**Dieter Schmalstieg**
dieter@cg.tuwien.ac.at
Vienna University of Technology
Favoritenstrasse 9-11/188
A-1040 Vienna, Austria
Tel. +43(1)58801-18852
Fax +43(1)58801-18898

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

**Gerd Hesina**
**Zsolt Szalavári**
Vienna University of Technology
Austria

**L. Miguel Encarnação**
Fraunhofer CRCG, Inc.
Providence, Rhode Island

**Michael Gervautz**
Imagination GmbH
Vienna, Austria

**Werner Purgathofer**
Vienna University of Technology
Austria

# The *Studierstube* Augmented Reality Project

## Abstract

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 3-D user interface metaphor as powerful as the desktop metaphor for 2-D. 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, and ubiquitous computing—to the interface as needed. The environment is controlled by the Personal Interaction Panel, a two-handed, pen-and-pad interface that has versatile uses for interacting with the virtual environment. *Studierstube* also borrows elements from the desktop, such as multitasking and multi-windowing. The resulting software architecture is a user interface management system for complex augmented reality applications. The presentation is complemented by selected application examples.

## 1 Introduction

*Studierstube* is the German word for the study room in which 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 3-D user interfaces for future work environments. Most current virtual reality systems 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 3-D user interface management: how to use three-dimensional interactive media in a general work environment in which a variety of tasks are performed simultaneously. In essence, we are searching for a 3-D user interface metaphor as powerful as the desktop metaphor for 2-D.

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 3-D information manipulation as a daily routine. Today's information workers are required to perform a large variety of tasks, but communication between human coworkers has an equally significant role. Consequently,

Presence, *Vol. 11, No. 1, February 2002,* 33–54

*Studierstube* tries to support productivity, typically associated with the desktop metaphor, as well as collaboration, typically associated with computer-supported cooperative work applications.

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 3-D work environment.

In the first part of this paper, we review the core user interface technologies of the *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 3-D interface that unites aspects of multiple user interface paradigms, in particular augmented reality, ubiquitous computing, and the desktop metaphor. It is illustrated by reviewing some selected demonstrations that were built using *Studierstube*, and some conclusions are drawn.

## 2 Related Work

The current architecture of *Studierstube* has absorbed many different influences and is utilizing— and 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.

Weiser (1991) introduced the concept of ubiquitous computing as a future paradigm on interaction with computers. Computers are constantly available in our surroundings by embedding them into everyday items, making access to information almost transparent. In contrast, augmented reality systems focus on the use of personal displays (such as see-through HMDs) to enhance a user's perception by overlaying computer-generated images onto a user's view of the real world.

The Shared Space project (Billinghurst, Weghorst, & Furness, 1996, 1998), at University of Washington's HITLab has—together with *Studierstube*—pioneered the use of collaborative augmented reality. The foundation of the group's recent work is ARToolKit, a library for the efficient and reliable optical tracking of card-shaped markers (Kato & Billinghurst, 1999), which is now freely available. Using ARToolKit, the group has worked on many innovative applications blending AR with other components. For example, Billinghurst, Bowskill, Jessop, and Morphett (1998) present a wearable augmented video conferencing space that can be used for remote collaboration. Through the arrangement of video windows in the user's surrounding together with spatialized audio, the sense of presence of the remote participants is heightened. In the Magic Book scenario (Billinghurst, Kato, & Poupyrev, 2001), users can browse 3-D virtual worlds using a real book with marked pages. They may also "teleport" into the worlds presented in the book and explore transitions to and from immersive virtual reality. VOMAR (Kato, Billinghurst, Poupyrev, Imamoto, & Tachibana, 2000) explores the tangible manipulation of virtual worlds using natural gestures in an interior design example.

The Computer Graphics and User Interfaces lab at Columbia University has a long-term reputation for augmented reality research. In one of the seminal works on augmented reality, Feiner, MacIntyre, and Seligman (1993) describe a system capable of guiding a user through complex tasks by providing knowledge-driven annotations of objects in the environment. The group at Columbia recently presented the EMMIE system (Butz, Höllerer, Feiner, MacIntyre, & Beshers, 1999), which is probably the closest relative to *Studierstube*. It combines augmented reality with ubiquitous computing by envel-

oping users and computers in a collaborative "ether" populated with graphical data items. The data is visualized using AR and ubiquitous computing devices such as HMDs, notebooks, personal digital assistants (PDAs), and projection walls. Every device is used according to its capabilities; for example, a handheld display presents hypertext information rather than 3-D graphics. Objects may be moved between devices by natural drag and drop operations, and different operations will be available depending on the used device. EMMIE can also be interfaced with remote mobile AR users (Höllerer, Feiner, Terauchi, Rashid, & Hallaway, 1999). The remote user's position can be visualized in EMMIE, and guiding information can be provided. EMMIE shares many basic intentions with our research, in particular the concurrent use of heterogeneous media in a collaborative work environment.

Rekimoto has developed a number of setups for multi-computer direct manipulation to bridge heterogeneous media. Similar to Fitzmaurice's earlier Chameleon work (1993), he used a handheld display—configured as a video see-through device—to provide an augmented window into the world (Rekimoto & Nagao, 1995). If position information is available for mobile users, personal annotations can be attached to places to be shared with other users (Rekimoto, Ayatsuka, & Hayashi, 1998). Other work deals with bridging the space between possibly heterogeneous devices, a precursor to EMMIE's "ether". Rekimoto (1997) uses a stylus to drag and drop data across display boundaries. Privacy can be kept by using personal—for example, handheld—devices, while information on a projection screen is available to other users. Later, this idea was picked up in Augmented Surfaces (Rekimoto & Saitoh, 1999), wherein a projection table provides the shared space to connect heterogeneous devices. Some objects are passive and simply recall associated data when placed on the Active Surface, whereas other objects such as laptop computers act independently and can, for example, be receptors of information dragged across the Active Surface.

The Tangible Media Group at MIT has developed a number of heterogeneous user interfaces based on the theme of tangible (physical) objects (Ishii & Ullmer, 1997). The group's work is based on the thesis that interaction should be based on physical objects augmented with computer-generated information and function. For example, the metaDESK (Ullmer & Ishii, 1997) is a back-projection table that presents information that can be manipulated with tracked props on its surface. It also offers a combination of viewing modalities, such as a separate handheld display used to observe the scene in a different dimension (3-D graphics), and with an independent viewpoint. The luminous room (Underkoffler, Ullmer, & Ishii, 1999) is a similar framework in which tangible objects are manipulated on a table surface to create complex application setups such as a simulated optical workbench. Although such a system is naturally collaborative by allowing multiple users to independently manipulate tangible objects, the system also allows remote collaboration using multiple separate display surfaces. Another tangible platform, mediaBlocks (Ullmer, Ishii, & Glas, 1998) are physical surrogates for data items. Manipulation of the physical objects on or in a number of receptor platforms such as a whiteboard, printer, or flat screen allow the associated information to be manipulated accordingly.

The Office of the Future project at UNC (Raskar et al., 1998) is concerned with the seamless embedding of computer-controlled displays into a conventional office environment. Their approach uses advanced display techniques such as front projection with multiple projectors on irregular and potentially overlapping surfaces to present computer-generated images almost everywhere. Scene analysis using structured light patterns is used to obtain the necessary geometric information of the real environment. The resulting system provides what Raskar, Welch, and Fuchs (1998) call spatially augmented reality. It is complementary to other forms of augmentation in that it neither requires see-though nor back-projection displays.

Heterogeneous user interfaces are typically explored in conjunction with collaborative tasks and/or conferencing situations, because it is natural that multiple computers are operated by multiple users, for example, in an operations center or a meeting room into which a user may bring a laptop or PDA. An issue that inevitably

arises in such situations is that of privacy: users do not necessarily desire all their data to be public (Butz, Beshers, & Feiner, 1998). A solution for the privacy issue is possible for every architecture that supports independent display to multiple users. So-called *subjective views* can be employed for displaying selected data to only one user if they are useless or distracting to other users (such as local highlighting or annotations) or if privacy is desired. This concept can be implemented on separate desktop displays (Smith & Mariani, 1997), handheld displays (Rekimoto et al., 1998), HMDs (Butz et al., 1999) or time-interlacing displays such as the two-user workbench (Agrawala et al., 1997).

Finally, some sources of inspiration for ideas presented in this paper—CRYSTAL and SPLINE—are not directly related to AR research. CRYSTAL (Tsao & Lumsden, 1997) is a single-user, multi-application platform. It is agnostic in terms of display media, but it pioneers the use of 3-D windows and multitasking of applications in virtual environments. Multiple applications can execute simultaneously in their own independent 3-D containers and communicate by message passing. As the authors point out, the coexistence of multiple applications is typically not supported in virtual reality systems due to performance reasons and development complexity, and CRYSTAL aims to overcome that.

SPLINE (Barrus, Waters, & Anderson, 1996) is a distributed multiuser environment. From it the term *locale* is borrowed, which in SPLINE is used to describe nonoverlapping places. Locales in SPLINE are typically physical places such as a room, hallway, or town square. Although SPLINE is neither an AR system nor a 3-D work environment (according to our use of the term), it allows multiple users to participate in multiple activities simultaneously, which resembles multitasking of applications.

## 3 Interaction in Augmented Reality

The *Studierstube* system as described by Schmalstieg, Fuhrmann, Szalavári, and Gervautz (1996) and Szalavári, Fuhrmann, Schmalstieg, and Gervautz (1998) was one of the first collaborative augmented reality systems. Multiple users gather in a room and can experience a shared virtual space 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. Copresence of users in the same room allows natural interaction (such as talking and gesturing) during a discussion. The combination of real-world experience with the inspection of virtual scenes yields a powerful tool for collaboration. Figure 1 shows two users exploring a scientific data set (Fuhrmann, Löffelmann, Schmalstieg, & Gervautz, 1998).

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

### 3.1 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 handheld props, a pen and a panel, both equipped with trackers. Via the see-through HMD, the props are augmented with computer-gener-

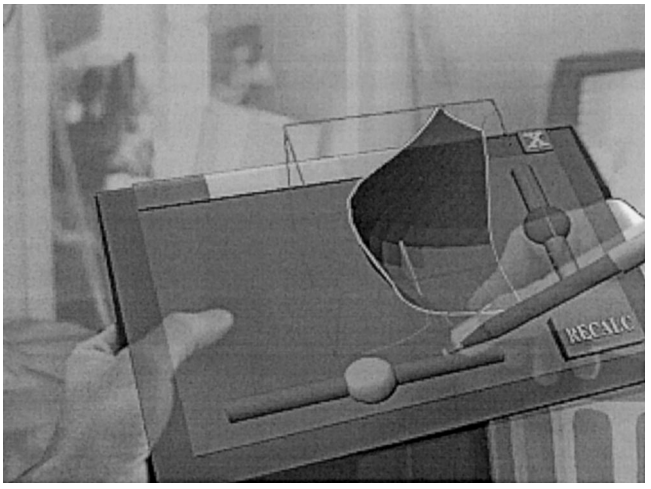

**Figure 2.** *The Personal Interaction Panel allows two-handed interaction with 2-D and 3-D widgets in augmented reality.*

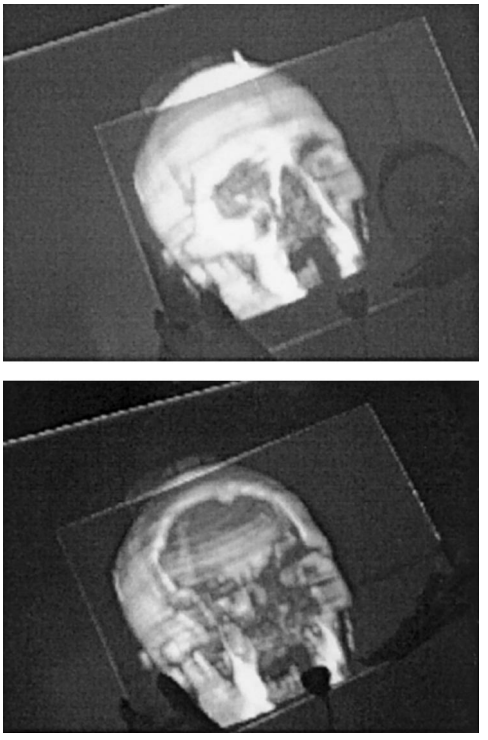

**Figure 3.** *The panel is used to position a clipping plane that cuts away a portion from the volumetric scan of a human skull.*

ated 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, Brooks, & Sequin, 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.

The asymmetric, two-handed interaction exploits Guiard's observations (1987) that humans often use the nondominant hand (holding the panel) to provide a frame of reference for the fine-grained manipulations performed 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 two dimensions in 3-D (figure 2). Many of the artifacts we encounter in real life—such as TV remote controls or button panels on household items—are essentially two-dimensional. The PIP approach, with its tactile feedback on the panel's surface, resembles those real-world artifacts better than naive 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 "2.5-D" widgets like buttons, sliders, or dials provide the tools of interaction.

Beyond system control functions, both props allow direct manipulation in three dimensions and lend themselves to a multitude of gesture-based interactions. For example, figure 3 shows how the pad is used in a medical application (Wohlfahrter, Encarnação, & Schmalstieg, 2000) to clip away portions of a medical data set to look inside in a manner similar to Goble, Hinckley, Pausch, Snell, and Kassel (1995). For more uses of the PIP, see, for example, Schmalstieg et al. (1999), Encarnação, Bimber, Schmalstieg, and Berton (2000), Encar-

nação, Bimber, Schmalstieg, and Chandler (1999), and Stoev, Schmalstieg, and Strasser (2001).

### 3.2 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 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 and 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. 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. For example, this idea was exploited by Szalavári, Eckstein, and Gervautz (1998) to prevent users of collaborative games from cheating. (See figure 4.)

## 4 Convergence of User Interface Metaphors

During the work on the original *Studierstube* architecture, we rapidly discovered new promising avenues of research that could not be investigated using the initial limited design. From approximately 1998 on, we therefore concentrated our efforts at reengineering and extending the initial solutions to construct a second-generation platform that built on what we had learned.

As a first attempt towards an extended platform, support for the Virtual Table (VT), a workbench-like device, was added to *Studierstube* (Schmalstieg, Encarnação, & Szalavári, 1999). The VT uses a large stereo back-projection screen and can provide stereoscopic graphics viewed with shutter glasses for a single head-tracked user. Through the use of transparent props made from Plexiglas that allow see-through graphics, *Studierstube's* original PIP-based interaction style was transposed to the new display hardware. (See figure 5.)

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 3-D 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. (See figure 6.) In particular, we wanted to converge AR with elements from ubiquitious 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, 1991) 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 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.

Although 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-dimensional widgets are not the only element of the desktop metaphor that we consider useful in a 3-D work environment. Desktop users have long grown accustomed to multitasking of applications that complement each

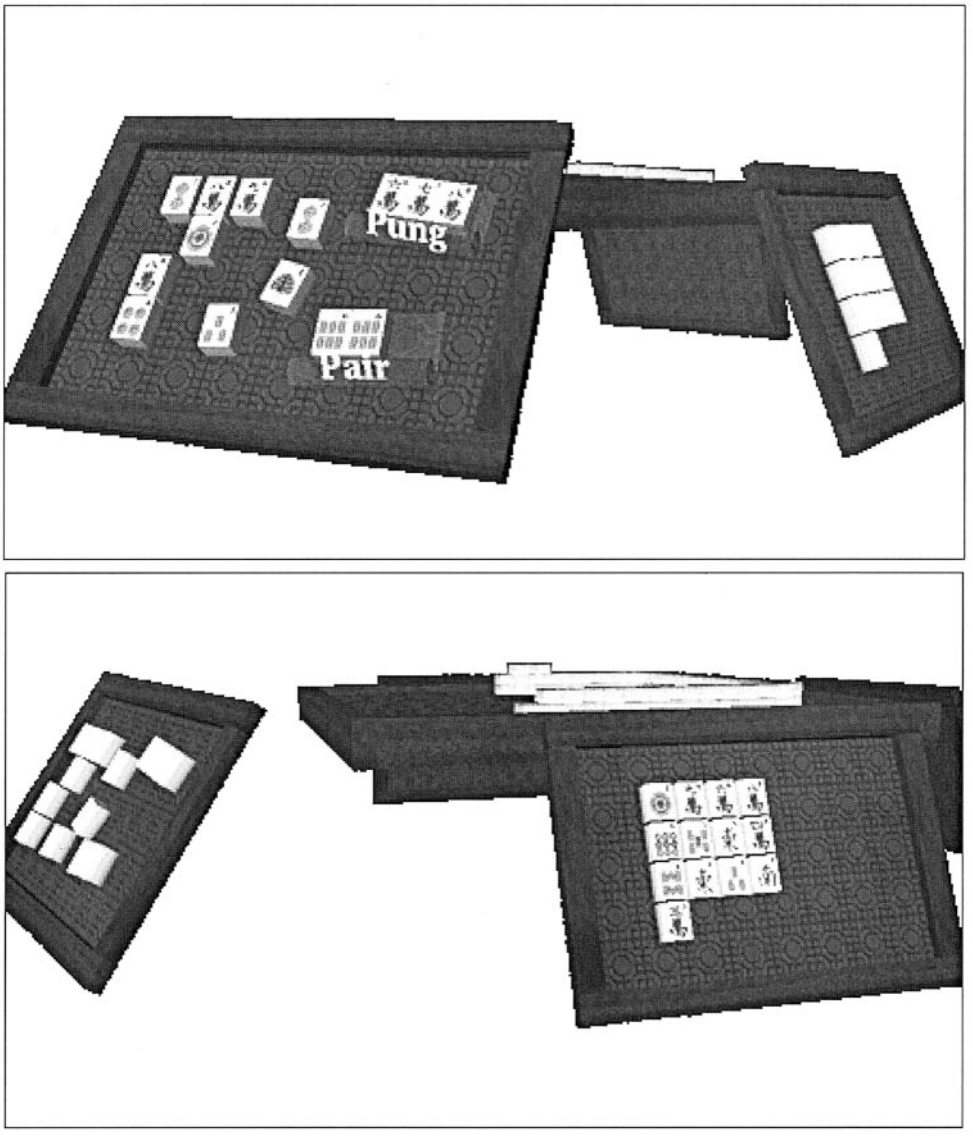


**Figure 4.** *Personal displays secure privacy when playing Mah-jongg. The left player (upper view) cannot see his opponent's tile labels and vice versa (lower view).*

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)—that is, working with multiple documents or data sets simultaneously—allowing comparison and exchange of data among documents. The use of 2-D windows associated with documents allows the convenient arrangement of multiple documents according to a user's preferences. Although these properties are established in the desktop world, they are not exclusive to it and are indeed useful to enhance productivity in a 3-D work environment as well.

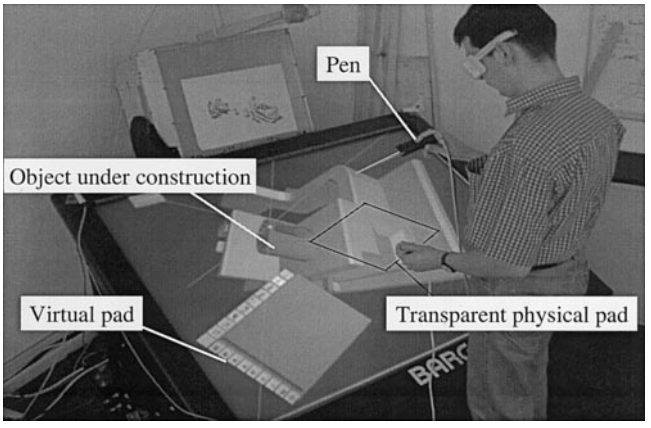


**Figure 5.** *Transparent pen and pad for the Virtual Table are almost invisible and replaced by computer graphics in the user's perception (image courtesy André Stork)*

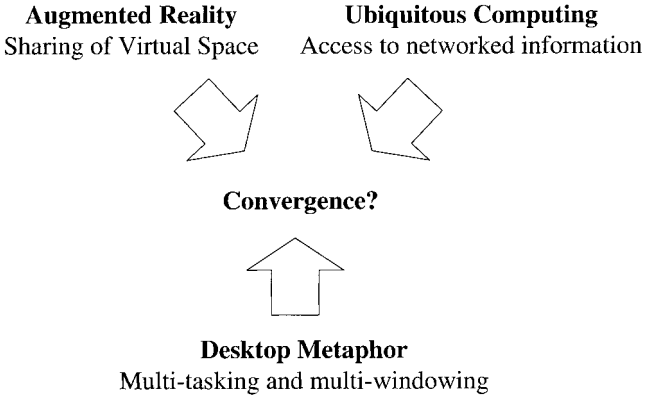


**Figure 6.** *The latest Studierstube* platform combines the best elements from augmented reality, ubiquitous computing, and the desktop metaphor.

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

### 4.1 Users

Support for multiple collaborating users is a fundamental property of the *Studierstube* architecture. Although 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 incorporates multiple host computers, typically one per user. However, *Studierstube* also allows multiple users to interact with a single host (for example, via a large screen or multiheaded 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.

### 4.2 Application Objects

The building blocks for organizing information in *Studierstube* are called *application objects*. An application object encloses application data itself, the data's graphical 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 interact with only the graphical representation, so the notion of an application is completely hidden from the user. In particular, users never have to "start" an application; they simply create or access an object of a specific application type. Conceptually, applications are always "on" (Kato et al., 2000).

In a desktop system, the data representation of a document is typically a single 2-D window. Analogously, in our three-dimensional user interface, an application object's graphical representation is a three-dimensional structure contained in a box-shaped volume: a 3D-window. (See figure 7.) Note that, unlike its 2-D counterpart, an application object can be shared by any number of users.

Every application object is an instance of a particular application type. Different types can exist concurrently, resulting in applications multitasking. Moreover, *Studierstube* also allows multiple instances of the same application type, thereby implementing an MDI. Multiple application instances are aware of each other and can share features and data. For example, consider the miniature stages of the storyboarding application (see section 8), which share the "slide sorter" view.

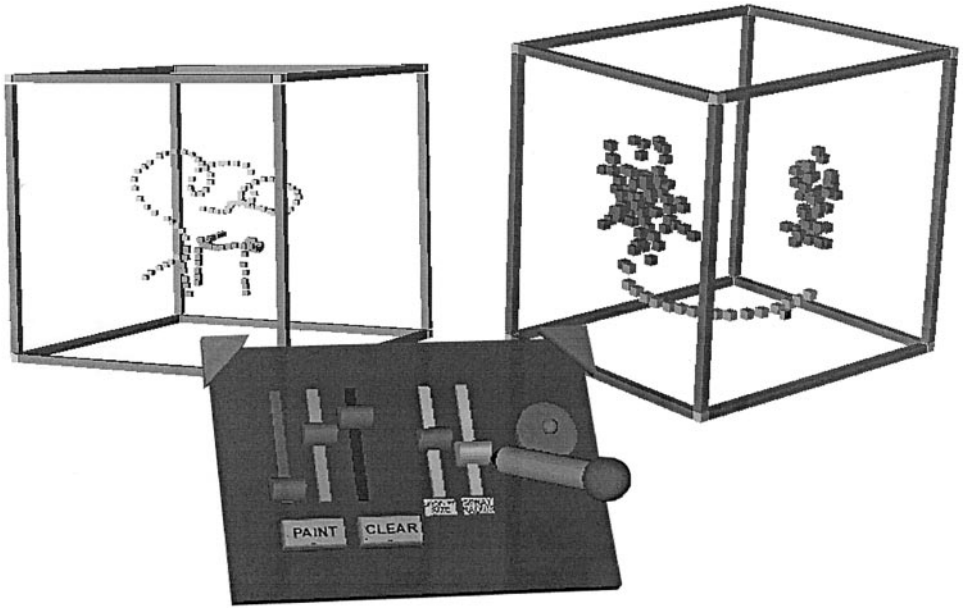


**Figure 7.** *Multiple document interface in 3-D. The right window has the user's focus, indicated by the dark window frame, and can be manipulated with the control elements on the PIP.*

### 4.3 Locales

Locales correspond to coordinate systems in the virtual environment (Barrus et al., 1996). 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 shared by multiple displays. Every application object can—but need not—be replicated in every locale; that is, it can appear, at most, once in every locale. All replicas of a particular application object are kept synchronized by *Studierstube's* distribution mechanism. (See section 7.)

### 4.4 Application versus Locale

At first glance, it may not be obvious why a separation of application objects 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 application and vice versa. All displays share the same physical locale. This approach is simple to understand and easy to implement, but 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 of the 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 a large environment is structured in a way that forces users to walk around in order to access frequently required data. The larger the user group, the more likely it becomes that two users who 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 because the position of a particular data item will often be inaccessible to a remote user.

In contrast, *Studierstube* separates application objects

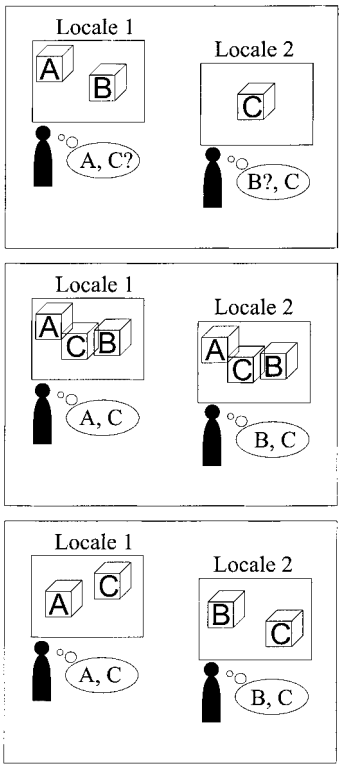


**Figure 8.** *Top: A global arrangement of items cannot fulfill all needs. Middle: Full replication of all items leads to display clutter. Bottom: On-demand replication of items allows the convenient customization of locales.*

and locales for increased flexibility. Every display uses a separate locale, that is, a scene with an independent coordinate system. An application object is placed in a locale by assigning to its 3-D windows a particular position within the locale. This approach allows for several strategies regarding the arrangement of application objects in the relevant locales.

A strategy of making an application object 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 application objects. (See figure 8, top.) In contrast, a strategy of replicating every application object in every locale guarantees convenient access, but quickly leads to display clutter. (See figure 8, middle.)

Therefore, replication of an application object in a given locale is optimal: there may be at most one replica of a given application object in any locale. This strategy allows a user to arrange a convenient working set of application objects in his or her preferred display. (See figure 8, bottom.) If the displays are connected to separate hosts in a distributed system, only those hosts that replicate an application object need to synchronize the corresponding 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. This is used for the initial collaborative augmented reality scenario as depicted in figure 1, and also corresponds to the virtual ether of EMMIE (Butz et al., 1999). Figure 9 illustrates the use of this feature for the 3-D drag and drop of an application object across display boundaries in a *Studierstube* environment composed of two adjacent desktop computers.

Although the space composed from stationary displays can usually be embedded in a single locale, mobile displays (such as notebooks or handheld displays) are better managed using a separate locale per display. In a scenario with two notebooks, applications could be moved from one locale to the next for collaborative work. In general, mobile users will require user-stabilized "mobile locales" that overlap in collaborative work situations. More examples on the use of application objects and locales are given in section 9.

## 5 Implementation of the User Interface

### 5.1 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 with-

**Figure 9.** *Application migration through dragging one of multiple 3-D application windows across display and host boundaries.*

out 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 3-D events—and a runtime framework that provides the necessary environment for *Studierstube* applications to execute. (See figure 10.) Together, these components form a well-defined API, which extends the OIV API and also offers a convenient programming model to the application programmer. (See section 8.)

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 multitasking of different applications (for example, a medical visualization and a 3-D modeler) and also an MDI.

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

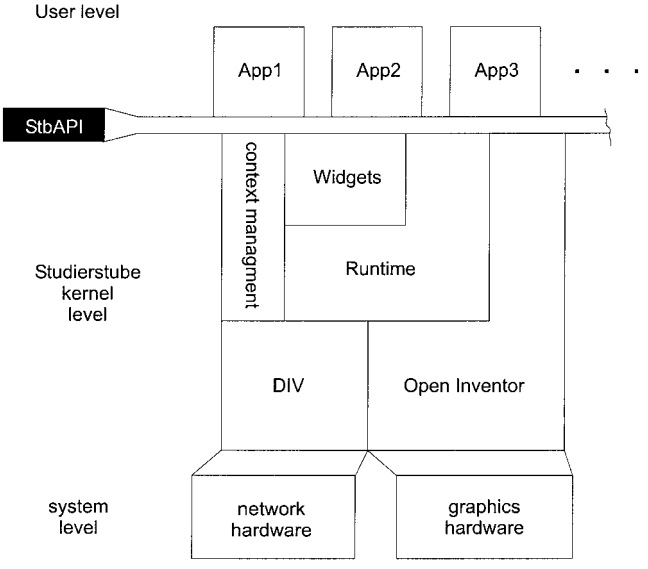


**Figure 10.** *The Studierstube* software is composed of an interaction toolkit and runtime system. The latter is responsible for managing application objects and distribution.

### 5.2 Three-Dimensional Windows

The use of windows as an abstraction and interaction metaphor is an established convention in 2-D

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 it also defines its exact boundaries.

An application object is normally represented in the scene by a 3-D window, although it may span multiple windows. The 3-D 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. Although nested windows are possible, 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, and so on). The color of the decoration also indicates whether a window is active (and hence receives 3-D events from that user). Like their 2-D counterparts, 3-D 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 application objects of the same type will maintain structurally similar windows, but this decision is at the discretion of the application programmer.

### 5.3 PIP Sheets

*Studierstube* applications are controlled either via direct manipulation of the data presented in 3-D windows, or via a mixture of 2-D and 3-D 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). However, the scene graph may also contain geometries (such as 2-D and 3-D icons) that convey the user interface state or can be used merely as decoration.

Every type of application object defines a PIP sheet template, a kind of application resource. For every application object 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 application object. However, widgets can also be shared by all users and/or all application objects. Consequently, *Studierstube's* 3-D event routing involves multiplexing between windows and users' PIP sheets.

## 6 Hardware Support

### 6.1 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, and so forth. After some consideration, we implemented an OIV-compatible viewer with a plugin architecture for camera control and display mode. The following display modes are currently 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 stereo**: Images for left/right eye are output on two different channels.
- **Anaglyph stereo**: Superimposed images for left/right eye are coded in red/green or red/blue.
- **Mono**: The same image is presented to both eyes.

The following camera control modes are currently supported:

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

This approach, together with a general off-axis camera implementation, allows runtime configuration of

**Table 1.** *All Combinations of Camera Control and Display Modes Have Distinct Uses*

| | Tracked Display | Tracked Head | Desktop |
|---|---|---|---|
| Field sequential | Glasstron (stereo) | Virtual Table or projector | Fishtank VR with shutter glasses |
| Line interleaved | i-glasses | VREX projector | i-glasses w/o head tracking |
| Dual screen | i-glasses Protec; Saab/Ericsson AddVisor | Single user dual-projector passive stereo w/head track | Multiuser dual-projector passive stereo |
| Anaglyph | — | Virtual Table or projector | Cheap Fishtank VR |
| Mono | Glasstron (mono) | Virtual Table (mono) | Desktop viewer |

almost any available display hardware. Table 1 shows an overview of some devices that we have evaluated so far.

### 6.2 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. Although 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 have very similar basic components, but they 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**: Many necessary configuration options can be characterized as filters, that is, 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 processes are running as services on dedicated hosts to share the computational load and distribute the available data via unicast and multicast mechanisms, thereby implementing a decoupled simulation scheme (Shaw, Green, Liang, & Sun, 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, 2001a), which is available as open source. It is based on a graph structure composed of linked nodes: source nodes deliver tracker data, and sink nodes consume data for further processing (for example, 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 (Du-

Charme, 1998) description of the graph, standard XML tools can be applied to author, compile, document, and script the OpenTracker architecture.

## 7 Distributed Execution

The distribution of *Studierstube* requires that, for each replica of an application object, 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.

### 7.1 Distributed Shared Scene Graph

Toward that aim, Distributed Open Inventor (DIV) was developed (Hesina, Schmalstieg, Fuhrmann, & Purgathofer, 1999) as an extension—more a kind of plugin—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, that is, 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 that propagates the changes via the network to all hosts that have a replica of the application object's scene graph, where the modifications are duplicated on the remote scene graph by a DIV listener. (See figure 11.)

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.

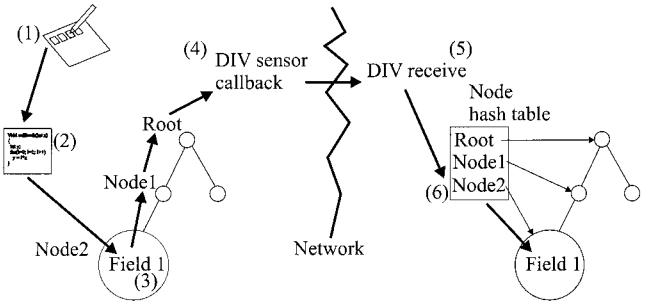


**Figure 11.** *Example of a field update in a master-slave configuration. (1) User triggers an action by pressing a button. (2) Corresponding callback is executed and modified field1 of node2. (3) Event notification is propagated upwards in scene graph and observed by sensor. (4) Sensor transmits message to slave host. (5) Receiver picks up message and looks up corresponding node in internal hash table. (6) Slave node is modified.*

### 7.2 Distributed Application Object 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 application object scene graphs.

These locally varied scene graphs allow for the management of locales by resolving distributed consistency on a per application object basis. There exists exactly one workstation that owns a particular application object and will be responsible for processing all relevant interaction concerning the application. This host's replica is called the *master* application object. All other hosts may replicate the application object as a *slave*. The slaves' data and representation (window, PIP sheet, and so on) stay synchronized over the whole life span of the application object for every replica.

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

The roles that application objects may assume (master

or slave) affect the behavior of the application object. The application part of a master is active and modifies application object data directly according to the users' input. In contrast, a slave is dormant and does not react to user input. For example, no callbacks are executed if widgets are triggered. Instead, a slave relies on updates to be transmitted via DIV. When something changes the scene graph of the master application object, DIV will pick up the change and propagate it to all slaves to keep them in sync with the master. This process happens transparently within the application, which uses only the master's scene graph.

Note that application object replicas can swap roles (such as by exchanging master and slave to achieve load balancing), but at any time there may be only one master copy per replicated application object.

Because the low-level replication of application object data is taken care of by DIV, the high-level application object 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 application objects they own or subscribe to, and it determines policy issues, such as load balancing.

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.

## 8 Application Programmer's Interface

The *Studierstube* API imposes a certain programming model on applications, which is embedded in a foundation class and 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 application objects (MDI).

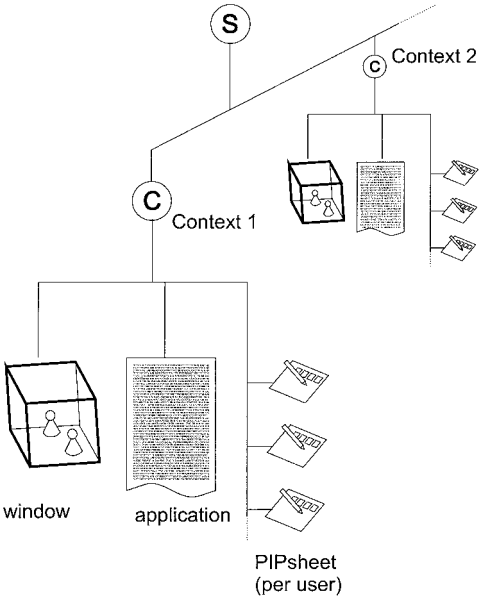


**Figure 12.** *An application object is implemented as a node in the scene graph, as are windows and PIP sheets. This allows for the organization of all relevant data in the system in a single hierarchical data structure.*

Each application object 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 application object itself a node in the scene graph. Such application object nodes are implemented as OIV kit classes. Kits are special nodes that can store both fields, that is, 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 application object are at least one 3-D window node, which is itself an OIV kit and contains the application object'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 (see figure 12), which can be conveniently managed by the *Studierstube* framework.

To create a useful application with all of the aforementioned properties, a programmer need only create a subclass of the foundation class and overload the 3-D window and PIP sheet creation methods to return cus-

tom scene graphs. Typically, most of the remaining application code consists of callback methods responding to certain 3-D events such as a button press or a 3-D 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 application object class as a field or node, or otherwise it will not be distributed. However, this is not a restriction in practice because 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 an application object to operate in either master and slave mode has implications on how application objects can be distributed: it is not necessary to store all master application objects of a particular type at one host. Some master application objects may reside on one host, some on another host. In this case, there usually will be corresponding slave application objects 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.

## 9 Examples

This section describes two experimental applications that were designed to explore the possibilities of heterogeneous user interfaces involving the convergence of augmented reality with other media. Section 9.1 presents Storyboard, a heterogeneous environment for collaborative design of film sequences. Section 9.2 introduces a mobile collaborative AR platform that leverages heterogeneous tracking to allow a roaming user to collaborate with an existing *Studierstube* installation. Both scenarios make use of multiple locales.

### 9.1 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 film design tool, and it allows multiple users to concurrently work on a storyboard for a film or drama. Individual scenes are represented by their stage sets, which resemble worlds in miniature (Pausch, Burnette, Brockway, & Weiblen, 1995).

Every scene is represented by its own application object and embedded in a 3-D window. Users can manipulate the position of props in the scene as well as the number and placement of actors (represented by colored boardgame figures), and finally the position of the camera. (See figure 13.)

All application objects share an additional large slideshow window, which shows a 2-D image of the selected scene from the current camera position. By flipping through the scenes in the given sequence, the resulting slideshow conveys the visual composition of the film.

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

The test configuration consisted of three hosts (SGI Indigo2 and O2 running IRIX, Intergraph TZ1 Wildcat running Windows NT), two users, and two locales. (See figure 14.) It was designed to show the convergence of multiple users (real ones as well as virtual ones), application objects, locales, 3-D windows, hosts, displays, and operating systems.

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

Users were able to perform some private editing on their local application objects and then update the slideshow/sorter to discuss the results. Typically, each user would work on his or her own set of scenes. However,

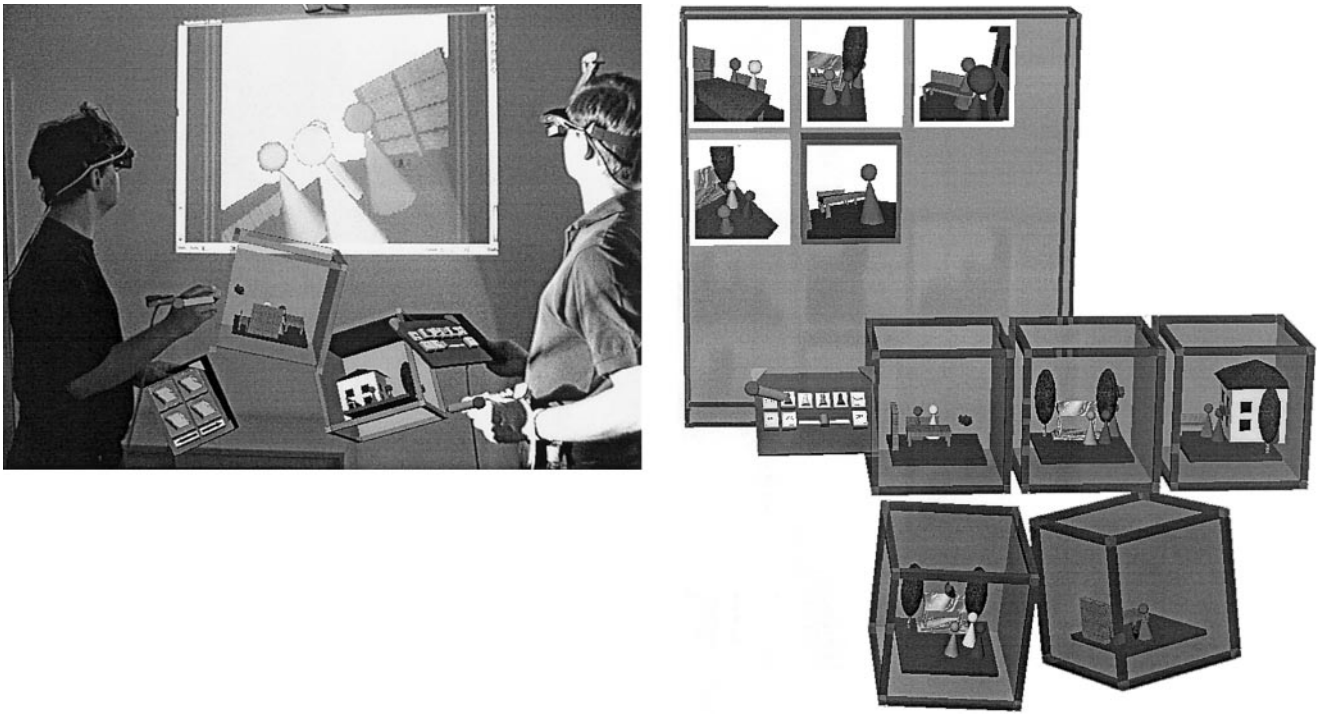

**Figure 13.** *Left: Storyboard application with two users and two stage sets as seen from a third "virtual" user perspective, used for video documentation. The video projection is visible in the background. Right: Slide sorter view with five corresponding stage sets.*

we chose to make all application objects 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 film 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 to 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.

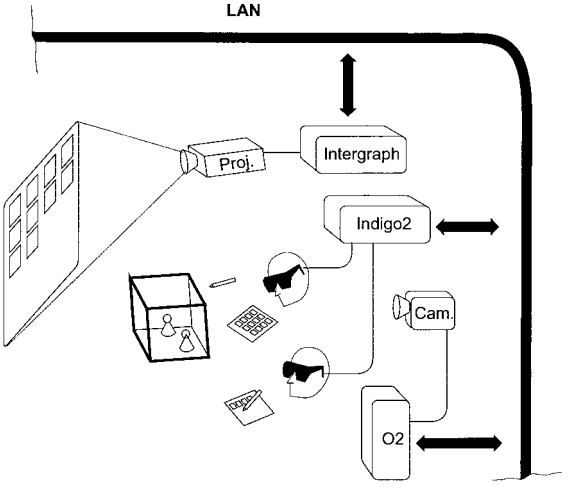


**Figure 14.** *Heterogeneous displays: two users simultaneously see shared graphics (via their see-through HMDs) and a large-screen projection.*

### 9.2 Mobile Augmented Reality

A recent addition to the *Studierstube* environment is the capability to execute on mobile hardware. The previously implemented hardware abstraction capabilities as detailed in section 6 allowed us to assemble a mobile *Studierstube* kit from commercial components: a notebook with GeForce2 Go graphics accelerator drives a stereocopic i-glasses HMD. Tracker data is fused from an optical tracker using ARToolKit (Kato et al., 2000) with a helmet-mounted camera, an inertial tracker (Intertrax2), and a tablet (Wacom). This setup allows untethered operation of the *Studierstube* software and connects to a distributed environment via wireless ethernet. For details, refer to Reitmayr and Schmalstieg (2001b).

Using head- and body-mounted orientation tracking, the mobile system presents information (that is, application objects) using a body-stabilized display. As the user moves about, the relative position of application objects to the user's body remains constant unless directly manipulated. Expressed in terms established earlier, every user gets a personal locale that is body stabilized. When multiple mobile users meet, or if a mobile user enters a stationary *Studierstube* environment, a situation with multiple overlapping locales naturally occurs. For collaboration, it is desirable that a locale shared between the participating users is established to allow direct interaction on shared data.

This goal can be achieved in one of several ways. If a situated display (such as a projection screen or VT) is available, it provides a natural frame of reference for sharing. If all displaying is done on HMDs, the users' personal locales can be anchored to a world-stabilized reference point. In the example described in the next paragraph, this was done by attaching a tracked marker to a table. It may be desirable to split a user's application objects among a body-stabilized personal locale and a world-stabilized shared locale. To accommodate this situation, the restriction of one locale per display can be overcome by embedding all application objects in the shared world-stabilized locale, but actively coupling the position of selected application objects to a user's relative body position expressed in world-stabilized coordinates.

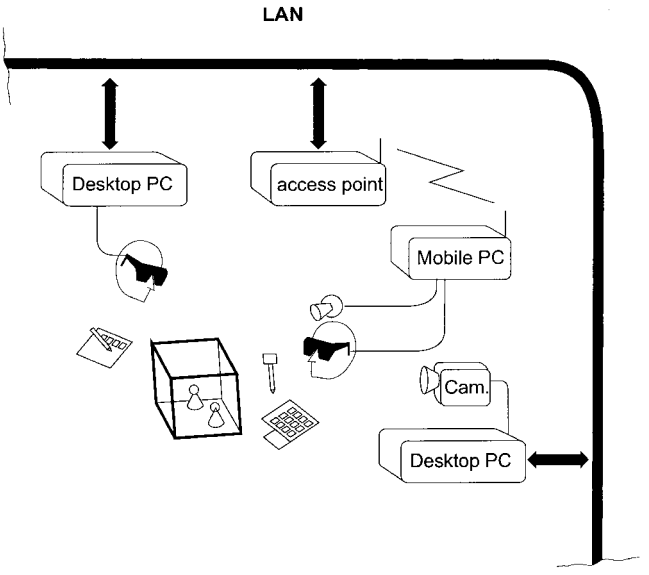


**Figure 15.** *Hardware setup for collaboration of mobile and stationary user.*

In an experimental setup, we investigated collaboration between a mobile and a stationary user. Two locales were used as motivated previously: a body-stabilized locale for the mobile user, and a world-stabilized locale for shared applications. The stationary user is tracked using a tethered magnetic tracking system. The user's display, a stereoscopic Sony Glasstron HMD, is driven by a desktop PC rendering host. The mobile user is equipped with the tracking setup previously described. A third rendering host featuring a video camera was used to render a third person's view over a video image to document the setup on video. Figure 15 displays the configuration.

The stationary user's tracking is straightforward, but the mobile user has to be correctly registered within the stationary user's coordinate system to use the same locale as the stationary user. This is achieved by an additional optical marker that is placed within the environment. Its position within the reference frame of the magnetic tracker is registered. This establishes an anchor for transforming the locations generated by the optical tracking of the mobile user to the magnetic reference frame.

**Figure 16.** *Collaborative game of chess between a mobile and a stationary user.*

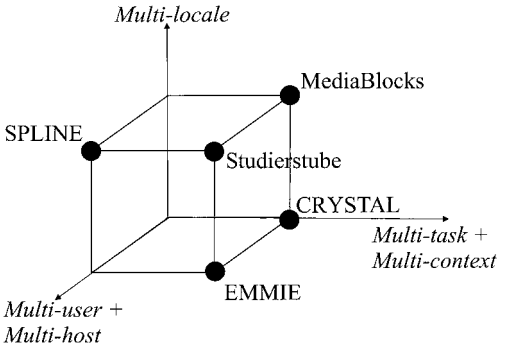


**Figure 17.** *Extended taxonomy for multiple dimensions of user interfaces with some related work (adapted from CRYSTAL).*

An OpenTracker standalone server running on the mobile unit tracks the anchor marker's location relative to the camera. Inverting this location information yields the camera's location relative to the marker. Using this information and the registration data of the marker, it computes the user's location and subsequently that of the interaction props within the magnetic tracker reference frame. It then sends the location to the other hosts via a second multicast group. Note that all three hosts receive tracker data from both multicast groups (optical/inertial and magnetic) and use OpenTracker components to combine the data for correct display and interaction of all tracked elements.

All three hosts are configured individually for the different needs of the output devices (for example, the Sony Glasstron HMD uses a field-interleaved stereo video signal, whereas the i-glasses device uses a line-interleaved stereo signal). Each run an instance of the shared chess application and use the tracking data communicated via the two multicast groups to render the user's actions properly. Figure 16 shows a collaborative game of chess from the point of view of the third host's camera.

## 10 Conclusions and Future Work

As observed by Tsao & Lumsden (1997), to be successful for everyday productivity work situations, virtual environment systems must allow multi tasking and multi-context operation. By *multi tasking*, they mean that the virtual environment can be reconfigured to execute different applications; that is, there is a separation of VR system software and application software.

Multi-context operation goes beyond that by allowing multiple applications to execute concurrently rather than sequentially. They also point out that this resembles a development earlier experienced for 2-D user interfaces, which evolved from single-application text consoles to multi-application windowing systems. It is no surprise that by "judicious borrowing," many useful results from 2-D user interfaces become applicable to 3-D, as is evident with *Studierstube's* PIP, 3-D windows, or 3-D event system.

However, CRYSTAL does not incorporate true multiuser operation, and consequently has no need for multiple locales. Extending the taxonomy from CRYSTAL, figure 17 compares some relevant work. For example, MIT's mediaBlocks (Ullmer et al., 1998) allow a user to work with different manipulators, which are dedicated devices for specific applications, and the mediaBlocks themselves are a very elegant embedding for context data. However, although principally possible, no multiuser scenarios were demonstrated.

In contrast, SPLINE (Barrus et al., 1996) is designed towards multiuser interaction. Although SPLINE completely immerses a user in a purely virtual world and thus does not meet our definition of a work environment, it features multiple locales that correspond to activities (for example, chat takes place in a street cafe, while train rides take place on a train).

The closest relative to our work is Columbia's EMMIE (Butz et al., 1999). Except for explicit support of locales, EMMIE shares many basic intentions with our research, in particular the concurrent use of heterogeneous media in a collaborative work environment. Like ourselves, the authors of EMMIE believe that future user interfaces will require a broader design approach that integrates multiple user interface dimensions before a successor to the desktop metaphor can emerge.

Although *Studierstube* is definitely just a prototype on the way to truly usable future 3-D work environments, it is unique in its comprehensive support for all three concepts previously discussed: users, tasks (applications), and locales. This strategy allows the integration of diverse approaches to user interfaces as needed, for example, augmented reality, situated and ubiquitous computing, and desktop computing. The resulting software architecture is a powerful user interface management system that is capable not only of collaborative augmented reality, but of very diverse heterogeneous user interface styles and applications.

Future work will emphasize mobile augmented reality. Mobile users should be able to fully explore the capabilities of the system independent of location, but also be able to use situated infrastructure. Our goal is to allow users to take 3-D application objects "on the road" and even dock into a geographically separate environment without having to shut down live applications.

## References


Agrawala, M., Beers, A. C., Fröhlich, B., McDowall, I., Hanrahan, P., & Bolas, M. (1997). The two-user responsive workbench: Support for collaboration through individual views of a shared space. *Proceedings SIGGRAPH '97,* 327–332.

Barrus, J., Waters R., & Anderson R. (1996). Locales and beacons: Precise and efficient support for large multi-user virtual environments. *Proceedings IEEE Virtual Reality Annual International Symposium (VRAIS'96),* 204–213.

Billinghurst, M., Weghorst, S., & Furness, T. III. (1996). Shared space: An augmented reality approach for computer supported collaborative work. Extended abstract in *Proceedings of Collaborative Virtual Environments (CVE'96)*. Nottingham, UK.

———. (1998). Shared space: An augmented reality approach for computer supported collaborative work. *Virtual Reality: Systems, Development and Applications, 3*(1), 25–36.

Billinghurst, M., Bowskill, J, Jessop, M, & Morphett, J. (1998). A wearable spatial conferencing space. *Proceedings International Symposium on Wearable Computing (ISWC'98),* 76–83.

Billinghurst, M., Kato, H., & Poupyrev, I (2001). The MagicBook: An interface that moves seamlessly between reality and virtuality. *IEEE Computer Graphics and Applications* (May/June), 2–4.

Butz, A., Beshers, C., & Feiner, S. (1998). Of vampire mirrors and privacy lamps: Privacy management in multi-user augmented environments. *Proceedings ACM User Interface Software and Technology (UIST'98),* 171–172.

Butz, A., Höllerer, T., Feiner, S., MacIntyre, B., & Beshers, C., (1999). Enveloping users and computers in a collaborative 3D augmented reality. *Proceedings International Workshop on Augmented Reality (IWAR'99),* 35–44.

Ducharme, B. (1998). *XML: The annotated specification*. Upper Saddle River, NJ: Prentice Hall PTR.

Encarnação, L. M., Bimber, O., Schmalstieg, D., & Berton, R. J. (2000). Seamless 3D interaction for virtual tables, projection planes, and caves. In Mapper, D. G. (Ed.) SPIE - The International Society for Optical Engineering: Cockpit Displays 2000. *Proceedings* Displays for Defense Applications. Bellingham: SPIE, 2000, p. 177–188 (SPIE Proceedings Series 4022).

Encarnação, L. M., Bimber, O., Schmalstieg, D., & Chandler, S. (1999). A translucent sketchpad for the virtual table exploring motion-based gesture recognition. *Computer Graphics Forum,* (September), 277–286.

Feiner, S., MacIntyre, B., & Seligmann, D. (1993). Knowledge-based augmented reality. *Communications of the ACM, 36*(7), 53–62.

Fitzmaurice, G. (1993). Situated information spaces and spa-

tially aware palmtop computers. *Communications of the ACM, 36*(7), 39–49.

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

Fuhrmann, A., Löffelmann, H., Schmalstieg, D., & Gervautz, M. (1998): Collaborative visualization in augmented reality. *IEEE Computer Graphics & Applications, 18*(4), 54–59.

Goble, J., Hinckley, K., Pausch, R., Snell, J., & Kassel, N. (1995). Two-handed spatial interface tools for neurosurgical planning. *IEEE Computer, 28*(7), 20–26.

Goethe, J. W. von (1808). *Faust* (D. Luke, Trans.). Oxford: Oxford University Press.

Guiard, Y. (1987). Assymetric division of labor in human skilled bimanual action: The kinematic chain as model. *Journal of Motor Behaviour, 19*(4), 486–517.

Hesina, G., Schmalstieg, D, Fuhrmann, A, & Purgathofer, W. (1999). Distributed open inventor: A practical approach to distributed 3D graphics. *Proceedings Virtual Reality Software and Technology (VRST'99),* 74–81.

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

Ishii, H., & Ullmer, B. (1997). Tangible bits: Towards seamless interfaces between people, bits and atoms. *Proceedings ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '97),* 234–241.

Kato, H., & Billinghurst, M. (1999). Marker tracking and HMD calibration for a video-based augmented reality conferencing system. *Proceedings International Workshop on Augmented Reality (IWAR'99),* 85–94.

Kato, H., Billinghurst, M., Poupyrev, I., Imamoto, K., & Tachibana, K. (2000). Virtual object manipulation on a table-top AR environment. *Proceedings International Symposium on Augmented Reality (ISAR'00),* 111–119.

Mine, M., Brooks F., Jr., & Sequin, C. (1997). Moving objects in space: Exploiting proprioception in virtual-environment interaction. *Proceedings SIGGRAPH '97,* 19–26.

Pausch, R., Burnette, T., Brockway, D., & Weiblen, M. (1995). Navigation and locomotion in virtual worlds via flight into hand-held miniatures. *Proceedings SIGGRAPH '95,* 399–401.

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

Raskar, R. Welch, G., & Fuchs, H. (1998). Spatially augmented reality. *Proceedings International Workshop on Augmented Reality (IWAR'98),* 63–72.

Reitmayr, G., & Schmalstieg, D. (2001a). An open software architecture for virtual reality interaction. *ACM Symposium on Virtual Reality Software and Technology 2001 (VRST 2001),* 47–54.

———. (2001b). Mobile collaborative augmented reality. *Proceedings International Symposium on Augmented Reality (ISAR '01),* 114–123.

Rekimoto, J. (1997). Pick-and-drop: A direct manipulation technique for multiple computer environments. *Proceedings ACM User Interface Software and Technology (UIST'97),* 31–39.

Rekimoto, J., Ayatsuka, Y., & Hayashi, K. (1998). Augmentable reality: Situated communication through physical and digital spaces. *Proceedings International Symposium on Wearable Computers (ISWC'98),* 68–75.

Rekimoto, J., & Nagao, K. (1995). The world through the computer: Computer augmented interaction with real world environments. *Proceedings of ACM User Interface Software and Technology (UIST'95),* 29–36.

Rekimoto J., & Saitoh, M. (1999). Augmented surfaces: A spatially continuous workspace for hybrid computing environments. *Proceedings ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'99),* 378–385.

Schmalstieg, D., Fuhrmann, A., Szalavári, Zs., & Gervautz, M. (1996). Studierstube-Collaborative augmented reality. *Proceedings Collaborative Virtual Environments (CVE'96).*

Schmalstieg, D., Encarnação, L. M., & Szalavári, Zs. (1999). Using transparent props for interaction with the virtual table. *Proceedings ACM SIGGRAPH Symposium on Interactive 3D Graphics (I3D'99),* 147–154.

Schmalstieg, D., Fuhrmann, A., & Hesina, G. (2000). Bridging multiple user interface dimensions with augmented reality. *Proceedings International Symposium on Augmented Reality (ISAR'00),* 20–30.

Shaw, C., Green, M., Liang, J., & Sun, Y. (1993). Decoupled simulation in virtual reality with the MR toolkit. *ACM Transactions on Information Systems, 11*(3), 287–317.

Smith, G., & Mariani, J. (1997). Using subjective views to enhance 3D applications. *Proceedings ACM Virtual Reality Software and Technology (VRST '97),* 139–146.

Stoev, S., Schmalstieg, D., & Strasser, W. (2001). Through-the-lens techniques for remote object manipulation, motion

and navigation in virtual environments. *Proceedings of the Joint Immersive Projection Technology / EUROGRAPHICS Workshop on Virtual Environments (IPT/EGVE 2001),* 51–60.

Strauss, P., & Carey, R. (1992). An object oriented 3D graphics toolkit. *Proceedings SIGGRAPH '92,* 341–347.

Szalavári, Zs., Eckstein, E., & Gervautz, M. (1998). Collaborative gaming in augmented reality. *Proceedings ACM Virtual Reality Software and Technology (VRST'98),* 195–204.

Szalavári, Zs., Fuhrmann A., Schmalstieg, D., & Gervautz, M. (1998). Studierstube: An environment for collaboration in augmented reality. *Virtual Reality Systems, Development and Applications, 3*(1), 37–49.

Szalavári, Zs., & Gervautz, M. (1997). The Personal Interaction Panel: A two-handed interface for augmented reality. *Computer Graphics Forum 16*(3), 335–346.

Tsao, J., & Lumsden, C. (1997). CRYSTAL: Building multicontext virtual environments. *Presence: Teleoperators and Virtual Environments, 6*(1), 57–72.

Ullmer, B., & Ishii, H. (1997). The metaDESK: Models and prototypes for tangible user interfaces. *Proceedings of ACM User Interface Software and Technology (UIST'97),* 223–232.

Ullmer, B., Ishii, H., & Glas, D. (1998). mediaBlocks: Physical containers, transports, and controls for online media. *Proceedings SIGGRAPH '98,* 379–386.

Underkoffler, J., Ullmer, B., & Ishii, H. (1999). Emancipated pixels: Real-world graphics in the luminous room. *Proceedings SIGGRAPH 1999,* 385–392.

Weiser, M. (1991). The computer for the twenty-first century. *Scientific American, 265*(3) 94–104.

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

Wloka, M., & Greenfield, E. (1995). The virtual tricoder: A uniform interface for virtual reality. *Proceedings of ACM User Interface Software and Technology (UIST'95),* 39–40.

Wohlfahrter, W., Encarnação, L. M., & Schmalstieg, D. (2000). Interactive volume exploration on the StudyDesk. *Proceedings of the 4th International Projection Technology Workshop (IPT'00).*