The original paper is published in CCAI : the journal for the integrated study of artificial intelligence, cognitive science and applied epistemology, Rijks Universiteit Gent, Belgium

A Virtual Reality User Interface for a Design Information system

Marc K.D. Coomans

Eindhoven University of Technology, Faculty of
Architecture, Building and Planning, The Netherlands
and
Vartec NV, Gent, Belgium

Abstract

The computer is a tool, a complex artefact that is used to extend our reach. A computer system can provide several kinds of services, but against these services stands a supplementary task that the user must deal with: the communication with the computer system. We argued that Virtual Reality (VR) can fundamentally improve the user interface by rendering on the common experiential skills of all users. We present the theoretical basis for this, referring to Donald Norman's theory. We show that VR provides at least theoretically, the means to take a big step in the direction of an ideal user interface. As an example of a innovative application of VR in user interface design, we presented the VR-DIS system; an interdisciplinary design system for the building and construction industry. We discuss the issues underlying the design of its VR interface.

1. Introduction

The computer is a tool, a complex artefact that is used to extend our reach. A computer system can provide several kinds of services:

- an external medium to represent abstracted information, supporting the user's reasoning.
- external memory (storage of quantifiable information)
- calculation and reasoning (logical reasoning)

As against these services stands a supplementary task that the user must deal with: the communication with the computer system. The use of the computer itself requires some mental effort, determining how it must be controlled to exploit the provided services. In the ideal situation however, it would be invisible to the user. The computer would provide its services in ways that perfectly match the users thinking. How can this be realised?

We believe that Virtual Reality (VR) can fundamentally improve the user interface of many business applications. We discuss the theoretical basis for these expectations, and compare the capabilities of VR with the features of an ideal interface. There after, we present the design of the VR-DIS project as an example of an innovative VR user interface development. We discuss the principles that underlie the design decisions.

2. The Ideal user interface

An inspiring theoretical framework for finding possible answers to this question is provided by Donald Norman (Norman,1988) who points out two important kinds of cognition:

- Experiential cognition
- Reflective cognition

Experiential cognition is a mode of cognition with which we can effectively respond on percieved events around us, without apparent effort or delay. It can also be called reactive or reflexive cognition. It appears to flow naturally, but years of experience or training may be required to make it possible. Take the example of typing. When we start learning to type, at first we have to think about finding keys, and so on. With practice, the typing skill becomes autonomous. This means that we no longer have to think consciously about it. Clearly, skilled performance, such as typing, requires a lot of mental processing, but that processing occurs independently of conscious thought. Experiential thought is essential to skilled behaviour.

Reflective thought is that of concepts, planning, reconsideration, comparison and decision making. It is this kind of cognition that leads to new ideas and novel responses. While experiential cognition appears quick and without apparent effort, it can only address information already existing somewhere in our memory. Reflective cognition on the other hand, can address new situations, but the price one pays is that the process is slow and laborious. Reflective though requires the ability to store temporary results, to make inferences from stored knowledge, and to follow chains of reasoning backward and forward. This process takes time and effort.

In the ideal case, a computer interface requires itself no mental effort to be use. Based on the previous notions of cognition, this could theoretically be realised in an interface that can be read and controlled by pure experiential cognition.

3. The ideal medium

Experiential cognition requires training. Complex motor skills can even take years to acquire. For obvious reasons, must computer interfaces can't be based on a motor skill that requires the users to go through such a tedious training program. An new interface can get around this problem when it is being based on an experiential skill that the users already have.

Some very powerful experiential skills are acquired by everybody during childhood. People's most effective universal skills are: our perception capabilities, especially the perception of spatially structured visual information, further the human motor capabilities, and the human language capabilities in the form of either talking or writing. VR technology now provides the means to exploit the human skills of 3D visual perception and 3D motor actions better than any other user interface technology did before. We subsequently discuss how VR facilities the exploitation of motor and perception skills.

First generation of computer systems could only be controlled by typing commands on a keyboard. The language capabilities of these computer systems were limited to a small specific vocabulary and strict syntactical rules. These command languages are very unlike natural human languages. It requires considerable effort to use them. The computer mouse has added a very effective input channel to the computer. It exploits the human motor capabilities, though in a very limited way.

VR technology now offers to exploit the human motor capabilities more fully. Some new devices make it possible to track 3D hand and head gestures, while others track the 3D forces that are performed on them. This enables us to build user interfaces that let a user input data by moving objects, connecting objects to each other, moulding objects, and so on. Of course, the success of such an innovative interaction paradigm stands or falls with what is usually called the interaction metaphor. This is the mapping between the application's semantics and the user's actions.

The computer's imaging capabilities have evolved together with their control possibilities. Computer imaging capabilities have evolved from pure text displays, to 2D graphical displays, to the interactive 3D graphical displays of the current VR systems. Virtual worlds can be rendered with photo-realistic quality, all interactively controllable by the user. The combination of 3D computer imaging and 3D motor control can potentially reduce the effort needed to control a computer application to the same ease as that of controlling objects in our every day surroundings.

4. The VR-DIS user Interface

As an example of an innovative VR user interface, we discuss the VR-DIS project. VR-DIS is a large research project at the Eindhoven University of Technology. The acronym VR-DIS stands for Virtual Reality - Design Information System. The goal of the project is the development of an integrated design system for the building and construction industry. The innovative nature concerns the integration of information from different building and construction disciplines within a dynamic design process. It aims at including the design expertise of the architects, structural engineers, HVAC engineers, planners, facility managers, etc. Consequently, large amounts and many different kinds of data need to be communicated to the users.

In order to communicate such an unusual large amount of information in an effective way to the user-designer, a broad and easy comprehensible interface medium is required. In the VR-DIS system, VR has been chosen as the primary User interface technology.

The development of a VR user interface currently involves the design of all aspects of the interface. Unlike conventional windows and mouse interfaces, VR has yet to evolve a dominant interaction paradigm. The application of VR interface technology currently requires thoughfull design of the data visualisation, the interaction method, the input and output devices.

4.1 Data visualisation

In VR-DIS, the data visualisation is based on the notion of mixed-representations. Mixed representation interfaces are interfaces that combine one or more descriptive representations with one or more pictorial representations of the same data (Coomans, et al.,1998). Such a combination provides a more complete view of hybrid data. It makes it possible to develop a single, rich interface that can be used by a multi-disciplinary design team.

In the current stage of the VR-DIS project we focus on the early architectural design stage. For this early stage, we have chosen to combine two views : a "feature view", and a "scale model view".

The feature view is a spatially structured verbal representation of the database that underlies the VR-DIS system. This database is organised in "features", hence the name. The modelling of architectural design information in features has been developed by van Leeuwen (vanLeeuwen,1997). 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 are presented in a web or graph-like representation. See left part of figure 1.

The scale model view is a pictorial representation of the design information, based on the physical appearance of the designed building. This view will feature a high level of verisimilitude. See right part of figure 1. Besides the native perceptible characteristics of the building-design, this view will also be used to visualise information that is normally invisible, in a pictorial way. For example: heat loss data can be visualised on the scale model by giving all external walls a color between red, for a lot of heat loss, and blue, for no heat loss.

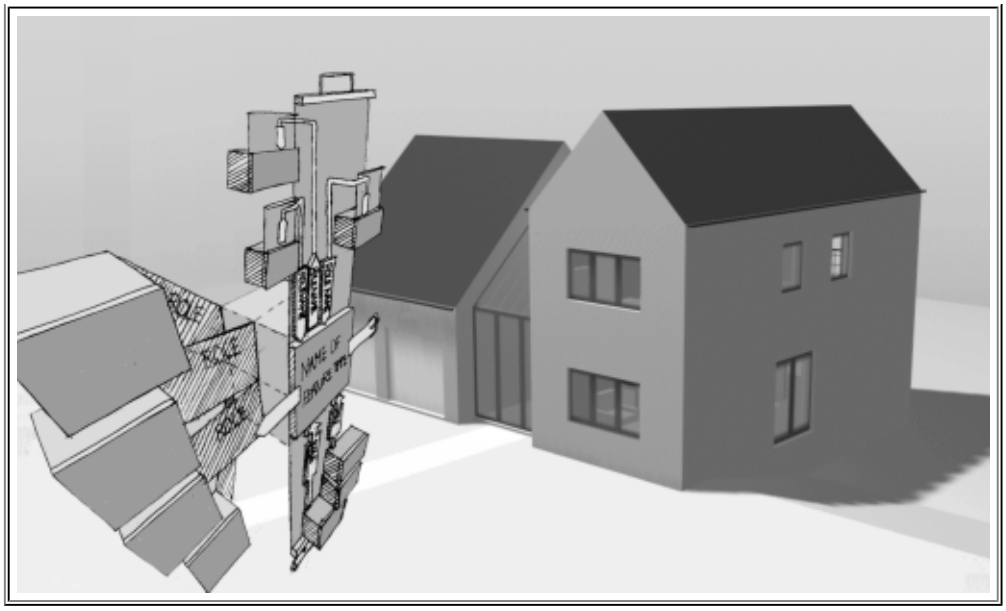


Figure 1: Integrated feature and mock-up view

4.2 Interaction method and devices for system output

In principle, the interaction method and the hardware devices are separate things. In practice however, they can hardly be discussed separately because the characteristics of the one determines the other. Here we discuss the output subsystem. In 4.3, we discuss the input subsystem.

The VR-DIS graphics subsystem makes use of what Ware has termed "fish tank VR" (Ware, et al., 1993). Fish tank VR is characterised by in real time generated MarcCoomans_CCAI98_images of a 3D scene, viewed on a monitor, with the perspective projection of the MarcCoomans_CCAI98_images coupled to the head position of the observer. The MarcCoomans_CCAI98_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.

This fish tank functionality is currently being prototyped in the VR-DIS system; see figure 2. We expect that fish tank VR can produce a similar feeling of being deeply engaged as immersive VR can. (Immersive VR is the form of VR that makes use of head-mounted-displays or all-side projection rooms, such as CAVE™s). While immersive VR brings the user in the virtual environment, fish tank VR is expected to bring the virtual objects to the viewer in an equally convincing way.

Figure 2 – Mock-up of the VR-DIS Fish tank VR system.

4.3 Interaction method and devices for system input

The selection of the input devices and interaction method 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. See figure 3.

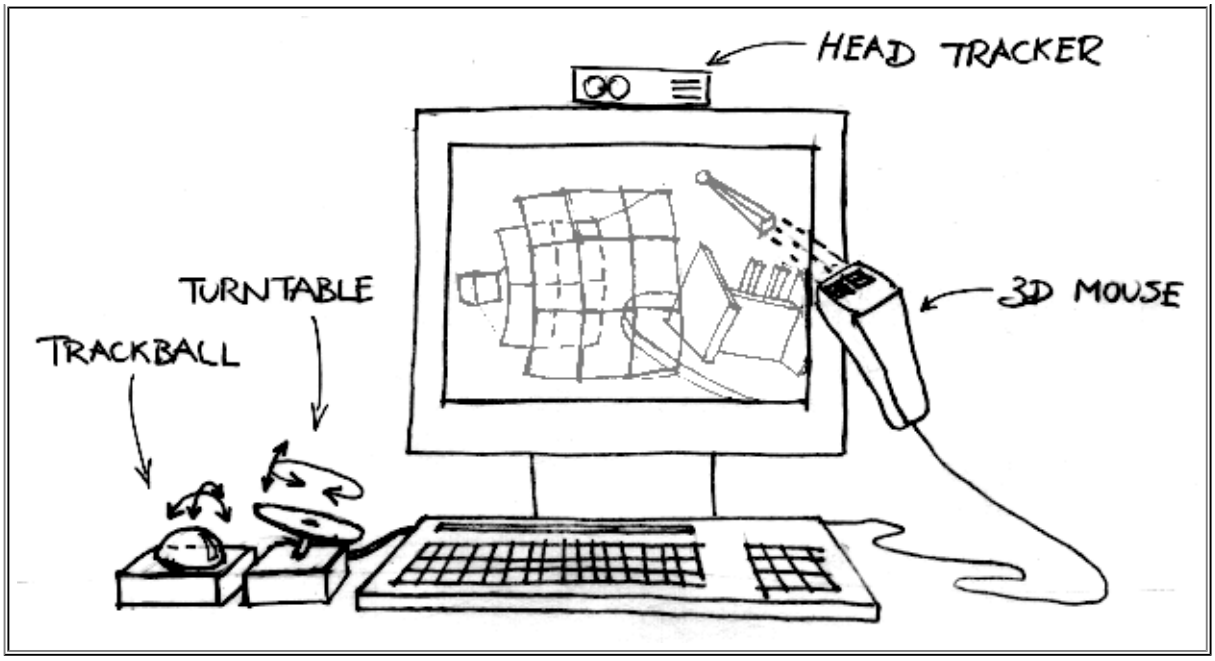


Figure 3 –VR-DIS interaction devices.

The trackball is used to manipulate the "feature box", the main tool of the feature view. 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 future 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 quickly responding virtual pen to 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.

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.

5. The VR-DIS feature view

In the previous section, we discussed the design-considerations that underlie the global VR-DIS interface. Now we will discuss in more detail the design of the feature view.

5.1 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 (van Leeuwen,1997) 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.

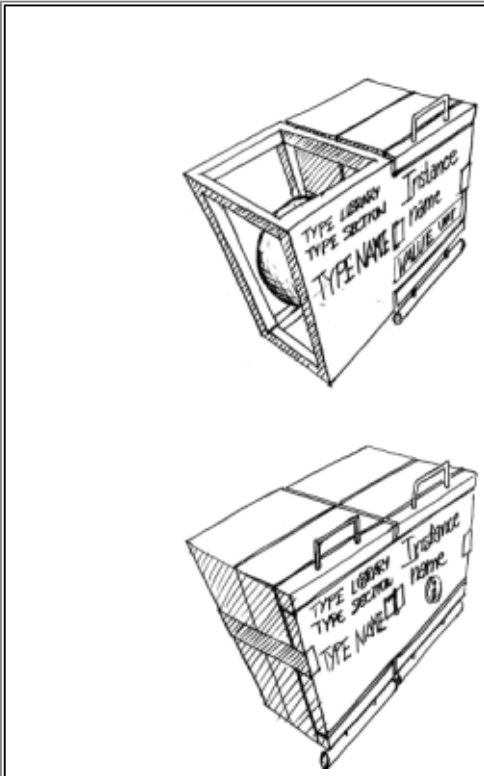
The visualisation design of the feature model can be split in two parts: (1) the visualisation of single features and their relations, and (2) the graph visualisation technique that can cope with the scale of the feature models.

5.2 Visualisation of single features and their relations

A feature is visualised as shown in figure 4. 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 4 – Visualisation of a Simple and a Complex Feature, in initial state

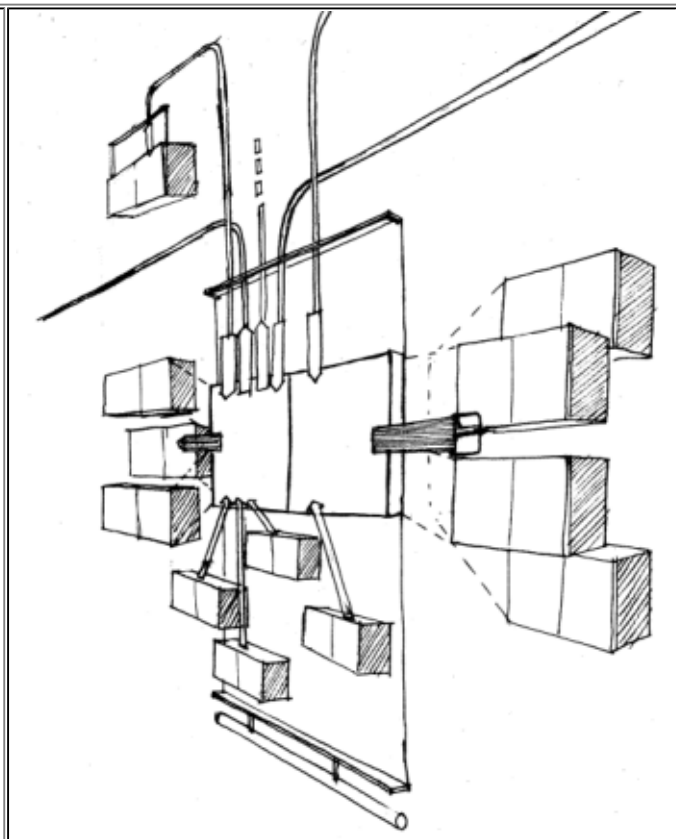


Figure 5 – Visualisation of a Complex Feature, in exploded state

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 5). 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.

5.3 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 MarcCoomans_CCAI98_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". Figure 6 shows a screen-dump of the first prototype of 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.

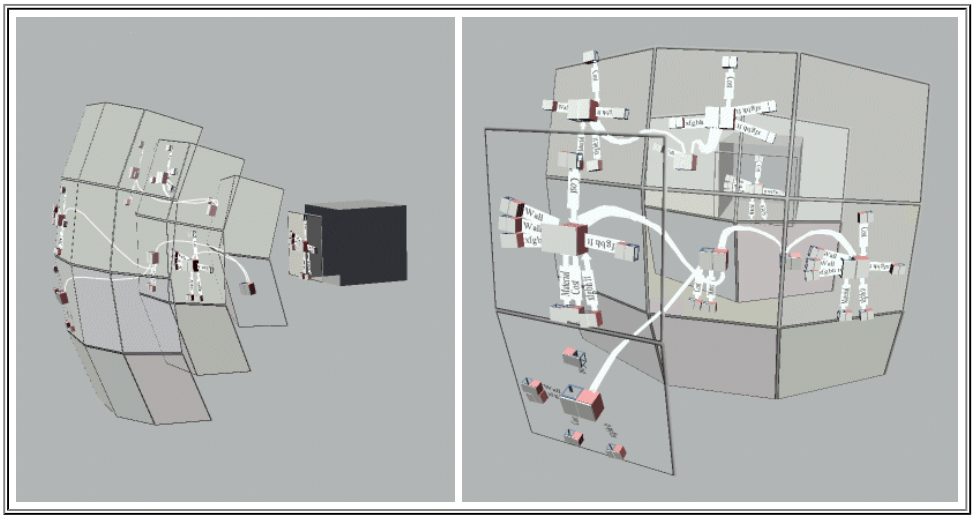


Figure 6 – Feature Tool

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.

5.4 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.)

6. Implementation and further work

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.

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, and the scale model view will be worked out.

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.

7. Conclusions

We argued that Virtual Reality (VR) can fundamentally improve the user interface of many business applications. We presented the theoretical basis for this, referring to Donald Norman's theory. We showed that that VR provides at least theoretically, the means to take a big step in the direction of an ideal user interface.

As an example of an innovative VR user interface development, we presented the VR-DIS system. We discussed the issues underlying the design. VR-DIS makes 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 VR-DIS system is characterised by a mixed representation of the task domain. We currently support the early architectural design stage by a combination of an pictorial "scale model view" and a descriptive "feature view". These views are integrated in a single virtual environment. 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 VR technology can much better support many user tasks than conventional interface techniques. We expect that VR provides the means to develop easily usable user interfaces for computer applications that deal with complex task domains.

References

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

Norman, D. A., (1988), The psychology of everyday things, Basic Books, New York, 1988, 257 p.

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.

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

Eindhoven University of Technology.