted from the CAD. The idea is to bring other classes of data to the VR model in order to build a more complex virtual environment.

An example we are currently working on is in the integration of a particle system to simulate, for example, the effects of an oil

spillage. Another example, which is the long term goal of ENVIRON, is to integrate to the virtual environment not only the CAD model, but also the model coming from GIS (Geographic Information System) systems, representing the surface of a region, and seismic data sets. In the specific case of oil & gas industry, the integrated virtual environment would be an oil exploration area, including the CAD models of ships, platforms and undersea equipments, the GIS models that represent the oil exploration basin, and the representation of the subterranean reservoir.

4 Conclusions

ENVIRON is part of a research initiative to create an infrastructure for the "immediate" generation of VR environments from the CAD models, a task that currently requires a significant effort from the VR teams in industry.

In this context, ENVIRON has been designed to be an extensible tool, with flexibility to receive new functionalities and to incorporate plugins, according to the different necessities of the industry. This conception is the opposite of that of the available commercial solutions, normally offered as "black boxes", with enhancements implemented by the developer on demand. We are trying to continuously enhance ENVIRON with the advances in the field. One of the next steps is to develop a collaborative version of ENVIRON to enable not only remote users to collaborate, but also co-located users, interacting simultaneously via PDAs.

Among the main problems of the CAD to VR conversion mentioned in section 1, ENVIRON has been particularly well succeeded in the treatment of geometry, specially of complex surfaces. Regarding loss of semantics, the connection to the databases is an important step in this direction. Regarding scene realism, ENVIRON already offers some important resources, such as GPU programming with specialized shaders, and the possibility to use models from 3ds Max.

Regarding performance, ENVIRON was developed based on CAD models of the oil & gas industry. More specifically, models of platforms and FPSOs (Floating, Production, Storage and Offloading), which are production and storage platforms adapted over oil ship shells. The data were furnished by Petrobras (Brazilian Oil & Gas Company) and were generated by PDS. Although ENVIRON performance is good for a great part of available CAD models, this is perhaps the subject that still deserves more attention, especially because ENVIRON is designed to run in desktop computers and to visualize CAD models with increasing complexity.

Finally, the last problem mentioned in section 1, which is the conversion from the VR model back to the CAD model, has not been treated in this research yet.

The problems presented in the CAD to VR path have not been totally solved by any commercial or academic solution yet. This fact restricts, but doesn't impede the use of these systems in industry. The process is still executed in parts, some of them requiring manually interventions, which are specific for the working model. Nevertheless, there are already advantages in the use of VR models generated from CAD models in processes like construction simulation, training, and equipment maintenance.

The problems presented will be minimized when standards for the CAD to VR conversion are established. Meanwhile, the effort presented in this work, as well as other efforts occurring in parallel are valid in the search for these solutions.

Acknowledgements

Virtual Reality research at Tecgraf/PUC-Rio is mainly supported by CENPES/PETROBRAS and FINEP (CT-Petro project).

References

AUTODESK. 2005. 3ds Max. http://www.autodesk.com.

AVEVA GROUP. 2005. XMpLant. *http://www.aveva.com/ media_centre/library/datasheets/vnet_xmplant.pdf*.

BAXTER, W., SUD, A., GOVINDARAJU, N., AND MANOCHA, D. 2002. Gigawalk: Interactive walkthrough of complex environments. In *Eurographics Workshop on Rendering*, 203–214.

BENTLEY. 1995. MicroStation 95 Ref. Guide, ch 18. Bentley Systems Incorporated: Intergraph Standard File Formats.

BENTLEY. 1999. MDL Function Reference Manual. Bentley Systems Incorporated.

BERTA, J. 1999. Integrating VR and CAD. *IEEE Computer Graphics and Applications 19*, 5, 14–19.

BIERBAUM, A. D. 2000. *VR Juggler: A Virtual Platform for Virtual Reality Application Development*. Master's thesis, Iowa State University.

FIORENTINO, M., MONNO, G., AND UVA, A. E. 2004. Smart tools for virtual reality based cad. In *ADM-AIAS International Conference*.

INTERGRAPH. 2005. PDS - Plant Design System. *http://ppm.intergraph.com/pds/*.

KELSO, J., ARSENAULT, L., SATTERFIELD, S., AND KRIZ, R. 2002. DIVERSE: A Framework for Building Extensible and Reconfigurable Device Independent Virtual Environments. In *Proceedings of IEEE Virtual Reality*, 183–192.

LAVOIE, P. 1999. NURBS++: The Nurbs Package - User's Reference Manual Version 3.0. http://libnurbs.sourceforge.net.

NIEDERAUER, C., HOUSTON, M., AGRAWALA, M., AND HUMPHREYS, G. 2003. Non-invasive interactive visualization of dynamic architectural environments. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, ACM Press, 55–58.

OMG. 2005. Computer Aided Design Service, V 1.2. *http://www.omg.org/technology/documents/formal/cad.htm*, Object Management Group, January.

OSFIELD, R., AND BURNS, D. 2005. Open Scene Graph. http://www.openscenegraph.org.

PAILLOT, D., MERIENNE, F., AND THIVENT, S. 2003. Cad/-cae visualization in virtual environment for automotive industry. In *EGVE '03: Proceedings of the workshop on Virtual environments 2003*, 315–316.

PTC. 2005. Division Reality. *http://www.ptc.com/*, Parametric Technology Corporation.

SHOU, L., CHIONH, J., HUANG, Z., RUAN, Y., AND TAN, K.-L. 2001. Walking through a very large virtual environment in real-time. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, 401–410.

STANNEY, K. M., Ed. 2002. *Handbook of Virtual Environments: Design, Implementation and Applications*. Lawrence Erlbaum Associates Inc., ch. 62 – Engineering Applications – O. H. Riedel, D. Rantzau and R. Breining.

TESSENDORF, J. 2002. Simulating ocean water. In *SIGGRAPH 2002 Course Notes #9 (Simulating Nature: Realistic and Interactive Techniques)*, ACM Press.

VAHL, M., AND LUKAS, U. 2003. Integration of virtual reality and cad based on omg's cad services interface. In *European Concurrent Engineering Conference (EUROSIS - European Multidisciplinary Society for Modelling and Simulation Technology)*, 54–61.

VRCONTEXT. 2005. Walkinside. *http://www.walkinside.com/*.

RAPOSO, A. B., CORSEUIL, E. T. L., WAGNER, G. N., SANTOS, I. H. F., GATTASS, M. Towards the use of cad models in VR applications. **Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications - VRCIA**, p. 67-74. Hong-Kong, China, 2006. ACM Press.