Virtual Reality System for Industrial Training

David Martínez Oliveira, Sandra Castro Cao, Xulio Fernández Hermida, Fernando Martín Rodríguez
ETSIT Communication and Signal Processing Deparment
University of Vigo
Vigo, Spain
Email: dmartin@uvigo.es, sandra.castro@uvigo.es, xuliofh@uvigo.es, fmartin@tsc.uvigo.es

***Abstract*— In this paper, a Virtual Reality Training system for maintenance of industrial equipment is presented. Since equipment are usually designed with CAD tools, it is feasible to create the Virtual Reality model. Despite of the virtual reality system itself, seting up an industrial related training system implies several processes in order to transform initial Computer Aided Design model into interactive models suitable for training processes. Our virtual training system is being developed following two main implementation approachs. On one hand a commercial tool (Eon Professional) is being used. On the other, the same system is being developed using open-source tools. Both approaches are described in this paper.**

I. Introduction

The application of Virtual Reality[1] (VR from now on) technologies in training processes has been shown as an interested topic [7] in the industrial environment. There are several and quite different reasons behind this interest.

In first place, VR technologies try to provide *realistic* representations of real world, so they are shown as a cheaper solution to currently training based on real mock-up models.

In second place, the use of Computer Aided Design (CAD) has become generalised in the industrial environment last years. This implies that 3D models are available and so, a lot of work is ready in order to build VR environments.

Finally, 3D representation of the model and interactivity with it seems to be a more natural learning media than plain documents, blueprints or fixed videos, making interaction between the student and the model richer.

This communication improvement gets even more outstanding if the VR environment provides simulation facilities, allowing the trainee to interact with the system in an realistic way, even before the equipment is fully designed.

Therefore, VR technology within an industrial environment makes tasks like training (using virtual mock-up models) or simply communication (using VR simulations) much more easy and rich.

VR technologies involve several knowledge fields, allowing different levels of implementations which fit to different user requirements. So, VR systems can run on a VR stereoscopic cave with a haptic interface or can be run in a simple desktop computer driven by a mouse. It all depends on the intended actions to be carried out. Independently of the chosen environment, which will determine the level of user immersion in the system, the interaction and simulation of the models is a very powerful tool from early design stages to final customer training systems.

Note that these last features of VR systems, interactivity and simulation, are what make them different from directly using the CAD tools. There is a third feature of VR systems to take into account: simplicity. A VR tool, within the context stated above, must be very simple and usable by non special trained personnel. By the fact of being working with a realistic model and using an intuitive interface (just the user hands) the user has the sensation of being almost in the real world.

Once the advantages of VR technologies in the industrial environment have been stated and the difference between VR and classical CAD tools has been made clear, it is time to expose the problem arising on setting up a VR simulation from a CAD model.

There are three main problems arising from this transformation, i.e. conversion from CAD to a virtual model, realistic user interaction and content generation tools. These problems are faced in Section II, III and IV.

After stating the implicit problems of setting up a VR environment from CAD information, two different approaches to implementation of this kind of system are proposed in Section IV.

II. From CAD to Virtual Models

As pointed out in the introduction, the first problem to be faced, is the conversion of CAD models to what is commonly called virtual models. In this process, a simplified version of the CAD model is generated to allow its interactive management by the VR system, where speed is more important that accuracy in order to provide a realistic response of the system.

The complexity of this transformation will depend on the way the CAD model was generated. For our purpose, the CAD model must be organized in function of the *objects* which will be finally manipulated by the VR system. The CAD designer main concern is drawing the parts of the machine or equipment. For instance, if two pieces of the machine are jointed by a screw, it is very possible that the pieces do not have a hole for the screw because that hole is not neccesary for the intended use of that CAD model.

There is a lot of work to be done in this stage. There are also very different strategies to follow depending on the CAD information available. This section describes the processing of transformation followed in our system. This processing of the CAD model can be considered quiet general at least for complex machines composed of several different pieces.

A. Choosing an Interchange File Format

The first problem to cover is choosing a suitable file format to carry out the CAD to VR transformation. Three main approaches can be stated for this task:

- Work on the native CAD file format. This approach implies supporting several CAD formats which is in general a complex and expensive task. It may be considered when a small number of CAD formats will be managed. That is, if all the CAD models are in a unique format, it does not worth to translate to an intermediate format, being easier to work with this native one.
- Using IGES[2] file format. The IGES file format was introduced as a flexible solution for CAD data interchange. However, its flexibility makes it to derive in several different versions which are not completely compatible among them. Different CAD tools export different IGES versions and, in practice the situation is similar to the previous case, where *different* CAD formats need to be supported.
- Using STEP[3]. STEP is another file format oriented to the CAD data interchange. This format is fully standardised by the IEEE and a lot of work is being put on its development. From all the CAD file formats tried in our development, STEP was the one providing us the best organization of the CAD model for our purposes.

Once the CAD input file format was chosen, it must be transformed in a VR oriented file format. For our development VRML[4] and 3DS[6] (Autodesk 3DStudio) file formats where chosen. Both formats provide a mesh-based representation of the objects which allows fast rendering in current graphics hardware. This initial transformation from STEP to these file formats was carried out by EonCAD Tool.

EonCAD is a software tool to convert among different 3D file formats, supporting several input and output filters (as DWG, STEP, IGES, VRML, ...). This tool is comercialised by EON Reality Inc, and now it can be bought directly from Deep Exploration Inc. as Deep Exploration 4.0.

B. Generating a model for VR interactivity

The previous step produces a whole model representation as unrelated meshes composed of vertexs. This means that if a given model uses some kind of screw en several parts of the model, an independent vertex mesh is generated for each screw. In order to set up an interactive model for our objectives (training, manipulation, etc...), this set of vertex meshes must be parametrised.

The parametrisation process will produce a model for VR interactivity (sketched in Figure 1) composed of the following elements:

- A set of unique meshes required to represent the model
- A set of primitive objects. Each primitive object is composed of:
  - A reference to the mesh it belongs to
  - The 3D position of this mesh within the model
  - The 3D orientation of this mesh within the model
  - A set of metadata information which will be discussed later in this paper.
- An optional set of logical objects, where each logical object is composed of a list of primitive objects

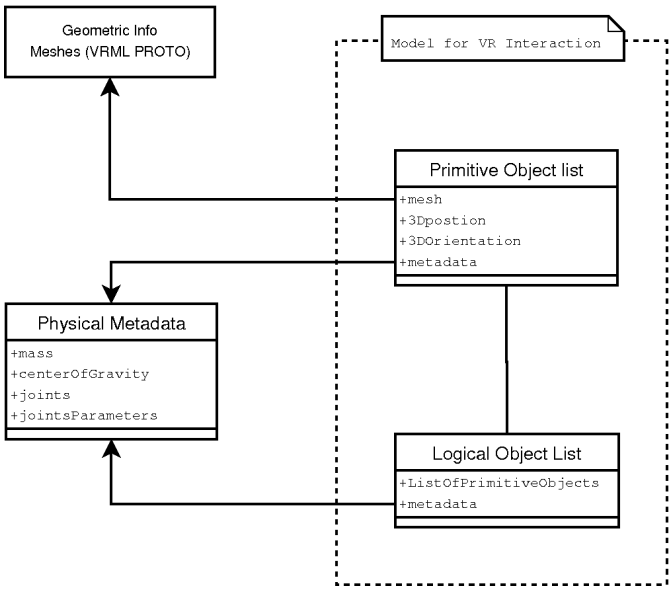

Fig. 1. Interactive Model Data Structure Diagram

1) Matching the model to the VR Training system: Note that in our data structure definition, primitive objects and logical objects are arbitrary entities which will depend on the training process to be produced. The availability of primitive objects (the minimal unit of interest in the model) will make definition of logical objects easier. Logical objects, are simple higher level units only making sense in a given training process.

So, data structures can vary in several ways between two end cases. One of them considers any mesh in the model as a primitive object, despite of the fact that several meshes are rotated versions of one unique item (for instance a screw).

The other one builds a full parametrised model of the model which will provide to the final training application a higher level of control on the model, allowing it to work with types of parts of the model, instead of concrete parts of it.

In other words, depending on the set of primitive objects generated, the final application can produce these two kinds of instruction:

- Put screw 1023 in hole 1 of plate 25
- Put a metric 8 screw in hole 1 of plate 25

So, full parametrization of meshes into a set of primitive objects will produce more natural training sequences. Additionally, full parametrisation of primitive objects will provide the highest interaction level with the final user, in the sense that any unit of the model may be manipulated.

Note that the set of logical objects will depend on the final application. This set will represent the units of interaction in the final system, so it is completely arbitrary since this kind of information is commonly not included in the CAD model. The main concern of CAD designers is, in general, not related to

produce a training tool. CAD designer produce CAD data for concrete tasks related to specific processes in the construction of the equipment.

So, in order to produce a complete interactive model the following algorithm is applied:

```
for each mesh_i in model
  calculate statistics st_i
  for each object_i in vmodel
    if st_i == object_i.stats
       mesh_i is a member of object_i
       add it to vmodel as primitive object
         position and orientation from st_i
    if st_i != object_i.stats
      mesh_i is a unique mesh
      add mesh_i to unique mesh list
      create primitive object from st_i
```

This simple algorithm will produce a virtual model with all the information to define the set of logical objects required for any given training application.

In this algorithm, `mesh_i` represent each of the meshes in the model to be manipulated. For each mesh a set of statistic values (`st_i` moments up to order 4) are calculated to obtain the parameters which will allow us to identify instances of a given geometric structure independently of its orientation and position.

A first filter based on the number of vertex of each mesh is applied for a coarse grained classification and then statistic parameters are used for fine grained classification. Higher order moments allows us to determine object orientation respect to a default reference orientation (orientation respect z axis).

At the end of the process, a list of unique geometric meshes is obtained along with a set of list of positions and orientations of each instance of each mesh in the model. This data is named `object_i` . The whole virtual model containing the list of all instances of `object_i` and the `object_i` full list of primitive objects is named `vmodel`

As can be seen, there is no automatic procedure to group primitive objects into logical objects. This is an arbitrary procedure which requires human interaction. Therefore, in order to complete the processing of CAD data, special authoring tools are required to allow the user to include the missing information within the interactive model which will be used in the final application. This last step is discussed in Section III

C. Output File Format

Now, that the data structures appropriated for the training system development have been created they must be saved on a file in order to be used by the training tool.

During development three major choices were studied:

- Proprietary Format
- VRML
- X3D[5]

Current implementation is working on a simple proprietary format, however a solution based on community standards will be better, making easier the access to our system.

A first choice is to store our virtual model using VRML. In this case, the following rules will apply.

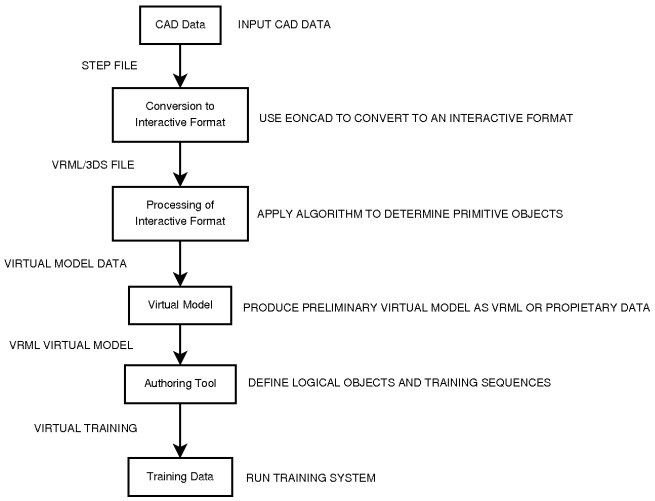

Fig. 2. Block diagram of the process to get a Virtual Model from CAD data

- Each mesh in the virtual model is stored in the VRML file as a prototype.
- Each primitive object is stored as a prototype reference in the main model definition of the VRML file.
- If logical objects are defined (this is a manual process), they are stored in the VRML file as prototypes, since they will be managed as a whole in the final application.

For each logical object definition, the set of primitive objects associated to it are removed from the main VRML model and substituted by a unique reference to the new logical object prototype.

Since VRML is being substituted by the X3D specification, it makes sense to use this last format for our training system. However, X3D is not a very common format nowadays. An in-deep analysis of this format and its conformance with our VR training system requirements is under study.

The whole transformation process is sketched in figure Figure 2.

III. Contents Generation

Next step in the generation process of a VR training application is to fulfill all the missing information which could not be extracted from CAD data or inferred in any way. This information is summarised below:

- Definition of the logical objects of interest for the final application
- User interaction points and animations. This information will allow the final application to control the user interaction, defining which parts of the model will be interactive, when they will switch active and which response or possible interaction will result from user action.
- Addition of physical information to each logical object if physical simulation will be required in the final application. This process involves the addition of metadata to each logical object (mass, center of gravity, ...) and the parametrization of interesting joints among logical objects

The first two sets of interactive information just require the author tool (or content generation tool) to provide an interface to the instructor to fulfill the missing data. Third set of information is more complex and requires a more sophisticated interface.

So, an authoring tool for VR training systems should provide, as a minimum, the following features:

- Load of interactive models generated according to the procedure described in Section II
- Allow user to add physical properties to primitive objects
- Allow user to group primitive objects into logical objects
- Allow user to define joints between logical objects
- Allow user to define interactive regions in the model (typically logical objects)
- Allow user to define animation on logical objects and associate them to interaction regions
- Allow user to define a sequence of simulation and associate messages to the different steps in the sequence.

For the knowledge of the authors, this kind of tool does not exists at the moment of writing this text. A list of available solutions is listed in Section V about related works.

Our current development covers the first set of information and allows the definition of a simplified set of user interaction and result animation, where simple animation actions can be associated to logical objects in a sequential way. At this moment a physical simulation engine is being integrated in the system to improve the user interaction with it. But up to now, this beta version is not fully functional.

IV. Implementation. Two Approaches

After setting up all the functionality and in order to address the main problems related to application of VR technologies to training processes in industrial environments, our current implementation of an author tool will be discussed using a real world use case.

In this use case a ship fuel warmer CAD model was manipulated to produce a VR training environment for maintenance procedures of this ship piece. Figure 3 show the full CAD model within one of our training tools.

The original CAD model was provided using the STEP file format and then transformed to a 3DS file using Eon Reality Deep Exploration tool. From this point on, two main approaches where followed to produce a VR Training environment.

First approach is based on the Eon Professional tool which provides a powerful graphical interface to build scenegraph based applications. The second approach builds the VR training tool from scratch using open-source tools. Next sections describe in detail each one of these two approaches.

A. Eon Professional

It has been developed a VR training tool based on EON Reality´s tools. The first step was to generate the interactive model of the equipment using the CAD transforation tool from Deep Exploration. This application allows to translate among a number of file formats, including EON´s proprietary

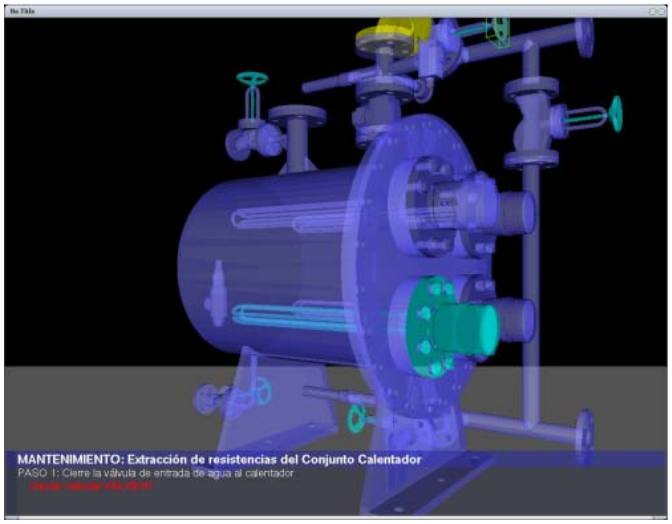

Fig. 3. Ship Fuel Warmer shown using our VR Training Tool Prototype

one, eoz (the project working format for EON Studio or EON Professional). From the CAD model in STEP format, it has been transformed in an eoz file, reorganizing the nodes in the model tree in order to group the meshes corresponding to individual objects, and grouping objects to obtain logical objects. The aim was to identify "operational objects", that is, objects that the user could interact with during the training sequence.

Once the interactive model was obtained, the interactive sequence is created into EON Professional. This tool presents a visual environment where objects, behaviors and user interfaces are treated as nodes in the scene tree. These nodes are adjusted via a series of parameters, defining their properties and functions. The underlying philosophy of the application is to generate an event - oriented programation of the simulation, defining what will be the response of the virtual environment to the user actions and/or previous internal events.

Taking this into account, the training sequence was created observing the following directives:

- The sequence is divided in sequential steps, so next step is started only if the previous one is correctly finished
- In each step only the objects involved with the tasks to fulfill will be manipulable by the user
- If one step has more than one task to fulfill without precedence order, all the combinations in the order will be valid

The training sequence was created manually using the visual environment of EON Professional (see Figure 4). Although very intuitive and easy to use, when the complexity of the simulation grows (due to large models and/or complex training sequences), the manual configuration becomes hard. Automation of the training sequences' configuration is under study.

B. Open Source

The open source version of the VR Training tool was programmed from scratch implementing the 3DS file format

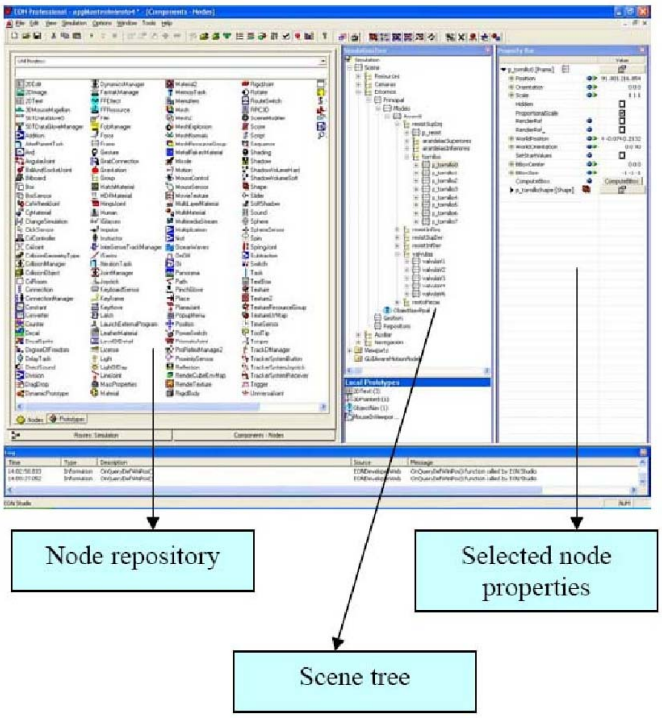

Fig. 4. Eon Professional Main Screen

parser as well as the rendering routines. The tool allows to define logical objects from the loaded 3DS model. However, this first version does not locate every possible primitive object. This feature is in development and current version treats each mesh in the model as a primitive object, which introduce some level of redundancy in data structures but it is a completely valid approach for our application.

The application parses a simple text file which defines the different steps involved in the training sequence. For each step the file provides a simple text description plus a set of actions which must be carried out by the user in order to advance to next step in the sequence.

Each action involves the following items:

- A Brief description of the action itself
- A reference to the logical object involved in the action the user must perform
- An animation sequence of the logical object, involving translation and rotations.

The main executable file is 100Kb long, and its memory requirement depends on the size of the model being loaded (training sequence size is negligible compared with model size) since all the memory allocations are dynamic.

The whole system can be distributed in a small USB memory stick including the whole operating system. A slightly modified version of *Damn Small Linux* was adapted to hold the application and the 3D driver required for hardware accelerated graphics. So the whole system can be distributed in a 128Mb memory stick supposing the 3D model involved in the training sequence can be hold in it.

This distribution solution has several advantages respect to traditional software installation.

- The whole system is distributed with the application so there is no problem about unmet dependencies
- It can be run on any machine with a minimal hardware requirements despite the installed operating system or any other software requirement
- Data and software are distributed together in an unique device
- If training data is classified information it will never go out of the USB device, which should be marked according to data classification

V. Related Works

In [7] it was presented a system of virtual training in the area of aeronautical engineering. This system was very similar to the one presented in this paper, allowing the user to follow a maintenance sequence using a virtual model of the equipment and the maintenance manual. The system was programmed from scratch without the use of any commercial tool.

Version 8 of Adobe Acrobat 3D[8] allows to include 3D models within .pdf files as well as fixed animations of them. Although it is not a training tool, documents generated this way improve traditional training based on handbooks.

VI. Conclusions

In this paper a comprehensive analysis of the steps to change a CAD model into a Virtual Interactive model was stated. Our main interest is to generate interactive models suitable for their use on training systems, however the process sketched in this paper can be used and extended for other kinds of VR applications.

A set of software elements to carry out parts of the defined procedure were developed and tested against real world models producing the first prototypes of functional VR training tool. This training tool was developed following two different approaches: the first one based on commercial tools and the second one based on open-source tools.

VII. Future Lines

There are two main future lines to follow from this point on. In first place, further research on how CAD models are developed will allow us to produce a simple set of guide lines which CAD designer may follow in order to make more easy the transition from CAD to VR. This is the first step to a full integration of VR technologies within the production procedures of current companies.

This guide lines will be augmented with extensions to currently available CAD files which, as a minimum, will allow to introduce VR side information within the production cycles. This is very important in order to avoid repetition of manual processes when in design stage, where models change quickly, and on later engineering changes on advanced product stages.

Even when our main research target at this point is training systems, the technology being developed can be applied to

several other fields where similar model processing is required. Some of the most important ones are:

- Ergonomic analysis
- Collaborative Design
- Large Structures build planing

The second future work line covers the development of a fully training system along with a complete authoring tool to help industrial producers generate VR manuals for their customers, including interactive maintenance procedures, virtual repair and any other task interesting for them.

The development of our tool is planned to be improved in the following ways:

- Support of VRML and X3D file formats for virtual model input data
- Integration of standard scenegraphs for rendering
- Integration of VRPN[9] system to support different user input interfaces

The inclusion of these features (current work in progress) will allow our virtual training tool to work on different user environments ranging from simple desktop computers to full VR systems including stereographic screens, 3D tracking systems and any other VR related hardware.

REFERENCES

[1] Grigore C. Burdea, Philippe Coiffet. *Virtual Reality Technology*. Wiley-IEEE Press; 2 edition (2003)

[2] National Bureau of Standards. *NBSIR 80-1978: Digital Representation for Communication of Product Definition Data*. 1980

[3] ISO. *ISO 10303. Standard for the Exchange of Product model data.*. 2004

[4] Web3D Consortium. *VRML97 (ISO/IEC 14772-1:1997)*. 1994

[5] Web3D Consortium. *X3D (ISO/IEC 19775-1)*. 2005

[6] Autodesk Ltd. "3D-Studio File Format (.3ds)". 1997

[7] Cristina Sánchez y Victoria García (I.T.P.). "REVIEN. Realidad Virtual Aplicada al Entrenamiento."*IV Congress in Virtual Reality Applications*, 15-16 June, 2006

[8] Adobe Web page. "Acrobat 3D" `http://www.adobe.com/products/acrobat3d/`.

[9] Russell M. Taylor II. "Virtual Reality Peripheral Network". University of North Carolina. `http://www.cs.unc.edu/Research/vrpn/`