A VR-User Interface for Design by Features

M.K.D. Coomans and H.J.P. Timmermans
Eindhoven University of Technology
Faculty of Architecture, Building and Planning
Eindhoven, The Netherlands

ABSTRACT — We present the design of a Virtual Reality based user interface (VR-UI). It is the interface for the VR-DIS system, a design application for the Building and Construction industry (VR-DIS stands for Virtual Reality - Design Information System). The interface is characterised by a mixed representation of the task domain: an analogue "mock-up view" is being integrated with a descriptive "feature view". It uses a Fish Tank VR configuration which integrates the virtual objects in the designer's normal working environment. The issues underlying the design of the feature view are discussed, as well as the choice of input devices.

1. INTRODUCTION

Design tasks, in particular architectural design tasks, have been found hard to support by means of computers. We point out two main reasons.

One reason is that the design objects provided by conventional CAD systems do not reflect the designer's thinking. The designer's elements are task specific and differ between individuals. Moreover, during a particular design process, these elements of design are not static but are invariably subject to change (van Leeuwen, et al.,1997). Design is a problem solving process that requires the computer to handle the information in a dynamic way.

A second reason is that design decisions are stored in subsequent graphic representations. Graphic representations are a generally acknowledged medium through which the architect develops the design (Achten,1997). Design representations are instrumental to both visual reasoning (idea generation) and evaluation. In the conceptualisation phase of the design process designers typically use sketches. Sketches are quick and easy, and support visual reasoning (Goldschmidt,1994). The current graphical computer systems lack this "perceptual interactivity" of the sketch. To support design evaluation, many aspects of the design (appearance, costs, physical behaviour, etc.) can currently be simulated in dedicated computer applications. Unfortunately, this split-up into unrelated applications results in different, unrelated data presentations and faulty mutual feed-back functionality.

VR-DIS is a research project in which possible solutions for these problems are investigated. VR-DIS stands for Virtual Reality - Design Information System. Key aspects of the research program are information modelling using features, and Virtual Reality (VR) user interfaces. Feature based information handling is expected to provide a bridge between the user's dynamic design concepts, and the information storage capabilities of the computer. The medium of Virtual Reality is expected to constitute the medium through which successful multi-aspect visualisations can be offered, and with which an intuitive and quick interaction method can be developed.

The VR-DIS system architecture consists of three types of components: (1) information storage components, (2) information processing components such as constraint solvers and performance simulation units, and (3) user interfaces. This paper focuses on the design of the user interface. First, we describe the general interface design. Then, we focus on the user interface component that presents design information in the formal 'feature model' format.

2. THE VR-DIS USER INTERFACE

Unlike conventional windows and mouse interfaces, VR has yet to evolve a dominant interaction paradigm. Currently, the design of a VR user interface requires the design of all aspects of the interface: the data visualisation, the interaction method, and the hardware devices through which the physical communication will take place. Each of these issues will subsequently be discussed. In the following section, we start with the data representation.

2.1 Mixed data representation

The different disciplines involved in a building and construction design process use different design methods and handle different design data. In order to support these different design approaches, it seems necessary to develop a variety of user interfaces. However, the feature modelling technique of VR-DIS may be instrumental in developing a user interface component that is generically applicable through the VR-DIS system.

To delimit the part of the user interface that can be developed generically and the part that will be discipline-specific, we must distinguish between two kinds of information representation: descriptive and analogue representations. Most data can be represented in both a descriptive, linguistic way and in an analogue, pictorial way. The concept "red", for example, can be communicated to a user by presenting the word "red", but can also be communicated by presenting a red surface. However, many data types have a certain preference for the one or the other. For example: the way things or situations look, sound, feel, or smell can often only with great difficulty be rendered linguistically, whereas other types of information such as abstract states of affairs and relationships can hardly be rendered at all using non-linguistic representations [Bernsen95]. The distinction between descriptive and analogue information representations seems generally valid. Human Computer Interaction modality analysis distinguishes between linguistic and non-linguistic interface representations [Bernsen95]. In cognitive psychology, an analogue distinction is made between verbal and pictorial mental representations.

The type of representation that serves best in a specific part of a computer interface can be expected to change, depending on the specific task that the user has at hand. For example, thermal isolation data is best communicated verbally when specific details are being fine-tuned, but, in the early design stages, a designer needs a quick rough insight in the thermal performance of the different parts of the building. In this latter case, a pictorial visualisation may serve better. We can conclude that VR-DIS cannot be served by a single representation. It seems essential to combine one or more descriptive representations with one or more pictorial representations. Such interfaces are called *mixed representation interfaces*. (See also (Coomans, et al.,1998).)

In VR-DIS, it's essential that the designer is in direct contact with the feature model. He/she must be able to adapt and expand the model continuously on the basis of his/her design thoughts. The feature model must at all times closely reflect the designer's thought on the different aspects of the design. Insight in the feature model as a whole can only be provided by a visualisation of the feature models themselves: features and their relations represented in a web or graph-like representation. We refer to this as the "*feature view"*. The feature view offers a descriptive representation of the design.

Because all design disciplines share the same feature model, an interface based on this data representation is valuable for all of them. The feature view is a generic component of the necessary interfaces for VR-DIS. For each design activity, the feature view should be integrated with other interface components, that are based on other data representations that are valuable for those design tasks, resulting in mixed representation interfaces. The feature view shall typically be integrated in at least one analogue representation. To support the early architectural design stage, we supply a combination of the feature view with a "*mock-up view*". The mock-up view is a pictorial representation of the design information, based on the physical appearance of the design. This view will feature a high level of verisimilitude.

2.2 Fish Tank VR

Virtual Reality is usually correlated with head mounted displays (HMD), gloves and CAVEs. With these devices, the user is immersed in the virtual environment. This popular form of VR has the major advantage of a wide field-of-view which can give the feeling of existing in the graphical world. Against this advantage there are several disadvantages.

Firstly, the cost of the view filling display is to block out the every day world of desks, chairs, filling cabinets, and passing colleagues. One consequence is that the inhabitants of VR have to have handlers to make sure that they do not hurt themselves. Another consequence is that designers are separated from the physical artefacts that can be found on their desks and that aid them in the problem solving process: the magazine picture that triggers a new idea, the product documentation sheets, project documents, reference books, and so on (which are all still in paper format). There is also a social aspect: immersed users are cut away from their colleagues. Colleagues who happen to pass by, are no longer able to provide some quick comments on the ongoing work.

The second disadvantage is that immersive VR typically requires the user to standup right such that he/she can interactively control the viewpoint by physically moving and turning around. This standing posture enables very intuitive control, but it is also tiring after a while. It is only suitable for shorter design sessions. The main working posture should be a seated posture.

A third and final disadvantage is specific for HMDs. In comparison with desktop computer monitors, HMDs have poor graphics resolution. HMDs feature lower absolute resolutions, but are at the same time placed just in front of the eyes such that they fill a wider angle of our view. One pixel of a HMD typically fills a view angle up to 6 times the angle filled by a pixel of a monitor.

Because of the many disadvantages of immersive VR set-ups, we have selected a different VR configuration for our design information system. The VR-DIS user interface makes use of what Ware has termed “fish tank VR” (Ware, et al., 1993).

Fish tank VR is characterised by in-real-time generated images of a 3D scene, viewed on a monitor, with the perspective projection of the images coupled to the head position of the observer. The images can either be stereoscopic or monoscopic. The front of the screen’s tube is experienced as if it is the glass front side of a fish tank that can be looked through. The virtual objects (the “fish”) are viewed inside the tank. The perspective projection reacts to the viewer’s head motion in such a way that the correct perspective to the virtual objects is obtained from any viewing location.

The advantages of fish tank VR are the high quality graphics provided by the monitor, the seated working posture, and the integration of the VR workspace with the everyday workspace. On top of that, the perspective–head movement coupling provides depth perception, and a feeling of scale. When we move our heads while looking at our environment, we perceive movement parallax that our perception system links and compares with the performed head movement. Ware (Ware, et al., 1993) has shown that movement parallax is at least as important as stereoscopic viewing for depth perception. Smets (Smets, 1992) has suggested that movement parallax is sufficient for tele-presence, the feeling of being physically present with the virtual objects. We expect that fish tank VR can present virtual objects on a desktop in such a convincing way that users can work with them as if they are real. We expect that fish tank VR can produce a similar feeling of being deeply engaged as immersive VR can. While immersive VR brings the user in the virtual environment, fish tank VR brings the virtual objects to the viewer.

The main disadvantage of fish tank VR is that it requires a quickly responding VR system. In order to obtain a strong percept of 3D space, the computer system must compute the adapted view with a minimal delay and high precision (Liang, et al., 1991). A standard VR configuration typically features a responds time of about 0.10s. One reason is the faulty quality of the magnetic head sensors with delays up to 0.08s. Another is the fact that typical VR graphics engines are still much to slow for what we would like them to do.

2.3 Direct manipulation

Fish tank VR requires a different kind of user input than immersive VR. In immersive VR, the user can navigate through the virtual environment to look at an object from another side, or to see some more detail. With fish tank VR the user always stays at approximately the same side of the virtual environment. To see another side of an object, the user must manipulate the object. Therefore, the user must have the appropriate devices to manoeuvre all the objects of interest to him, instead of navigating himself through the virtual environment.

The selection of the direct manipulation input devices for VR-DIS has been based on two guidelines: (1) exploit physical mappings, and (2) use distinct devices for distinct functions. Users perform familiar motor actions (such as manipulating objects with the hands) independently of conscious thought. A user interface that exploits these human motor capabilities leaves the rest of the cognitive system free for other tasks such as determining what to do, instead of determining how to things must be done (Coomans, et al., 1998). For a direct manipulation interface, this demand renders into establishing physical mappings between input gestures and task domain representations. For example, moving a control to the right must move a display object to the right.

The exploitation of the human motor capabilities also renders into the use of specific input devices for distinct functions. The conventional computer interface uses one spatial input device (the mouse) for all graphical input. Consequently, operations that cannot be mapped on the mouse's tracking capabilities can only indirectly be executed through manipulating screen widgets such as sliders and buttons. Further, menu entries are required for selecting appropriate input modes. The many menus, buttons, and sliders on the desktop tend to interfere with the representation of the task objects, and disturb the user's reasoning process (Stuyver, et al.,1995). The alternative is to provide specific interface tools for common tasks. Although using task-specific input devices may limit generality, the haptic feedback from the physical tool provokes a visceral and enthusiastic response from users (Hinckley, et al.,1994).

We have currently two spatial input devices connected to the VR-DIS system : a trackball and a 3d flying mouse. A third, a turntable will be added in the near future.

The trackball is used to manipulate the "feature box", the main tool of the feature view (see later). The feature box only requires rotation in the left-right and up-down direction. We have deliberately chosen for a trackball (with a large ball) instead of a Spaceball™-like device. Spaceball-like devices track the magnitudes of the push, pull and rotation forces that are performed upon them. The performed force is translated to a corresponding movement of the virtual object that is controlled. The good thing of Spaceball-like devices is that there is a clear mapping between the direction of the force that must be performed, and the direction of the wanted motion. To move an object up, just pull the ball up; to rotate the object to the right, just rotate the ball a bit to the right. The bad thing of Spaceball-like devices is that there is no physical correspondence between the magnitude of the displacements of the ball, and that of the displacement of the virtual object. For example: it's unclear how long and how hard the ball must be turned in order to obtain a turn of about 90 degrees. One can only look at the computer screen and see when the object reaches the wanted orientation. The feedback is purely visual, not physical. With a trackball, the orientation of the ball can be correlated to the orientation of the virtual object. To turn the object 90 degrees, just rotate the ball 90 degrees.

In the near furture we will add another specific device for manipulating the scale model of the mock-up view. It will be based on Hennesey's turntable device (Hennessey, et al., 1995). The turntable is a disk that can be rotated on a central axis. As the disk is rotated, the scale model on the computer screen rotates in the exact proportion.

The second spatial device that we have already implemented is a 3D flying mouse. The flying mouse is used to grab, relocate, and activate feature-objects in the feature view, as well as building parts in the mock-up view. As with conventional mouse and arrow, the physical device must be represented by some pointer in the virtual world (a 3D arrow). When the virtual pointer touches or intersects an object, the object can be grabbed or activated. Most VR applications use a 3D model of the human hand as virtual index. The motion of the physical device is mapped by the virtual pointer. As with the 2D mouse and arrow, the motion of the virtual pointer and the motion of the physical device are often mutually scaled; the scale factor is sometimes even dynamically adapted (Poupyrev, et al., 1996). Such an inexact correlation between displacements is sufficient to exploit the human motor capabilities. However, in order to realise the through feeling that the virtual objects are physically present, all connections between the real and the virtual environment should be as strong as possible.

In order to enforce the Fish Tank metaphor, we have realised a strong perceptual link between the 3D flying mouse and its virtual index. We have coupled a quickle responding virtual pen to the the 3D mouse. The virtual pen is, so to speak, mounted on the front of the physical 3D mouse; see figure 1. The virtual pen follows the movement of the 3d mouse such that the illusion of there mechanical connection is at all times maintained. As with the head-view coupling, the illusion of the mechanical connection between the physical and the virtual object can only be maintained with a quick and accurate responds of the VR system.

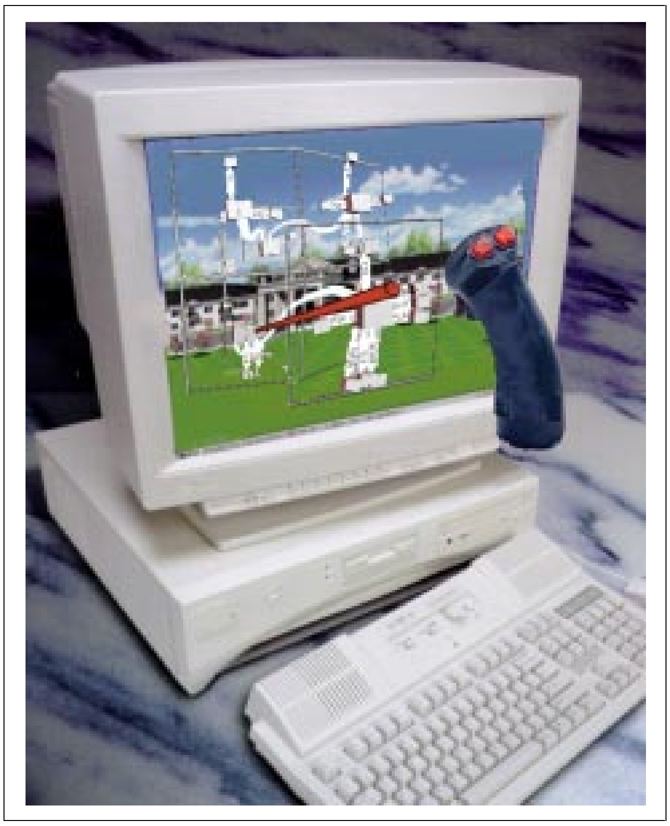

Figure 1 – The flying mouse with the virtual pen

The virtual pen collides with all other virtual objects; they never intersect. Further, the movement of the virtual pen is never constrained because the movement of the physical 3D mouse cannot be constrained either (because we have no force feedback equipment). Instead, touched objects are always pushed away by the pen. All objects feature an elastic connection with their rest position. When the pen is moved away from an object, the object turns dynamically back to its rest position.

The tip of the virtual pen is the active part. An object touched by the tip of the pen can be triggered by any of the 3D buttons of the 3D mouse. The mouse features a “grab” button with which the touched object is grabbed (temporarily connected to the pen), a “+” button and a “-“ button. The effect of the latter two depends on the type of the object that is triggered.

The position and orientation of the flying mouse is currently recorded by a magnetic tracker. As discussed before, magnetic trackers feature a long delay time which can be problematic for Fish Tank VR. We have reduce the effect through the use of a predictive filter, as suggested by Liang (Liang, et al., 1991).

2.4 Linguistic input

The VR-DIS system has also a linguistic input channel to specify values, names, descriptions and so. We are currently using a keyboard. We consider replacing the keyboard by a natural language recognition system.

3. THE DESIGN OF THE FEATURE VIEW

3.1 The design problem

The design information of particular design projects is stored in feature models. Feature models are shared by all design participants, and they will contain many different kinds of information: geometry data, lay-out data, appearance data, hydro-thermal data, cost information, ageing data, organisational data for the building process, and so on.

The single features are formal descriptions of specific characteristics or concepts in a design project [J.v.Leeuwen97]. Some examples: one feature might represent a particular wall, another might represent the building as a whole, there can be a feature representing the cost of a light switch, and yet another feature might even represent the principal’s demand that the meeting room should be "comfortable". The designers can make instances of predefined feature-types, but they can also define new types, reflecting the precise design concepts they have in mind. Each feature represents a single concept.

Features are linked together forming feature models. There are two main predefined relation types: composition relations, and specification relations. All other possible relations are termed "associations". Examples of associations are: "is supported by" (a beam supported by a column), "is accessible by" (the building by the road), and so on. As with the feature types, the designer can add his own kinds of relations if he feels that's appropriate. Features can be linked together on both the type and the instance level. Type level relations are used to form complex feature types, to be stored in a feature library. Instance level relations are used to describe the relations between the feature instances of a design project at hand.

The cost of the great flexibility and extensibility of this data model is that it lacks almost any structure. While designing with the VR-DIS system, feature models grow that are not ordered in any strict predefined way: no hierarchy, no matrix structure, no list structure. Feature models are just (chaotic) web-like data structures, consisting of features that are of an extensible list of types, and mutually connected by relations which are decompositions, specifications, or other types of relations. On top of this, in real design projects feature models can grow as large as (presumably) 10 to 100.000 features. It is clear that the flexibility, the extensibility, and the scale of the feature models constitute high demands for the representation through which the feature models are communicated to the user.

The visualisation design of the feature model can be split in two parts: (1) the visualisation of single features and their relations, and (2) the graph visualisation technique that can cope with the scale of the feature models.

3.2 Visualisation of single features and their relations

A feature is visualised as shown in figure 2. Each feature consists of a front plate and a rectangular area behind it. The front-plate features a textual representation of the feature's name and type. The rectangular area behind the front-plate represents the content of the feature. This area is subdivided in a left and a right half.

The left half represents the content that has been defined on the type level. In the case of a primitive ("simple") feature type, the content is something like a number, or a string. Such a primitive content is represented by a 3D icon. In the case of a complex feature type, the content is constituted by all the components (cf. component relations). In this case, the left half of the content area is filled with small visualisations of all the features that have been defined as components at the type level.

The right half of the content area represents the content that has been defined on the instance level. For all feature types, this area is filled with the visualisation of the components that have been added at the instance level.

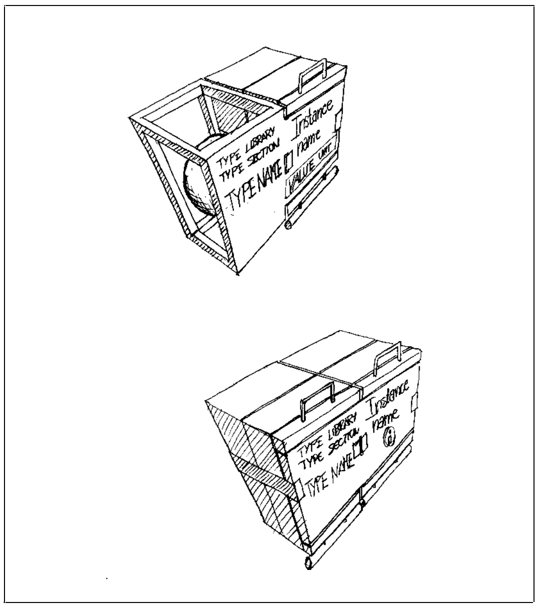

Figure 2 – Visualisation of a Simple and a Complex Feature, in initial state

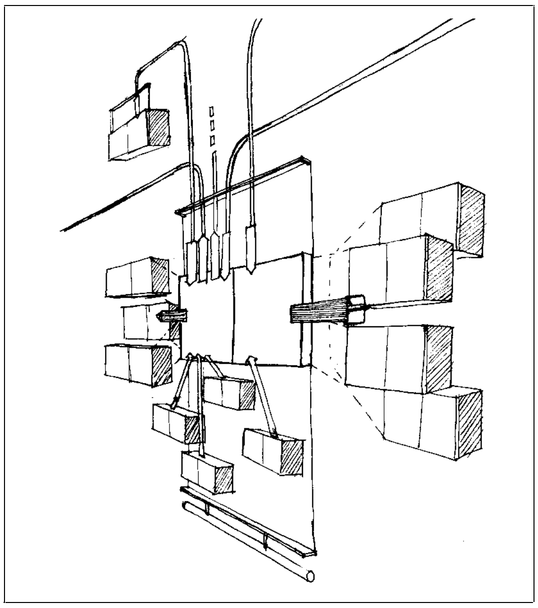

Figure 3 – Visualisation of a Complex Feature, in exploded state

When a feature has specification relations, those specifications are hidden in a drawer that is connected to the bottom of the front-plate. The drawer has a left type-level part, and a right instance-level part. The drawer can be opened by pulling the handle downwards. When the drawer is opened, all features appear that specificy the central feature (see figure 3). The specification relations themselves are visualised by arrows with an overprint of the relation's name. The top of the front-plate features a similar drawer that hides all associations of the feature. It can be opened in a similar manner.

Because the front-plate hides most of the component-features, those component features can be pulled to the side of the front-plate. When pulled to the side, the components are visualised similar to the specifications and associations: the component-relations are visualised as bands with an overprint of the relation's name.

Specifications, association, and components each appear in specific directions relative to the parent feature. This systematic grouping eases the "reading" of features, but it also eases feature browsing because we can map users' input gestures to these directions (see later).

The initial state of a feature is referred to as the *collapsed state*. The state in which some or all of it's relations are visualised is referred to as the *exploded state*.

3.3 Browsing feature models

Feature models are graphs: sets of nodes (features) with mutual edges (relations). The visualisation of graphs is one of the problems that is addressed by a specific discipline of the computer science community, known as "Graph Drawing". Unfortunately, the traditional drawing techniques of this discipline cannot be used for the visualisation of feature models. As Eades indicates (Eades, et al., 1997), traditional graph drawing techniques assume that the complete graph can reasonably be represented in a readable and understandable manner on the display medium. This assumption does not hold for feature models that can grow up to 100.000 nodes.

Very large data-sets require an interactive drawing technique. In subsequent steps, the user must specify from which part of the graph he/she wants to see more. This process is referred to as *browsing*. Eades and Graham (Eades, et al., 1997) (Graham, 1997) proposed an interactive drawing techniques for the visualisation of very large graphs. Both produce 2D images in which child-nodes are placed on circles around parent nodes. The spatial distribution of the child nodes around their parent depends on the number of the children's children, and on the browsing history (= which "uncle" and "nephew" nodes that have been looked at before). This layout mechanism makes the technique difficult to apply for feature models. We already organised child features around their parent on the basis of the type of their relation. Eades' and Graham's layout mechanisms suppose only one relation type. Their layout mechanisms cannot be superimposed on our's because the two would interfere and result in an unreadable graph.

In VR-DIS, we have chosen to visualise the browsing history in the third dimension. When a feature's relations are visualised, the feature first moves to a new transparent plane that is laid on top. Recursive requests for more detail results in multiple display layers. This mechanism has been inspired by Lieberman's multiple translucent layering technique (Lieberman, 1994). We have designed a specific virtual tool through which this browsing process can take place. We refer to this tool as the "feature box".

When starting, the feature box looks just like a real closed box The box represents the whole feature model of a specific design project. The box can be opened, after which a pyramidal area lights up. In the enlightened pyramid, the shell of a sphere emerges concentric around the box. The shell is orthogonally subdivided in a number of rectangular fields. In each field, a single feature and its relations can appear. The user can formulate type and name based queries to select the features of this first shell's fields. See figure 4.

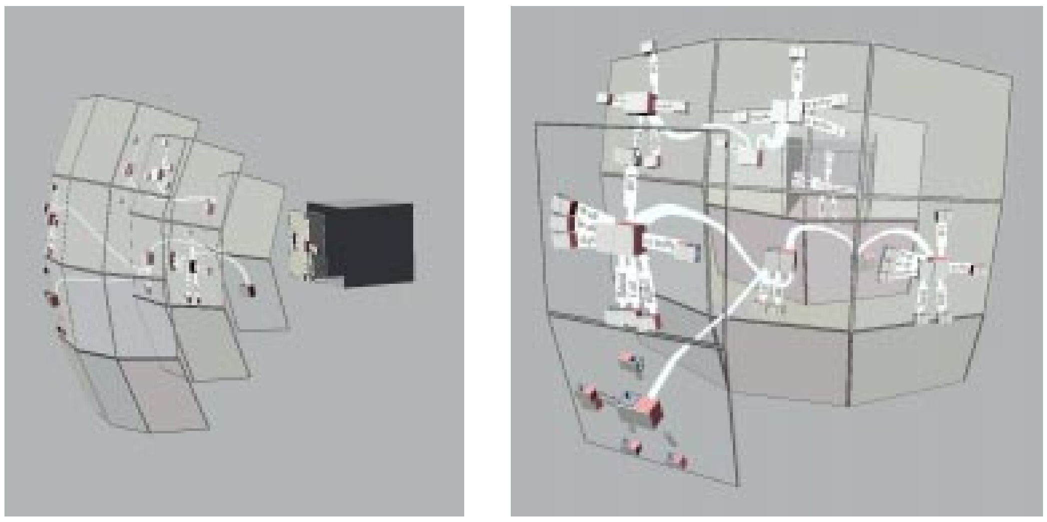

Figure 4 – Feature Tool

When a feature has components, specifications, and/or associations, they can appear around the original feature, all within a single field of the feature box's shell. The related features are shown smaller than the central feature. When the designer asks such a related, small feature to explode its relations, then it will first move to a new empty field of its own. This new field will be located at a shell that's concentric around the previous shell, but with a bigger radius. While moving to the new field, the relations to the features on the original field stay intact: arrows and bands are bend to S-like shapes. In this new field, the feature can then explode its own related features. These will again appear around their parent, unless the relation goes to a feature that is already visible on any of the other fields of the feature box. In that later case, the related feature stays where it is, and is connected by a bent arrow or band.

3.4 Control

The designer can manipulate the feature box as a whole by a trackball. Through the trackball, the designer can rotate the feature box's pyramidal area.

The designer uses the 3D mouse to change a feature from collapsed state to exploded state (making all the relations visible). He or she must first select the collapsed feature by pointing the virtual pen at it (the feature will now highlight) and then hit the mouse button marked "+" ("+" means "show more"). With the direction of the pen, the designer indicates the kind of relations that must be exploded. The direction-sensitivity corresponds with the directions in which the different relations are visualised: pointing left-right will explode compositions, pointing bottom-up will explode associations, pointing top-to-bottom will explode specifications; pointing straightforward explodes all relations together. The related features can be hidden again by hitting the mouse button marked "-" while pointing at the central feature. This hide commando is also direction dependent.

Because we mapped input gestures to the direction in which relation types are visualised, we avoided the use of small hot spots to select relation types. Hot spots are small buttons or handles on an object with which the object can be changed or manipulated in a specific way. Although this approach is popular in 2D graphics applications, it's a faulty technique in VR environments because freely movable 3D pointing devices are currently still too imprecise to point accurately at small places in

3D space. The selection of a small 3D hot spots would just be too cumbersome, too faulty. (Position sensors with mechanical linkages and and potentiometers perform better but they limit the user’s moving freedom.)

4. IMPLEMENTATION

The described VR interface to the VR-DIS feature models is currently in the first prototyping stage. We have a working prototype of the feature view developed using Sense8’s WorldUp™. It communicates with a feature model, which is currently an MS-ACCES™ database accessible through ODBC™. An alternative form fill-in type interface has been implemented as well. It has the same functionality as the VR-feature view. This conventional windows interface proves particularly useful to evaluate the advantages and disadvantages of the VR feature interface. The Fish Tank VR functionality has been prototyped in a separate application using WorldToolKit™.

Other prototype components of the VR-DIS system are the constraint solver, and an interface for viewing the shapes of the design (the mock-up view). De Vries en Jessurun (deVries et al. ,1998) discuss these other components.

5. FUTURE WORK

In a next development cycle, all prototypes will be integrated in a single multi-treaded application. The feature and mock-up view will be integrated in one as intended. The mock-up view will be worked out, and a natural language recognition system will be integrated.

We are currently setting up an experiment to evaluate the effectiveness of the the designed VR interface. We will also investigate the design-supporting qualities of the feature view.

6. CONCLUSIONS

We have presented the prototypes of the VR-DIS user interface and have discussed the issues underlying their design. VR-DIS make use of Fish Tank VR because it features high quality graphics, a seated working posture, and integration of the VR environment with the designer’s everyday workspace. The final VR-DIS system will feature several task-specific user interfaces of which each will be characterised by a mixed representation of the task domain. In this frame, we currently support the early architectural design stage by a combination of an analogue “mock-up view” and a descriptive “feature view”. We selected three input devices for object manipulation. Two of them, the trackball and the turntable, are dedicated to the manipulation of the core objects of respectively the feature view and the mock-up view. By adding specific devices for common tasks, we minimise the mental effort that is required for executing these tasks. We developed a specific virtual tool, the feature box, with which large feature models can be browsed through. In order to ease the control, the browsing process exploits a mapping between feature visualisation and input gestures.

We expect that a feature based design system, with the described type of interface can much better support design than conventional CAD systems. We expect that the VR-DIS system will provide design objects that closely reflect the designer's thinking, as well as an interface that is easy and quick enough to support visual reasoning.

REFERENCES

Achten, H.H. (1997). Generic representations - Knowledge representation for architectural design. G. Ang, L. Hendriks, P. Nicholson and M. Prins (eds.) 1997. *Journal of Architectural Management 13*

Coomans, M.K.D. and H.H. Achten (1998) Mixed Task Domain Representation in VR-DIS. Proceedings of the 3rd Asia-Pacific Conference on Human Computer Interaction, APCHI'98, Shonan village Center, Japan, July 15-17, 1998.

de Vries, B., and A. J. Jessurun (1998) An experimental design system for the very early design stage, in Proceedings of the 4th Conference on Design and Decision Support Systems in Architecture and Urban Planning, DDSS'98, July, 1998

Eades, P., F. Cohen and M.L. Huang (1997) Online Animated Graph Drawing for Web Navigation. G. Goos, J. Hartmanis and J. van Leeuwen (eds.) 1997. Graph drawing, Proceedings of the 5th International Symposium on Graph Drawing, GD'97, held in Rome, Italy, Sept. 18-20, 1997. Springer-Verlag, Berlin, pp. 330-335.

Goldschmidt (1994) On Visual Design Thinking: the vis kids of architecture, *Design Studies*, vol. 15, pp. 158-174.

Graham, J.W. (1997) NicheWorks – Interactive Visualization of Very Large Graphs, G. Goos, J. Hartmanis and J. van Leeuwen (eds.) 1997. Graph drawing, Proceedings of the 5th International Symposium, GD'97, held in Rome, Italy, Sept. 18-20, 1997. Springer-Verlag, Berlin, pp. 403-414.

Hennessey, J.M. and M. Gribnau (1995), Demonstration of: Two-handed modeling of three-dimensional computerized objects with the help of specialisaed input devices. Proceedings of HCI'95. Huddersfield, August 1995. Cambridge University Press, Cambridge, 1995.

Hinckley, K., R. Pausch, J.C. Goble and N.F. Kassell (1994) Passive Real-World Interface Props for Neurosurgical Visualisation. … (eds.) Human factors in Computing Systems. Proceedings of CHI'94. ACM, pp.452-458.

Liang, J., C. Shaw, and M. Green (1991) On Temporal-Spatial Realism in the Virtual Reality Environment. Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology 1991, pp.19-25

Lieberman H. (1994) Powers of Ten Thousand: Navigating in Large Information Spaces. Proceedings of the ACM Symposium on User Interface Software and Technology, 1994. p.15-16

Poupyrev, I., Billinghurst, M., Weghorst, S., & Ichikawa, T. (1996). The Go-Go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR. In Proceedings of UIST '96, ACM, New York, 1996, pp. 79-80

Smets, G.J.F. (1992) Designing for telepresence: the interdependence of movement and visual perception implemented, in IFAC Man-Machine Systems, the Hague, The Netherlands, 1992, pp. 169-175

Stuyver R., and J. Hennessey (1995), A support tool for the conceptual Phase of Design. M.A.R. Kirby, A.J. Dix, and J.E. Finlay (eds.), People and Computers X. Cambridge University Press, Cambridge, pp.235-245

van Leeuwen, J.P. and H. Wagter (1997). Architectural Design-by-Features. Junge, Richard (ed.) 1997. CAAD futures 1997. Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures held in Munich, Germany, 4-6 August 1997. Kluwer Academic Publishers, Dordrecht, p. 97-115.

Ware, C., K. Arthur and K.S. Booth, Fish Tank Virtual Reality, INTERCHI'93, Proceedings of the international conference on human computer interaction … , 24-29 April 1993, ACM, pp. 37-42