EUROGRAPHICS 2001 / A. Chalmers and T.-M. Rhyne
(Guest Editors)

Volume 20 (2001). Number 3

Interactive Content for Presentations in Virtual Reality

Anton.L.Fuhrmann,

Jan Přikryl and Robert F. Tobler

VRVis Research Center for Virtual Reality and Visualization, Vienna, Austria

Werner Purgathofer

Vienna University of Technology, Austria

Abstract:

We develop concepts for presenting interactive content in form of a slideshow in a virtual environment, similar to conventional desktop presentation software. We demonstrate how traditional content like text and images can be integrated into 3D models and embedded applications to form a seamless presentation combining the advantages of traditional presentation methods with 3D interaction techniques and different 3D output devices. We demonstrate how different combinations of output devices can be used for presenter and audience, and discuss their various advantages.

1. Introduction

Virtual Reality systems are per definition well suited for the presentation of interactive 3D content. At the moment, there are two ways to actually implement such a presentation: an authoring tool can be used to construct a VRML world with limited interaction and scripting possibilities, or a fully-fledged VR application can be written, using one of the available generic toolkits or environments.

We propose a framework capable of simple authoring of content and generic 3D interaction, and extendable to include complex interactions and simulations by dynamically embedding applications tailored to specific presentation needs.

One of our main concerns is scalability: simple content and interactions should be simple to author, but this simplicity should not prevent us from integrating complex content or interactions using the necessary effort.

2. Related Work

Generic commercial [1] and academic systems like DIVE [2], EMMIE [3], and our own system Studierstube [4] can be used to implement applications for specific demonstration purposes. VRML [5] implements a generic file format for the description of interactive 3d-content to be displayed from within a standard web browser. It is possible to produce

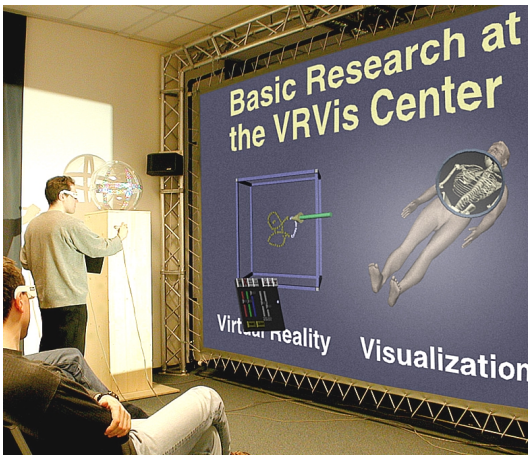

Figure 1: *3D painting in an embedded painting application & medical visualization w/magic lens*

transitions between different scenes similar to our slide transitions (→ section 5.2) by defining multiple viewpoints in VRML, but the execution of this concept is browser-dependent and not controllable. Java3D [8] takes scene description one step further towards VR by integrating concepts describing properties of head-mounted or projection based displays and interaction devices. It would indeed be possible to implement most of the concepts presented here in Java3D, although execution speed concerns remain.

For VRML authoring applications exist, that allow simple modeling and scripting of content, but most generic VR systems require users to integrate their content by programming a completely new application.

While these applications are necessary for the integration of new functionality into a virtual environment, the presentation of existing applications or models should in our opinion not require the development of new software or the adaptation of existing, stand-alone VR applications.

3. Concepts

Traditional presentation software allows the user to quickly design slides containing typeset text and graphics, enhance them with multimedia content like videos and sounds, and order them in a sequence connected by slide transitions. The result of this procedure is a sequential slideshow, which can be presented on a projection screen in front of an audience. Divergences from a strictly linear sequence can be used to integrate optional content, and interactions within slides allow some variation in the presentation. To demonstrate "live" content, the presentation is usually interrupted and replaced by the application to be demonstrated.

To implement a presentation system in VR, we have to transfer and extend these concepts from the conventional computer desktop to the virtual environment (VE).

- **3D Interaction**
  Interaction must be extended into three dimensions and six degrees of freedom.
  New interaction methods should not necessarily require programming skills.
- **3D Slides**
  We extend the flat slide concept into a slide containing a *volume* filled with 3D content.
  This extension requires new, three-dimensional content types.
- **Presentation Scenarios**
  We have to support different hardware and interaction concepts and evaluate their applicability for a range of presentation scenarios.
- **Content Elements for Interaction**
  To implement simple, generic 3D interaction within the authoring of content, we integrate interaction elements directly into the content description.
- **Embedded Applications**
  For complex interactions and simulations we propose the embedding of applications into 3D slides. These applications should not need to be specifically developed for the presentation.

These concepts are explained in detail in the following sections.

4. Three-Dimensional Interaction

Interaction on the desktop is performed via keyboard and mouse or by wireless devices like a projector's remote control.

Interaction in our VE happens via tracked interaction devices delivering absolute coordinates. The position of the presenter therefore has to be taken into account when planning interactions. Interaction via the keyboard may be used, but its availability depends on the setup used.

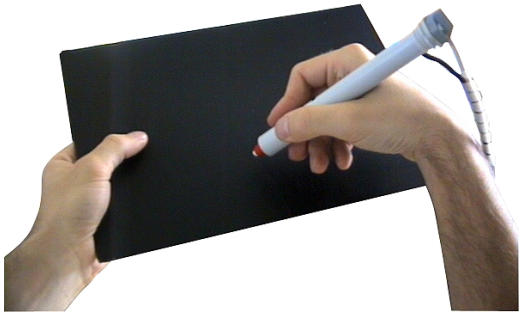

Figure 2: PIP - real tracked pen and pad devices

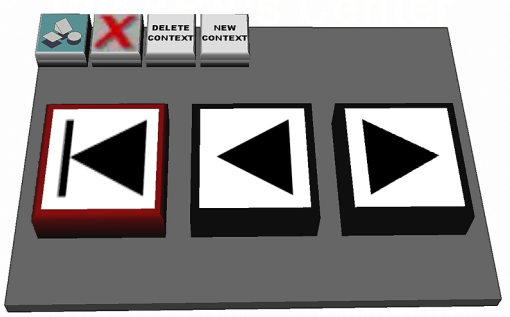

Figure 3: PIP with slideshow controls

We implement complex control of the presentation – i.e. more complex than requesting the next slide – by using the *Personal Interaction Panel* (PIP) [4], a simple tracked pad-and-pen combination on which we display the necessary interaction elements. Figure 2 shows the real counterparts to the virtual elements. The PIP is usually controlled by the presenter, who employs it for the slide show controls start, forward

and backward (Figure 3) and for interaction with embedded applications (Figure 9, →section 8). The other controls visible on the top edge are Studierstube system controls. The pen can be used as a 6DoF interaction device on its own, and implements our main means for *direct interaction* (→ section 7).

5. Three-Dimensional Slides

On the "flat" screen, each slide covers the whole area of the desktop and takes its reference coordinate system from the desktops extensions. In the VE, a slide could theoretically be positioned at any orientation anywhere in space. We need a "natural" coordinate system, which defines the *volume* the slide may use.

5.1. Slide Reference Frames

Studierstube may be used on different hardware setups: projection screen, virtual table and HMD are all supported. This imposes serious differences on the way the content has to be displayed. On a projection screen and a virtual table the presentation has to be aligned with the display surface, whereas with an head-mounted display (HMD) setup the presentation can be displayed anywhere in the working volume imposed by the tracking device.

5.2. Slide Transitions

The transition from one slide to the next is implemented on the desktop as more or less complex transition between two images. The possibly least distracting method is the simple switch between the display of the first and the following slide.

In 3D this transition becomes more irritating, since the switch can be performed between a nearly flat slide and one with extremely protruding 3D content, resulting in accommodation problems or – in extreme cases – jumping back of the audience (an effect much used in spectacular 3D movies is not necessarily an improvement for a presentation). Transitions can be easily implemented as animated linear transforms of the slide geometry, thereby implementing translations, rotations and scales along author-defined curves. Image-processing transitions are also possible, e.g. the fade from one slide to the other, but may lead to annoying z-buffer artifacts depending on the implementation.

Figure 5 shows a simple transition: the first slide on the left is rotated back using left screen edge as a "hinge", while the next slide (also shown in Figure 10) rotates in around the right edge.

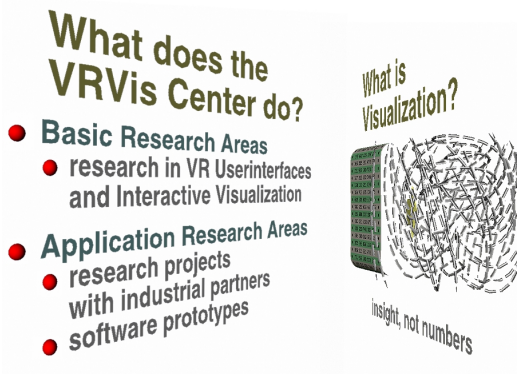

Figure 5: Slide transition "Door"-style

6. Presentation Scenarios

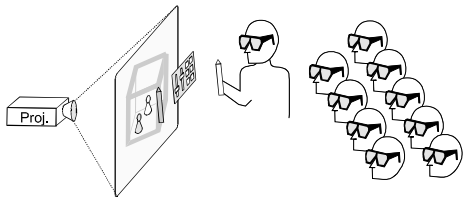

Figure 4: Frontal presentation scenario: all users view stereo projection

Presentations on desktop systems are always performed in the same setting: one presenter and one or more spectators. In a virtual environment, this does not necessarily have to be so: Studierstube implements a concept supporting multiple users or views in different locales [9], thereby separating content and interaction in a manner similar to Java3D [8], but slightly more flexible.

6.1. Frontal Presentation Scenario

The use of a stereo back-projection wall (Figure 4) is ideally suited for presentations to large audiences: The stereo-effect is set up to work correctly for a spectator in the middle of the audience, thereby resulting in a sufficiently convincing spatial appearance of the content for most positions.

The presenter is usually positioned in front of the audience, slightly left or right of the screen. When interacting with the presentation the presenter faces the screen, in the same position a teacher would assume on a blackboard (Figure 1).

The screen is in most cases larger than the reach of the presenter, which can be solved in two different ways:

– The presenter walks in front of the screen to the interaction element he wants to use.

– The gestures of the presenter are scaled to implement a larger working volume.

In the first case the presenter may occlude parts of the screen, but interaction may be more transparent for the audience.

In the second case the presenter is essentially in the same position as when using a mouse on a desktop presentation. Interaction is intuitive as soon as the hand-to-eye offset is learned, and the audience soon understands the correlation between the gestures of the presenter and the cursor movement.

The main disadvantage of this scenario lies in the differences between hand-to-eye coordination with an offset on a 2D desktop and in 3D: the distorted stereo-view the presenter perceives from his position does not allow precise interaction in the working volume. A possible remedy for this is described in the next sections.

6.2. Multi-User Scenario

Demonstrations for smaller groups (2-3 protagonists) can be performed using head-mounted displays (HMDs) (Figure 7). This *multi-user* setup implies several differences to a conventional presentation: the different spectators may see completely different views of the same slide. In 2D this would only result in differently distorted views of the screen with essentially the same content, but in 3D this may lead to occlusion of important features. On the other hand, a setup where each user may choose his own viewpoint, and – more importantly – where each user may *interact* with the presentation has obvious advantages for scenarios where a group of people discusses a common topic.

The completely separated data-paths to each participant additionally enable finer distinctions between what is presented to each user. Content may be displayed on demand, and individual users can select not only their viewpoints, but also which aspects of the content to view. A scenario using one common projection surface (projection wall or CAVE) is not able to supply these user-specific views.

This scenario is also much more symmetrical than the others: while one user may assume the role of a presenter, initiating and guiding the presentation, there is no real technical reason, why the roles should not change during the presentation. A case where a workgroup presents results to their manager, for example, would consist of multiple presenters and only one spectator.

The disadvantages of this scenario are more of a technical nature: head-mounted displays are more expensive per user than shutterglasses or polarized glasses, and they deliver in most cases display quality inferior to projection-based setups both in resolution and image stability.

6.3. Combined Scenario

A combination of these two presentation setups makes sense, too: the presenter wears an HMD, while the audience follows his actions on a stereo projection screen (Figure 6). The main advantage in this setup lies in the correct viewpoint, which can be displayed in the HMD. In the projection-only setup, the viewpoint is static, i.e. calculated somewhere sufficiently near the center of the auditorium to give an acceptable stereo effect for the whole audience. Under normal circumstances, this means a distorted view for the presenter, who stands in front of the audience. Almost the same effect can be seen at the TV weather report: the presenter tries to compensate for his different view of the scene by learning a different hand-to-eye coordination.

In the combined setup these difficulties do not arise: the presenter sees his personal presentation within the reach of his arm and may see and manipulate the applets or the PIP with correct perspective. Furthermore, additional information can be displayed

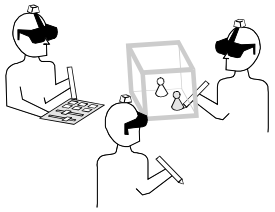

Figure 7: Multi-user scenario: all users use HMDs

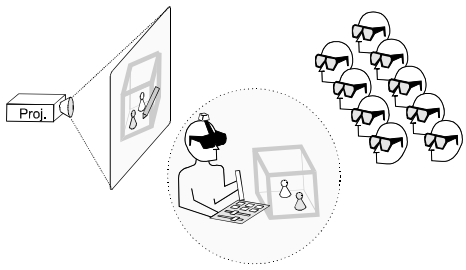

Figure 6: Combined scenario: presenter uses HMD, spectators view projection

for the presenter only, like annotations or a virtual tele-prompter. To implement this, we have to let the presenters interactions take place in his own *locale*, i.e. a local coordinate system / viewer / scene combination [9].

The disadvantages of the combined setup arise from the necessary interaction between presenter and audience. First, the HMD covers parts of the presenters face, making conversations slightly awkward, and second, the different locales for interaction of the presenter and display for the audience decouple the direct interaction between audience and presenter: when a spectator points at the screen, the presenter has to be able to quickly identify the selected point in his "private universe".

On the other hand the combined setup seems to be the ideal strategy for the presentation of complex 3D interactions in applets. The "over-the-shoulder" view this setup presents to the audience, and the correct perspective and point of view it gives the presenter represent in this special case the best for both parties.

7. Content Elements for Interaction

Not all interactive content needs the complexity of an applet. Simple 2D interaction elements (widgets) suffice for many cases. E.g. to selectively display different aspects of an architectural visualization like walls, wiring, or plumbing, a control with the functionality of 2D radio-buttons or check-boxes would be enough. We have integrated most of the standard 2D widgets including sliders and dials into the presentation system, where they can be used to control VRML models and animations.

We extend these purely 2D interaction methods – which nevertheless are operated using a 3D interaction device - by widgets implementing the most common 3D interactions. Movement is implemented by 3D-Dragger widgets able to position and orient a model by click-and-drag operations in 3D using 3 or 6 degrees-of-freedom (Figure 10, right). Rotations can be performed using a virtual trackball (Figure 8).

In combination these widgets account for most simple interactions one wants to integrate in a presentation of static – in the sense of precomputed, but possibly keyframe animated – content. Our implementation enables the author to integrate these interaction elements in a VRML scene instead of the standard VRML sensors, which support 2D interaction on the screen only.

8. Embedded Applications

The integration of "live" content in the form of running applications is one of the most flexible features of our presentation concept. The integration of applications as content elements into the layout of a slide allows us for example to explain a new visualization method on a slide and then demonstrating the method on the following slide. Applications in this context are fully-fledged Studierstube applications with all capabilities thereof. Their 3D output volumes (3D Windows) [9] are positioned relative to the slide. This allows to arrange the layout of the slide correctly and to integrate the applet into the slide transitions.

The user interacts with these applications directly using the pen in their working volume, or indirectly via widgets on the PIP.

8.1. Direct Interaction

The most intuitive control can be executed over an applet when *direct interaction* is used. In this case the interaction device – e.g. the pen – is placed inside the applets 3D window and is used to manipulate geometry or indicate actions by click or drag operations exactly like one would interact with a

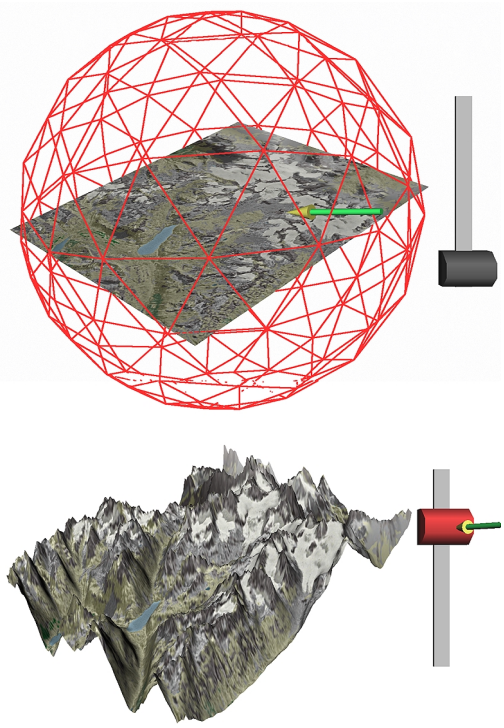

Figure 8: Interaction with widgets: rotation via trackball (top), scaling via slider (bottom)

conventional application. This kind of interaction is highly intuitive when applied correctly, i.e. when an correlation between the gesture and the results is easily recognizable.

An example for this is given in Figure 1, where the presenter uses a simple embedded application to paint or spray in three dimensions. The window of the application is shown as perspectively distorted box and can be moved and resized.

8.2. Interaction via widgets

Not all parameters of an application are suited for direct interaction. When we want to control numerical parameters for example, a slider or a dial makes more sense. These controls could be attached in the applets working volume, which makes sense when the direct relation between the sliders position and the resulting changes in the applets output should be shown side by side. In many cases where more widgets have to be used, or when the parameterization only concerns the presenter, we place these interaction elements on the PIP (Figure 9).

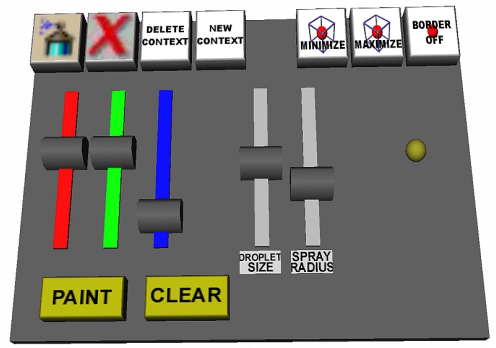

Figure 9: PIP with sliders and buttons to control the 3D painting application.

This is used for example in Figure 1, where the presenter uses sliders displayed on the PIP by the abovementioned embedded painting application to vary the color and size of the sprayed spheres, or to clear the canvas. These sliders and additional buttons are displayed on the PIP (Figure 9) when the user manipulates the embedded application.

8.3. Focussing Strategy

This presents us with the problem how the PIP can be shared by the presentation application – which uses it for the control of all slides – and the embedded application, which needs to use it only when it is visible. To solve this conflict, we implemented a *focusing strategy*, similar to equivalent strategies in 2D window managers. The PIP only shows the controls of that part of the presentation, which has the *input focus*, e.g. the presenter or an applet. Focus changes have to be implemented via a click-to-focus strategy, otherwise the PIP would change back from application focus every time the pen leaves the applets working volume and enters the presenters volume (i.e. anywhere else).

9. Authoring

We use VRML as main authoring interface to our application, since it has become something of a de facto standard for the exchange of 3D data. The VRML 2.0 standard file format allows us to integrate 3D content in form of geometry or animations into our presentations. Extended by the special nodes we implemented (3D-widgets, stereo textures, and embedded applications) it serves as an easy scripting language for interaction as well as static content.

Authoring can be coarsely divided into three separate procedures:

- Slide content and layout
  (including simple interactions with widgets)
- Presentation design
  (transitions and slide sequence)
- Application design

Slide content can be generated by hand, using a VRML capable modeler, or as direct file output of some user-specific software module. The VRML for static geometry is sufficiently simple to quickly implement an output routine into most databases or applications. Converters from different standard file formats (DXF, IGES, etc.) into VRML are also available.

Although VRML is very well suited for describing 3D models, it lacks some features necessary for creating 3 dimensional presentation slides. In order to integrate slide-layout functionality, and presentation styles, we developed a small macro facility called PYM (Python Macros) [PYM].

This macro facility together with a specially developed package of macros implements standard layout operations, such as paragraph styles, automatic line-breaks, and parameterized transitions. PYM expansion produces a VRML file for each slide, which contains special nodes for interaction elements and embedded applications.

An example for a presentation slide before the PYM macros have been expanded might look as follows:

```
#include "wrl_slide.pym"
```

```
@[ Title( text=[ "What is Visualization?" ])]@
@[ VRMLexternal( name =
       "ScrollWithNumbers.wrl",
       pos  = V_CENTER + H_FIRST_QUARTER ) ]@
@[ VRMLexternal( name = "Arrow.wrl",
       pos = V_CENTER + H_CENTER ) ]@
@[ DragableObject(
       VRMLexternal( name =
       "AnimatedVortex.wrl",
       pos  = V_CENTER + H_THIRD_QUARTER )) ]@
@[ NoBulletLine(
       text = [ "insight, not numbers" ] ,
       pos = V_BOTTOM + H_CENTER ) ]@
@[ MasterSlide( NOBACKGROUND ) ]@
```

This definition – consisting of some text, two included non-interactive objects (“ScrollWithNumbers” and “Arrow”) and one dragable object (“AnimatedVortex”) - expands to the slide depicted in Figure 10: The “AnimatedVortex” geometry on the right - which contains a streamline visualization of a vortex with animated textures – shows a highlight in form of a bounding box when the pen is inside and signals thus its interactivity.

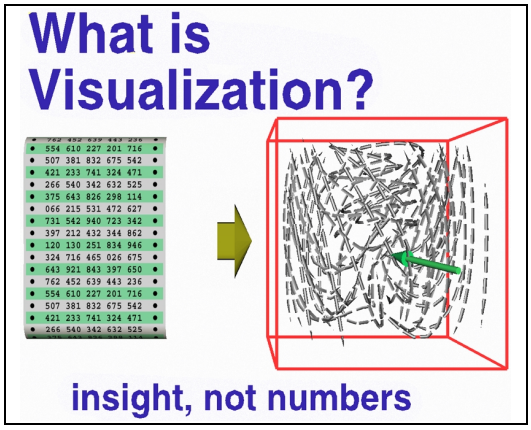

Figure 10: Slide with draggable element on the right.

The overall sequence of the slideshow is also defined using our macro package. This makes it possible to easily specify the transitions between the slides, and choose individual parameters for each transition. PYM handles named parameters and default parameters, so that only parameters that override the defaults have to be specified.

Here is the source for a small example slide show:

```
#include "wrl_show.pym"
@[ Sequence(
       [ "VRVis": "vrvis.wrl",
         "VR": "vr.wrl",
         "Vis": "vis.wrl",
         "Center": "center.wrl",
       ])]@
@[ Transition(
       slide1 = "VRVis",
       slide2 = "VR",
       animation = "CenterRotation" ) ]@
@[ Transition(
       slide1 = "VR",
       slide2 = "Vis",
       animation = "HorizontalFlip" ) ]@
@[ Transition(
       slide1 = "Vis",
       slide2 = "Center",
       animation = "VerticalFlip", ) ]@

@[ SlideShow( pen_object = "arrow.wrl" ) ]@
```

The above code describes a presentation consisting of four slides named “VRVis”, “VR”, “Vis”, and “Center”. This sequence of slides is connected by three different transitions, “CenterRotation”, "HorizontalFlip", and "VerticalFlip". The last line sets as optional argument the appearance of the pen as an arrow-shaped pointer.

While the necessary scripting is not more complicated than writing simple HTML-pages, we plan to implement some graphical interface producing this code.

The last procedure – application design – is an optional part of the presentation design workflow. It is in most cases not necessary to implement a specific application to present. As mentioned previously in section 7, many interactions, especially 3D-specific 6DoF movement and parameterizations via sliders or buttons can be implemented as VRML scripts (VRML routes, to be precise). An example for this is depicted in Figure 8, where the output of an geographic information system can be rotated via a virtual trackball widget (top), and an exaggerated scale of the mountains can be adjusted on the slider attached to the scene (bottom).

10. Implementation Details

As basis for our system, we use *Studierstube* [4], our generic virtual environment. We implemented the slideshow application *“Presenter”* as a Studierstube applet in C++. Studierstube is based on the open source distribution of SGIs OpenInventor.

The 3D painting application is a previously existing Studierstube demo application and was embedded in the slideshow without modifications.

All applications – even the presenter application itself – are dynamically loaded modules, which can be executed standalone or in combination with each other.

The presenter application has been developed and tested on PCs with hardware 3D-accelerator (GeForce 2) using both Windows 2000 and Linux. The presentation environment consists of an SGI Onyx2

executing the IRIX version of Studierstube. Applications source code can be used with Studierstube executing in any of the supported operating systems.

11. Results and Future Work

We have been employing the presentation system for some months now, both for demonstrations of new applications inside our company and for public relation purposes. Especially when demonstrating new interaction concepts it proved to be a valuable and elegant tool, since conventional content (text, diagrams) and the actual VR applications were embedded in one seamless presentation.

We used the presentation application in different setups (HMD and head-tracked setup on the virtual table) to explain and demonstrate interaction as "hands-on experience" for single users. The imposed sequence of applications and explanations supported the educational flow of our demonstrations very well and kept users from "getting lost in the interface".

Interaction without head-tracking during demonstrations in the frontal presentation scenario proved to be slightly difficult, but since this scenario implies a trained presenter and no interaction from the audience we were able to compensate for this problem after some training.

We plan to provide the presentation system with an graphical authoring interface from within the application. This interface should make it possible to choose from existing content in form of models or text, place them in 3D within slides, and select transition effects.

Acknowledgments

Special thanks to Rainer Splechtna for his ingenious implementation and thanks to Michael and Peter who appeared as extras in the video.

Links

For further material concerning this project visit: **http://www.vrvis.at/br1/projects/presentation/**

References

1. Ron Fosner: Virtual Reality and the WorldToolKit for Windows. Dr. Dobb's Journal of Software Tools, 20(1), p. 78, 80, 82, 102-105, January 1995.
2. C. Carlsson, O. Hagsand: DIVE- A platform for multi-user virtual environments. Computers & Graphics, Vol. 17, No. 6, pp. 663-669 (1993).
3. A. Butz, T. Höllerer, S. Feiner, B. MacIntyre, C. Beshers, Enveloping Users and Computers in a Collaborative 3D Augmented Reality, In: Proc. IWAR '99 (Int. Workshop on Augmented Reality), San Francisco, CA, October 20-21, 1999, pp. 35-44
4. D. Schmalstieg, A. Fuhrmann, G. Hesina, Zs. Szalavari, L. M. Encarnação, M. Gervautz, W. Purgathofer: "The Studierstube Augmented Reality Project". Submitted for publication. Available as technical report TR-186-2-00-22, ftp://ftp.cg.tuwien.ac.at/pub/TR/00/TR-186-2-00-22Paper.pdf
5. Rikk Carey, Gavin Bell: The Annotated Vrml 2.0 Reference Manual. Addison-Wesley, 1997.
6. D. B. Conner, S.S. Snibbe, K. P. Herndon, D. C. Robbins, R. C. Zeleznik, and A. van Dam. Three-Dimensional Widgets. Proc.SIGGRAPH Symposium on Interactive 3D Graphics, 25(2):183-188, 1992.
7. D. Schmalstieg, M. Encarnação, Zs. Szalavári: Using Transparent Props For Interaction With The Virtual Table. Proceedings of SIGGRAPH Symposium on Interactive 3D Graphics '99, Atlanta, GI, April 26-28, 1999.
8. Henry A. Sowizral, Kevin Rushforth, Michael Deering: The Java 3D Specification. Addison-Wesley, 1998.
9. D. Schmalstieg, A. Fuhrmann, G. Hesina: Bridging Multiple User Interface Dimensions with Augmented Reality. Proceedings of the 3rd International Symposium on Augmented Reality (ISAR 2000), pp. 20-30, Munich, Germany, Oct. 5-6, 2000.
10. Robert F. Tobler: PYM - A Macro Preprocessor based on Python. Proceedings of the 9th International Python Conference, Long Beach, California, March 2001.