The *Studierstube* Augmented Reality Project

Dieter Schmalstieg
dieter@cg.tuwien.ac.at
Vienna University of Technology, Austria

Anton Fuhrmann
VRVis Research Center for Virtual Reality and Visualization, Vienna, Austria*

Gerd Hesina
Vienna University of Technology, Austria

Zsolt Szalavári
Vienna University of Technology, Austria

L. Miguel Encarnação
Fraunhofer CRCG, Inc., Providence, Rhode Island, U.S.

Michael Gervautz
Imagination GmbH, Vienna, Austria*

Werner Purgathofer
Vienna University of Technology, Austria

* Work done while at Vienna University of Technology

Abstract

This paper describes *Studierstube*, an augmented reality system developed over the past four years at Vienna University of Technology, Austria, in extensive collaboration with Fraunhofer CRCG, Inc. in Providence, Rhode Island, U.S. Our starting point for developing the *Studierstube* system was the belief that augmented reality, the less obtrusive cousin of virtual reality, has a better chance of becoming a viable user interface for applications requiring manipulation of complex three-dimensional information as a daily routine. In essence, we are searching for a 3D user interface metaphor as powerful as the desktop metaphor for 2D. At the heart of the *Studierstube* system, collaborative augmented reality is used to embed computer-generated images into the real work environment. In the first part of this paper, we review the user interface of the initial *Studierstube* system, in particular the implementation of collaborative augmented reality, and the Personal Interaction Panel, a two-handed interface for interaction with the system. In the second part, an extended *Studierstube* system based on a heterogeneous distributed architecture is presented. This system allows the user to combine multiple approaches--augmented reality, projection displays, ubiquitous computing--to the interface as needed. The environment is controlled by the Personal Interaction Panel, a two-handed pen-and-pad interface, which has versatile uses for interacting with the virtual environment. *Studierstube* also borrows elements from the desktop, such as multi-tasking and multi-windowing. The resulting software architecture resembles in some ways what could be called an “augmented reality operating system.” The presentation is complemented by selected application examples.

1. Introduction

Studierstube is the German term for the "study room" where Goethe's famous character, Faust, tries to acquire knowledge and enlightenment (Goethe, 1808). We chose this term as the working title for our efforts to develop 3D user interfaces for future work environments. Most virtual reality systems of today are tailored to the needs of a single, very specific application that is highly specialized for that purpose. In contrast, the *Studierstube* project tries to address the question of how to use three-dimensional interactive media in a general work environment, where a variety of tasks are carried out simultaneously. In essence, we are searching for a 3D user interface metaphor as powerful as the desktop metaphor for 2D.

Our starting point for developing *Studierstube* was the belief that augmented reality (AR), the less obtrusive cousin of virtual reality (VR), has a better chance than VR of becoming a viable user interface for applications requiring information manipulation as a daily routine. Today's information workers are required to carry out a large variety of tasks, but communication between human co-workers has an equally significant role. Consequently, *Studierstube* tries to support productivity, typically associated with the desktop metaphor, as well as collaboration, typically associated with computer supported cooperative work applications. To fulfill these needs, the framework therefore has taken on many functions of a conventional operating system in addition to being a graphical application.

At the heart of the *Studierstube* system, collaborative AR is used to embed computer-generated images into the real work environment. AR uses display technologies such as see-through head-mounted displays (HMDs) or projection screens to combine computer graphics with a user's view of the real world. By allowing multiple users to share the same virtual environment, computer supported cooperative work in three dimensions is enabled.

This paper gives an overview of the various avenues of research that were investigated in the course of the last four years, and how they relate to each other. The intent of this paper is to provide a summary of this rather extensive project as well as an introduction to the approach of blending augmented reality with elements from other user interface paradigms to create a new design for a convincing 3D work environment. In the first part of this paper, we review the core user interface technologies of the initial *Studierstube* work, in particular the implementation of collaborative augmented reality, and the Personal Interaction Panel, a two handed-interface for interaction with the system.

In the second part, we present an extended collaborative 3D interface that unites aspects of multiple user interface paradigms: augmented reality, ubiquitous computing, and the desktop metaphor. In the third part, we illustrate our work by reviewing some selected experimental applications that were built using *Studierstube*. Finally, we discuss how *Studierstube* is related to previous work, and draw conclusions.

2. Interaction in augmented reality

The initial *Studierstube* system as described in (Schmalstieg et al., 1996) and (Szalavári et al., 1998a) was among the first collaborative augmented reality systems to allow multiple users to gather in a room and experience a shared virtual space that can be populated with three-dimensional data. Head-tracked HMDs allow each user to choose an individual viewpoint while retaining full stereoscopic graphics. This is achieved by rendering the same virtual scene for every user's viewpoint (or more precisely, for every user's eyes), while taking the users' tracked head positions into account.

Collaborators may have different preferences concerning the chosen visual representation of the data, or they may be interested in different aspects. It is also possible to render customized views of the virtual scene for every user that differ in aspects other than the viewpoint (for example, individual highlighting or annotations). At the same time, co-presence of users in the same room allows natural interaction (talking, gesturing etc.) during a discussion. The combination of real world experience with the visualization of virtual scenes yields a powerful tool for collaboration (Figure 1).

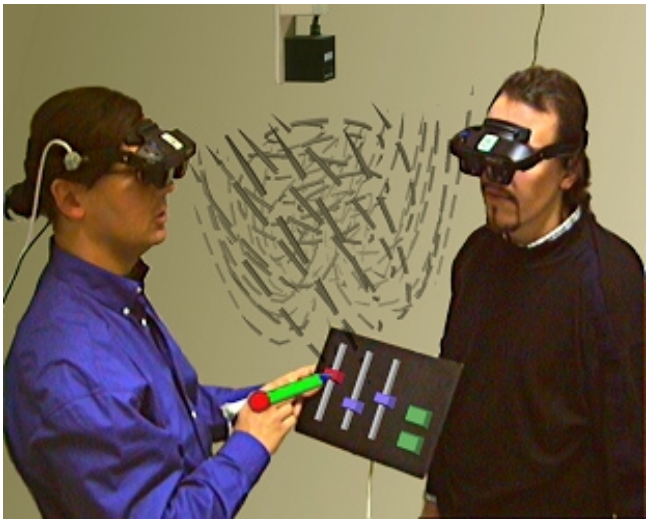

Figure 1: Two collaborators wearing see-through displays are examining a flow visualization data set

2.1 The Personal Interaction Panel

The Personal Interaction Panel (PIP) is a two-handed interface used to control *Studierstube* applications (Szalavári & Gervautz, 1997). It is composed of two lightweight hand-held props, a pen and a panel, both equipped with magnetic trackers. Via the see-through HMD, the props are augmented with computer generated images, thus instantly turning them into application-defined interaction tools similar in spirit to the virtual tricorder of Wloka & Greenfield (1995), only using two hands rather than one. The pen and panel are the primary interaction devices.

The props' familiar shapes, the fact that a user can still see his or her own hands, and the passive tactile feedback experienced when the pen touches the panel make the device convenient and easy to use. Proprioception (Mine et al., 1997) is readily exploited by the fact that users quickly learn how to handle the props and can remember their positions and shapes. A further advantage is that users rarely complain about fatigue as they can easily lower their arms and look down on the props.

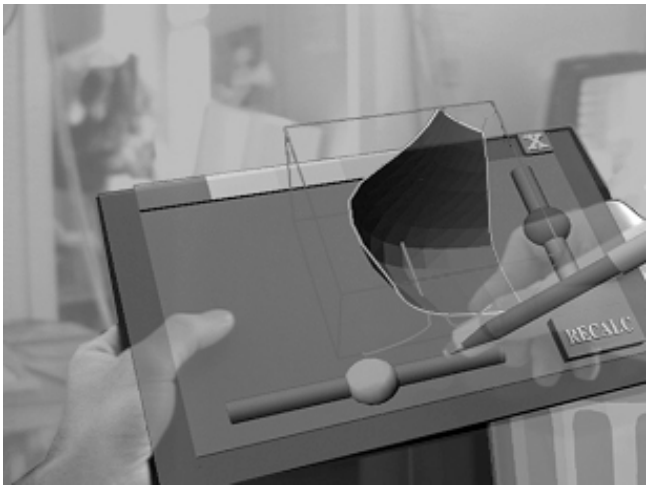

Figure 2: The Personal Interaction Panel allows two-handed interaction with 2D and 3D widgets in augmented reality

The asymmetric two-handed interaction exploits Guiard's observations (1987) that humans often use the non-dominant hand (holding the panel) to provide a frame of reference for the fine-grained manipulations carried out with the dominant hand (holding the pen). Many of the interaction styles we have designed take advantage of this fact.

However, the panel not only provides a frame of reference, but also a natural embedding of 2D in 3D (Figure 2). Many of the artifacts we encounter in real life, such as TV remote controls or button panels on household items such as microwave ovens, are essentially two-

dimensional. The PIP approach with its tactile feedback on the panel's surface resembles those real world artifacts better than naïve VR approaches such as flying menus. Consequently, the PIP provides a way to transpose many useful widgets and interaction styles from the desktop metaphor into augmented reality. Such "2.5D" widgets such as buttons, sliders or dials provide the bread-and-butter of interaction.

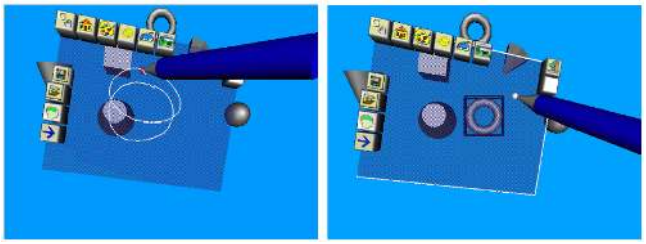

Figure 3: A gesture is used to create a torus in CADesk

However, the PIP's direct and expressive interaction language has much more to offer:

- **Object manipulation**: The pen is used as a six-degree-of-freedom pointer for object manipulation in three dimensions. Objects can either be manipulated directly in the virtual space, on the panel, or in any combination of the two. A user can instantly establish such combinations by overlaying the fixed-world frame of reference with the frame of reference defined by the panel, for example, by dragging and dropping objects from a palette to the virtual scene.
- **Gestural interaction**: Perhaps the most fundamental function of a pen and panel is gesturing, i.e., writing and drawing. As noted by (Poupyrev et al., 1998), using the panel as a surface for the gestures is an efficient mode of input in virtual environments, and even more so in AR where a user can see his or her hands while gesturing. Delimiting the area for gestures on the panel's surface allows simultaneous symbolic input and direct object manipulation. Figure 3 shows CADesk (Encarnação et al., 1999a), a solid modeling tool that has been enhanced with gesture-based interaction using the *Studierstube* framework (Encarnação et al., 1999b).

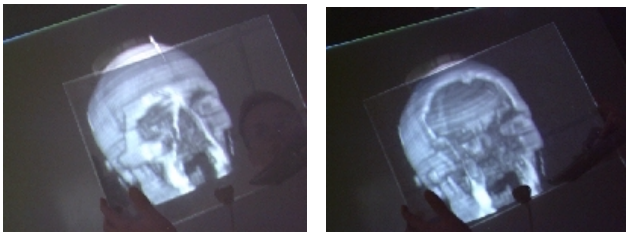

Figure 4: The panel is used to position a clipping plane that cuts away a portion from the volumetric scan of a human skull

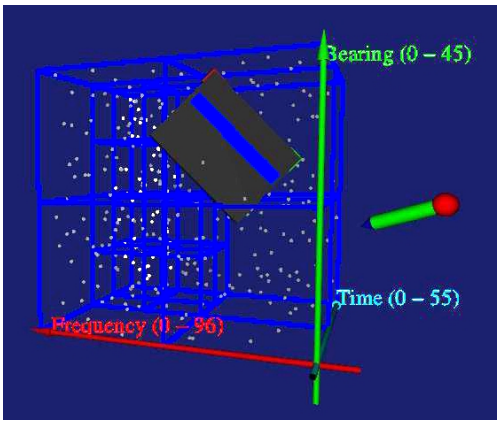

Figure 5: The panel is swept through an aggregation of particle data. During the sweep, a filter is applied to the underlying raw data, which produces aural feedback that can assist the user in detecting structures in the data sets that are not visible to the human eye.

- **Surface tool**: The panel, a two-dimensional physical shape that extends in three-dimensional space, can be interpreted as a hand-held plane or planar artifact. It can be used as a screen showing still images, animations, flat user interfaces (compare Angus & Sowizral, 1995), or live images taken from the real or virtual environment (like the screen of a digital camcorder). For example, in the MediDesk application (Wohlfahrter et al., 2000), the panel can be used to slice a volumetric model to obtain "X-ray plates" (Figure 4). Map-type tools such as worlds-in-miniature (Pausch et al., 1995) can use the panel as a ground plane. The panel

can also be used to apply filters to the data samples penetrated when sweeping the panel through a data set (Encarnação et al., 2000). Such filters can produce new visual representations of the underlying data sets or other kinds of feedback, such as sonification (Figure 5).

2.2 Privacy in Augmented Reality

The *personal* in Personal Interaction Panel was chosen to emphasize how its use allows users to leverage the advantages of collaborative augmented reality: Holding and manipulating the PIP puts a user in control of the application. If only one PIP is used, contention for control is resolved using social protocols such as passing on the PIP. In contrast, giving each user a separate PIP allows concurrent work. Although using multiple PIPs requires the system software to resolve the resulting consistency issues, users can freely interact with one or multiple data sets, because every user gets a separate set of controls on his or her PIP. Fuhrmann & Schmalstieg (1999) describe how interface elements can, but need not be shared by users or application instances.

The concept of personal interaction in collaborative environments is tied to the issue of privacy – users do not necessarily desire all their data to be public (Butz et al., 1998). Fortunately, a display architecture that supports independent per-user displays such as ours can be configured to use subjective views (Smith & Mariani, 1997) with per-user variations to a common scene graph. One user may display additional information that is not visible for the user's collaborators, for example if the additional information is confusing or distracting for other users, or if privacy is desired (consider highlighting or private annotations). We found the PIP to be a natural tool for guarding such private information: For privacy, a user can make information on the panel invisible to others. This idea was explored in (Szalavári et al., 1998b) for collaborative games to prevent users from cheating (Figure 6).

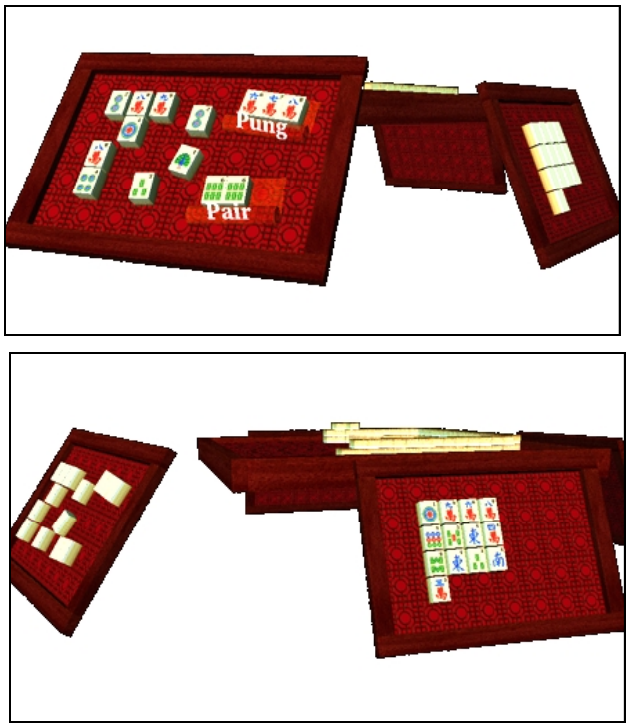

Figure 6: Personal displays secure privacy when playing Mahjongg – the left player (top view) cannot see his opponent's tile labels and vice versa (bottom view)

2.3 Augmented Reality for the Virtual Table platform

Normally, AR is associated with see-through or video-based HMDs. Unlike HMDs, large stereo back-projection screens viewed with shutter glasses, such as used in CAVE (Cruz-Neira et al., 1993), wall, or workbench (Krüger et al., 1995) setups, offer significantly better viewing quality, but cannot produce augmentation, as opaque physical objects will always occlude the back projection[1]. To overcome this restriction, we developed a setup that achieves a kind of inverse augmented reality,

[1] Note that this discussion does not consider front projection, which is capable of producing so-called spatially augmented reality, but suffers from a different set of technical complexities.

or *augmented VR*, for the Virtual Table (VT), a workbench-like device, through the use of transparent pen and panel props made from Plexiglas (Schmalstieg et al., 1999).

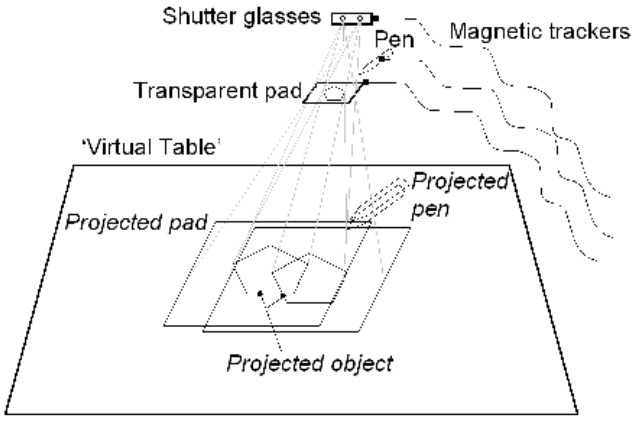

Figure 7: The Personal Interaction Panel combines tactile feedback from physical props with overlaid graphics to form a two-handed general-purpose interaction tool for the Virtual Table.

Using the information from the trackers mounted to shutter glasses and props, the workstation computes stereoscopic off-axis projection images that are perspectively correct for the user's head position. This property is essential for the use of AR as well as augmented VR, since the physical props and their virtual counterparts have to appear aligned in 3D (Figure 7). Additional users with shutter glasses can share the view with the leading user, but they experience some level of perspective distortion. Also the virtual panel will not coincide with its physical counterpart.

The material for the pen and pad was selected for minimal reflectivity, so that with dimmed lights – the usual setup for working with the VT – the props become almost invisible. While they retain their tactile property, in the user's perception they are replaced by the graphics from the VT (Figure 8).

Our observations and informal user studies indicate that virtual objects can even appear floating above the Plexiglas surface, and that conflicting depth cues resulting from such scenarios are not perceived as disturbing. Minor conflicts occur only if virtual objects protrude from the outline of the prop as seen by the user because of the depth discontinuity. The most severe problem is occlusion from the user's hands. Graphical elements on the pad are placed in a way so that such occlusions are minimized, but they can never be completely avoided.

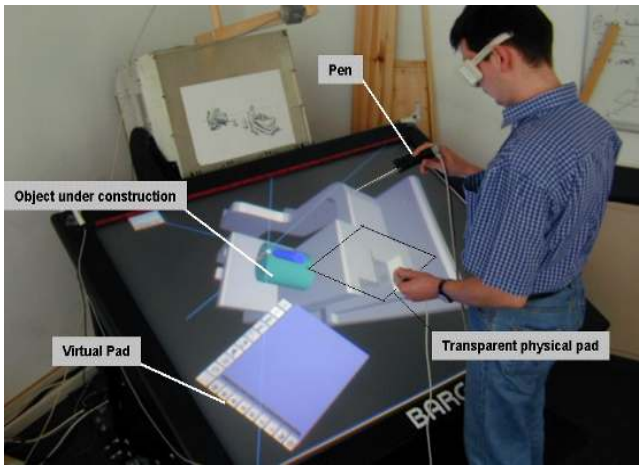

Figure 8: Transparent pen and pad for the Virtual Table are almost invisible and replaced by computer graphics in the user's perception (Stork & de Amicis, 2000)

Using the transparent props, the *Studierstube* software was ported to the VT platform. Applications could now be authored once and displayed on different platforms. One lesson we learned in the process was that the format and properties of the display strongly influence application design, much like a movie converted from Cinemascope to TV must be edited for content.

It was only after a working prototype of the VT setup was finished that we realized that a *transparent* panel affords new interaction styles because the user can see *through* it:

- **Through-the-plane tools**: The panel is interpreted as a two-dimensional frame defining a frustum-shaped volume. A single object or set of objects contained in that volume instantly becomes subject to further

manipulation – either by offering context sensitive tools such as widgets placed at the panel's border, or by 2D gestural interaction on the panel's surface. For example, Figure 9 shows the application of a „lasso" tool for object selection.

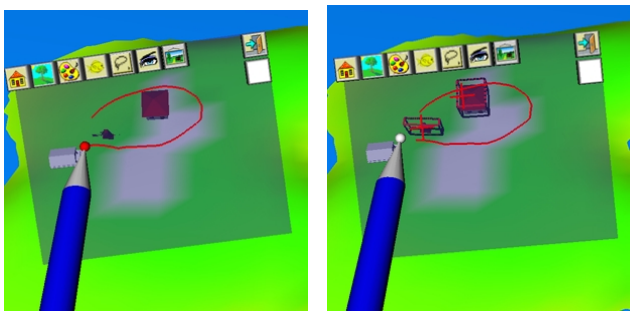

Figure 9: The lasso tool allows users to select objects in 3D by sweeping an outline in 2D on the pad. All objects whose 2D projection from the current viewpoint is contained in the outline are selected.

- **Through-the-window tools**: The transparent panel is interpreted as a window into a different or modified virtual environment. This idea includes 3D *magic lenses* (Viega et al., 1996) such as X-ray lenses (Figure 10), that are essentially modified versions of the main scene, but also SEAMS (Schmalstieg & Schaufler, 1998), which are portals to different scenes or different portions of the same scene. A recent extension to the window tools is proposed in (Stoev et al., 2000): The panel acts as a lens into a separate locale of the virtual environment, the pen is used to move the scene underneath.

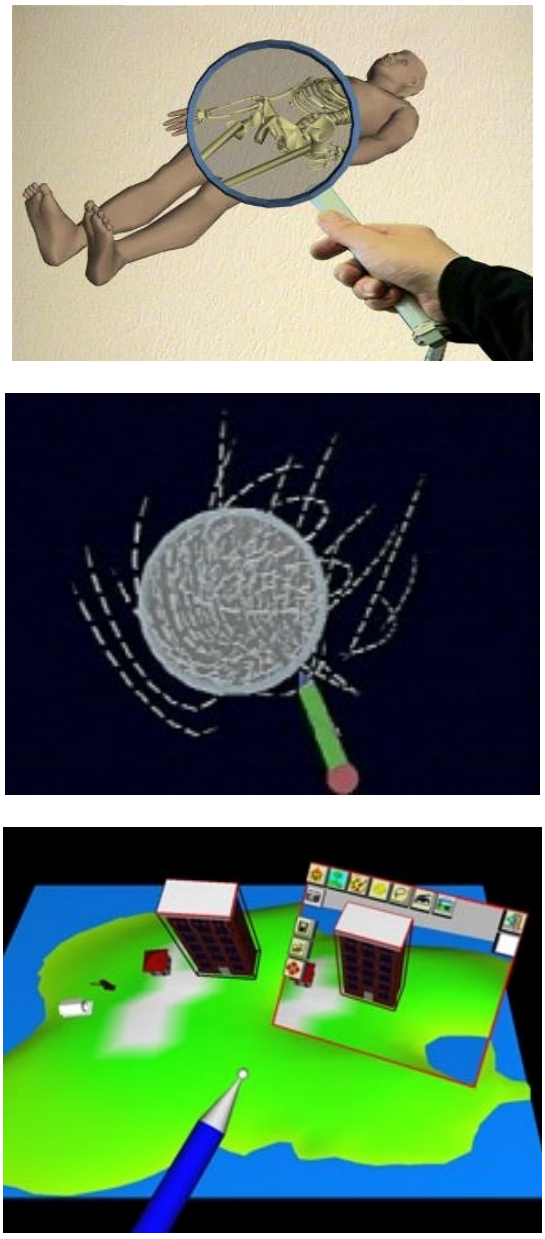

Figure 10: Different applications of through-the-window tools: (top) X-ray lens, (middle) focus lens that locally increases density of streamlines in a flow visualization, (bottom) portal to a different version of a scene

3. Convergence of user interface metaphors

During the work on the original *Studierstube* architecture, we rapidly discovered new

promising avenues of research, which could not be investigated using the initial limited design. From about 1998 on, we therefore concentrated our efforts at re-engineering and extending the initial solutions to construct a second-generation platform building on what we had learned. The support for the VT platform, as detailed in the last section, was the first outcome of this work.

It gradually became clear that augmented reality – even in a collaborative flavor – was not sufficient to address all the user interface requirements for the next generation 3D work environment we had in mind. We needed to mix and match elements from different user interface metaphors. A vision of converging different user interface paradigms evolved (Figure 11). In particular, we wanted to converge AR with elements from *ubiquitious computing* and the *desktop metaphor*.

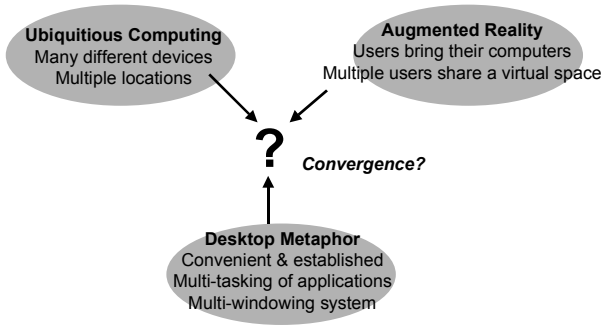

Figure 11: The latest *Studierstube* platform combines the best elements from augmented reality, ubiquitous computing, and the desktop metaphor

In contrast to AR, which is characterized by users carrying computing and display tools to augment their environment, ubiquitous computing (Weiser, 1990) denotes the idea of embedding many commodity computing devices into the environment, thus making continuous access to networked resources a reality. The VT platform, although hardly a commodity, is an instance of such a situated device. Yet there are other devices such as personal digital assistants (PDAs) that blur the boundaries between AR and ubiquitous computing. We are interested in exploring possible combinations of a multitude of simultaneously or alternatively employed displays, input, and computing infrastructures.

While new paradigms such as AR and ubiquitous computing enable radical redesign of human-computer interaction, it is also very useful to transpose knowledge from established paradigms, in particular from the desktop, into new interaction environments. Two-dimentional widgets are not the only element of the desktop metaphor that we consider useful in a 3D work environment. Desktop users have long grown accustomed to multi-tasking of applications that complement each other in function. In contrast, many VR software toolkits allow the development of multiple applications for the same execution environment using an abstract application programmer's interface (API); however, the execution environment usually cannot run multiple applications concurrently. Another convenient feature of desktop applications is that many of them support a multiple document interface (MDI), i.e. working with multiple documents or data sets simultaneously, allowing comparison and exchange of data among documents. The use of 2D windows associated with documents allows convenient arrangement of multiple documents according to a user's preferences. While these properties are established in the desktop world, they are not exclusive to it and indeed useful to enhance productivity in a 3D work environment as well.

The latest version of the *Studierstube* software framework explores how to transpose these properties into a virtual environment (Schmalstieg et al., 2000). The design is built on three key elements: users, contexts, and locales.

3.1 Users

Support for multiple collaborating users is a fundamental property of the *Studierstube* architecture. While we are most interested in computer-supported face-to-face collaboration, this definition also encompasses remote collaboration. Collaboration of multiple users implies that the system will typically incorporate multiple host computers – one per user. However, *Studierstube* also allows multiple users to interact with a single host (e.g. via a large screen or a multi-headed display), and a single user to interact with multiple computers at once (by simultaneous use of multiple displays). This design is realized as a distributed system composed of different computing, input (PIP) and output (display) devices that can be operated simultaneously.

3.2 Contexts

The building blocks for organizing information in *Studierstube* are called *contexts*. A context encloses the data itself, the data's representation and an application that operates on the data. It therefore roughly corresponds to an object-oriented implementation of a document in a conventional desktop system. Users only interact within those contexts, so the notion of an application is completely hidden from the user. In particular, users never have to "start" an application; they simply open a context of a specific type. Conceptually, applications are always "on" (Kato et al., 2000).

In a desktop system, the data representation of a document is typically a single 2D window. Analogously, in our three-dimensional user interface, a context's representation is defined as a three-dimensional structure contained in a box-shaped volume – a 3D-window (Figure 12). Note that unlike its 2D counterpart, a context can be shared by any group of users.

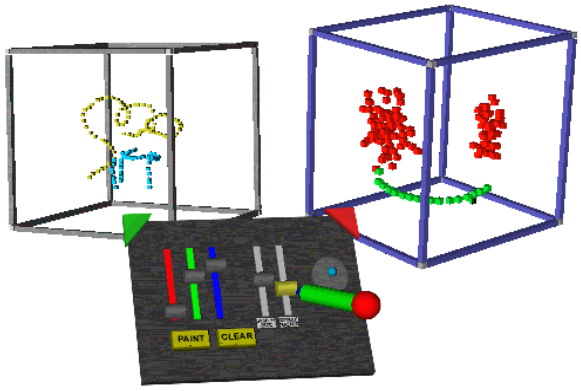

Figure 12: Multiple document interface in 3D – the right window has the user's focus – indicated by the dark window frame – and can be manipulated with the control elements on the PIP.

Every context is an instance of a particular application type. Contexts of different types can exist concurrently, which results in multi-tasking of multiple applications. Moreover, *Studierstube* also allows multiple contexts of the same type, thereby implementing an MDI. Multiple contexts of the same type are aware of each other and can share features and data. For example, consider the miniature stages of the Storyboarding application (section 8), which share the "slide sorter" view.

3.3 Locales

Locales correspond to coordinate systems in the virtual environment. They usually coincide with physical places, such as a lab or conference room or part of a room, but they can also be portable and linked to a user's position or used arbitrarily—even overlapping locales in the same physical space are allowed and used. By convention, every display used in a *Studierstube* environment shows the content of exactly one locale, but one locale can be assigned to multiple displays. Every context can—but need not—be replicated in every locale, i.e. it can appear, at most, once in every locale. All replicas of a particular context are kept synchronized by

Studierstube's distribution mechanism (section 6).

3.4 Context vs. locale

At first glance, it may not be obvious why a separation of contexts and locales is necessary. For example, the EMMIE system (Butz et al., 1999) envelops users and computers in a single environment called "ether," which is populated by graphical data items. An item's locale also defines its context and vice versa. All displays share the same physical locale. While this approach is simple to understand and easy to implement, the interaction design does not scale well with the number of data items and users: As the number of data items increases, it becomes increasingly difficult to arrange them so that all users have convenient access to all data items that they are interested in. Data items may be occluded or out of reach for convenient interaction. Even a fully untethered setup of displays and devices may be inconvenient if the environment is structured in a way that forces users to walk around in order to access frequently required data. The larger the user group is, the more likely it becomes that two users that are not in close proximity will compete for a particular data item, making optimal placement difficult or impossible. Moreover, remote collaboration is ruled out by the single locale approach, as the position of a particular data item will often be inaccessible to a remote user.

In contrast, *Studierstube* separates contexts and locales for increased flexibility. Every display uses a separate locale, i.e., a scene with an independent coordinate system. A context is placed in a locale by assigning to the context's 3D-windows a particular position within the locale. This approach allows for several strategies regarding the arrangement of contexts in the relevant locales.

A strategy of making a context available exclusively in one locale is equivalent to the single locale approach, with the exception that the locale is broken up into disjointed parts. Again, users may not be able to access desired contexts (Figure 13, top). In contrast, a strategy of replicating every context in every locale guarantees convenient access to a context, but quickly leads to display clutter (Figure 13, middle).

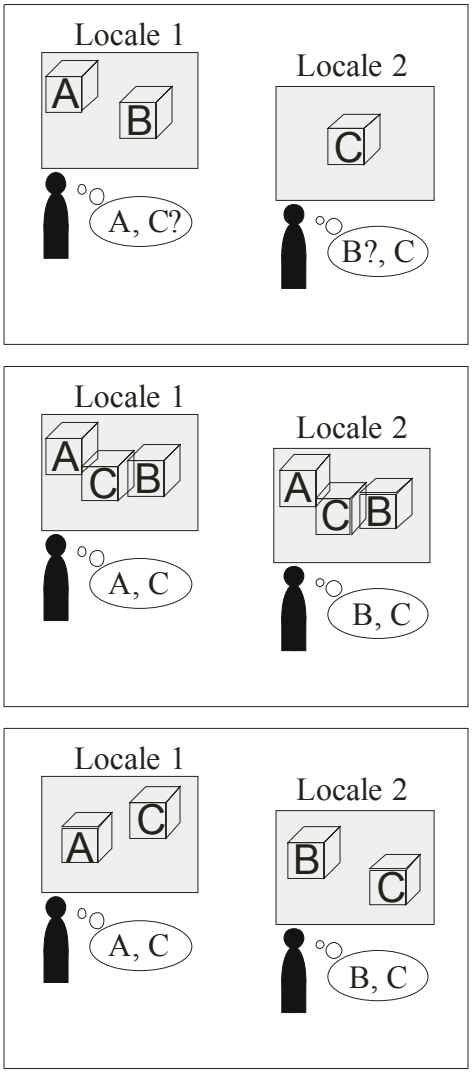

Figure 13: (top) A global arrangement of items cannot fulfill all needs. (middle) Full replication of all items leads to display clutter. (bottom) On-demand replication of items allows convenient customization of locales.

Therefore replication of a context in a given locale is optional: There may be at most one replica of a given context in a given locale. This strategy allows a user to arrange a convenient working set of contexts in his or her preferred display (Figure 13, bottom). If the displays are connected to separate hosts in a distributed system, only those hosts that replicate a context need to synchronize the context's data. If it can be assumed that working sets typically do not exceed a particular size, the system will scale well.

Yet in many situations it is desirable to share position and configuration over display boundaries. *Studierstube* thus allows locales to be shared over displays. More precisely, multiple displays can have independent points of view, but show images of an identical scene graph.

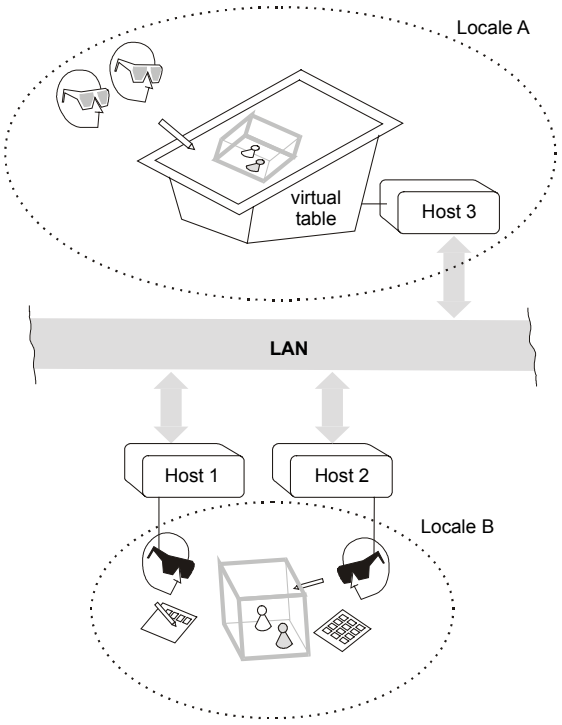

Figure 14: Multiple locales can simultaneously exist in *Studierstube*. They can be used to configure different output devices and to support remote collaboration.

This allows for collaborative augmented reality settings as introduced in section 2, but even for more complex setups such as a large projection screen display augmented by graphics from a see-through HMD. Figure 14 shows a non-trivial example involving one context, two locales, three displays, and four users.

4. Implementation of the user interface

4.1 Software architecture

Studierstube's software development environment is realized as a collection of C++ classes built on top of the Open Inventor (OIV) toolkit (Strauss & Carey, 1992). The rich graphical environment of OIV allows rapid prototyping of new interaction styles. The file format of OIV enables convenient scripting, overcoming many of the shortcomings of compiled languages without compromising performance. At the core of OIV is an object-oriented scene graph storing both geometric information and active interaction objects. Our implementation approach has been to extend OIV as needed, while staying within OIV's strong design philosophy (Wernecke, 1994).

This has led to the development of two intertwined components: A toolkit of extensions of the OIV class hierarchy—mostly interaction widgets capable of responding to 3D events—and a runtime framework which provides the necessary environment for *Studierstube* applications to execute (Figure 15). Together these components form a well-defined application programmer's interface (API), which extends the OIV API, and also offers a convenient programming model to the application programmer (section 7).

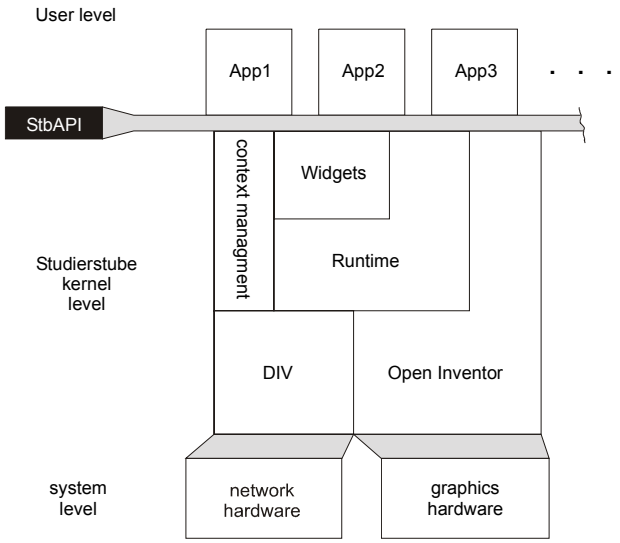

Figure 15: The *Studierstube* software is composed of an interaction toolkit and runtime system. The latter is responsible for managing context and distribution.

Applications are written and compiled as separate shared objects, and dynamically loaded into the runtime framework. A safeguard mechanism makes sure that only one instance of each application's code is loaded into the system at any time. Besides decoupling application development from system development, dynamic loading of objects also simplifies distribution, as application components can be loaded by each host whenever needed. All these features are not unique to *Studierstube*, but they are rarely found in virtual environment software.

By using this dynamic loading mechanism, *Studierstube* supports multi-tasking of *different* applications (e.g. a medical visualization and a 3D modeler) and also an MDI.

Depending on the semantics of the associated application, ownership of a context may or may not privilege a user to perform certain operations on the information (such as object deletion). Per default, users present in the same locale will share a context. Per default, a context is visible to all users and can be manipulated by any user in the locale.

4.2 Three-dimentional windows

The use of windows as an abstraction and interaction metaphor is an established convention in 2D GUIs. Its extension to three dimensions can be achieved in a straightforward manner (Tsao & Lumsden, 1997): 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 virtual environment. It supplies the user with the same means of positioning and resizing the display volume and also defines its exact boundaries.

A context is normally represented in the scene by a 3D window, although a context is allowed to span multiple windows. The 3D-window class is a container associated with a user-specified scene graph. This scene graph is normally rendered with clipping planes set to the faces of the containing box so that the content of the window does not protrude from the window's volume. Nested windows are possible, although we have found little use for them. The window is normally rendered with an associated "decoration" that visually defines the window's boundaries and allows it to be manipulated with the pen (move, resize etc). The color of the decoration also indicates whether a window is active (and hence receives 3D events from that user). Like their 2D counterparts, 3D-windows can be minimized (replaced by a three-dimensional icon on the PIP to save space in a cluttered display), and maximized (scaled to fill the whole work area). Typically, multiple contexts of the same type will maintain structurally similar windows, but this decision is at the discretion of the application programmer.

4.3 PIP sheets

Studierstube applications are controlled either via direct manipulation of the data presented in 3D-windows, or via a mixture of 2D and 3D widgets on the PIP. A set of controls on the PIP— a *PIP sheet*—is implemented as an

OIV scene graph composed primarily of *Studierstube* interaction widgets (such as buttons, etc.). However, the scene graph may also contain geometries (e. g., 2D and 3D icons) that convey the user interface state or can be used merely as decoration.

Every type of context defines a PIP sheet template, a kind of application resource. For every context and user, a separate PIP sheet is instantiated. Each interaction widget on the PIP sheet can therefore have a separate state. For example, the current paint color in an artistic spraying application can be set individually by every user for every context. However, widgets can also be shared by all users and/or all contexts. Consequently, *Studierstube's* 3D event routing involves a kind of multiplexer between windows and users' PIP sheets.

5. Hardware support

5.1 Displays

Studierstube is intended as an application framework that allows the use of a variety of displays, including projection based devices and HMDs. There are several ways of determining camera position, creating stereo images, setting a video mode etc. After some consideration, we implemented an OIV compatible viewer with a plug-in architecture for camera control and display mode.

The following display modes are supported:

- Field sequential stereo: Images for left/right eye output in consecutive frames
- Line interleaved stereo: Images for left/right eye occupy odd/even lines in a single frame
- Dual screen: Images for left/right eye are output on two different channels
- Mono: The same image is presented to both eyes

The following camera control modes are supported:

- Tracked display: Viewpoint and display surface are moving together and are tracked (usually HMD)
- Tracker head: A user's viewpoint (head) is tracked, but the display surface is fixed (such as a workbench or wall)
- Desktop: The viewpoint is either assumed stationary, or can be manipulated with a mouse

This approach, together with a general off-axis camera implementation, allows runtime configuration of almost any available display hardware. Table 1 shows an overview of some devices that have evaluated so far.

	Tracked display	**Tracked head**	**Desktop**
Field sequential	Sony Glasstron	Virtual Table	Fishtank VR with shutter glasses
Line interleaved	i-glasses	VREX VR2210 projector	i-glasses w/o head tracking
Dual screen	i-glasses Protec	Single user dual-projector passive stereo w/head track.	Multi-user dual-projector passive stereo
Mono	i-glasses (mono)	Virtual Table (mono)	Desktop viewer

Table 1: All combinations of camera control and display modes have distinct uses.

5.2 Tracking

A software system like *Studierstube* that works in a heterogeneous distributed infrastructure and is used in several research labs with a variety of tracking devices requires an abstract tracking interface. The approach taken by most commercial software toolkits is to implement a device driver model, thereby providing an abstract interface to the tracking devices, while hiding hardware dependent code inside the supplied device drivers. While such a model is certainly superior to hard-coded device

support, we found it insufficient for our needs in various aspects:

- **Configurability**: Typical setups for tracking in virtual environments are very similar in the basic components, but differ in essential details such as the placement of tracker sources or the number and arrangement of sensors. The architecture allows the configuration of all of those parameters through simple scripting mechanisms.
- **Filtering**: There are many necessary configuration options that can be characterized as filters, i.e., modifications of the original data. Examples include geometric transformations of filter data, prediction, distortion compensation, and sensor fusion from different sources.
- **Distributed execution and decoupled simulation**: Processing of tracker data can become computationally intensive, and it should therefore be possible to distribute this work over multiple CPUs. Moreover, tracker data should be simultaneously available to multiple users in a network. This can be achieved by implementing the tracking system as a loose ensemble of communicating processes, some running as service processes on dedicated hosts that share the computational load and distribute the available data via unicast and multicast mechanisms, thereby implementing a decoupled simulation scheme (Shaw et al., 1993).
- **Extensibility**: As a research system, *Studierstube* is frequently extended with new experimental features. A modular, object-oriented architecture allows the rapid development of new features and uses them together with existing ones.

The latest version of tracking support in *Studierstube* is implemented as an object-oriented framework called OpenTracker (Reitmayr & Schmalstieg, 2000), which is available as open source. It is based on a graph structure composed of linked nodes: source nodes deliver tracker data, sink nodes consume data for further processing (e. g. to set a viewpoint), while intermediate nodes act as filters. By adding new types of nodes, the system can easily be extended. Nodes can reside on different hosts and propagate data over a network for decoupled simulation. By using an XML (Bray et al., 2000) description of the graph, standard XML tools can be applied to author, compile, document, and script the OpenTracker architecture.

6. Distributed execution

The distribution of *Studierstube* requires that for each replica of a context, all graphical and application-specific data is locally available. In general, applications written with OIV encode all relevant information in the scene graph, so replicating the scene graph at each participating host already solves most of the problem.

6.1 Distributed shared scene graph

Toward that aim, Distributed Open Inventor (DIV) was developed (Hesina et al., 1999) as an extension—more a kind of plug-in—to OIV. The DIV toolkit extends OIV with the concept of a distributed shared scene graph, similar to distributed shared memory. From the application programmer's perspective, multiple workstations share a common scene graph. Any operation applied to a part of the shared scene graph will be reflected by the other participating hosts. All this happens to the application programmer in an almost completely transparent manner by capturing and distributing OIV's notification events.

Modifications to a scene graph can either be updates of a node's fields, i.e., attribute values, or changes to the graph's topology, such as adding or removing children. All these changes to the scene graph are picked up by an OIV sensor and reported to a DIV observer which propagates the

changes via the network to all hosts that have a replica of the context's scene graph, where the modifications are duplicated on the remote scene graph by a DIV listener (Figure 16).

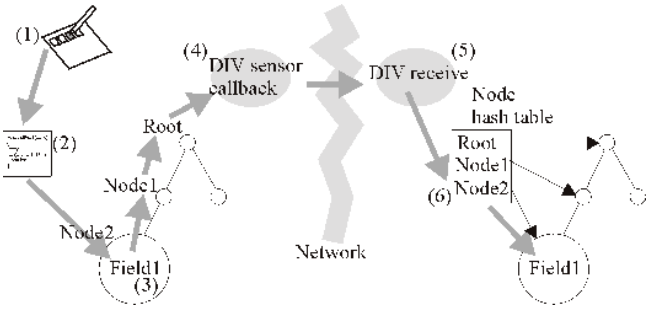

Figure 16: Example of a field update in a master-slave configuration. (1) User triggers an action by pressing a button. (2) Corresponding callback is executed and modified field1 of node2. (3) Event notification is propagated upwards in scene graph and observed by sensor. (4) Sensor transmits message to slave host. (5) Receiver picks up message and looks up corresponding node in internal hash table. (6) Slave node is modified.

On top of this master/slave mechanism for replication, several network topology schemes can be built. A simple reliable multicasting scheme based on time stamps is used to achieve consistency.

6.2 Distributed context management

A scene graph shared with DIV need not be replicated in full—only some portions can be shared, allowing local variations. In particular, every host will build its own scene graph from the set of replicated context scene graphs.

These locally varied scene graphs allow for the management of locales by resolving distributed consistency on a *per-context* basis. There exists exactly one workstation, which owns a particular context and will be responsible for processing all relevant interaction concerning the application. This host's replica is called the *master context*. All other hosts may replicate the context as a *slave context*.

The slave contexts' data and representation (window, PIP sheet etc.) stay synchronized over the whole life span of the context for every replica.

The replication on a per-context basis provides coarse-grained parallelism. At the same time the programming model stays simple and the programmer is relieved of solving difficult concurrency issues since all relevant computation can be performed in a single address space.

The roles that contexts may assume (master or slave) affect the status of the context's application part. The application part of a master context is active and modifies context data directly according to the users' input. In contrast, a slave context's application is dormant and does not react to user input. For example, no callbacks are executed if widgets are triggered. Instead, a slave context relies on updates to be transmitted via DIV. When the application part changes the scene graph of the master context, DIV will pick up the change and propagate it to all slave contexts to keep them in sync with the master context. This process happens transparently within the application, which uses only the master context's scene graph.
Note that context replicas can swap roles (e. g., by exchanging master and slave contexts to achieve load balancing), but at any time there may only be one master copy per replicated context.

Because the low-level replication of context data is taken care of by DIV, the high-level context management protocol is fairly simple: A dedicated session manager process serves as a mediator among hosts as well as a known point of contact for newcomers. The session manager does not have a heavy workload compared to the hosts running the *Studierstube* user interface, but it maintains important directory services. It maintains a list of all active hosts and which contexts they own or subscribe to, and it determines policy issues, such as load balancing, etc.

Finally, input is managed separately by dedicated device servers (typically PCs running Linux), which also perform the necessary filtering and prediction. The tracker data is then multicast in the LAN, so it is simultaneously available to all hosts for rendering.

7. Application programmer's interface

The *Studierstube* API imposes a certain programming model on applications, which is embedded in a foundation class, from which all *Studierstube* applications are derived. By overloading certain polymorphic methods of the foundation class, a programmer can customize the behavior of the application. The structure imposed by the foundation class supports multiple contexts.

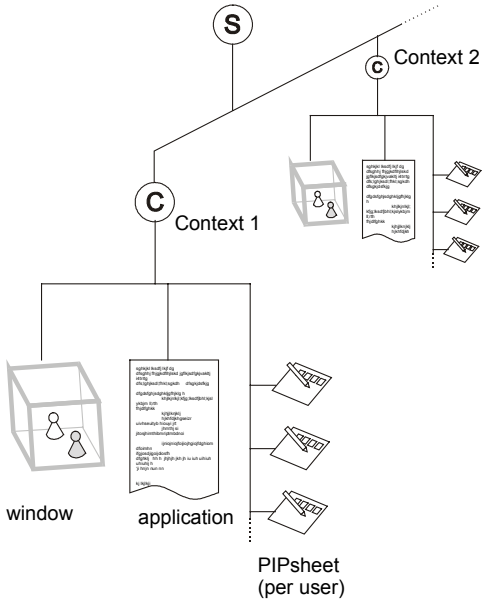

Figure 17: A context is implemented as a node in the scene graph, as are windows and PIP sheets. This allows for the organization of all relevant data in the system in a single hierarchical data structure.

Each context can be operated in both master mode (normal application processing) and slave mode (same data model, but all changes occur remotely through DIV). The key to achieving all of this is to make the context itself a *node* in the scene graph. Such context nodes are implemented as OIV *kit* classes. Kits are special nodes that can store both fields, i.e., simple attributes, and child nodes, both of which will be considered part of the scene graph and thus implicitly be distributed by DIV. Default parts of every context are at least one 3D-window node, which is itself an OIV kit and contains the context's "client area" scene graph, and a set of PIP sheets (one for each participating user). In other words, data, representation, and application are all embedded in a single scene graph (Figure 17), which can be conveniently managed by the *Studierstube* framework.

To create a useful application with all the properties mentioned above, a programmer need only create a subclass of the foundation class and overload the 3D-window and PIP sheet creation methods to return custom scene graphs. Typically, most of the remaining application code will consist of callback methods responding to certain 3D events such as a button press or a 3D direct manipulation event. Although the programmer has the freedom to use anything that the OIV and *Studierstube* toolkits offer, any instance data is required to be stored in the derived context class as a field or node, or otherwise it will not be distributed. However, this is not a restriction in practice, as all basic data types are available in both scalar and vector formats as fields, and new types can be created should the existing ones turn out to be insufficient (a situation that has not occurred to us yet).

Note that allowing a context to operate in either master and slave mode has implications on how contexts can be distributed: It is not necessary to store all master contexts of a particular type at one host. Some master contexts may reside on one host, some on another host—in that case, there usually will be corresponding slave contexts at the respective other host, which are also instances of the same kit class, but

initialized to function as slaves. In essence, *Studierstube's* API provides a distributed multiple document interface.

8. Applications

To evaluate the *Studierstube* platform, a number of applications were developed and are still being developed. They cover a variety of fields, for example, scientific visualization (Fuhrmann et al., 1998), CAD (Encarnação et al, 1999a), and landscape design (Schmalstieg et al., 1999). In this section, three application examples are chosen to highlight the platform's strengths: Section 8.1 discusses storyboard, a multi-user design system, section 8.2 presents MediDesk, a medical visualization tool, and section 8.3 describes Construct3D, a geometry education tool.

8.1 Storyboard design

To demonstrate the possibilities of a heterogeneous virtual environment, we chose the application scenario of *storyboard design*. This application is a prototype of a cinematic design tool. It allows multiple users to concurrently work on a storyboard for a movie or drama. Individual scenes are represented by their stage sets, which resemble *worlds in miniature* (Pausch et al., 1995).

Every scene is represented by its own context and embedded in a 3D-window. Users can manipulate the position of props in the scene as well as the number and placement of actors (represented by colored board game figures), and finally the position of the camera (Figure 18).

All contexts share an additional large *slide show* window, which shows a 2D image of the selected scene from the current camera position. By flipping through the scenes in the given sequence, the resulting slide show conveys the visual composition of the movie.

Alternatively, a user may change the slide show to a "slide sorter" view inspired by current presentation graphics tools, where each scene is represented by a smaller 2D image, and the sequence can be rearranged by simple drag and drop operations. The slide sorter comes closest to the traditional storyboard used in cinematography. It appears on the PIP for easy manipulation as well as on the larger projection screen.

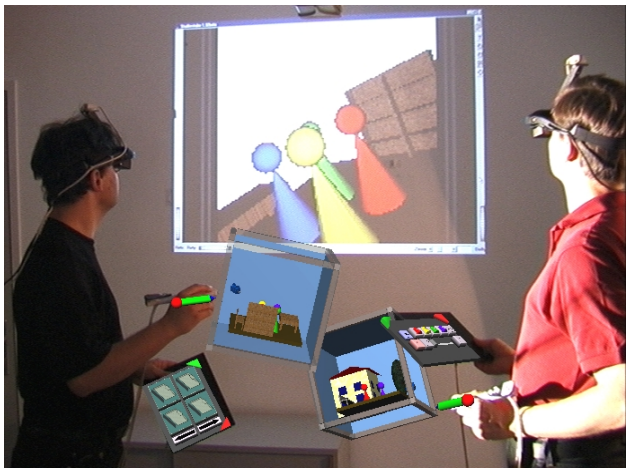

Figure 18: Storyboard application with two users and two contexts as seen from a third "virtual" user perspective, used for video documentation. In the background the video projection is visible.

The test configuration consisted of three hosts (SGI Indigo2 and O2 running IRIX, Intergraph TZ1 Wildcat running Windows NT), two users, and two locales (Figure 19). It was designed to show the convergence of multiple users (real ones as well as virtual ones), contexts, locales, 3D-windows, hosts, displays and operating systems.

The two users were wearing HMDs, both connected to the Indigo2's multi-channel output, and seeing head-tracked stereoscopic graphics. They were also fitted with a pen and panel each. The Intergraph workstation was driving an LCD video projector to generate a monoscopic image of the slide show on the projection screen (without viewpoint tracking), which complemented the presentation of the HMDs.

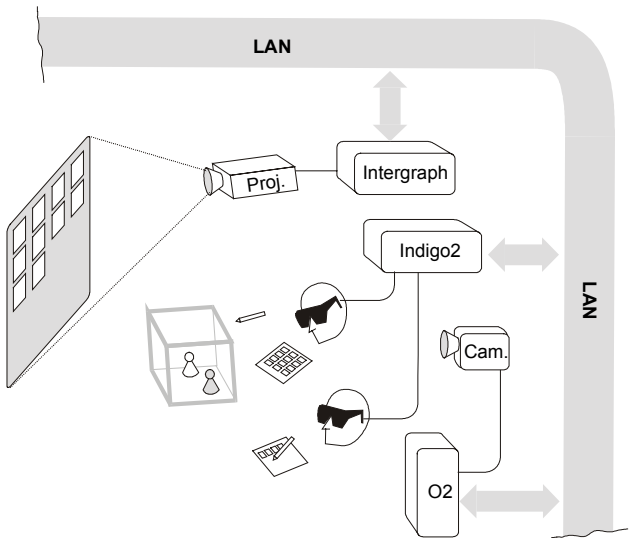

Figure 19: Heterogeneous displays—two users simultaneously see shared graphics (via their see-through HMDs) and a large screen projection.

Users were able to perform some private editing on their local contexts and then update the slide show/sorter to discuss the results. Typically, each user would work on his or her own set of scenes. However, we chose to make all contexts visible to both users so collaborative work on a single scene was also possible. The slide sorter view was shared between both users so global changes to the order of scenes in the movie were immediately recognizable.

The third host—the O2—was configured to combine the graphical output (monoscopic) from *Studierstube* with a live video texture obtained from a video camera pointed at the users and projection screen. The O2 was configured to render images for a virtual user whose position was identical with the physical camera. This feature was used to document the system on video.

The configuration demonstrates the use of overlapping locales: The first locale is shared by the two users to experience the miniature stages at the same position. This locale is also shared by the O2, which behaves like a passive observer of the same virtual space, while a second separate locale was used for the Intergraph driving the projection screen, which could be freely repositioned without affecting the remainder of the system.

8.2 Medical visualization

MediDesk is an application for interactive volume rendering in the *Studierstube* system (Wohlfahrter et al., 2000). As the name suggests, its primary use lies in the field of medical visualization. Users can load volumetric data sets (typically CT or MRI scans), which are rendered using OpenGL Volumizer (Eckel, 1998). Volumizer allows interactive manipulation of volume data, although it requires a high-end SGI workstation for reasonable performance.

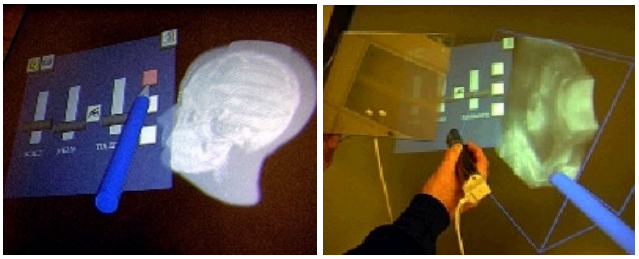

Figure 20: MediDesk allows interactive manipulation of volumetric data, such as CT scans.

As with most *Studierstube* applications, a set of buttons and sliders on the panel allows a user to control the application, such as altering transfer function parameters (Figure 20). The backside of the panel serves special purposes for volume manipulation: This allows for the design of an intuitive interface for volume rendering, a style inspired by a medical doctor's X-ray workplace.

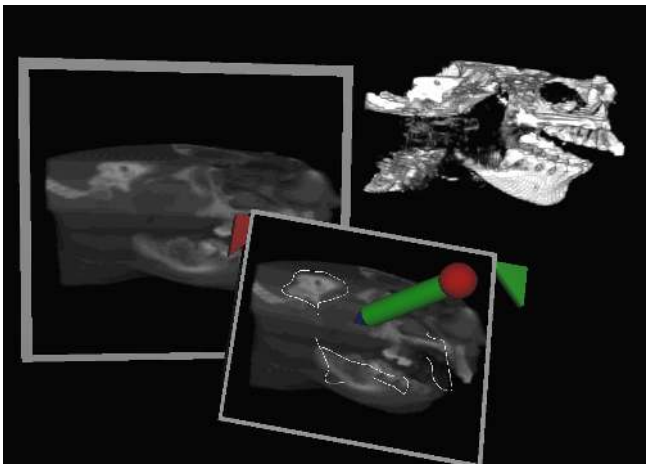

Figure 21: The lower half of the image shows the annotating of a virtual "X-ray" print taken from the volume on the upper right.

The use of two-handed interaction for manipulation of medical data has been found advantageous in the past (Goble et al., 1995). The PIP allows a similar approach: The volumetric data set can be positioned with the pen, while the panel acts as a clipping plane. The user may also freeze one or multiple clipping planes in space to inspect isolated regions of interest. Alternatively, cross-sections can be extracted from the volume with the panel and subsequently appear (as textures) on the pad, where they can be annotated with the pen as if the panel were a notepad. These virtual "X-ray" prints can be attached to a physical wall for reference (Figure 21).

8.3 Geometry education

Construct3D is a prototype application for exploring the use of collaborative augmented reality in mathematics and geometry education (Kaufmann et al., 2000). More specifically, we were interested how constructive geometry education, which still uses traditional pen-and-paper drawing methods to teach high school and college students the basics of three-dimensional space, could be implemented in *Studierstube*. It is important to note that this differs from typical computer aided design (CAD) tasks. Users trained in desktop CAD tools may have a different background and a different set of expectations than students involved in pen and paper exercises.

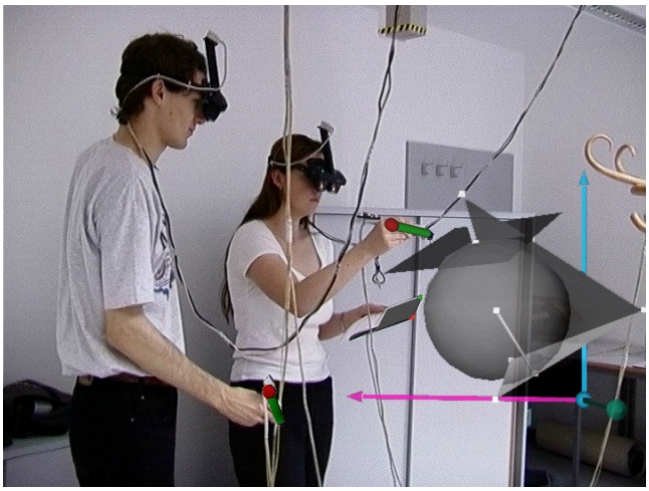

Figure 22: A tutor teaches a student how to geometrically construct 3D entities with Construct3D.

To assess the usability of a 3D tool like *Studierstube*, we found geometry education an interesting application field because is not so much concerned with the final result of the modeling, but rather with the process of construction itself and its mathematical foundation. We tried to evaluate the advantages of actually *seeing* three-dimensional objects, as opposed to calculating and constructing them using two-dimensional views. We speculated that AR would allow a student to enhance, enrich and complement the mental pictures of complex spatial problems and relationships that students form in their minds when working with three-dimensional objects. By working directly in 3D space, it may be possible to comprehend the task better and faster than with traditional methods.

We therefore aimed not at creating a professional 3D modeling package but rather at developing a simple and intuitive 3D construction tool in an immersive AR environment for educational purposes. The main goal was to keep the user interface as simple as possible to facilitate learning and efficient use. The main

areas of application of the system in mathematics and geometry education were vector analysis and descriptive geometry.

Construct3D uses the PIP to offer a palette of geometric objects (point, line, plane, box, sphere, cone and cylinder) that can be input using direct manipulation for coordinate specification (point and click). A coordinate skitter (Bier, 1986) aids accurate positioning. The modeling process is constructive in the sense that more complex primitives can be assembled from simpler ones (e. g., a plane can be defined by indicating a previously created point and line). Audio feedback guides the construction process. The use of transparency for primitives allowed users to observe necessary details, such as intersections.

With this application, an informal user study with 14 subjects was conducted. The test session consists of two parts. The first part required each participant to solve a construction example from mathematics education with the help of a tutor in Construct3D (Figure 22). The example stems from vector analysis as taught in 10^{th} grade in Austria. For high school students, calculating the results would be lengthy and rather complex. In the second part, all subjects completed a brief survey. The survey contains an informal section about VR in general and questions about Construct3D.

In general, speculations that AR is a useful tool for geometry education were confirmed. The subjects were able to perform the task after a few minutes of initial instruction. The majority of comments regarding the AR interface were encouraging. Some questions arose about how larger groups of students could work together (we partly relate this comment to the current tethered setup that has a rather limited working volume). Some comments addressed the technical quality (such as tracking or frame rate). Most students consider AR a useful complement (but not replacement) to traditional pen and paper education. Figure 22 also shows how unplanned uses of the environment can arise—one student spontaneously placed the printed task description on the PIP, thus "augmenting" her PIP with a physical layer of information.

9. Related work

The current architecture of *Studierstube* has absorbed many different influences and is utilizing—partially enhancing—many different ideas. The most influential areas are augmented reality, computer supported cooperative work, ubiquitous computing, and heterogeneous user interfaces. Here the discussion is limited to some of the most influential work:

The Shared Space (Billinghurst et al., 1996; Billinghurst et al., 1998b) project at University of Washington's HITLab has—together with *Studierstube*—pioneered the use of collaborative augmented reality. Since then, HITLab has worked on many innovative applications blending AR with other components into a heterogeneous environment: Easily deployable optical tracking allows to utilize tangible objects for instant augmentation (Kato et al., 2000), for example, to build wearable augmented video conferencing spaces (Billinghurst et al., 1998a) or hybrids of AR and immersive virtual worlds.

The Computer Graphics and User Interfaces lab at Columbia University has a long reputation for augmented reality research (Feiner et al., 1993). Their EMMIE system (Butz et al., 1999) is probably the closest relative to *Studierstube*. It envelops computers and users in a collaborative "ether" populated with graphical data items provided by AR and ubiquitous computing devices such as HMDs, notebooks, PDAs, and projection walls. Communication between stationary and mobile AR users is facilitated as well (Höllerer et al., 1999). Except for the locale concept, EMMIE shares many basic intentions with our research, in particular concurrent use of heterogeneous media in a collaborative work environment. Like us, (Butz et al., 1999) believe

that future user interfaces will require a broader design approach integrating multiple user interface dimensions before a successor to the desktop metaphor can emerge.

Rekimoto has developed a number of setups for multi-computer direct manipulation to bridge heterogeneous media. In (Rekimoto, 1997), a stylus is used to drag and drop data across display boundaries, while Hyperdragging (Rekimoto & Saitoh, 1999) describes a similar concept that merges multiple heterogeneous displays to create a hybrid virtual environment.

The Tangible Media Group at MIT has developed a number of heterogeneous user interfaces based on the theme of tangible (physical) objects (Ishii & Ulmer, 1997). For example, the metaDESK (Ulmer & Ishii, 1997) combines tangible objects with multiple displays, implicitly defining two views into one locale. The luminous room (Underkoffler, 1999) allows remote collaboration using embedded displays, while mediaBLOCKS (Ulmer & Ishii, 1998) are tangible containers that roughly correspond to contexts in *Studierstube*.

The Office of the Future project at UNC (Raskar et al., 1998a) is concerned with the seamless embedding of computer controlled displays into a conventional office environment. This system uses sophisticated front projection to implement spatially augmented reality (Raskar, 1998b), an interesting variety of AR.

CRYSTAL (Tsao & Lumsden, 1997) is a single-user multi-application platform. While it is agnostic in terms of display media, it pioneers the use of 3D-windows and multi-tasking of applications in virtual environments.

Finally, SPLINE (Barrus et al., 1996) is a distributed multi-user environment. From it the term "locale" is borrowed, which in SPLINE is used to describe non-overlapping places. While SPLINE is neither an AR system nor a 3D work environment (according to our use of the term), it allows multiple users to participate in multiple activities (i.e., applications) simultaneously.

10. Conclusions and future work

Studierstube is a prototype system for building innovative user interfaces that use collaborative augmented reality. It is based on a heterogeneous distributed system based on a shared scene graph and a 3D interaction toolkit. This architecture allows for the amalgamation of multiple approaches to user interfaces as needed: augmented reality, projection displays, ubiquitous computing. The environment is controlled by a two-handed pen-and-pad interface, the Personal Interaction Panel, which has versatile uses for interacting with the virtual environment. We also borrow elements from the desktop, such as multi-tasking and multi-windowing. The resulting software architecture resembles in some ways what could be called an "augmented reality operating system."

Research that is currently in its initial phase will investigate the possibilities of mobile collaborative augmented reality. The name *Studierstube* ("study room") may be no longer appropriate for a portable or wearable AR platform, but a mobile 3D information platform has exciting new possibilities, such as ad-hoc networking for instant collaboration of augmented users. Our goal is to allow users to take 3D contexts "on the road" and even dock into a geographically separate environment without having to shut down live applications.

Acknowledgments

The *Studierstube* project is sponsored by the Austrian Science Fund *FWF* under contracts no. P12074-MAT and P14470-INF, by the Fraunhofer IGD/CRCG Student and Scholar Exchange Program, and the Fraunhofer IGD TRADE Virtual Table 1998-1999 Program. Special thanks to Oliver Bimber, Pedro Branco, Hannes Kaufmann, Markus Krutz, Clemens

Pecinovsky, Roman Rath, Gerhard Reitmayr, Gernot Schaufler, Rainer Splechtna, Stan Stoev, André Stork, Hermann Wurnig, and Andreas Zajic for their contributions, and to M. Eduard Gröller for his spiritual guidance.

Web information

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

References

(Angus & Sowizral, 1995) I. Angus, H. Sowizral. Embedding the 2D Interaction Metaphor in a Real 3D Virtual Environment. Proceedings SPIE, vol. 2409, pp. 282-293, 1995.

(Barrus et al., 1996) J. Barrus, R. Waters, R. Anderson. Locales and Beacons: Precise and Efficient Support for Large Multi-User Virtual Environments. Proc. VRAIS '96, pp. 204-213, 1996.

(Bier, 1986) E. Bier. Skitters and Jacks: Interactive 3D Positioning Tools. Proc. 1986 ACM Workshop on Interactive 3D Graphics, Chapel Hill, NC, pp. 183-196, 1986.

(Billinghurst et al., 1996) M. Billinghurst, S. Weghorst, T. Furness III. Shared Space: An Augmented Reality Approach for Computer Supported Collaborative Work, Extended abstract in Proc. of Collaborative Virtual Environments '(CVE'96), Nottingham, UK, 1996.

(Billinghurst et al., 1998a) M. Billinghurst, J. Bowskill, M. Jessop, J. Morphett. A Wearable Spatial Conferencing Space, Proc. ISWC '98, pp. 76-83, 1998.

(Billinghurst et al., 1998b) M. Billinghurst, S. Weghorst, T. Furness III. Shared Space: An Augmented Reality Approach for Computer Supported Collaborative Work, Virtual Reality: Virtual Reality - Systems, Development and Applications, 3(1), pp. 25-36, 1998.

(Bray et al., 2000) T. Bray, J. Paoli, C. Sperberg-McQueen et al. Extensible Markup Language (XML) 1.0. http://www.w3.org/TR/REC-xml/, 2000.

(Butz et al., 1998) A. Butz, C. Beshers, S. Feiner. Of Vampire Mirrors and Privacy Lamps: Privacy Management in Multi-User Augmented Environments. Proc. ACM UIST'98, pp. 171-172, Nov. 1998.

(Butz et al., 1999) A. Butz, T. Höllerer, S. Feiner, B. MacIntyre, C. Beshers. Enveloping Computers and Users in a Collaborative 3D Augmented Reality, Proc. IWAR '99, pp. 1999.

(Cruz-Neira et al., 1993) C. Cruz-Neira, D. Sandin, T. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. Proceedings of SIGGRAPH'93, pp. 135-142, 1993.

(Eckel, 1998) G. Eckel. OpenGL Volumizer Programming Guide. SGI Inc., 1998.

(Encarnação et al., 1999a) L. M. Encarnação, A. Stork, D. Schmalstieg, O. Bimber. The Virtual Table – A Future CAD Workspace. Proceedings of the 1999 CTS (Autofact) Conference, Detroit MI, Sept. 1999.

(Encarnação et al., 1999b) L. M. Encarnação, O. Bimber, D. Schmalstieg, S. Chandler. A Translucent Sketchpad for the Virtual Table Exploring Motion-based Gesture Recognition. Computer Graphics Forum, pp. 277-286, Sept. 1999.

(Encarnação et al., 2000) L. M. Encarnação, R. J.Barton III, P. Stephenson, P. Branco, J. Tesch, J. F. Mulhearn. Interactive exploration of the underwater sonar space in a semi-immersive environment. Submitted for publication, 2000.

(Feiner et al., 1993) S. Feiner, B. MacIntyre, D. Seligmann. Knowledge-Based Augmented Reality. Communications of the ACM, vol. 36, no. 7, pp. 53-62, 1993.

(Fuhrmann et al., 1998) A. Fuhrmann, H. Löffelmann, D. Schmalstieg, M. Gervautz. Collaborative Visualization in Augmented Reality. IEEE Computer Graphics & Applications, Vol. 18, No. 4, pp. 54-59, IEEE Computer Society, 1998.

(Fuhrmann & Schmalstieg, 1999) A. Fuhrmann, D. Schmalstieg. Multi-Context Augmented Reality. Technical report TR-186-2-99-14, Vienna University of Technology, 1999. Available from ftp://ftp.cg.tuwien.ac.at/pub/TR/99/TR-186-2-99-14Paper.pdf

(Goble et al., 1995) J. Goble, K. Hinckley, R. Pausch, J. Snell, N. Kassel. Two-Handed Spatial Interface Tools for Neurosurgical Planning. IEEE Computer, 28(7):20-26, 1995.

(Goethe, 1808) J. W. von Goethe. Faust. Drama, first published 1808. English translation by D. Luke published by Oxford University Press, 1998.

(Guiard, 1987) Y. Guiard. Assymetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as Model. Journal of Motor Behaviour, 19(4):486-517, 1987.

(Hesina et al., 1999) Hesina G., D. Schmalstieg, A. Fuhrmann, W. Purgathofer. Distributed Open Inventor: A Practical Approach to Distributed 3D Graphics, Proc. VRST '99, London, pp. 74-81, Dec. 1999.

(Höllerer et al., 1999) Höllerer T., S. Feiner, T. Terauchi, G. Rashid, D. Hallaway. Exploring MARS: Developing indoor and outdoor user interfaces to a mobile augmented reality systems, Computers & Graphics, 23(6), pp. 779-785, 1999.

(Ishii & Ulmer, 1997) Ishii H., B. Ulmer. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms, Proc. CHI '97, pp. 234-241, 1997.

(Kato et al., 2000) H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, K. Tachibana. Virtual Object Manipulation on a Table-Top AR Environment. Proc. IEEE and ACM International Symposium on Augmented Reality, 2000.

(Kaufmann et al., 2000) H. Kaufmann, D. Schmalstieg, M. Wagner. Construct3D: A Virtual Reality Application for Mathematics and Geometry Education. To appear in: Journal of Education and Information Technologies, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

(Krüger et al., 1995) W. Krüger, C. Bohn, B. Fröhlich, H. Schüth, W. Strauss, G. Wesche. The Responsive Workbench: A Virtual Work Environment. IEEE Computer, vol. 28, no.7, pp. 42-48, 1995.

(Mine et al., 1997) M. Mine, F. Brooks Jr., C. Sequin. Moving Objects in Space: Exploiting Proprioception In Virtual-Environment Interaction. Proc. SIGGRAPH '97, pp. 19-26, 1997.

(Pausch et al., 1995) R. Pausch, T. Burnette, D. Brockway, M. Weiblen. Navigation and Locomotion in Virtual Worlds via Flight into Hand-Held Miniatures, Proc. SIGGRAPH '95, pp. 399-401, 1995.

(Poupyrev et al., 1998) I. Poupyrev, N. Tomokazu, S. Weghorst. Virtual Notepad: Handwriting in Immersive VR. Proc. of VRAIS'98, 1998.

(Raskar et al., 1998a) R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays, Proc. SIGGRAPH '98, pp. 179-188, 1998.

(Raskar et al., 1998b) R. Raskar, G. Welch, H. Fuchs. Spatially Augmented Reality. In: Proceedings of the First IEEE Workshop on Augmented Reality (IWAR'98), San Francisco, CA, USA, A.K. Peters Ltd., 1998.

(Reitmayr & Schmalstieg, 2000) G. Reitmayr, D. Schmalstieg. OpenTracker - An Open Software Architecture for Reconfigurable Tracking based on XML. To appear as a poster in: IEEE Virtual Reality 2001, Yokohama, Japan, March 13-17, 2001. Extended version available as technical report TR-186-2-00-18, Vienna University of Technology, June 2000.

(Rekimoto & Saitoh, 1999) J. Rekimoto, M. Saitoh. Augmented Surfaces: A Spatially Continuous Workspace for Hybrid Computing Environments, Proceedings of CHI'99, pp.378-385, 1999.

(Rekimoto, 1997) J. Rekimoto. Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments, Proc. UIST '97, pp. 31-39, 1997.

(Schmalstieg & Schaufler, 1998) D. Schmalstieg, G. Schaufler. Sewing virtual worlds together with SEAMS: A mechanism to construct complex virtual environments. Presence, 8(4): 449-461, Aug. 1998.

(Schmalstieg et al., 1996) D. Schmalstieg, A. Fuhrmann, Zs. Szalavari, M. Gervautz. Studierstube - Collaborative Augmented Reality, Proc. Collaborative Virtual Environments '96, Nottingham, UK, Sep. 1996.

(Schmalstieg et al., 1999) Schmalstieg D., L. M. Encarnação, Zs. Szalavári. Using Transparent Props For Interaction With The Virtual Table, Proc. SIGGRAPH Symp. on Interactive 3D Graphics '99, pp. 147-154, Atlanta, GI, April 1999 (patent pending).

(Schmalstieg et al., 2000) 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.

(Shaw et al., 1993) C. Shaw, M. Green, J. Liang, Y. Sun. Decoupled Simulation in Virtual Reality with the MR Toolkit. ACM Trans. on Information Systems, 11(3):287-317, 1993.

(Smith & Mariani, 1997) G. Smith, J. Mariani. Using Subjective Views to Enhance 3D Applications, Proc. VRST '97, pp. 139-146, New York, NY, Sep. 1997.

(Stoev et al., 2000) S. Stoev, D. Schmalstieg, W. Strasser. Through-the-lens techniques for remote object manipulation, motion and navigation in virtual environments. Submitted for publication, 2000.

(Stork & de Amicis, 2000) A. Stork, R. de Amicis. ARCADE/VT - a Virtual Table-centric modeling system. Proc. of 4th International Immersive Projection Technology Workshop (IPT 2000), June 19-20, Ames, Iowa, 2000.

(Strauss & Carey, 1992) P. Strauss, R. Carey. An object oriented 3D graphics toolkit, Proc. SIGGRAPH '92, pp. 341-347, 1992.

(Szalavári & Gervautz, 1997) Zs. Szalavári, M. Gervautz. The Personal Interaction Panel - A Two-Handed Interface for Augmented Reality, Computer Graphics Forum, 16(3), pp. 335-346, Sep. 1997.

(Szalavári et al., 1998a) Zs. Szalavári, A. Fuhrmann, D. Schmalstieg, M. Gervautz. Studierstube - An Environment for Collaboration in Augmented Reality, Virtual Reality - Systems, Development and Applications, 3(1), pp. 37-49, 1998.

(Szalavári et al., 1998b) Zs. Szalavári, E. Eckstein, M. Gervautz. Collaborative Gaming in Augmented Reality Proceedings of VRST'98, pp.195-204, Taipei, Taiwan, November 2-5, 1998.

(Tsao & Lumsden, 1997) J. Tsao, C. Lumsden. CRYSTAL: Building Multicontext Virtual Environments, Presence, 6(1), pp. 57-72, 1997.

(Ulmer & Ishii, 1997) B. Ullmer, H. Ishii. The metaDESK: Models and Prototypes for Tangible User Interfaces. In Proceedings of ACM UIST'97, Banff, Alberta, Canada, pp. 223-232, 1997.

(Ulmer & Ishii, 1998) B. Ullmer, H. Ishii, D. Glas. mediaBlocks: Physical Containers, Transports, and Controls for Online Media, Proc. SIGGRAPH '98, pp. 379-386, July 1998.

(Underkoffler et al., 1999) J. Underkoffler, B. Ullmer, H. Ishii. Emancipated Pixels: Real-World Graphics in the Luminous Room. Proc. SIGGRAPH 1999, pp. 385-392, 1999.

(Viega et al., 1996) J. Viega, M. Conway, G. Williams, R. Pausch. 3D Magic Lenses. In Proceedings of ACM UIST'96, pp. 51-58. ACM, 1996.

(Weiser, 1991) M. Weiser. The Computer for the twenty-first century. Scientific American, pp. 94-104, 1991.

(Wernecke, 1994) J. Wernecke. The Inventor Toolmaker : Extending Open Inventor, Release 2. Addison-Wesley, 1994.

(Wloka & Greenfield, 1995) M. Wloka, E. Greenfield: The Virtual Tricoder: A Uniform Interface for Virtual Reality. Proceedings of ACM UIST'95, pp. 39-40, 1995.

(Wohlfahrter et al., 2000) W. Wohlfahrter, L. M. Encarnação, D. Schmalstieg. Interactive Volume Exploration on the StudyDesk. Proceedings of the 4th International Projection Technology Workshop, Ames, Iowa, USA, June 19-20, 2000.