GRAPHICS '99 / P. Brunet and R.Scopigno
(Guest Editors)

Volume 18 (1999), number 3

Concept and Implementation of a Collaborative Workspace for Augmented Reality

Anton Fuhrmann and Dieter Schmalstieg

Institute of Computer Graphics, Vienna University of Technology
Vienna, Karlsplatz 13/186/2, A-1040, Austria
{fuhrmann|schmalstieg}@cg.tuwien.ac.at

Abstract

In this paper we present Studierstube Workspace*, an application framework which supports multiple users, multiple applications and multiple 3D and 2D interaction contexts in an augmented environment. We develop a design concept for collaborative working environments in virtual reality, especially the concept of* multi-user aware applications *and show how our implementation fits into this concept. Implementation details and application samples conclude the paper.*

Keywords: computer graphics, virtual reality, augmented reality, user interface, Workspace, multi-user aware applications

1. Introduction

Technical progress in recent years gives reason to believe that virtual reality (VR) has a good potential as the user interface of the future. At the moment, VR applications are usually tailored to the needs of a very specific domain, such as a theme park ride or a virtual mock-up for design inspection. We believe that augmented reality (AR), the less obtrusive cousin of VR, has a better chance to become a viable user interface for everyday applications, where a large variety of tasks has to be covered by a single system.

Studierstube[22] is a a collaborative augmented reality system allowing multiple users to gather in a room and experience the sensation of a shared virtual space that can be populated with three-dimensional data Figure 1. Head-tracked see-through head-mounted displays (HMDs) allow each user to choose an individual viewpoint while retaining full stereoscopic graphics. A two-handed pen-and-pad interface, the personal interaction panel (PIP), is used to control the application[28].

Studierstube is intended to be a collaborative AR user interface in which a variety of tasks can be performed. Such a user-interface is opposed to a dedicated application that is designed for only a single purpose (e. g., a driving simulator[19]). The envisioned interface must satisfy three essential requirements:

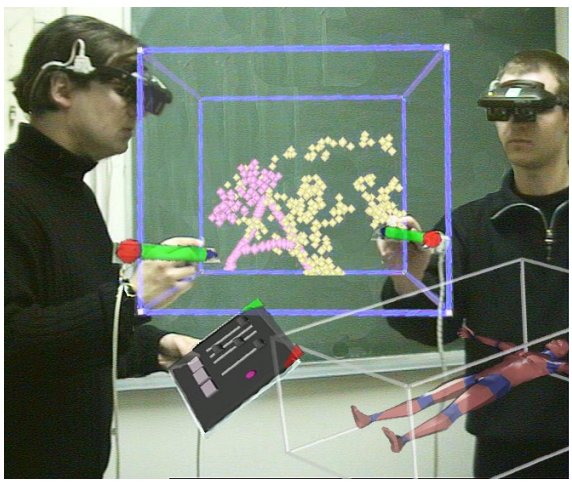

Figure 1: *Collaborative work in STUDIERSTUBE Workspace:3D painting application window (focussed, middle) object viewer window(unfocussed, lower right)*

- *Augmentation*: It must be applicable to a shared augmented reality environment. This has two important consequences: Firstly, the organisational principles for the interface must be appropriate to 3D (and it is not trivial to transfer user interface elements from 2D). Secondly, there can be only one common three-dimensional space which is shared among all participants and imposes some spatial constraints on the interface design.
- *Collaboration*: The discussed system is designed for multiple concurrent users and must support collaborative work. While the co-presence of users in the same room allows many issues to be resolved using social protocols, the user interface design must incorporate appropriate "groupware" mechanisms so that technical concurrency issues as well conflicts between users competing for an application can be resolved.
- *Multi-application*: A Workspace has very much in common with modern multi-tasking desktop GUIs. A user should be able to interact with multiple concurrently executing applications in turn. To further complicate requirements, multiple users must be able to work with any desired application, and even work with the same application (the same instance) at once.

Clearly, such a demanding set of features for a user interface requires a powerful user interface metaphor that is equally expressive in 3D as the desktop metaphor in 2D. We propose as our metaphor a "Workspace" which can be customized by the users for their needs. This paper gives an analysis of the design space for collaborative AR Workspaces. As a result of that analysis, we present a user interface design for such a Workspace that fulfils the three requirements listed above. Finally, we discuss our prototype implementation of a development framework for the Studierstube Workspace together with initial experiences and observations.

2. Related work

Our Workspace design builds upon legacy knowledge from very different fields. In the following, we list some of the work that we consider most influential for the system presented in this paper.

Several research groups have created augmented reality applications, either using video composition[3] or see-through HMDs[12]. While those systems are intended for individual users, other work has focused on building collaborative augmented environments: the shared space project[4] and the Studierstube system[26] upon which our Workspace approach is built.

Prominent attempts to create collaborative semi-immersive settings are the CAVE™[8] and the virtual workbench[18]. While these systems are frequently used by multiple simultaneous users, they are not strictly group systems: Only one "leading" user sees correct head-tracked stereo graphics, while for the remaining users the images are often severely distorted. Recently, a workbench extension for two users was presented[1].

A different area which has inspired our design are systems for collaborative work. While such groupware[21] is already being integrated into commercial desktop environments, support for collaboration in virtual environments is currently an active area of research. However, most work focuses on tools for remote collaboration in areas such as collaborative design[9,20], collaborative visualization[35,22,15] and tele-meetings[6,17]. While such remote systems are not directly related to our approach of physical co-presence, they explore many useful ideas on how people collaborate.

A separate area of work are graphical user interfaces. The desktop metaphor that was originally conceived in the Xerox STAR project[29] is ubiquitous in today's work environments such as X-Windows[16]. The challenge lies in bridging the gap from the widely accepted document-centric 2D workstyle to a spatial 3D Workspace. While the use of 3D makes radically new interaction styles possible[7], judicious borrowing from 2D often eases the transition from the desktop to the virtual world. In particular, there are some attempts to organize content as well as re-use existing 2D developments[11,10,2]. In some sense, the PIP[32] used in our system also falls into that category as a container for 2D as well as 3D user interface elements. The cognitive coprocessor architecture[24] uses so-called 3D Rooms to provide multiple virtual Workspaces for a user.

The most directly related approach to ours is the CRYSTAL system[34] that allows a user to organize his or her work in 3D windows. The taxonomy in Figure 2 was originally introduced for the CRYSTAL system, which supports multiple contexts and applications, but lacks true multi-user support. Note that "CRYSTAL in the CAVE" is not really a multi-user application, as only one user can be active and

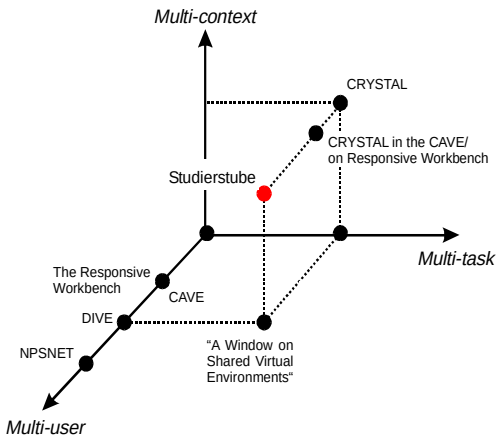

Figure 2: *Taxonomy of virtual environments (adapted from: Tsao & Lumsden[34])*

the other users are passive observers. In contrast, the Studierstube Workspace design introduced here aims at the construction of a system that is scalable in all three properties.

Finally, from a software engineer's perspective, development frameworks for user interfaces are notoriously huge and complex. A modular implementation as well as support for rapid prototyping via scripting is essential. In the area of 3D user interfaces, current prominent examples with such features that have guided our design are ALICE[23], MR[28], Java3D[30] and the VRML97 ISO standard[5].

3. Design Analysis of the Workspace

This section attempts an analysis of the design space for a user interface that satisfies the requirements stated in the introduction, namely support for augmentation, collaboration, and multiple applications. To illustrate our choices, we will first describe a few example applications, that were chosen to characterize everyday activities that users may perform in the proposed work environment. From these examples, general guidelines for the design of the Workspace can be derived.

- *Calculator:* The calculator has a desktop-like interface on the PIP (figure). Only one user may enter data at one time. Results are displayed on the PIP, and so no additional interaction or output elements are necessary.
- *3D object viewer*: The 3D equivalent to an image viewer application on a desktop is a simple object viewer. It supports the loading and display of multiple static objects in the Workspace. It does not support any interaction with the displayed objects, but allows to resize and move their representations via their surrounding 3D frames. Every object is self-contained in such a frame, and every user may move or resize it. Only one user at a time may open a new object, but the "object open" operation is accessible for every user.
- *3D Paint Application:* Contrary to the "object viewer" application, where one of many static 3D objects can be manipulated only by one user at a time, a 3D paint application should allow many users to manipulate a shared "scratch volume" at the same time. It shall be the equivalent to a blackboard: a space where a lot of users may display their ideas and add or modify the ideas of others. Each user should be able to specify his own painting tools properties - colour, width etc. - without modifying the painting mode of others.
- *Collaborative games:* A typical collaborative application is an implementation of any multi-user (as opposed to a solitaire) game. In a game of chess, white and black pieces may only be moved alternating, but every user can see the complete game state. A different example is a war game scenario consisting of terrain and different forces, where the (static) terrain exists as a real miniature model on which the forces are augmented. In this case all users can interact with each other and the forces at the same time. Every user may only move his own forces. Collaborations between different parties may result in shared access to forces and to information on enemy territories outside of the own range.
- *Educational demonstration:* In an educational setting a Workspace can be thought as an extension of the classroom into virtuality. The standard setting of one presenter and many viewers/students is of course a collaborative one, albeit a highly asymmetrical. Examples for such an asymmetry are a presentation where only the presenter may change parameters or a test situation where the teacher may see the work of each student, but where students should not be able to copy each others results.

From the mentioned examples it can be seen that the semantics of applications in the Workspace can be vastly different. Some applications are rather oriented towards a single user or a group of isolated users, while others only make sense with multiple present users. In some cases the roles of the users are symmetric, while other cases exhibit asymmetry. Nevertheless, the design paradigm underlying these applications can easily be summarized: Every interaction path should support multiple instances.

The Workspace supplies the junction point as well as it acts as an implicit "traffic control". To perform this task, the underlying framework has to support multiplicity in four distinct ways:

- *Multiple users:* Clearly the basic requirement for a collaborative setting
- *Multiple applications:* Only the support of multiple active applications at the same time enables a productivity scenario comparable to desktop environments.
- *Multiple input contexts:* By input context we mean the interaction state for input as perceived by a particular user. The same application may present the same interface elements - normally on the PIP - to different users, but the state of the elements (e. g., a selected colour) is presented on a per-user basis. Different users then see different colours on their PIP, and a change of colour by one user does not affect the chosen colour of another user.
- *Multiple output contexts:* A single application may have more than one output context, normally represented as a 3D window. These contexts may be homogeneous (i.e. representing the same object in the applications context, e. g., a different view of the same building in a CAD system), or heterogeneous (e. g., a windows that shows attributes of an object selected in another window). Both the homogeneous and the heterogeneous case will use multiple 3D windows to present the different sets of information.

In the following sections we will detail how the Workspace implements these properties.

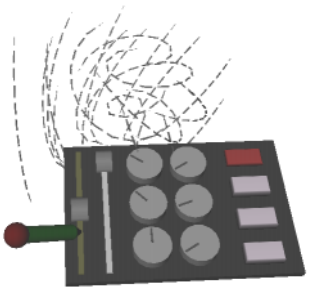

Figure 3: *use of 2D widgets on the PIP to parameterize a flow visualization*

4. 3D interaction elements

A 3D Workspace requires interface elements for efficient manipulation and customization by the user[7]. One of our design goals was the extension of as many 2D GUI mechanisms into our VE as possible. As pointed out previously[2], there are two important reasons for this approach:

Firstly, since many 2D elements have been developed and refined over the last years and have proven their worth, it would be counterproductive to completely discard the well-developed solutions for these elements.

Secondly most users are accustomed to the use of 2D GUIs. An approach that builds upon widely accepted user interface elements is more likely to ease the migration from conventional desktop computing to VE Workspaces.

While we do build upon the legacy of 2D GUIs, we do not want to impose unnecessary constraints upon our interface: we use 2D techniques where they prove to be of advantage - either by the nature of the data to be input or by the conventions associated with a specific operation - and extend them or completely abandon them when appropriate.

4.1 2D widgets in AR

For the straightforward integration of conventional 2D interface elements like buttons, sliders etc. we use PIP, a simple board with attached tracking sensor, which functions as base for the virtual 2D interface elements (Figure 3) in two ways: it allows the user to position the 2D interface conveniently using the non-dominant hand and additionally provides the necessary haptic feedback when using a slider or button. “Floating menus” or similar virtual interfaces lack this feedback and can therefore only be used when in the field of view. Every application may display its own interface in the form of a PIP “sheet”, which appears on the PIP when the application is in focus. The PIP and pen are our primary interaction devices. Both of them are tracked with 6DOF and provide the means for flexible interaction in 2D and 3D. The PIP cannot only be used as passive base but also as interaction and selection tool in itself. Its use as “virtual camera” to make 3D-snapshots of the scene as well as “fishnet” metaphor for selecting objects by sweeping it through space has been demonstrated[27].

4.2 3D Windows in AR

The use of windows as abstraction and interaction metaphor for an output context is a long-time convention in 2D GUIs. It’s extension to three dimensions seems logical[34] and can be achieved quite straightforward: Using a box instead of a rectangle seems to be the easiest way of preserving the well-known properties of desktop windows when migrating into a VE. It supplies the user with the same means of positioning and resizing the display area and also defines its exact boundaries (Figure 1). Obvious differences of these 3D windows (“boxes”) to their desktop counterparts can in many cases be resolved easily. Positioning a box by grabbing a designated part of its geometry may of course include the rotation of the window to an arbitrary orientation. Resizing is achieved by grabbing a corner and repositioning it with 3DOF, thereby changing all measurements of the box in one movement.

Differences appear in a case easily resolved in two dimensions: overlapping display contexts. On the desktop this is easily handled by a 2½D approach, rendering one window “on top” of another, effectively implementing a stacking order of windows. In three dimensions the equivalent solution would be a similar explicit order in which the windows are rendered - possibly based on importance or least recently used criteria - where each window clears the volume it defines before rendering its contents. This leads to an unambiguous assignment of each point in the working volume of the virtual environment to at most one box.

5. Multi-user aware applications

The proposed multiplicity of interaction paths implies some changes in application design. Full support of all possible interaction paths leads to some extensions to the applications interface. A multi-user aware application (MUA) has to take into account additional information concerning which user is interacting with it and keep user-dependent information. While in some cases multiple instances of single-user applications are sufficient (e.g. calculator and object viewer as described in section 3.) only a MUA can support a fine-grained collaborative workstyle where both users can interact with the application without extensive context switches.

While simple single-user applications shall be able to function in the Workspace without any knowledge of the multi-user setup, a multi-user aware application has to recognize and support one or all of the following situations:

5.1 Different internal states for each user

Depending on the applications requirements possibly extensive user-context information has to be kept for each user. Many standard operations allow different semantics when extended to multiple users: Shall a cut-and-paste sequence access a shared clipboard or should there be a clipboard for every user? Is there one common or are there many individual command histories?

In these abstract cases both variations make sense and can only be decided on a per-application or per-environment basis. The Workspace supplies all relevant user-specific information to the application program via one of its manager classes (section 7.2) and leaves this policy decisions to the application.

5.2 Multiple input foci on one 3D window

The same window receives at the "same" time input from more than one user. While idempotent operations are possible, in many cases input from different users requests special consideration. An example would be two "dragging" operation at the same time. Our Workspace implementation supplies all MUAs with all necessary information to differentiate between actions executed by different users via the 3D event system (section 7.1). The application is left to decide whether to support a concurrent workstyle or lock the application when one user is already accessing it.

5.3 Multiple users input into same PIP sheet

Since the PIP as our primary input device has to be used in MUAs too, special considerations for its behaviour are necessary. Normally, every application displays all of its 2D interface elements on its own "sheet" on the PIP, meaning a combination of widgets only displayed on a users PIP when the users inputs focus resides within the application. Since a user can acquire an additional focus of a MUA while another user is already working with it - e.g. by selecting an output window of the application - the same PIP sheet will appear on the PIPs of both users.

A MUA has to manage a different state of its associated PIP sheet for every user. Since this state may be visible - e.g. in the position of a slider - it is not sufficient to create multiple renderings of the same sheet on different PIPs, but also to supply each sheets instance with its own state. Again, a Workspace manager exists which allows to distinguish between PIPs on a per-user basis.

6. Hardware Setup

A typical Studierstube per-user setup consists of one semi-transparent headmounted display (HMD), one tracked pen and one tracked PIP (Figure 1). At the moment we are using virtual-IO i-glasses as HMD and an Ascension "Flock of Birds" magnetic tracker. Pens and PIP have been custom made for our setup. Rendering is done on SGI and InterGraph workstations, an additional Linux workstation services the magnetic tracker and distributes the tracker data via a LAN.

6.1 Personal Interaction Panel

The Personal Interaction Panel (PIP)[32] consists of a magnetic tracked pen and clipboard which contains augmented information presented to the user by see- through HMDs. It allows 2D interaction and three dimensional direct manipulation to be done in parallel. Unlike many other interfaces it implements a 2D interface in 3D rather than 2D besides 3D. Using the PIP is similar to using a notebook's flat surface in the real world. Our evaluations have shown that most test persons were familiar with the interface in a very short time and found the two- handed interaction metaphor natural and appealing.

7. Implementation

The Workspace software development environment is realized as a collection of C++ classes which extend the Open Inventor (OIV) Toolkit[31]. The rich graphical environment of OIV allows rapid prototyping of new interaction styles. We also use the file format of OIV to enable convenient scripting of the properties of an application and to include our custom classes. A further advantage of OIV is its availability on IRIX and Windows NT, which allows for some flexibility in the selection of equipment.

OIV represents scenes using scenegraphs - directed acyclic graphs - which can not only contain geometry but also active interaction objects.

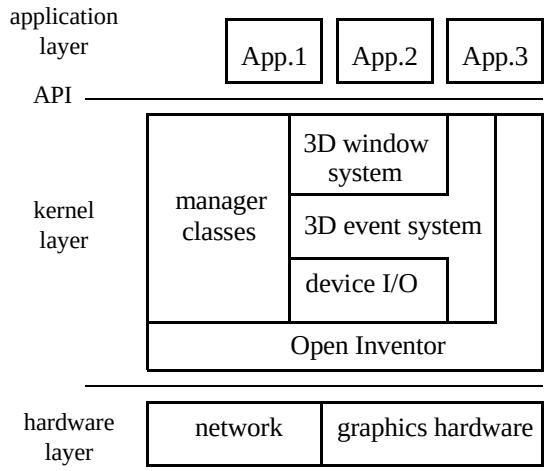

Figure 4: *Studierstube Workspace architecture*

7.1 3D event system

Open Inventor's event system has been extended to process 3D events, which is necessary for choreographing complex 3D interactions. Since OIV does not support 3D interaction with 6DOF but normally projects 2D input into 3D space, we had to implement a new 3D event class, which distributes events containing 6DOF information through the scenegraph using bounding box information. A new class hierarchy of 3D interaction objects allows for the easy integration of active components in a scenegraph.

These objects respond to events generated by a 6DOF input device - in most cases the pen - by altering their appearance, position or internal state. They can filter events depending on user or type or "grab" an input device, redirecting all input to only one object.

This base classes have been subclassed for the PIP to implement standard 2D widget behaviour (buttons, radiobuttons, sliders and dials, see figure 3). Additional features like highlighting and flyover help are available for these widgets.

The more general mechanism for full 6DOF interaction are in most cases implemented by the Studierstube applications. This is done similar to 2D GUIs by opening a 3D window object and attaching a window callback function which receives all events generated in the windows volume (move, drag and click). Some commonly used types of 3D interaction like move and resize of passive objects have been integrated into special scenegraph components, which allow the scripting of simple applications in OIV ASCII files.

7.2 Studierstube manager classes

The *manager* class in Studierstube Workspace give access to high-level interaction concepts. While the basic interaction element classes implement widgets like sliders, buttons or 6DOF draggers and do not depend on any interaction concepts besides the 3D event system (section 7.1), these classes implement the specific Workspace concepts like support for multiple application, multiple windows and multiple users.

7.3 Application manager

The Workspace *job management* allows loading a new application into Workspace and starting and stopping of applications. This function is mostly used by the application loader sheet (section 8.1) but may be used by any application to start a helper application like an object viewer or to stop another application. Starting and stopping applications has to be executed synchronously (between screen updates) with the application, which the application manager achieves this by sending an *EXIT* message via the *message manager* (section 7.6). Upon receiving this message, the application may perform necessary clean up functions and then exit.

7.4 Resource manager

The Workspace *resource management* implements inquiry and setting of Workspace and device attributes as listed in table 1.

resource	attributes
workspace	dimensions, number of users
pen	associated user, geometry
HMD	associated user, geometry, calibration
PIP	associated user, geometry, sheets, active properties: (fishnet, snapshot)

Table 1: *resource attributes*

Some of this attributes, like PIP sheets and pen geometry, are obviously used by most applications, but some of them, like number of users, concern only MUAs. HMD geometry for example is only set when an application wants to attach augmented information to a user and HMD calibration only is accessed by the Workspaces calibration utility.

The ability to attach active components to a PIP sheet as demonstrated in the landscaping application (section 8.6) in the form of a "fishnet" - to select by sweeping (Figure 10)- or to use it as a magic lens can be accessed via this manager.

7.5 Window manager

The Workspace *window management* implements the creation and destruction of window objects and the setting of window attributes.

Callback functions for rendering and event processing and extensions like "drag-and-drop" between windows can be specified via this manager.

window attribute	content
focus	on / off
title bar display	on /off
representation	minimized, 2D, 3D, maximized
cursor	cursor geometry

Table 2: *3D window attributes*

The window manager furthermore manages the states of the displayed windows. The most important attributes of 3D windows are listed in table 2.

The *representation* of a 3D window can be maximized, 3D window, 2D window or minimized.

- *Minimized* windows are only accessible via its application icon in the application loader and consume no space in the working volume.
- *2D* windows are displayed as a flat frame through which the applications geometry can be seen.
- *3D (normal)* window is displayed with a surrounding frame, which allows positioning and resizing via dragging of the edges respective corners.
- *Maximized* windows do not have the frame of 3D windows, effectively consuming the whole display volume. Since only one application window may be displayed in this state, maximizing forces all users to work with the same application.

Displaying an application in a *2D window* reduces space consumption too, but still allows viewing the applications display in realtime. All application output is still rendered in 3D, thereby generating the effect of watching it through a "window in space" or "magic mirror", which can be placed like a framed picture somewhere in the Workspace. This mechanism is very efficient when an applications output has to be watched while working in another window. It is implemented using our SEAM interaction element[25]. Interaction with applications in this state can only be accomplished via the PIP.

7.6 Message manager

The Workspace *application-level message passing* implements a general communication mechanism between studierstube Workspace objects, mainly between applications themselves or between applications and managers (table 3). *System* events implement task management and are generated by the *application manager* (section 7.3). They are routed to special methods of the application. *Window* and *application* messages are generated by the window manager or another application and are passed through the specified receiver windows window-function (section 7.5).

sender	message
application manager	init, exit, idle
window manager	open, close, size, move, drag&drop
application	application-specific

Table 3: *Workspace messages*

8. Results

This section gives a short overview of essential features of our Studierstube Workspace implementation. We want to show how the Workspace concept has been applied in our VE and how standard tasks can be performed. Note that these examples only represent a small aspect of possible situations and policies, and are presented here as demonstration of features discussed before and as proof of concept.

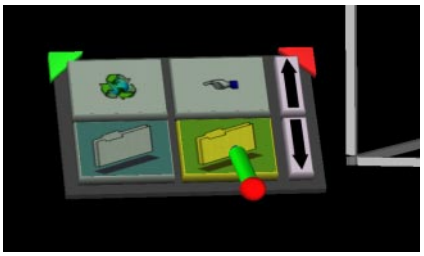

Figure 5: *application sheet on the PIP*

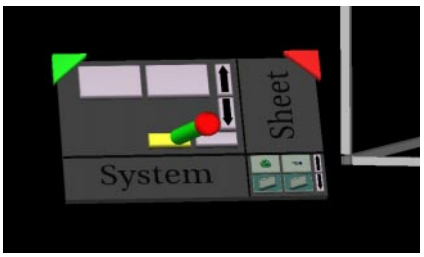

Figure 6: *system sheet on the PIP*

8.1 A sample Workspace session

The user enters the Workspace by putting on the head-mounted display (HMD) and picking up pen and PIP. Normally the PIP is held along an edge using the non-dominant hand. The pen is held like a real pen and allows interaction on the PIP in a familiar pen-and-clipboard manner as well as 3D interaction with 6DOF.

The PIP is the main mechanism for abstract interaction in the Workspace, e.g. input of values and selection of abstract properties as opposed to direct interaction like 3D dragging and pointing. It is not only used as application input device displaying the input elements of running applications but also as primary control device for the Workspace. Directly accessible via short-cut corners (red and green triangles in figures 5 & 6) are two special sheets which always available, the system sheet and the application sheet.

The application sheet - selected via the green corner - functions as a simple shell and enables the user to browse the filesystem and start Studierstube applications or switch focus between running applications. This module has been implemented along the lines of the familiar "file open" dialog from desktop applications. Running applications are shown as icons and allow explicit focus changes (Figure 5). Implicit focus change occurs when clicking into a non-focused window.

The system sheet gives access to 3D window configuration methods. A 3D window can be switched to a maximized, 3D, 2D or minimized representation (section 7.5). Further options like display of title bar or opaque background are accessible too.

8.2 Application examples

The following applications demonstrate the variety of tasks that can be performed within the framework of our Workspace implementation. Large parts of the relevant interface appearance and behaviour have been implemented using the predefined interaction classes, in some cases even only by scripting inside the scene description files.

8.3 Calculator

The simplest application is this simulation of a desktop calculator (figure). It has been written as a test program for 2D interaction elements and demonstrates nicely how the PIP is used for 2D interaction with the “button” interaction elements. This application is frequently used to instruct new Studierstube users on the usage of the PIP. All visible geometry and most of the interaction behaviour has been defined in an Open Inventor scene description file using simple scripting in a text editor.

8.4 X-ray viewer

Almost as simple is this integration of the SEAM[25] interaction element in a medical simulation. The SEAM acts as a magic lens, giving the user the ability to look under the skin of a patient. Positioning of the lens is done by dragging its frame over the patient, allowing a real-time cutaway view of skin and skeleton at the same time.

Since all the positioning is done by a “dragger” interaction element, the application consists only of an initialization of body and skeleton representation and the SEAM with its frame coupled to the dragger.

8.5 3D flow visualization

Here we implemented our geometry and texture based flow visualization technique[14] in Studierstube. Parametrization of the simulation and the visualization was done via the PIP. Additionally SEAMs were used as “magic lenses” and “magic boxes” to navigate inside a sparse representation of the 3D flow and select areas where the flow should be depicted with high detail.

Again all real-time interaction have been defined using the standard interaction elements and only the application specific parts - simulation, visualization and animation of the results - had to be implemented.

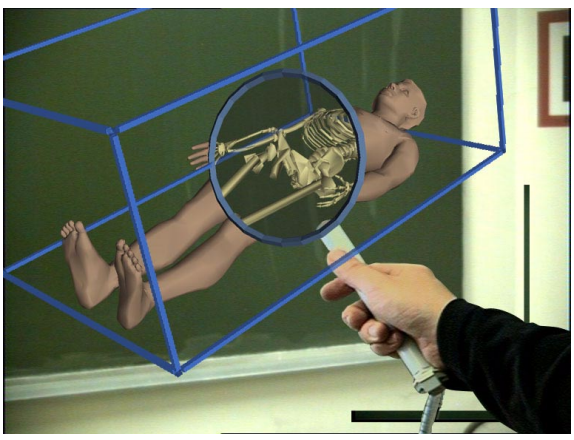

Figure 7: *medical simulation: X-ray lens*

8.6 Landscaping

In our landscaping application[27] we have integrated a variety of interface elements and methods. It is a CAD-like application specifically designed for the development of landscaping solutions in suburban environments. Simple interactions like object placement (houses, trees) are integrated as well as specialized metaphors like a cable TV tool that provides the user with X-ray vision (Figure 9). The user can look under the surface of the landscape representing an island (using wireframe rendering) and uses the pen to lay wire and connect houses to a cable TV network.
Interesting here is the direct application of the PIP as 3D interaction device: Sweeping the PIP like a fishnet through the scene (Figure 10) selects all objects which were “caught” in the motion. Furthermore the PIP can be used as a camera, taking virtual snapshots of different states of the landscape and positioning them like notes somewhere in the Workspace. This snapshot tool allows the user to manage a collection of scenes that are viewed from different perspectives and in different stages of development.

8.7 3D painting application

The painting application (Figure 1) demonstrates how multi-user application function inside Studierstube Workspace. Upon start-up, it generates a single 3D window and PIP sheets for every user inside the Workspace. The sheets contain sliders for colour selection and brush size as well as buttons to select the brush type - spray or pen - and to clean the blackboard.
Every user may select a colour and brush according to his needs, which is displayed as “life size” icon on his PIP.

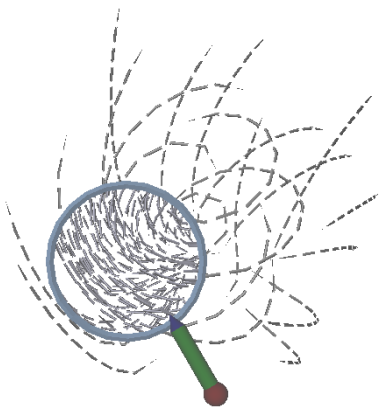

Figure 8: *flow visualization, area inside the lens rendered with high detail*

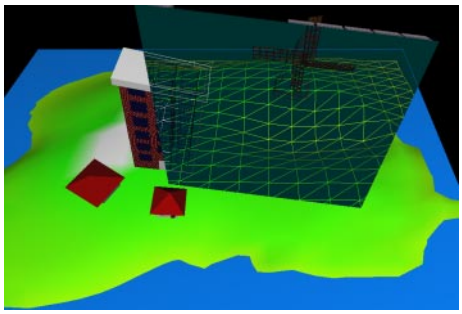

Figure 9: *using the PIP as "magic lens"*

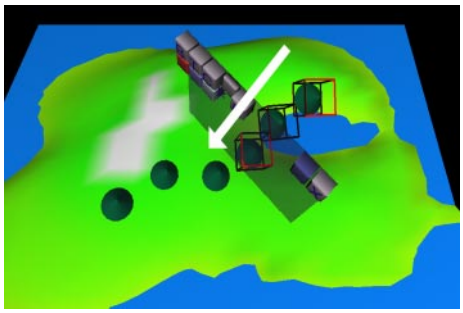

Figure 10: *"fishnet" selecting of objects*

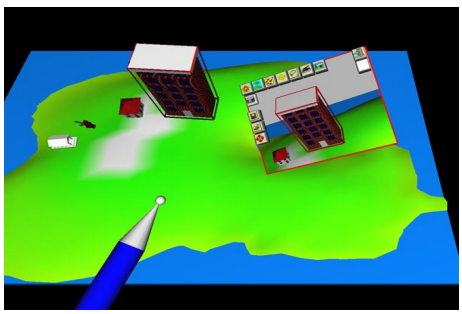

Figure 11: *taking and displaying snapshots with the PIP*

Painting is done by dragging a pen through the window. Depending on the selected brush type, a stream of 3D "dots" - emulating a spray can - is generated. Equipped with this simple interface, users are free to generate whatever three-dimensional pointilistic art they may have in mind.

The possibilities of multi-user collaborations appeared to most users when they inadvertently entered each others paint volume and resulted in impressive, if abstract works as well as in "paint fights".

In this application the advantages of AR could clearly be observed: since users could see each other as well as their creations, collisions of painters or paintings could easily be avoided (or deliberately provoked!) and collaboration on single paintings was enhanced by gestures and discussions.

9. Conclusion

This paper has presented design guidelines and implementation strategies for a collaborative user interface in augmented reality - the Workspace.

It allows multiple users to interact simultaneously with multiple concurrently executing applications. We also introduce the notion of a multi-user aware application (or MUA), which is able to deal with multiple concurrent users at once without the need for monopolization of all system resources.

Studierstube Workspace has proved to stand up against the demands of a wide variety of applications. Many Workspace-specific task like focus changes or application loading have been implemented in a way which enables users to transfer skills acquired in desktop environments into augmented reality.

The application programmer interface - while still in its early development phase - has enabled programmers to quickly integrate their code into the Workspace and supplement it with both 6DOF interaction techniques and conventional 2D graphical user interface elements on the PIP. Since the decisions on a consistent policy regarding multi-user applications can not be done at this early stage, the implementation of MUAs still depends heavily on explicit intervention on the application programmers side. The resulting variations in applications policy regarding user interaction are going to aid us in resolving this aspect for future versions of the Workspace.

10. Future work

An important aspect for a natural collaboration in the Workspace is user migration into and out of the environment. It should be possible for a user to completely leave the environment and for new users to join it. Some applications (especially MUAs) may need to recognize these changes. Not only has the user context to be established or removed, the time when this happens may be crucial. The initial PIP state of a user which joins an application may be considerably different from the one at the start of this application. The necessary message structure is going to be integrated in future releases of the message manager (section 7.6).

User-dependent access rights for visibility and interaction make sense in applications where an inherently asymmetrical relationship exist, as discussed in the case study "educational demonstration". Their implementation via different *layers*[33] is going to be included in one of our next releases.

Further information about this project can be found at:

http://www.cg.tuwien.ac.at/research/vr/studierstube/

Acknowledgements

This work has been supported by the Austrian Science Foundation (FWF) under project no. P-12074-MAT.

Special thanks to Hermann Wurnig and Gerd Hesina for their implementation work and to Michael Gervautz

References

1. M. Agrawala, A. C. Beers, B. Fröhlich, I. McDowall P. Hanrahan, and M. Bolas. The Two-User Responsive Workbench: Support for Collaboration Through Individual Views of a Shared Space. Proceedings SIGGRAPH '97, pages 327-332, 1997.
2. I. Angus and H. Sowizral. Embedding the 2D Interaction Metaphor in a Real 3D Virtual Environment. Proceedings SPIE, volume 2409, pages 282-293, 1995.
3. M. Bajura, H. Fuchs, and R. Ohbuchi. Merging Virtual Objects with the Real World: Seeing Ultrasound Imaginery within the Patient. Proceedings of SIGGRAPH'92, (2):203-210, 1992.
4. M. Billinghurst, S. Weghorst, and T. Furness III. Shared Space: An Augmented Reality Approach for Computer Supported Collaborative Work. Virtual Reality Research Development and Applications, 3(1):25-36, 1997.
5. Rikk Carey, Gavin Bell: The Annotated Vrml 2.0 Reference Manual. Addison-Wesley, 1997.
6. C. Carlsson, O. Hagsand: DIVE- A platform for multiuser virtual environments. Computers & Graphics, Vol. 17, No. 6, pp. 663-669 (1993)
7. 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.
8. C. Cruz-Neira, D. Sandin, and T. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. Proceedings of SIGGRAPH'93, pages 135-142, 1993.
9. J. Dias, R. Galli, A. Almeida, C. Belo, J. Rebordao: mWorld: A Multiuser 3D Virtual Environment. IEEE Computer Graphics and Applications, Vol. 17, No. 2, pp. 55-65, March-April 1997.
10. P. Dykstra: X11 in Virtual Environments: Combining Computer Interaction Methodologies. Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality, pp. 118-119, 1993.
11. S. Feiner, B. MacIntyre, M. Haupt, and Solomon. Windows on the World: 2D Windows for 3D Augmented Reality. Proceedings of UIST'93, pages 145-155, 1993.
12. S. Feiner, B. MacIntyre, D. Seligmann: Knowledge-Based Augmented Reality. Communications of the ACM, Vol. 36, No. 7, pp. 53-62 (1993)
13. A. Fuhrmann, H. Löffelmann, D. Schmalstieg: Collaborative Augmented Reality: Exploring Dynamical Systems. Proc. of IEEE Visualization 1997, pp. 459-462, November 1997.
14. A. Fuhrmann and E. Gröller: Real-Time Techniques for 3D Flow Visualization. Proc. of IEEE Visualization 1998, pp. 305-312, November 1998.
15. M. Gerald-Yamasaki: Cooperative visualization of computational fluid dynamics. Proceedings of Eurographics'93, pp. 497-508, 1993.
16. J. Gettis, P. Carlton, S. McGregor: The X Window System, version 11. Software Practice and Experience, 20(S2), October 1990.
17. S. Gibbs, C. Arapis, C. Breiteneder: TELEPORT - Towards immersive copresence. ACM Multimedia Systems Journal, 1998.
18. W. Krüger, C. Bohn, B. Fröhlich, H. Schüth, W. Strauss, and G. Wesche. The Responsive Workbench: A Virtual Work Environment. IEEE Computer, 28(7):42-48, 1995.
19. J. Kuhl, D. Evans, Y. Papelis, R. Romano, and G. Watson. The Iowa Driving Simulator: An Immersive Research Environment. IEEE Computer, 28(7): pp. 35-42, 1995.
20. V. Lehner, T. DeFanti: Distributed VR: Supporting Remote Collaboration in Vehicle Design. IEEE Computer Graphics and Applications, Vol. 17, No. 2, pp. 13-17, March-April 1997.
21. D. Marca, G. Bock: Groupware: Software for computer-supported cooperative work. IEEE Computer Society Press, 1992.
22. A. Pang, C. Wittenbrink: Collaborative Visualization with CSpray. IEEE Computer Graphics & Applications, 32-41, March-April 1997.
23. R. Pausch, T. Burnette, M. Conway, R. DeLine, R. Gossweiler: Alice: A Rapid Prototyping System For Virtual Reality. UIST'93 (1993)
24. G. Robertson, S. Card, and J. Mackinlay. The Cognitive Coprocessor Architecture for Interactive User Interfaces. Procedings of ACM CHI'89, pages 10-18, 1989.
25. G. Schaufler and D. Schmalstieg. Sewing Worlds Together With SEAMS. Technical Report TR-186-2-98-11, Institute of Computer Graphics 186-2, Technical University of Vienna, Vienna, Austria, August 1998.

26. D. Schmalstieg, A. Fuhrmann, Z. Szalavari, M. Gervautz: Studierstube - An Environment for Collaboration in Augmented Reality Extended abstract appeared in proceedings of Collaborative Virtual Environments '96, Nottingham, UK, Sep. 19-20, 1996. Full paper in: Virtual Reality - Systems, Development and Applications, Vol. 3, No. 1, pp. 37-49, 1998.

27. D. Schmalstieg, M.Encarnação, Zs. Szalavári: Using Transparent Props For Interaction With The Virtual Table. To appear in: Proceedings of SIGGRAPH Symposium on Interactive 3D Graphics '99, Atlanta, GI, April 26-28, 1999.

28. A. Shaw, M. Green, J. Liang, and Y. Sun: Decoupled simulation in virtual reality with the MR toolkit. ACM Transactions on Information Systems, Vol. 11, No. 3, pp. 287-317, 1993.

29. D. Smith, C. Irby, R. Kimbrall, W. Verplank, E. Harslem: Designing the Star user interface. BYTE, pp. 254-258, April 1983.

30. Henry A. Sowizral, Kevin Rushforth, Michael Deering: The Java 3D Specification. Addison-Wesley, 1998.

31. P. Strauss and R. Carey. An Object Oriented 3D Graphics Toolkit. Proceedings of SIGGRAPH'92, (2):341-347, 1992.

32. Zs. Szalavári., M. Gervautz: The Personal Interaction Panel - A Two-handed Interface for Augmented Reality. Proc. EUROGRAPHICS 97, Budapest, Hungary, pp. 335-346, 1997.

33. Zs. Szalavári., E. Eckstein and M. Gervautz: Collaborative Gaming in Augmented Reality. Proc. of VRST'98, Taipei, Taiwan, pp.195-204, November 2-5, 1998.

34. J. Tsao and Ch. J. Lumsden. CRYSTAL: Building Multicontext Virtual Environments. Presence, 6(1):57-72, 1997.

35. J. Wood, H. Wright and K. Brodlie: Collaborative Visualization: Proc. of IEEE Visualization, 253-259, 1997.