Eurographics Italian Chapter Conference (2007)
Raffaele De Amicis and Giuseppe Conti (Editors)

VR Studio: Solid Modelling in a Virtual Reality Environment

F.Cappello[1], T.Ingrassia[1], M.Romano[1]
[1]Università degli Studi di Palermo, Italy

Abstract

Today one of the most important industrial needs is to improve the design process, making it faster and more robust; of course such a requirement can be obtained by using virtual reality. In the field of mechanical design, in fact, there is a great demand for more intuitive and natural modelling systems to speed up the modelling process. All that can be reached by stereocopic visualization tecnology and 3D input systems. In this paper we present a software for 3D solid modelling in a virtual reality system. The CAD models, carried out by this software, can be studied in their natural scale thanks to a large screen display. Moreover the user is free to move his hands to sketch the models thanks to a 3D wireless input device.

Categories and Subject Descriptors : I.3.7 [Three-Dimensional Graphics and Realism]: Virtual Reality

1. Introduction

Since 1943, when the ENIAC (Electronic Numerical Integrator and Computer) was created, computers are used to increase men productivity.
Nowadays designers can rely on high performance computers and on very complete and complex software. Among all the existing software, some of the most importants, with no doubt, are the CAD (Computer Aided Design) systems, essential tools for every designer.

Figure 1 : *Design review process in a VR environment*

During the recent years, CAD software have reached great quality, but they still retain a limit in that they are mainly two-dimensional in their interface. All 3D models, in fact, are analyzed through 2D input and output devices (the mouse and the monitor display) but, of course, this solution does not seem the best one. Undoubtedly, it should be more natural, for the user, handling 3D CAD models as if they are really in front of him. Nowadays, to satisfy such a requirement, virtual reality (VR) tecnology [BC03] and 3D input devices [SM97] can be used.
Virtual Reality, in fact, is yet being used more and more for many indutrial applications related to the design review of virtual prototypes (figure 1), the study of post processing data of numerical analyses [CTL01], the simulation of assembly procedures [JCL97], etc.. Nevertheless many past and current research activities are also addressed to the integration of CAD technologies, both regarding the free-form modelling of curves and surfaces [SFDM02] [DBSL02][CILG06], and the solid modelling [DG97] [CLL94], in virtual reality environment [ESS*99][LPA94].
In this work, following the guidelines which have been emerging in the last years, related to which sees spreading of virtual reality applications in the design process, we present the system we have developed, called VR Studio, that integrates some typical CAD tools in a VR environment. The system here presented allows to create solid models by simply moving user hands in the three-dimensional space and to perceive their three-dimensionality thanks to the stereoscopic display tecnology adopted.

2. Hardware Setup

The application here presented was developed at the Virtual Reality Lab of the Department of Mechanics of the University of Palermo. VR Studio uses the most advanced technologies in the market, namely:

- A Onyx 4 Workstation of Silicon Graphics (SGI), with 4 MIPS CPUs, 4 GB of RAM and 2 ATI FireGL X graphics cards;

- 2 Barco projectors;
- A 3m x 2m Stewart large screen;
- An A.R.T. infrared tracking system.

The Onyx 4 Workstation is one of the most advanced system for the management of 3D computer graphics [FvD], thanks to the high-end hardware, the two video pipelines and the IRIX OS. For many years, up to the spreading of more and more powerful graphic cards at low costs, SGI workstations represented the undisputed leader in the computer graphics area.
VR Studio uses the A.R.T. Tracking System [ART], composed of a flystick (flying joystick), used as the 3D input device, three infrared cameras and a pc for data elaboration. On the flystick there are four buttons and a mini digital joystick. The main functions are carried out with the trigger (button 0), whereas the scene camera is controlled by the mini joystick.

3. Software Setup

The software architecture of VR Studio is divided in three modules:

- one to display the models and to interact with the user;
- one to create the geometries and to store them;
- one to read input data.

To develop these modules three high level APIs (Application Programming Interface) have been used: Open Inventor, Open Cascade and Trackd API (figure 2).

3.1 Open Inventor

Open Inventor [OI94] is a multiplatform C++ API based on OpenGL. It is a powerful tool for developing interactive 3D applications like VR Studio. Moreover, Open Inventor allows to use the hardware acceleration for stereoscopic visualization (Raw Stereo mode). The three-dimensional scene built with Open Inventor is described by means of a tree graph (the scene-graph) [Wer94]. The nodes of this tree can represent a surface, a shape, a transformation, a light, a camera, and so on. The scene is rendered [FvD] traversing this scene-graph.

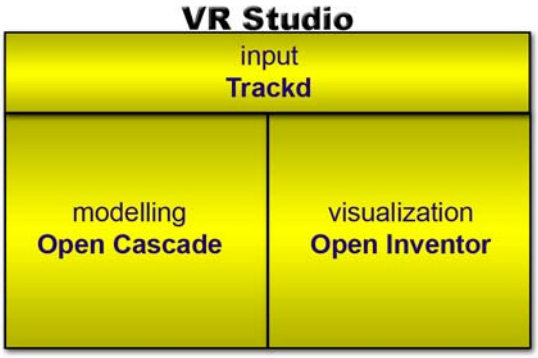

Figure 2 : *APIs used in VR Studio*

3.2 Open Cascade

Open Cascade [OC] is an open source platform to develop C++ 3D modelling applications. It is the modelling kernel of VR Studio. The user input data is sent to the modelling kernel, that creates the solids and store them in the VR Studio database. The model visualization is performed by Open Inventor.

3.3 Trackd

Trackd is a middleware that allows to capture data from several tracking systems and it makes it available to other applications. Trackd is composed of a server daemon, that runs on the pc, and of a client daemon, that runs on the IRIX workstation. In this case the input data, managed from the pc connected to the A.R.T. Tracking System, is sent to the VR Studio system through Trackd system.

4. Functionalities

VR Studio allows to create 3D models by means of typical functions like extrusion and revolution. The main available features are:

- Surface selection for 2D sketching of the profile,
- 2D sketching of the profile by segments and arcs,
- Point snapping to help the user drawing the 2D profile,
- Revolution and extrusion of the drawn profiles,
- Camera management by the 3D input device,
- Opening and saving of files in BREP and IGES formats.

All these features are accessible by 3D menus integrated in the immersive environment. The user can move a menu inside the scene by holding the trigger with the wand on the title bar and by dragging it.

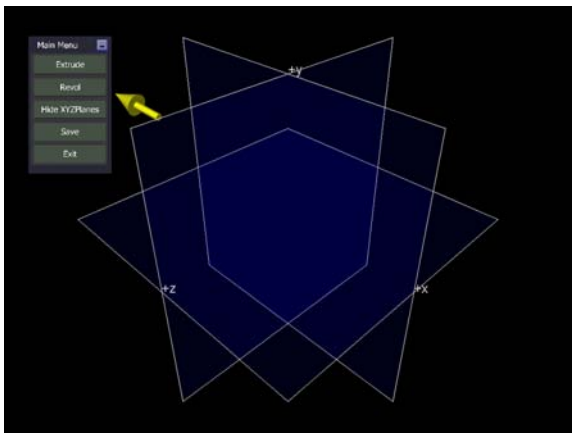

Figure 3 : *VR Studio workspace*

The menus, moreover, show also the user some information about the current operation (like the segment length or the extrusion depth) thus allowing a very precise modelling process. At the start of a working session, the workspace is

composed of three orthogonal main planes, the main menu and the wand (figure 3).

4.1 Surface selection

After selecting the operation to execute (extrusion or revolution) from the main menu, the user must select the plane surface that will be used to draw the 2D profile. At every 3D input device movement, VR Studio checks if the wand is over a plane. If this condition is satisfied, the selectable surface is highlighted and the user can select it by pressing the flystick trigger. VR Studio allows, also, to select plane surfaces of imported IGES models created with others CADs.

4.2 2D profile sketching

After having selected the plane surface, the user can draw the 2D profile for the creation of a revolved or extruded solid. The profile must be a closed wire. It can be composed of segments or arcs. For this purpose the user can select the kind of curve to draw by a 3D menu, called "2D Toolbox". This menu shows some essential informations to draw a profile with exact dimensions: this menu shows the user the length of the segment or the radius of the arc he/she is drawing. Moreover, in this menu, it is shown the angle between the last point and the current point, with its coordinates (figure 4).

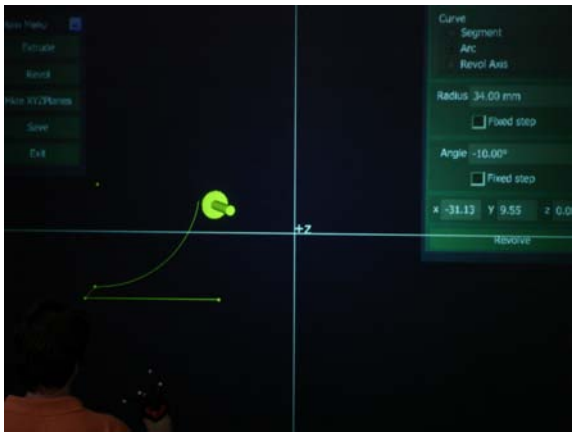

Figure 4 : *2D profile drawing*

Another help tool is the "Fixed Step" option that allows to round to one millimeters the lengths, and to five degrees the angles. Because of the hand vibrations, in fact, it is very difficult to maintain a stable position of the wand in the 3D space so, consequently, it is impossible to select the exact desired dimension.
In case of revolved solid modelling, through the 2D Toolbox menu, it is also possible to select a feature to draw the revolution axis; in this case self crossing profiles are not accepted.

4.3 Snapping

Another feature of VR Studio, made to improve its usability, is snapping. This is an essential function to draw a closed wire. During the profile drawing, VR Studio, at every wand movement, checks if close to the pointer there is a point, if so the wand changes color and, by pressing the flystick trigger, the user will be able to use that point.

4.4 Extrusion and Revolution

After drawing the profile, by pressing the "Extrude" or the "Revolve" button of the 2D Toolbox, VR Studio allows to complete the operation by selecting the extrusion depth or the revolution angle. Also in this case a new menu appears to show the length of extrusion or the extension of the revolution angle. Moreover, in this step, the scene camera rotates to help the user to better perceive the extrusion (revolution) direction and the depth (angle). During the extrusion or the revolution features, a copy of the previously drawn profile is translated along the extrusion or revolution direction at every wand movement, to let the user better imagine the final solid model. As soon as the user presses the flystick trigger the extrusion depth or the revolution angle is confirmed.

4.5 Camera management

In VR Studio the user can manage the position and the orientation of the scene camera using the flystick (figure 5). The possible operations are:

- Camera translation;
- Zoom;
- Camera rotation.

The first of this functions is accessible by the use of the mini joystick on the top of the flystick. Moving the joystick along the x-axis the camera is translated to left or right. Moving the joystick along the y-axis the camera is translated up or down.

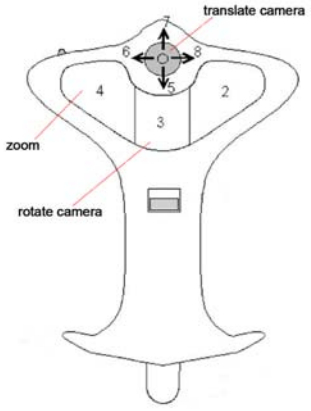

Figure 5 : *The Flystick*

The zoom is obtained translating the camera along its z-axis

(that is perpendicular to the screen display). This operation is executed holding the button 4 of the flystick, and bringing the flystick near to the screen or moving it away. In both cases, i.e. during camera translation and zoom, the scene camera orientation is not modified.
The camera rotation is implemented using a virtual trackball algorithm [Leh]. A sphere (not visible to the user) is mapped on the viewport and the wand position is projected on this sphere (figure 6). As the wand is moved holding the button 3 of the flystick, the camera is rotated to keep the same point on the sphere underneath the wand. When the wand is moved horizontally, a rotation about the Y-axis is required to keep the same point under the wand. Similarly, vertical changes in the wand position result in rotation about the X-axis. This interface provides a fairly intuitive method by which a model may be freely manipulated by applying a combination of rotations about the X and Y axes.

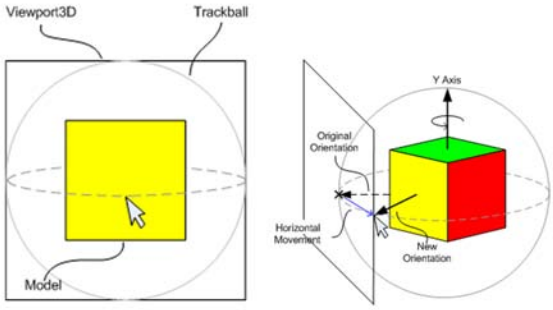

Figure 6 : - *Trackball and viewport*

4.6 File management

VR Studio allows saving and opening of files containing solid models. Supported formats are BREP, default file format used by Open Cascade, and IGES. The latter is very important because it is a common format used by most of commercial CADs. The file exchanging potentiality assures the interoperability of VR Studio with others CADs. This way the user can import and modify, in VR Studio, files created with other software, and vice versa. When the user opens an IGES file (created with another application), VR Studio analyzes all the surfaces that compose the imported model, identifying the plane surfaces. This will allow, afterwards, to select them to draw a 2D profile and to make very complex models.
To open a file the user must specify (in the command line) the option "-f" followed by the filename when launching the application. It is possible to save the model, when inside the virtual environment, by pressing the "Save" button. In this case the user will be shown a simple dialog box (figure 7), where the user can select the file format, and if to overwrite the current file or to save in a new file.

Figure 7 : *The save menu in VR Studio*

Since the user cannot access a keyboard, considering he/she is in the virtual environment, it was decided to save the file by adding a progressive number after the original filename. Of course VR Studio checks if already there are, inside the working directory, other files with the same current progressive number and with the same suffix.

5. Implementation

5.1 Main

The main function initializes Open Inventor and Open Inventor DialogViz (this is the module that manages the menus rendering). The initGlobalVar() function (declared in init.h) initializes the extern global variables. Before the description of these variables, we need to explain how Open Inventor builds a 3D scene. An Open Inventor scene is built through a tree graph (figure 8).
initGlobalVar() initializes the following variables:

- CameraManager *cameraManager – manages the camera that visualizes the 3D scene;
- SoSeparator *scene3d – this is the separator which children represent the 3D scene with the three main planes and the 3D solid models;
- SoSeparator *mainPlanes – this is the separator that contains the three main planes;
- SoSeparator *models – this is the separator that contains the 3D solid models created or imported;
- SoEventCallback *eventCallback – this is necessary to register callback functions that handle the user events;
- SoSeparator *onScreenSep – this separator contains all the objects that remain fixed on the screen; it has a own camera with a fixed position;
- SoPerspectiveCamera *onScreenCam – this is the camera that visualizes the onScreenSep child nodes;
- MainMenu *mainMenu – this is the object that manages all the VR Studio menus;
- Wand *wand – this object manages the wand;
- OCCShapeContainer – this is the object that contains the the created or imported models.

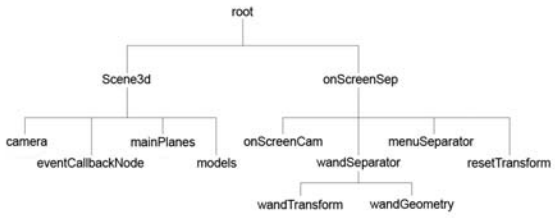

Figure 8 : *Initial Open Inventor scenegraph*

5.2 MainMenu and MainMenuAuditor

The MainMenu class represents the main menu of VR Studio (figure 9) that is loaded from the file MainMenu.iv. This is a text file that contains the tree graph of the menu. Every interactive menu must be linked to its "auditor". This one is used to invoke the right method when a pressing of a menu button takes place. For this purpose we have implemented the MainMenuAuditor class that is derived from SoDialogAuditor class of Open Inventor.

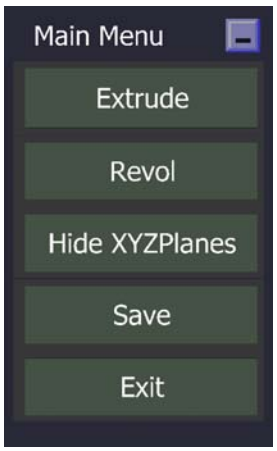

Figure 9 : *Main Menu*

5.3 TrackdReader

The TrackdReader class allows to read the flystick data from the trackd demon. The class constructor accepts as parameters the shared memory addresses where the flystick position data and the buttons status are stored. To refresh the data the methods refreshMatrix() and refreshButtonStatus were implemented. Furthermore the methods buttonChange() and getValuatorStatus() were set up. The former accepts as parameter the id button and returns -1 if the button was released, 1 if the button was pressed, and 0 if the button status was not changed. getValuatorStatus() provides instead information on the position of the mini joystick. It returns -1 or 1 if the joystick was moved along an axis specified by the passed parameter. The flystick position and rotation is stored in a 4x4 matrix of float declared as public member and so accessible by other classes.

5.4 Wand

This class represents the wand showed in the scene. It contains a pointer to a TrackdReader object and a pointer to a SoTimerSensor. The second object calls at regular times the trackingCB() function. The latter calls the methods refreshMatrix() and refreshButtonStatus() of TrackdReader, and it moves the wand on the screen applying the transformation matrix retrieved by TrackdReader, it checks the flystick button status and, if necessary, it launches an event to the application. Moreover it modifies the camera position of the 3D scene as described before. The wand geometry is loaded from a *.iv. The feature to rotate the camera can be disabled with enableRotatingCam() method. It was necessary to implement this method because in some cases, like the 2D drawing step, the use of this feature is not convenient.

5.5 OCCShapeContainer

This class is related to the modelling module of VR Studio: it contains a list (TopTools_ListOfShape object) of TopoDS_Shape, the shape object of Open Cascade. The methods addShape() and removeShape() are used to add or remove 3D objects from the VR Studio database. In this class we have implemented the methods readBrepFile(), readIGESFile(), saveBrepFile() and saveIGESFile() that are used for importing and saving the models.

5.6 Tool and Step

In VR Studio, to perform the operations of extrusion or revolution, it is necessary to follow a sequence of other simple operations as surface selection, 2D profile sketching, etc.. For this it is necessary to execute a series of steps. Basing on this concept we have created the Tool class, composed of an array of Step objects. These classes are two base classes, from which other specialized classes are derived. The main methods of Tool class are:

- getNumOfSteps() – returns the number of Steps that compose the Tool;
- goToNextStep() – calls switchOff() method of the current Step, it increments the index of the enabled Step, and it calls SwitchOn() of the new Step;
- goToPrevStep() – similar to the previous;
- terminate() – this method is called from the last Step to communicates to its Tool that all the operations are terminated;
- getStep() – returns the Step pointer specified by the index argument.

The main methods in Step, instead, are the following:

- eventHandler() – this is the method that handles the user events when the Step is enabled;
- switchOn() – this method is called by Tool to enable the Step; in this method is registered the eventHandler() as the method that responds to user events;
- switchOff() – this method is called by Tool to disable the Step, so the eventHandler() will not handle

the events;

- getParentTool() – returns the pointer of the own Tool.

Thanks to this architecture it is very simple to add new features in VR Studio, by deriving these two classes. The Tools and Steps for the operations of extrusion and revolution, in fact, are derived, in a very simple way, from these base classes (figure 10).

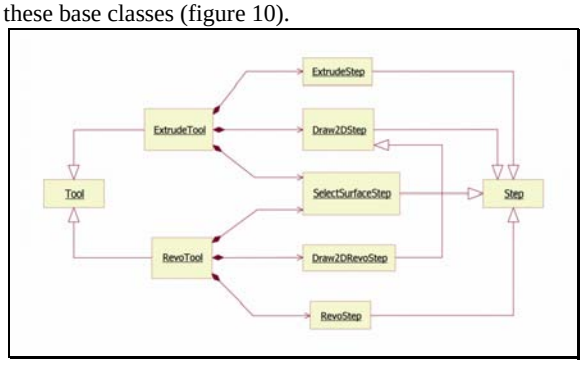

Figure 10 : *Diagram of the classes used to manage the extrusion and revolution tasks*

5.7 ExtrudeTool

ExtrudeTool derives from Tool class. It is composed of three Step objects, whose pointers are stored in the correct sequence in an array. The needed Steps to perform this operation are SelectSurfaceStep, Draw2DStep and ExtrudeStep. The constructor creates the SoSeparator temp object that is added as child node to the scene3d separator and that will contain the 2D profile. Some of the members of ExtrudeTool are a plane, that is the plane where will be drawn the 2D profile, and a pointer to the SoSeparator node that contains the selected plane surface. To access to these members the getProfilePlane() and getSelectedSurfRoot() methods can be used.

5.8 SelectSurfaceStep

This class is derived from Step, and allows the user to select a plane surface. The eventHandler creates a SoRayPickAction object at every wand movement; in this way the picked object list is checked and if an object of this list is a SoIndexedFaceSet and if the SoSeparator parent node have the name "PLANE_SURFACE" or "aMainPlane", that faceset is highlighted calling highlightSurface(). "PLANE_SURFACE" and "aMainPlane" represent the surfaces of the solid models and the three default orthogonal main planes.

5.9 Draw2DStep

This Step allows the user to draw a 2D profile. The switchOn() method builds the 2DToolbox menu and it rotates the camera so that the selected surface is viewed frontally. Further it registers the eventHandler and it builds the subgraph where the objects that will compose the closed profile will be placed. In the temp node are added:

- a SoCoordinate node with the drawn points coordinates,
- a SoIndexedLineSet that represents the segments,
- a SoSeparator with two children: a transformation node and a SoVRMLArc2D node.

The transformation node is useful to draw, correctly, a 3D arc. A SoVRMLArc2D arc, in fact, is always drawn at the origin of the viewport, and the angles are measured starting at the positive X-axis and sweeping towards the positive Y-axis. The arc extends from the startAngle counterclockwise to the endAngle. For these reasons, in order to draw an arc in the 3D space, the system needs to translate it to the selected plane and to rotate it so that its normal lies with the plane normal.

Arcs data is stored in two arrays: the first one contains the indexes of the three points that define the arc (center, initial and final points), while the second array contains the normals. These arrays, together with the SoCoordinate3 and SoIndexedLineSet nodes will be used in the next step to create a TopoDS_Wire, that is the Open Cascade profile. The snapping function is also managed by Draw2DStep: at every wand movements it is applied a SoRayPickAction to check if near the wand there is a SoPointSet object. In this case, the wand color changes, calling the setColor() method, and when the user presses the trigger, the system captures the index of that point, that will be used to draw the new segment or arc.

5.10 ExtrudeStep

This class creates a copy of the extruding profile that is translated along the normal following the wand movement. The switchOn() method, besides registering the eventHandler and rotating the camera, it builds the menu in which the extrusion depth will be shown. When the user presses the trigger, the solid will be created calling the createOCCShape() method. This creates a TopoDS_Wire by processing the child nodes of temp and the arrays of the previous step, which contains the arcs data. After creating the wire, the extrusion operation is performed by defining an instance of BRepPrimAPI_MakePrism Open Cascade class, which returns a TopoDS_Shape object that is added to the shapeContainer. Then the shape object is converted in several SoIndexedFaceSet by the occShape2FaceSets() function and it is showed by Open Inventor. This function, in fact, returns a root SoSeparator that will be added to the SoSeparator models. EstrudeStep is the last Step of ExtrudeTool, so it calls the method terminate() of ExtrudeTool.

5.11 RevoTool

RevoTool is very similar to ExtrudeTool. The Steps that compose it are: SelectSurfaceStep (identical to that of ExtrudeTool class), Draw2DRevoStep and RevoStep.

5.12 Draw2DrevoStep

This class derives from Draw2DStep but it has some

different methods to draw the revolution axis, represented by a SoIndexedLineSet of red color to distinguish it from the profile segments. The getRevoAxisCoord() method returns the coordinates of the points that describe the revolution axis.

5.13 RevoStep

The switchOn() method builds a copy of the 2D profile, rotates the camera, it builds the menu that shows the revolution angle, and it registers the eventHandler. In this step, at every wand movement, the copy of the profile rotates around the revolution axis. The procedure to create the solid is similar to that used by the ExtrudeStep, but in this case the createOCCShape() method creates a revolution solid thanks to an instance of BRepPrimAPI_MakeRevol class, to which is passed the closed profile to revolve (TopoDS_Wire), the revolution axis and the angle.

6. Conclusion and Future Works

The use of virtual reality for design process holds a great attraction for the designers. No doubt the spreading of this tecnologies is predictable thanks to the increasing of the systems perfomances and the decreasing of costs.
The results obtained with VR Studio were very satisfying. The choice to keep unchanged the standard procedures used in the conventional CADs, proves to be very appreciated from the users, that had no difficult to learn to use VR Studio for creating even complex models like transmission shafts (figure 11). The realistic perception of the three-dimensional models in the design process is a great step forward within the CAD software field, but there are still many things to set up.

Figure 11 : *Transmission shaft created in VR Studio*

The developed system, in fact, to be very user-friendly and powerful, needs more additional features, some related to the modelling tasks, like the boolean operations, the possibility of simulating assembly, some tools to modify the created solid geometries, etc.., others related to the improvement of the user- system interfacing. For this purpose, in fact, one of the future work will be related to the implementation of voice recognition commands to send numerical values to the application.
Thanks to the robust software architecture of VR Studio, at any time in the future, will not be more difficult to develop and add new features to improve this application make it a more and more complete tool for the industrial design.

References

[ART] A.R.T., http://www.ar-tracking.de

[Bja] Bjarne Stroustrup, The C++ Programming Language – Third Edition, Addison Wesley.

[BC03] Burdea G. C., Coiffet P.; "Virtual reality technology - 2nd edition"; Wiley-Interscience. A John Wiley & Sons Inc., Publication, Hoboken, New Jersey, 2003

[CILG06] Cappello F., Ingrassia T., La Cascia M., Gazziano G., "Free-form surfaces modelling in virtual reality", Proceedings of International Conference - Ingegraf 2006, Sitges - Barcelona (Spain) - May 2006.

[CLL94] Chapin W. L., Lacey T. A., Leifer L.; "DesignSpace: a manual interaction environment for computer aided design"; in Proceedings of the ACM SIGCHI 1994 Conference: CHI94 Human Factors in Computer, Systems, Boston, MA pp 33-34, 1994

[CTL01] Connell M., Tullberg O., Lu L., "The use of Virtual Reality in Interactive Finite Element Analysis by Java3D API", Applied Virtual Reality in Engineering and Construction, proceedings of the conference AVR II and CONVR, pp 60-68, 2001

[DG97] Dani T. H, Gadh R.; "COVIRDS: A Conceptual Virtual Design System"; Computer Aided Design, Vol. 29(8):555-563, 1997

[DBSL02] De Amicis R., Bruno F., Stork A., Luchi M. L.; "The Eraser Pen: A New Interaction Paradigm for Curve Sketching in 3D"; International Design Conference - DESIGN 2002 Dubrovnik- Croatia, May 14 - 17, 2002

[ESS*99] Encarnação L.M., Stork M., Schmalstieg D., Bimber o., Barton R. J.; "The Virtual Table – A Future CAD Workspace"; SME Computer Technology Solutions Conference, Sep. 13-16, Detroit, MN, USA, 1999

[FvD] Foley J.D., van Dam A.; "Computer Graphics Principles and Practice – Second edition in C"; Addison Wesley Publishing Company

[JCL97] Jayram S., Connacher H., Lyons K.; "Virtual Assembly using Virtual Reality Techiques"; Computer Aided Design vol. 29, No. 6, pp. 575-584, 1997

[LPA04] Lee K., Price M., Armstrong C., "ViSiCADE a Virtual simulation environment for seamless integration of CAD/CAE into VR ", ISSC conference, Belfast, june 2004

[Leh] Lehenbauer D.; "Rotating the Camera with the Mouse – Implementing a Virtual Trackball with the Windows Presentation Foundation"

[OC] Open Cascade; http://www.opencascade.org

[OI] Open Inventor Architecture Group; "The Inventor Reference Manual"; Addison-Wesley, New York, 1994.

[SFDM02] Stork A., Fiorentino M., De Amicis R., Monno G.; "Surface design in virtual reality as industrial application"; International Design Conference - DESIGN 2002 Dubrovnik- Croatia, May 14 - 17, 2002

[SM97] Stork A., Maidhof M.; "Efficient and Precise Solid Modelling using a 3D Input Device"; Proceedings of Fourth Symposium on Solid Modelling and Applications. pp. 25-33, 1997

[Wer94] Wernecke, J.; "The Inventor Mentor: Programming Object-Oriented 3d Graphics With Open Inventor" Release 2. Addison Wesley, 1994