Habilitationsschrift

Collaborative Augmented Reality

eingereicht an der Technischen Universität Wien
Technisch-Naturwissenschaftliche Fakultät

von

Dipl.-Ing. Dr. techn. Dieter Schmalstieg

Wien, Februar 2001 Dieter Schmalstieg

Inhalt

Daß ich erkenne, was die Welt
Im Innersten zusammenhält,
Schau alle Wirkenskraft und Samen,
Und tu nicht mehr in Worten kramen.

Johann Wolfgang von Goethe, *Faust*

Einleitung

Virtual Reality (Virtuelle Realität) beschreibt eine Klasse von Benutzerschnittstellen, die durch Einsatz von dreidimensionaler, oft stereoskopischer Computergraphik den Benutzer in eine möglichst überzeugende künstliche Umgebung eintauchen läßt. Solche Systeme übertreffen aufgrund ihrer 3D-Darstellung zwar herkömmmliche Präsentationsmittel, schirmen den Benutzer aber vollständig von der Außenwelt ab. Im Gegensatz dazu erlaubt *Augmented Reality* einen Kombination von Realität und virtueller Umgebung. Erreicht wird dies durch den Einsatz von halbdurchlässigen Datenbrillen, Videomischung oder speziellen Projektionstechniken.

Auf dieser Technik basiert die *Studierstube*, eine kollaborative computerunterstützte Arbeitsumgebung, die seit 1996 an der Technischen Universität Wien – ursprünglich unter der Leitung von Prof. Dr. Michael Gervautz, später unter meiner Leitung - entwickelt und weiterentwickelt wird. Namensgebend für das Projekt ist jener Raum, in dem Goethes Faust seinem Drang nach Wissen und Erkenntnis nachgeht.

In der Studierstube steht der kollaborative Aspekt im Vordergrund: Mehrere Benutzer können gemeinsam virtuelle Objekte betrachten. Interaktionen eines Benutzers mit den virtellen Objekten können von den anderen Benutzern beobachtet werden, während sie gleichzeitig den Erläuterungen des Kollegen in der realen Welt folgen können. Anders als bei Virtual Reality ist die natürliche Kommunikation – Sprache, Gestik, Mimik – weitgehend unbehindert.

Die Artikel in dieser Sammlung beleuchten einzelne Forschungsaspekte des Studierstube-Systems im Detail und geben gleichzeitig einen Abriß der Entwicklungsgeschichte des Projekts. Schwerpunkte sind zum einen technische Problemstellungen, wie Hardware- und Softwarekonzept, Darstellungs- und Netzwerktechnologie, zum anderen Probleme des Benutzerschnittstellen-Design, wie dreidimensionale Interaktion, Visualisierung und Kollaboration.

Im Forschungsprojekt Studierstube haben eine Reihe von Forschern in verschiedenen Rollen zusammengearbeitet. Diese Zusammenarbeit, die in zahlreiche gemeinsame

Publikationen der beteiligten Forscher gemündet hat, ist prinzipiell sehr wünschenswert. Für eine Habilitation ist jedoch die Ausweisung der Einzelforschungsleistung verlangt. Es wurden deswegen aus den zahlreichen Publikationen (über 20!), die – überwiegend unter meiner Beteiligung – im Zusammenhang mit der Studierstube-Forschung entstanden sind, jene ausgewählt, bei denen der überwiegende Anteil der Arbeit von mir selbst geleistet wurde, was sich auch traditionsgemäß in der Rolle des ersten Autors ausdrückt. Es folgt eine Kurzbesprechung der Artikel unter Berücksichtigung der Arbeitsaufteilung.

1. The Studierstube Augmented Reality Project

Dieser Artikel, zuletzt geschrieben, gibt einen Überblick über das ganze Projekt und dient somit als Einleitung. Besonderer Wert wird gelegt auf den Zusammenhang der einzelnen Bereiche und die zugrundeliegende Philosophie zur Gestaltung von Benutzerschnittstellen. Im Brennpunkt steht die These, daß sich Augmented Reality als Kerntechnologie zur Entwicklung einer Interaktionsumgebung eignet, welche ähnlich produktives Arbeiten in 3D gestattet wie die Desktop-Metapher in 2D. Diese Definition umfaßt einerseits Modell- und Informationsmanipulation, andererseits Kommunikation und Computer Supported Cooperative Work.

Die in diesem Artikel beschriebenen Forschungsarbeiten habe ich zwischen 1998 und 2000 geleitet und koordiniert. Der Schwerpunkt liegt auf Konzepten, die ich selbst, oft in Diskussionen mit Dr. Fuhrmann, entwickelt habe, wobei die technische Realisierung auf viele Personen aufgeteilt war, insbesondere von mir betreute Studierende. Den Text habe ich selbständig verfaßt.

2. "Studierstube" - An Environment for Collaboration in Augmented Reality

Dieser Artikel beschreibt die erste Phase der Versuche im Bereich Collaborative Augmented Reality. Neben einer allgemeinen Untersuchung der Eigenschaften und Möglichkeiten von Collaborative Augmented Reality wird der erste einfache Prototyp eines Augmented Reality-Systems für mehrere Benutzer beschrieben, welcher auch das Personal Interaction Panel, ein zweihändiges Eingabegerät basierend auf einer “Pen & Paper”-Metapher, umfaßt.

An den in diesem Artikel beschriebenen ersten Projektschritten habe ich noch unter der Leitung von Prof. Gervautz mitgewirkt. Meine Leistungen waren hier neben der Mitentwicklung der Konzepte noch verstärkt konkrete Implementierungsarbeiten, aber auch die Abfassung des ersten “extended abstract” mit dem gleichen Titel, das Ende 1996 mit mir als erstem Autor im Tagungsband der Konferenz “Collaborative Virtual Environments ‘96” in Nottingham, UK, erschien. Später wurden wir eingeladen, einen auf dieser Veröffentlichung basierenden Artikel für das “Virtual Reality”-Journal zu schreiben, der hier wiedergegeben ist. Die Endfassung dieses Artikels übernahm damals Dr. Szalavári.

3. Using Transparent Props For Interaction With The Virtual Table

In einer Zusammenarbeit mit dem Fraunhofer Center for Research in Computer Graphics in Providence, Rhode Island, USA, entstand eine Version der Studierstube-Software für den Virtual Table, ein Virtual Reality-System mit Rückprojektions-Darstellung. Besonderes Augenmerk wurde auf eine adaptierte Version des Personal Interaction Panel gelegt, die durch die Verwendung von transparentem Material mit der Rückprojektion verträglich ist. Es werden umfangreiche Benutzerschnittstellen-Studien mit diesem Eingabegerät beschrieben.

Dieser Text markiert für mich den Beginn der zweiten Projektphase, in der ich die Leitung der Studierstube-Forschung übernahm und wir die behandelten Themen über den ursprünglichen Kernbereich "Collaborative Augmented Reality" hinaus ausdehnten. Die beschriebenen Arbeiten habe ich im Sommer 1998 alleine durchgeführt. Dr. Encarnação ist dafür zu danken, daß er diese Arbeit ermöglicht hat, Dr. Szalavári für seine Dissertation über das "Personal Interaction Panel", auf die ich vom Konzept her aufgebaut habe.

4. Bridging Multiple User Interface Dimensions with Augmented Reality

Diese Arbeit beschäftigt sich mit der zweiten Generation des Studierstube-Systems, welche gegenüber der ersten Generation konzeptuell und technisch stark erweitert ist und insbesondere multimediale und multimodale Techniken betont. Beschrieben wird ein Baukasten zur Erstellung dreidimensionaler Benutzerschnittstellen für mehrere Benutzer, der auf einem verteilten graphischen System basiert und gleichzeitige Verwendung von verschiedenen Formen der Eingabe (Tracking) und Ausgabe (Displaytechnologie) sowie das Multitasking dreidimensionaler Applikationen erlaubt. Mit diesem System lassen sich Ideen aus Augmented Reality und Ubiquitous Computing, also die Verwendung vielfältiger, in die Arbeitsumgebung eingebetteter Informationstechnik, kombinieren.

Dieses "Systems Paper" beschreibt ein komplexes Software-System, zu dessen Realisierung neben mir auch Dr. Fuhrmann und DI Hesina wesentlich beigetragen haben. Die Beschreibung der zugrundeliegenden Konzepte und insgesamt die schriftliche Arbeit stammt jedoch ausschließlich von mir.

5. Sewing Virtual Worlds Together With SEAMS:
A Mechanism to Construct Complex Virtual Environments

Dieser Aufsatz beschreibt ein allgemeines Konzept zur Verschachtelung dreidimensionaler Szenen basierend auf der Idee eines "Portals" bzw. "Wurmlochs". Diese Aufgabe wird sowohl in Hinblick auf ihre theoretische Implikation zur Konstruktion von Benutzerschnittstellen als auch auf ihre praktische Umsetzung (Darstellung, Vernetzung) untersucht. Ursprünglich als unabhängige Arbeit begonnen, wurden die Ergebnisse später als Benutzerschnittstellen-Element in die Studierstube integriert.

Während die ursprüngliche Idee zu den SEAMS und eine zugehörige Machbarkeitsstudie von Dr. Schaufler stammt, trage ich die Verantwortung für die Weiterentwicklung der Idee, ihre Implementierung im Rahmen des Studierstube-Projekts, ihre Anwendung als Benutzerschnittstellen-Element für Augmented Reality und schließlich die Abfassung des Artikels.

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).

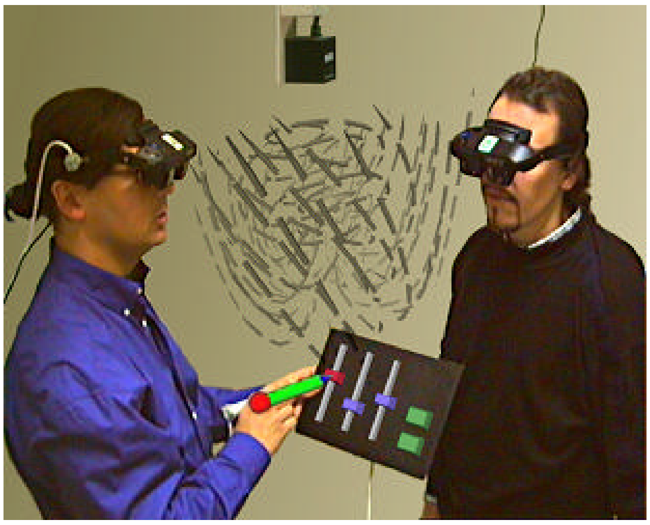


Figure 1: Two collaborators wearing see-through displays are examining a flow visualization data set

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.

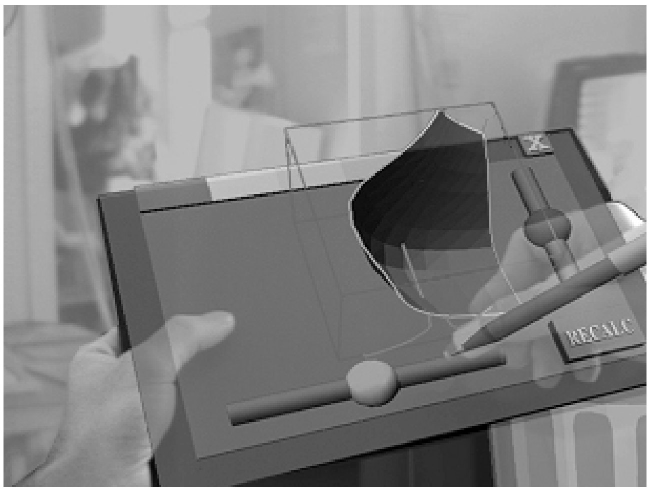


Figure 2: The Personal Interaction Panel allows two-handed interaction with 2D and 3D widgets in augmented reality

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.

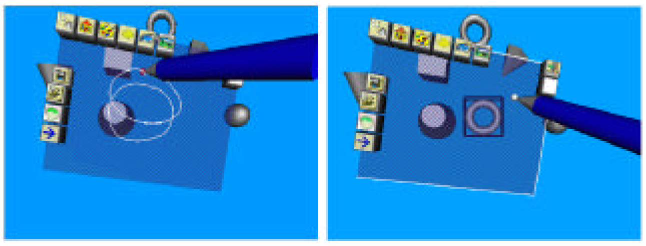


Figure 3: A gesture is used to create a torus in CADesk

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).

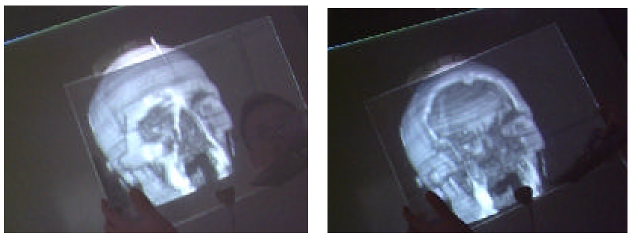


Figure 4: The panel is used to position a clipping plane that cuts away a portion from the volumetric scan of a human skull

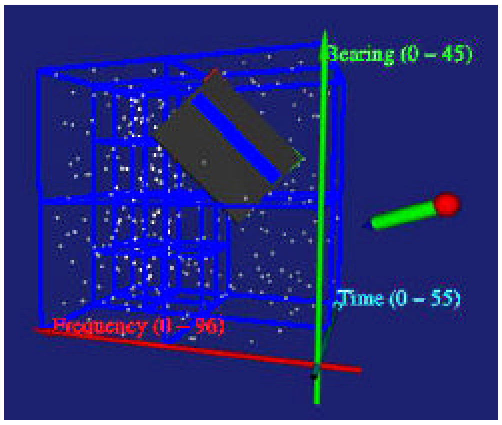


Figure 5: The panel is swept through an aggregation of particle data. During the sweep, a filter is applied to the underlying raw data, which produces aural feedback that can assist the user in detecting structures in the data sets that are not visible to the human eye.

- **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).

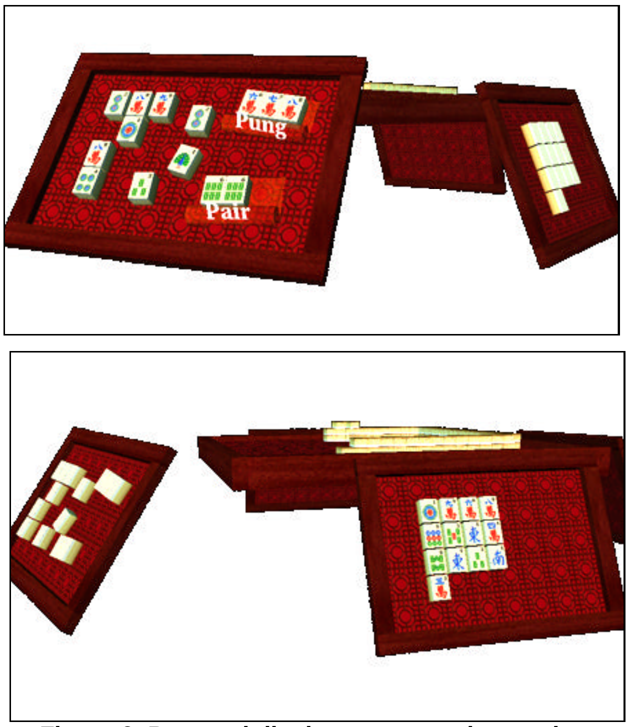


Figure 6: Personal displays secure privacy when playing Mahjongg – the left player (top view) cannot see his opponent's tile labels and vice versa (bottom view)

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).

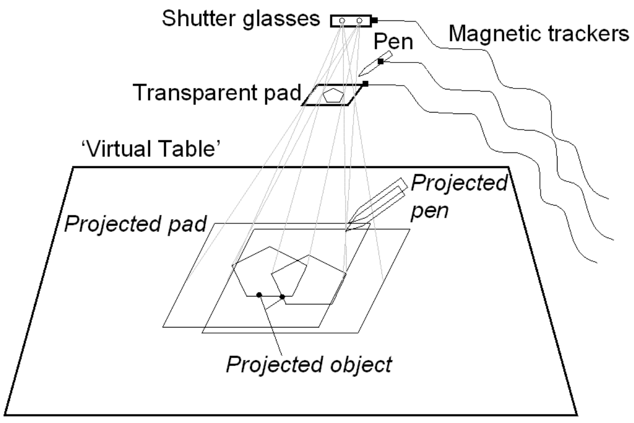


Figure 7: The Personal Interaction Panel combines tactile feedback from physical props with overlaid graphics to form a two-handed general-purpose interaction tool for the Virtual Table.

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.

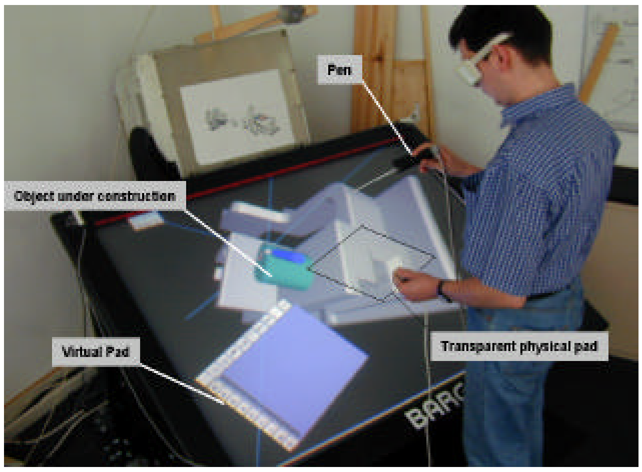


Figure 8: Transparent pen and pad for the Virtual Table are almost invisible and replaced by computer graphics in the user's perception (Stork & de Amicis, 2000)

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.

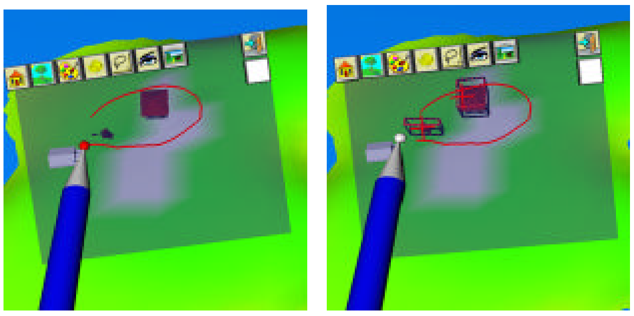


Figure 9: The lasso tool allows users to select objects in 3D by sweeping an outline in 2D on the pad. All objects whose 2D projection from the current viewpoint is contained in the outline are selected.

- **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.

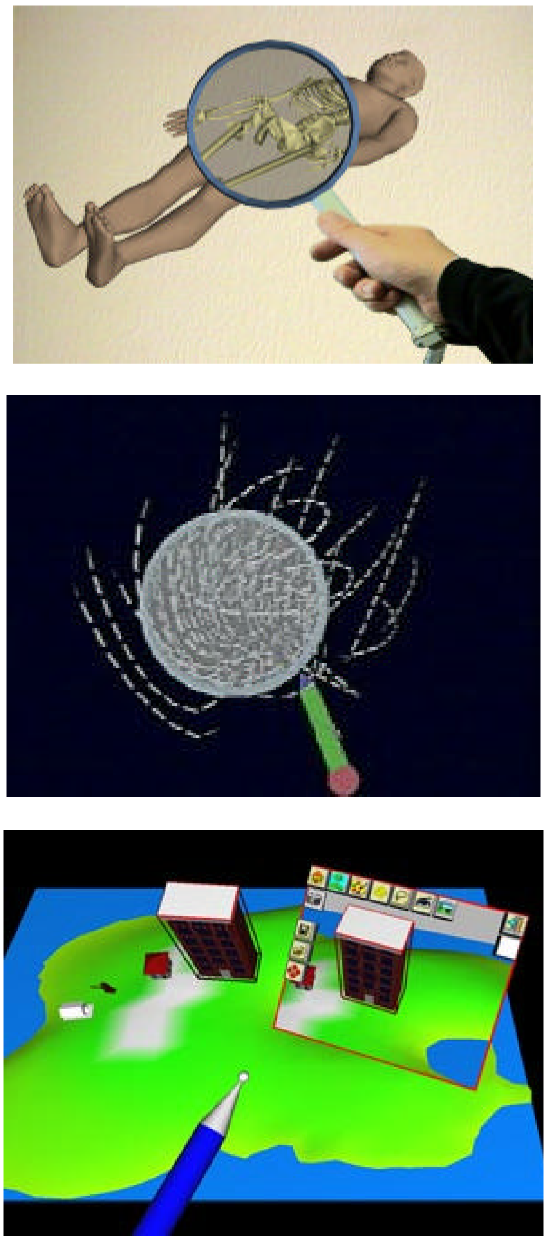


Figure 10: Different applications of through-the-window tools: (top) X-ray lens, (middle) focus lens that locally increases density of streamlines in a flow visualization, (bottom) portal to a different version of a scene

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 *ubiquitous computing* and the *desktop metaphor*.

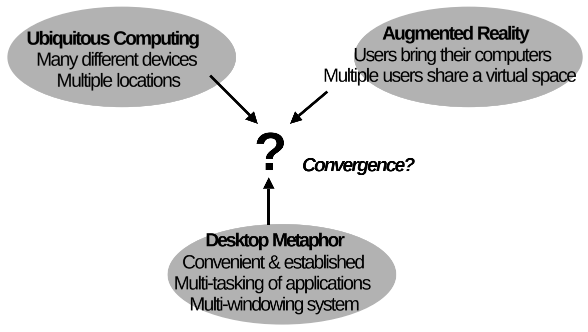


Figure 11: The latest *Studierstube* platform combines the best elements from augmented reality, ubiquitous computing, and the desktop metaphor

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-dimensional 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.

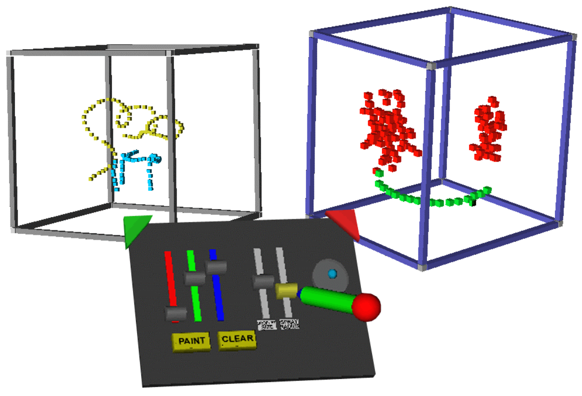


Figure 12: Multiple document interface in 3D – the right window has the user's focus – indicated by the dark window frame – and can be manipulated with the control elements on the PIP.

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).

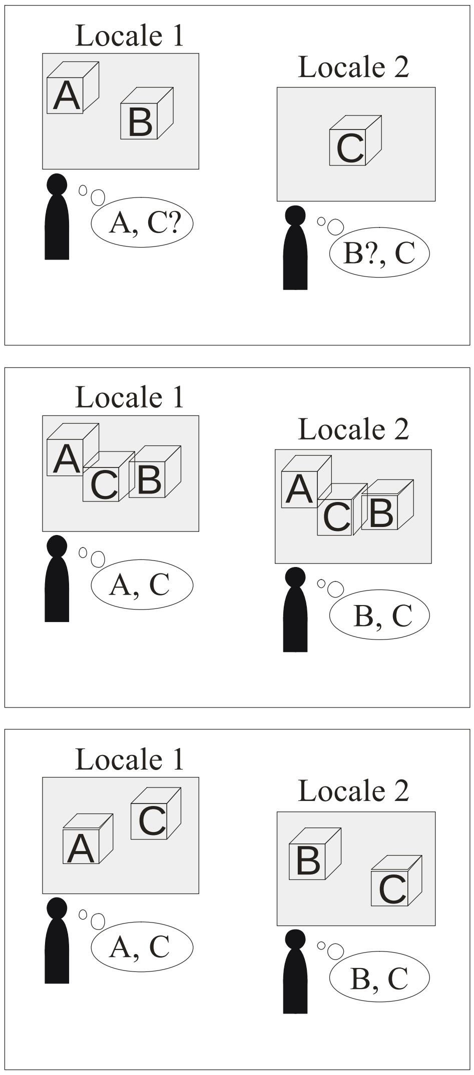


Figure 13: (top) A global arrangement of items cannot fulfill all needs. (middle) Full replication of all items leads to display clutter. (bottom) On-demand replication of items allows convenient customization of locales.

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.

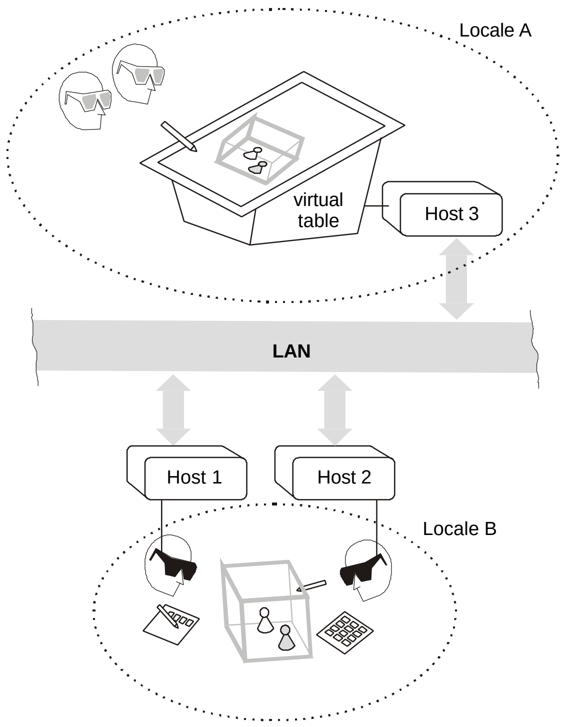


Figure 14: Multiple locales can simultaneously exist in *Studierstube*. They can be used to configure different output devices and to support remote collaboration.

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).

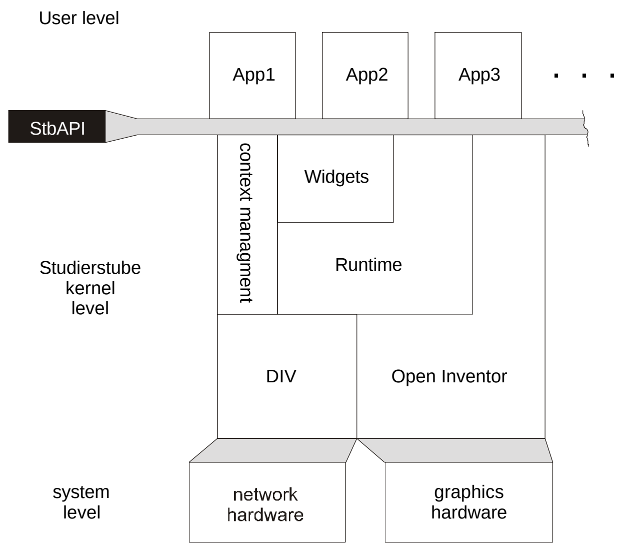


Figure 15: The *Studierstube* software is composed of an interaction toolkit and runtime system. The latter is responsible for managing context and distribution.

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-dimensional 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).

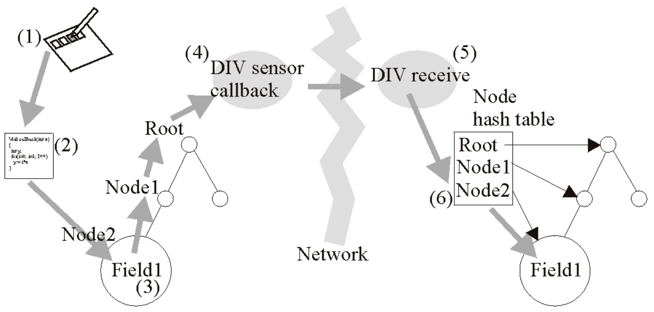


Figure 16: Example of a field update in a master-slave configuration. (1) User triggers an action by pressing a button. (2) Corresponding callback is executed and modified field1 of node2. (3) Event notification is propagated upwards in scene graph and observed by sensor. (4) Sensor transmits message 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.

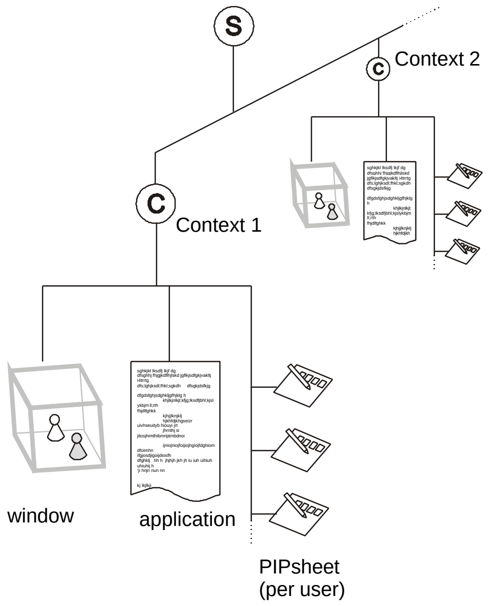


Figure 17: A context is implemented as a node in the scene graph, as are windows and PIP sheets. This allows for the organization of all relevant data in the system in a single hierarchical data structure.

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.

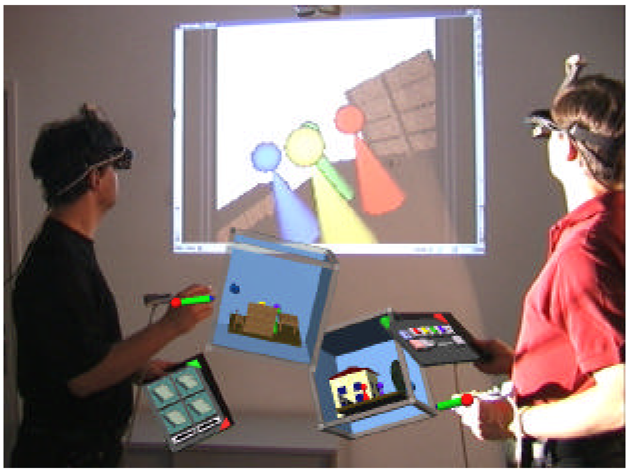


Figure 18: Storyboard application with two users and two contexts as seen from a third "virtual" user perspective, used for video documentation. In the background the video projection is visible.

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.

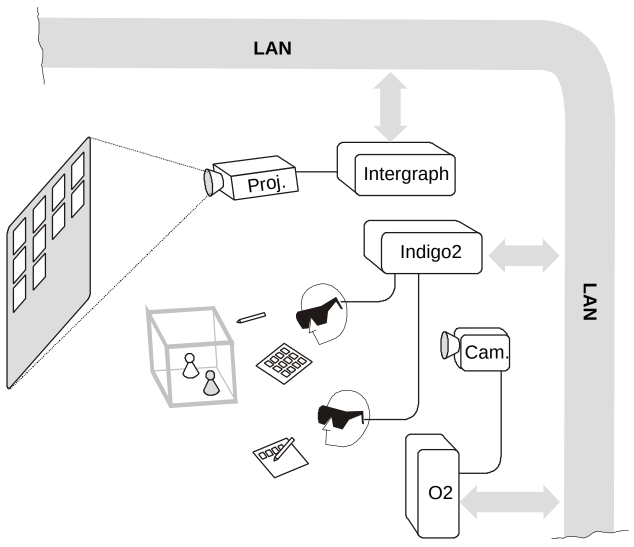


Figure 19: Heterogeneous displays—two users simultaneously see shared graphics (via their see-through HMDs) and a large screen projection.

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.

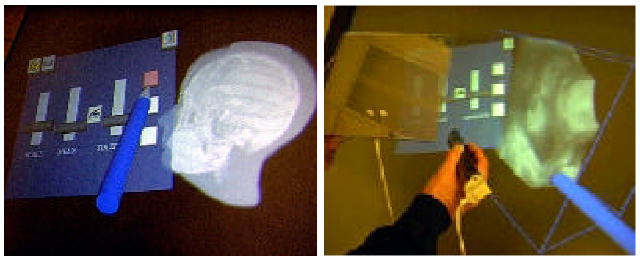


Figure 20: MediDesk allows interactive manipulation of volumetric data, such as CT scans.

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.

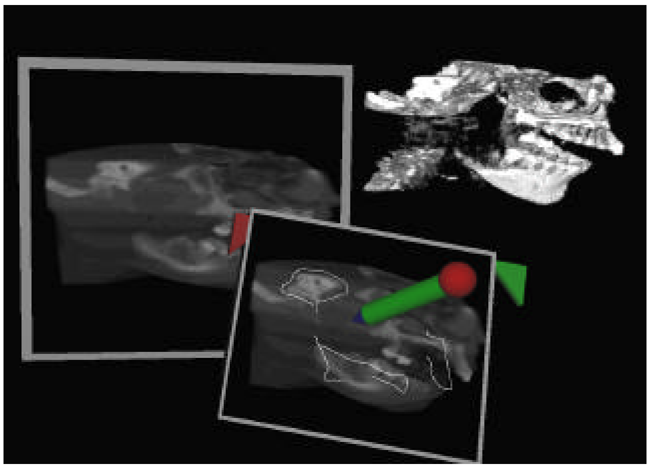


Figure 21: The lower half of the image shows the annotating of a virtual "X-ray" print taken from the volume on the upper right.

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.

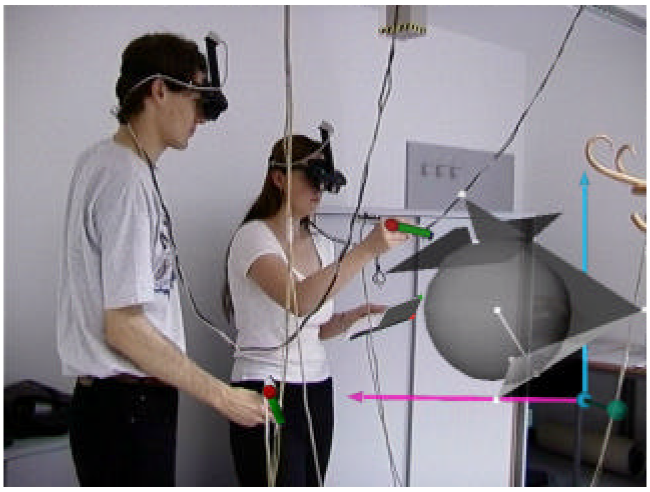


Figure 22: A tutor teaches a student how to geometrically construct 3D entities with Construct3D.

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 10th 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.

“Studierstube”

An Environment for Collaboration in Augmented Reality

Zsolt Szalavári, Dieter Schmalstieg, Anton Fuhrmann, Michael Gervautz

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

Abstract: We propose an architecture for multi-user augmented reality with applications in visualization, presentation and education, which we call “Studierstube”. Our system presents three-dimensional stereoscopic graphics simultaneously to group of users wearing light weight see-through head mounted displays. The displays do not affect natural communication and interaction, making working together very effective. Users see the same spatially aligned model, but can independently control their viewpoint and different layers of the data to be displayed. Major field of application serves computer supported cooperative work and enhances cooperation of experts. The paper presents the client server software architecture underlying this system and details that must be addressed to create a high-quality augmented reality setup.

Keywords: augmented reality, multi-user applications, collaboration, distributed graphics

1. Introduction

Daß ich erkenne, was die Welt
Im Innersten zusammenhält,
Schau alle Wirkenskraft und Samen,
Und tu nicht mehr in Worten kramen.

To realize what holds the world
Together in its core,
I see all seeds and force of act
And search for words no more.

Johann Wolfgang von Goethe, Faust

We selected the project name “Studierstube”, after the play *Faust* by Johann Wolfgang von Goethe, in which the leading character uses a study room for performing research and philosophy: the *Studierstube*.

This paper deals with an attempt to combine two very important recently developed evolving fields:

- The potential of combining of everyday's reality with the power of computer has gone through a dramatic evolution in the last years. The method for visual improvement or enrichment of the surrounding environment by overlaying spatially aligned computer-generated information onto a human's view, called *Augmented Reality* (AR), has potential for a broad range of applications, including mobile context-sensitive information systems, scientific visualization, in-place display of measurement data, medicine and surgical planning, education, training and entertainment.
- The primary goal to provide insight into a complicated problem by the enrichment of simulation data, that is mapped and rendered to a displayable image (Figure 1) has become important to numerous fields of science outside of computer graphics and augmented reality. Scientific visualization tries in recent time on projects of higher and higher complexity.

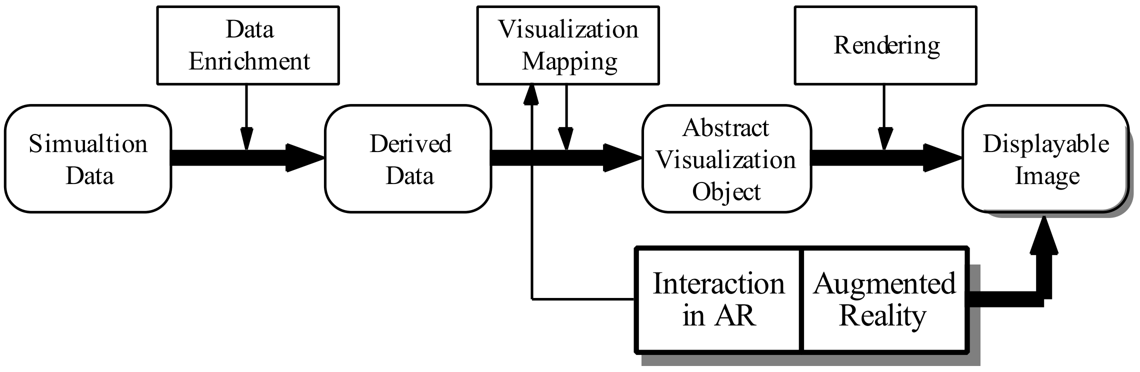


Figure 1: Visalization pipeline (Nielsen, 1990)

As a highly interdisciplinary field, scientific visualization frequently requires experts with different background to cooperate. Collaborators may have different preferences concerning the chosen visual representation of the data, or they may be interested in different aspects. An efficient collaboration requires that each of the researchers has a customized view of the data set. At the same time, presence in the same room is preferred because of the natural interaction during a discussion. These requirements can uniquely be fulfilled in an augmented reality system which combines real world experience of the collaborators and physical equipment with the visualisation of the synthetic data.

Compared to visualization in immersive virtual reality, augmented reality allows the use of detailed physical models, the properties of which cannot be met by their virtual counterparts: arbitrarily detailed visual representation, no visual or temporal artifacts and force-feedback for free. Only those aspects of the model that cannot be seen in reality have to be added by the computer system: For example, one could take the physical model of an airplane or airplane wing to investigate the flow around this object, which is simulated by computer and added to the display. Manipulation of the real world model (e. g. its orientation) is more intuitive and simpler to support than a purely virtual environment. A related example would be the use of a humanoid torso or puppet that is overlaid with medical information from inside the human body in the style of (Bajura, 1992).

This combination of conventional experimental work with scientific visualization and augmented reality technology leads to the concept of an augmented laboratory, which would provide a superior research environment in which to conduct experiments that are executed solely inside the computer, while maintaining a conventional and familiar work setup.

The “Studierstube” approach concentrates on the seamless combination of a physical world workspace and an augmented environment for multiple users in three dimensions, with unaffected social communication channels and an augmented user interface that supports natural handling of complex data at interactive rates. In this type of distributed multi-user systems adequate communication strategies for continuous synchronization and real-time performance are required that also allow the interaction with a shared geometric database.

2. Related Work

The evolution of augmented reality started in the early days of computer graphics, Sutherland pioneered research on head mounted displays (Sutherland, 1968). His work still inspires the virtual reality research community of today. Although only capable of simple vector drawings, his prototype head mounted display was the first binocular see-through system, effectively the first augmented reality system. Feiner et al. (Feiner, 1992) (Feiner, 1993) described a knowledge based augmented reality system. As a demonstration, they chose to configure the system to support people with the maintenance of laser printers. However, a lot of effort is required to generate accurate models, and extremely precise registration is required. Bajura et.al. (Bajura, 1992) described a medical visualization system based on augmented reality techniques. A see-through head mounted display (HMD), also developed at UNC (Holmgren, 1992), allows geometrically correct superposition of ultrasound data of the unborn onto the belly of the mother-to-be, so the gynecologist can examine the position of the unborn within the mother. Another medical application of AR has been presented by State et. al. (State, 1996) for ultrasound guided needle biopsy of the breast. Sharma and Molineros (Sharma, 1996) prensent a system for mechanical assembly guidance using annotations attached to real world scenery.

Scientific visualisation in virtual reality becomes increasingly a field of interest for many researchers. In the early 90ies at UNC within the GROPE project a group around Fred Brooks produced a haptic arm-like device and a large stereo display for the visualization and manipulation of chemical data (Brooks, 1990). Their nanomanipulator (Taylor, 1993) allows precise manipulation of a scanning tunneling microscope and works also with force feedback. Another important milestone for the combination of VR and Scientific Visualization was the development of the virtual wind tunnel at NASA-AMES by Steve Bryson. Using a BOOM device and a data glove as interaction tool (Bryson, 1991), scientists were able to see and interact with true stereoscopic images of a flow field visualization. A follow-up project, the distributed windtunnel (Bryson, 1993) was developed, which divided computation in a distributed system for better efficiency, and allowed multiple users to experience the simulation at the same time. Collaboration in a distributed virtual environment, not limited to scientific visualization has been proposed by Fahlén et. al. (Fahlén, 1993).

Not considering distribution of virtual environments on large scale, we can state that most existing augmented applications are single user setups, or do not exploit the multi-user character of their systems. The CAVE-System (Cruz-Neira, 1993 a) (Cruz-Neira, 1993 b) and the responsive workbench (Krüger, 1995) are the most prominent examples. The CAVE uses to view. In the CAVE users see stereoscopic 3-D scenes with LCD-shutter glasses on large projection walls surrounding them. One user is head-tracked, so that the images on all walls correspond to this viewer's position. The viewers have the impression to be surrounded by 3-D virtual scene. A disadvantage of this system is that the presented images fit to the head position only for one viewer; noticeable visual artifacts exist for all other viewers. The responsive workbench uses one display area, which is built in a table top. Like in the CAVE,viewers wear LCD shutter glasses and only one user can see the objects in correct stereoscopy. Furthermore, a relatively steep viewing angle is necessary to achieve a good 3D impression, i.e. the viewers have to stay close to the table.

3. The "Studierstube" approach

We propose a system capable of visualization of three-dimensional scientific data for multiple simultaneous viewers within one room. The choice of this setting limits the complexity of the problem, as the "real world" is limited to a room, but the "virtual world". Each viewer wears see-through HMDs providing a stereoscopic real-time display, and can freely walk around in order to observe the augmented environment from different viewpoints. The coarse determination of each user's viewpoint is provided by magnetic head-tracking and refined by optical tracking.

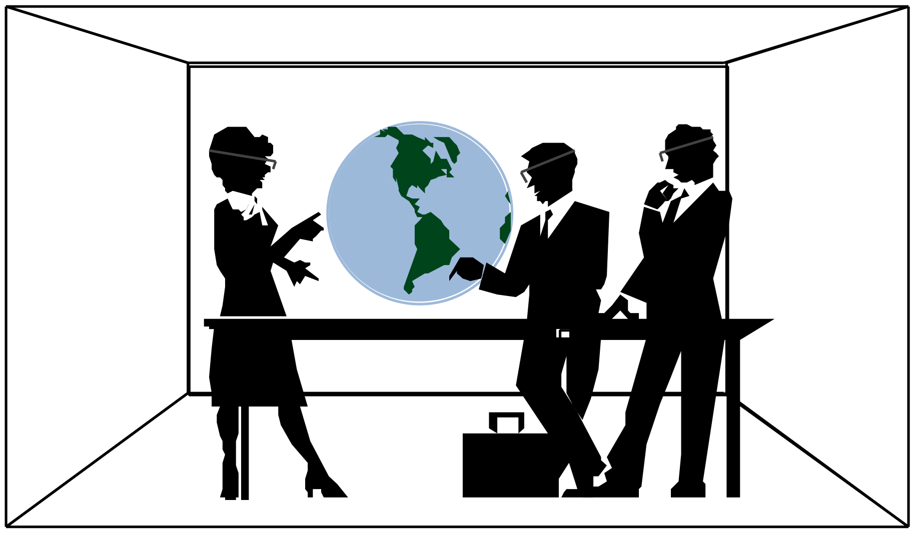


Figure 2. Three people wearing see-trough glasses at a meeting, viewing a virtual globe. Note that the table is an object in the real world, the globe just an image projected into the space by the head-set.

The mixture between real and virtual visual experience, created in our system by see-trough HMDs, is a key feature of our system. Thus, it is possible to move around freely without fear to bump into obstacles, as opposed to fully immersive displays, where only virtual objects can be perceived. This enables a work group to discuss the viewed object, because the participants are seeing one another and can therefore communicate in the usual way.

We exploit the possibilities of augmented reality with the methods of data layers, spatially linked annotations and mobile tracked objects on the example of visualization of three-dimensional data sets, as described in the later sections.

Interaction with the augmented part of the scenery is maintained using high-level interaction metaphors and tools, like the Personal Interaction Panel (PIP) (Szalavári, 1996). We incorporate this new two-handed input device that supports a multitude of interaction styles and is particularly well suited for augmented reality applications. It unifies general control functions of Studierstube, usual 3D manipulation tasks, as well as application specific interaction methods. The PIP is composed of position and orientation tracked lightweight, notebook sized hand-held panel and a pen and carries instant augmented elements for interaction.

3.1 Properties of our system

The following key properties summarize the attributes of our system:

Virtuality

Viewing and examining of objects that are not accessible directly or that do not exist in the real world can be carried out in this environment. Investigation of datasets using information visualization becomes a task of handling almost “real” objects. Size, complexity, physical properties are just parameters in a simulatation, no longer are they constraints for the analysis.

Augmentation

Real-world objects can be augmented with spatially aligned information. This allows smooth extension of real objects with virtual properties in design processes, like variations of new parts for an existing system. Superimposed information can also incorporate enhancing elements for real objects, like descriptions or advices in training or education situations, which we call annotations.

Multi-user support

A situation where multiple users congregate to discuss, design, or perform other types of joint work is generally categorized as CSCW (computer supported cooperate work). Much work has been devoted to the question how conventional software and desktop computers can be enhanced with measures to support effective group interaction. Fortunately, a benefit of augmented reality is that sophisticated groupware mechanisms are not really needed to perform real work. Normal human interactions (verbal, gestures, etc.) are easily possible in an augmented reality setup, and they are probably richer than any computer-governed interaction can ever be.

Independence

Unlike the CAVE and the Workbench, control is not limited to a guiding person, while other users act as passiv observers. Each user has the option to move freely and independently of the other users. In particular, each user may freely choose a viewpoint with stereoscopy for correct depth perception. But not only observation is independent, also interaction can be performed on a personal base. The semi-immersive character of our augmentation helps to keep human communication channels unblocked, improving the quality of collaboration.

Sharing vs. Individuality

Investigated objects are in general shared among users, in the sense of visibility, this means that all participants can see the same coherent model, consistent in its state over time. By presenting the visual sensation directly to each user with the lightweight see-trough HMDs, the displayed data set can also be different for each viewer, as required by the application's needs and the individual's choice. Personal preferences on different layers of information can be switched on and off, as described in the next sub-section.

Interaction and Interactivity

With the support of augmented tools like the proposed PIP, visualized data can be explored interactively. Changes inherent in the scientific simulation can be viewed immediately. The visual components of the panel in one users hand can be kept private, invisible for other users, or public, sharing even 3D information by direct visibility or projection to projection walls, as described in the next section.

3.2 Augmented features

We incorporated layers and annotations as augmented features to our system, and introduce an information sharing surcfaces - the projection wall. Furthermore we show uses of mobile tracked objects in an augmented environment.

Layers

We incorporate layers similar in concept to the ones found in technical illustration programs or CAD packages and the work of Fritzmaurice (Fritzmaurice, 1993). Data is separated into disjoint sets according to semantic considerations (e.g. floor plan with walls only - furniture - measurements). Display can be turned on and off for every layer individually. This concept is fundamental for allowing individuals to customize the display to their needs. Users may see the same model and at the same time *not* see the same model, as everyone sees a different set of aspects of the same thing. Aside from personal taste and interest, this is useful if professional people (e.g. an architect) talk to inexperienced people (e.g. customer), or if people with different interest (e.g. designer and engineer) collaborate.

Annotations

Augmentation is not necessarily limited to 3D graphics added to the physical world. General multi-media data can be useful (e.g. sound cues), but what we consider absolutely essential to support are textual annotations. While it is often true that illustrations and graphics make difficult concepts clearer than textual explanations can, for complicated models a legend that explains important parts and gives names is just as important. The system will provide a possibility to link text to specific 3D points of a model. The text is then displayed "in place", but in 2D overlaid onto the 3D image similar to (Rekimoto, 1995). As the user moves, his viewpoint, the text stays screen-aligned so that it is always clearly readable. The system takes care that multiple text elements do neither overlap nor occlude each other. By means of the layer mechanism, individual annotation sets can be switched on and off. The annotation concept will be especially useful if physical props (e.g. demonstration objects or mock-ups for education) are used, but it will also improve the quality of purely virtual presentations. Annotations can be created, edited and directly placed or moved in the augmentation with the Personal Interaction Panel. A three dimensional drag and drop mechanism gives a natural interactive feeling of handling spatially aligned multi-media data.

Projection walls

By allowing the users to keep individual items of data on their PIPs, sharing this data with other users becomes possible, similer to the “whiteboard” concept in (Fahlén, 1993). This can be done either directly by making modifications on the shared model itself, like placing public multi-media annotations in space, or by using projection walls. These walls are static, virtual objects in the Studierstube and are logically connected to a user’s PIP. Thereby, a single user interaction tool is transformed into a multi-user presentation space.

Tracked mobile objects

Static objects become part of the augmentation in a simple setup phase. Geometric properties such as size and position have to be registered for inclusion in an environment. To include a real world object completly in the system and an ongoing simulation, the system needs to have information about changes in position, orientation and state in addition to the static properties, so that they can interact with other parts of the augmentation.

For this reason we introduce tracked mobile objects, as functional part of our system, which can be moved, held in hand by users, passed on from user to user and so forth. Typically, the number of such objects will be small, but their role in the application will be significant. Main usage of mobile objects are manipulation tools such as the PIP, and physical models (mock-ups) that are augmented with supplementary information not physically available (e.g. isolines of stress). Technically, the position of these objects is determined by a dedicated tracking sensor, and a representation of the physical model is rendered in background color to resolve the occlusion problem among physical and virtual objects.

4. System Overview

We consider our system to work in a stationary environment, e.g. a room, so we can assume a sufficient network bandwith for communication between parts of our client-server approach, both for geometric and application data, as well as supporting information like tracker data. The representation of this data and communication concerning the changes, as well as interaction between users and system are crucial factors calling for detailed presentation.

4.1 Data representation and modeling

We propose three different kinds of 3D models, each for a different purpose:

Registration data

Registration data describes the geometry of the presentation room (walls, windows, doors etc.) along with features to be recognized and matched with images of the real world. Because this kind of data is completely static (does not change at all), it can be prepared for fast matching and the tracking correction algorithm. There is a trade-off between accuracy and time requirement which can be tuned by varying the complexity of the underlying model. Very simple models will be very fast in processing time but may fail for good registration. Complex models may lead to precise registration but the time required for that task may be large.

Data representing mobile objects

Within the our environment the system also supports mobile objects, which can have virtual data representing or supporting them. This type of data differs from static registration data, as it has to be update in real-time during operation.

Display data

Data presented or added to the environment is generally handled as display data. This data is shared between the Studierstube and the underlying simulation. Simualtion engines have to provide visual output in the same format, so that inclusion in the geometric database of the Studierstube is rather uncomplicated, but major changes to this database can still be controlled directly by the application.

For the purpose of our implementations we composed all geometry descriptions in the emerging public standard for 3-D geometry, Open Inventor (Strauss, 1992).

4.2 Client-Server Approach

We propose a software architecture for our augmented reality system, which is based on a client-server structure. A server holds a database of all data types, including registration, mobile object and display or application data. Users connect to the server via a network using client software. The client obtains a replica of the database from the server, which is used by the client locally to render the image presented to the user.

Except for special customizations, the view that concurrent users have of the scene (position, color of objects etc.) must be consistent. As there are multiple local copies of an object, if any change is made to the presented scene (e.g. color of an object changed), changes must be propagated to other replicas. This is done by sending a message to the server, which in turn distributes them to the other participants. As such update events only happen occasional (note that tracker data is handled separately!), the improved consistency outweighs the longer communication involving a server.

Tracker data is managed by a special tracker demon. The quality of tracking is crucial to the quality of the experience, so a separate machine is dedicated to the tracking. The tracker demon is continuously running, and clients can connect at will to obtain a stream of tracker data. Our system involves multiple tracked points (head tracking for multiple users, hand/pointer tracking, tracking of mobile objects). All the data from these tracked points influences the state of the scene and is therefore propagated to the connected clients as a bundle, which improves throughput and consistency of the data.

A simulation engine is required to provide the data for the scientific visualization task in our implementation. This data can be precomputed and loaded into the system at runtime. Simple visualizations such as analytical dynamical systems can be hand-coded. However, a capable simulation system is better suited to address the diverse needs of multiple visualisation tasks, and also eases development. In previous projects, we have used AVS (AVS, 1992) to create scientific visualization data. Its data flow concept allows export of the data in almost any desired format and lends itself naturally to an integration into our system architecture.

A loose coupling is defined between AVS as the computational back end and the visualization server that coordinates interaction with the model in the Studierstube. Visualization data is exported from AVS to the visualization server that takes care of distribution of the data among the Studierstube's clients. Computational steering is achieved by using special input modules for AVS that accept new values for simulation parameters from the Studierstube. If re-generation of the model or its parts with modified parameters is reasonably fast, real-time or near real-time steering can be achieved.

Modifications of the visualization data that do not involve the simulation (such as rotating the simulated model) can be carried out in a close loop by the Studierstube system alone and do not pass data between Studierstube and AVS. Such simple interactions are not affected by the performance penalty created by invoking a complex software system such as AVS and can therefore always be carried out with real-time response and high fidelity.

The proposed overall system architecture can be seen in Figure 3.

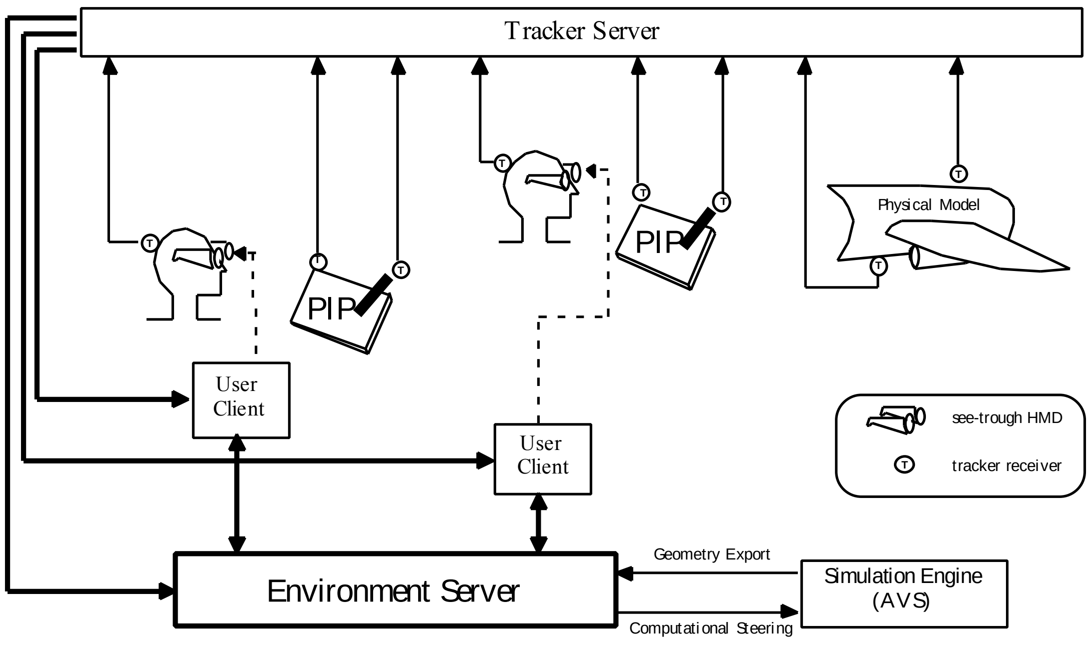


Figure 3. System architecture: The augmented reality environment is maintained by a server that takes care of the synchronisation needs of the clients and interoperates with the simulation backend. The clients are responsible for displaying the environment. A tracker server manages input devices.

The distribution and consistency of the shared geometric database plays a significant role in the quality of our system. To handle coherence and merge the interaction of multiple users and communication between server and simulation engine, we base the intercourse between all parts of our client-server environment on sophisticated protocols and message passing algorithms. We are currently developing an in-house communication standard for connection of three dimensional user interfaces to visualization.

4.3 Interaction tools

Interaction with the augmented part of the scene is performed with the Personal Interaction Panel is a unique tool that combines physical and virtual attributes: The physical nature of the pen and panel make it a very simple, yet effective and precise device for interaction, that supports tactile feedback and has good ergonomics. However, the surface of the panel is a virtually unlimited information display of computer generated (augmented) information.

There are many different possibilities to use the PIP as interaction tool in the augmented environment, we will show features for general tasks and in the application section those supporting our implementation of a scientific visualisation environment.

The pen alone can be used for any 3D pointing operation and direct manipulation, where a 3D mouse (6 degrees of freedom) is normally used. This feature is integrated with the extended PIP functionality, so that the PIP supports a superset of "standard" 3D operations in virtual and augmented reality.

A conventional 2D computer display can be projected onto the board, supporting a 2D desktop metaphor better than "flying menus" so traditional 2D user interaction and parameter manipulation is possible. In addition to "flat" 2D user interface elements, three-dimensional widgets that "float" above the panel's surface are supported (e.g., selection of a point on a sphere), clipboard funcionality and drag-and-drop in 3D can also be implemented.

Pen and panel can be used as video camera and control monitor, respectively: The direction of pointing of the pen orients a virtual camera, the resulting live image is shown on the PIP for immediate feedback or snapshots can be made by holding the panel against the virtual scene. The panel can also be used as a magnifying lens or filter in the style of (Bier, 1993) showing an enlarged or otherwise enhanced view (e.g. X-ray).

Multiple navigation metaphors are supported by two handed interaction as featured by the PIP: Among them are use of hand-held miniatures (compare (Pausch, 1995)) specifying direction of movement with the pen or "spaceship" control gadgets (2-D buttons or 3-D widgets) on the panel's surface.

The general controls for Studierstube can easily be made available by the PIP, so reconfiguration of the application can largely be achieved without leaving the augmented environment. For example, loading a new model can be done with a file selector presented on the PIP.

5. Implementation

Our current limited implementation of the described system above consists of an environment for two users. The hardware configuration includes i-Glasses head mounted see-trough displays and a Polhemus Fastrak tracking device connected to a tracker server PC. Tracker data is transmitted over Ethernet using TCP/IP protocols and multicast technology. Rendering is done on Silicon Graphics workstations (Maximum Impact graphics) using Open Inventor libraries. The hardware of the Personal Interaction Panel consists of a lightweight wooden panel and a plastic pointer, both tracked in position and orientation with Fastrak receivers. From our current implementation we can conclude, that for high-fidelity augmented reality, precise registration of the real world with the augmented display is crucial, and our current static registration is barely sufficient. Nevertheless, our experiences show that users feel comfortable and working in the environment is pleasant. Concerning the tracking problem enhancement of registration by hybrid tracking technology is currently under developed in cooperation with the Vision Group of the Graz University of Technology as part of a parallel project.

6. Applications

The “Studierstube” approach is powerful enough to support a wide range of different applications, like every kind of industrial, commercial or didactic presentation, scientific visualization and computer supported cooperative work (CSCW), allowing communication among participants in the most natural way, while displaying the subject of their work in front of them.

6.1 Applications in Scientific Visualization

As described above, we want to set the focus of our applications to scientific visualization. Augmented reality for scientific visualization can provide an intuitive, even transparent, interface for computational steering. Consequently, a test case is needed that is simple enough for interactive steering even on conventional workstations, yet complex enough to be interesting to researchers working in the field. Therefore, we will investigate analytically defined dynamical systems. The simulation of such systems requires the numerical approximation of differential equations, which can be done fast enough for computational steering and therefore suits our needs. Following a previous cooperation with researchers from econometrics, we will initially concentrate on population models, where for example the interaction between population growth, economic activity and environmental impact is modeled (Gröller, 1996). Changing certain parameters of such systems only slightly may have significant impact on the long term behavior, making interactive computation steering essential for the understanding of such systems.

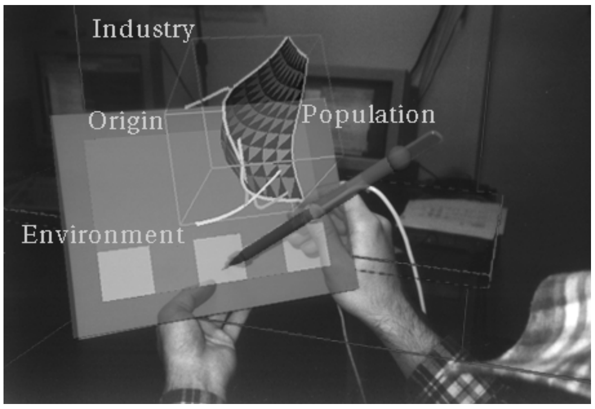


Figure 4. The Wonderland Model (Gröller, 1996) on the Personal Interaction Panel in Studierstube

Standard methods for flow visualization including stream lines, stream surfaces, particles, tufts, iconic representations of local properties (glyphs) will be tested to illustrate the flow. The use of a true three-dimensional display is beneficial to the understanding, because with 2-D images it is difficult to grasp the dynamics. Nevertheless, so far mostly 2-D sections have been used.

We will enhance the expressive power with display and interface techniques exploiting the augmented reality setup. This will include using the PIP metaphor (see above) custom tailored for interaction with the dynamical system. The PIP will be used as a probing tool to define 2-D cross sections, Poincare sections, 2-D projections and to specify the origin of particles introduced into the flow, and at the same time can serve as a screen for the resulting images. Phase space representations can be displayed right on top of the PIP and exchanged among researchers. The augmented reality setup also allows the use of an additional high-resolution CRT monitor for the display of high quality 2-D images (e.g., the mentioned cross sections) without leaving the augmented environment.

The PIP's pen can also be used as a probing tool to display local properties of the visualization data at the point indicated by the pen's tip. Corresponding values of the parameters and graphical representations (glyphs) can be displayed on the panel with interactive update.

Besides the PIP, designated physical tools can be used to control application parameters with high fidelity and force feedback. For example, parameters of a 3-D phase space can be defined by position and orientation tracked mobile objects, like telescopic antennas corresponding to the principal axes, allowing free choice of orientation and scaling.

7. Conclusions and Future Work

We presented a collaborative augmented environment setup supporting interactive scientific visualization for multiple users. Our system provides 3D display of synthetic data and augmentation of physical objects with geometrically aligned information. Co-workers wear position and orientation tracked see-trough head mounted displays, allowing independent choice of viewpoint. Interaction is performed using the Personal Interaction Panel, a two-handed interface for augmented reality.

The system provides a natural working atmosphere, by enriching reality with spatially aligned information while leaving natural communication channels in principle unaffected. Annotations enhance understandability of the discussed topic while customization of different data layers support cooperation of experts from different fields. Direct exploration and modification in visualization provides improved insight in complex problems.

Although experiments with unskilled users show promising results on acceptance, enhanced registration and correct matching of real environment and ovarlaid graphics is required using hybrid tracking technology. Our restricted implementation supporting two users should be extended to a number of participants, allowing more complex collaborative situations. Connection to external modules with standardised protocols for image and interaction data will provide a wide variety of different applications.

8. Acknowledgements

This work has been supported by the Austrian Science Fund (FWF) under project number P-12074-MAT.

9. References

AVS (1992). *AVS Developers Guide - Release 4*. Advanced Visualisation Systems Inc.

Bajura, M., Fuchs, H. and Ohbuchi, R. (1992). *Merging Virtual Objects with the Real World: Seeing Ultrasound Imaginery within the Patient*. In proceedings of SIGGRAPH’92: 203-210.

Bier, E., Stone, M., Pier, K., Buxton, W. and DeRose, T. (1993). *Toolglass and Magic Lenses: The See-through Interface*. In proceedings of SIGGRAPH’93: 73-80.

Brooks, F. Jr. et. al. (1990). *Project GROPE - Haptic Displays for Scientific Visualisation*. In proceedings of SIGGRAPH’90: 177-185.

Bryson, S. (1991). *The Virtual Wind Tunnel*. In proceedings of IEEE Visualization’91: 17-25.

Bryson, S. (1993). *The Distributed Virtual Windtunnel*. In SIGGRAPH’93 Course Notes 43: 3.1-3.10.

Cruz-Neira, C., Sandin, D. and DeFanti, T. (1993 a). *Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE*. In proceedings of SIGGRAPH’93: 135-142.

Cruz-Neira, C. et al. (1993 b). *Scientists in Wonderland: A Report on Visualization Applications in the CAVE Virtual Reality Environment*. In proceedings of the IEEE 1993 Symposium on Research Frontiers in Virtual Reality: 59-67.

Fahlén, L.E., Brown, C.G., Ståhl, O. and Carlsson, C. (1993). *A Space Based Model for User Interaction in Shared Synthetic Environments*. In proceedings of INTERCHI’93: 43-48.

Feiner, S., MacIntyre, B. and Seligmann, D. (1992). *Annotating the Real World with Knowledge-Based Graphics on a See-Through Head-Mounted Display*. In proceedings of Graphics Interface’92: 78-85.

Feiner, S., MacIntyre, B. and Seligmann, D. (1993). *Knowledge-Based Augmented Reality*. Communications of the ACM 36(7): 53-62.

Fritzmaurice, G.W. (1993). *Situated Information Spaces and Spatially aware Palmtop Computers*. Communications of the ACM 39(7): 39-49.

Gröller, E., Wegenkittl, R., Milik, A., Prskawetz, A., Feichtinger, G. and Sanderson, W.C. (1996). *The Geometry of Wonderland*. Accepted for publication in the journal Chaos, Solitons & Fractals.

Holmgren, D. (1992). *Design and Construction of a 30-Degree See-Through Head-Mounted-Display*. Technical Report 92-030 at the University of North Carolina, available at ftp://ftp.cs.unc.edu./pub/technical-reports/92-030.ps.Z .

Krüger, W., Bohn, C., Fröhlich, B., Schüth, H., Strauss, W. and Wesche, G. (1995). *The Responsive Workbench: A Virtual Work Environment*. IEEE Computer 28(7): 42-48.

Nielsen, G., Shriver, B. and Rosenblum, L. (eds.) (1990). *Visualization in Scientific Computing* (Los Alamitos/California: IEEE Computer Society Press).

Pausch, R., Burnette, T., Brockway, D. and Weiblen, M. (1995). *Navigation and Locomotion in Virtual Worlds via Flight into Hand-Held Miniatures*. In proceedings of SIGGRAPH'95: 399-401.

Rekimoto, J. and Nagao, K. (1995). *The World through the Computer: Computer Augmented Interactions with Real World Environments.* In proceedings of UIST '95: 29-36.

Sharma, R., Molineros, J. (1996). *Interactive Visualization and Augmentation of Mechanical Assembly Sequences*. Proceedings of Graphics Interface '96: 230-237.

State, A., Livingston, M.A., Garrett, F., Hirota, G., Whitton, M.C., Pisano, E.D. and Fuchs, H. (1996). *Technologies for Augmented Reality Systems: Realizing Ultrasound-Guided Needle Biopsies*. In proceedings of SIGGRAPH'96: 439-446.

Strauss, P. and Carrey, R. (1992). *An Object Oriented 3D Graphics Toolkit.* In proceedings of SIGGRAPH'92: 341-347.

Sutherland, I. (1968). *A Head-Mounted Three Dimensional Display*. Fall Joint Computer Conference, In proceedings of AFIPS Conference 33: 757-764.

Szalavári, Zs. and Gervautz, M. (1996). *The Personal Interaction Panel - A Two-handed Interface for Augmented Reality*. Technical Report TR186-2-1996-20 at the Institute of Computer Graphics, Vienna University of Technology, available at ftp://ftp.cg.tuwien.ac.at/pub/TR/96/TR-186-2-96-20Paper.ps.gz .

Taylor, R. M. et. al. (1993). *The Nanomanipulator: A Virtual Reality Interface for a Scanning Tunneling Microscope*. In proceedings of SIGGRAPH'93: 127-134.

Presented at 1999 ACM Symposium on Interactive 3D Graphics (I3DG'99),
April 26-28, 1999, Atlanta, GA

Using Transparent Props For Interaction With The Virtual Table

Dieter Schmalstieg[1], L. Miguel Encarnação[2], and Zsolt Szalavári[3]

[1]Vienna University of Technology, Austria
[2]Fraunhofer CRCG, Inc., Providence, RI, USA
[3]Imagination GmbH, Vienna, Austria

Abstract

The Virtual Table presents stereoscopic graphics to a user in a workbench-like setting. This paper reports on a user interface and new interaction techniques for the Virtual Table based on transparent props— a tracked hand-held pen and a pad. These props, but in particular the pad, are augmented with 3D graphics from the Virtual Table's display. This configuration creates a very powerful and flexible interface for two-handed interaction that can be applied to other back-projected stereographic displays as well: the pad can serve as a palette for tools and controls as well as a window-like see-through interface, a plane-shaped and through-the-plane tool, supporting a variety of new interaction techniques.

1. INTRODUCTION

While the desktop metaphor is well-understood and represents an effective approach to human-computer interaction for document-oriented 2D tasks, transplanting it to 3D reveals inherent limitations (e.g. [8]). In contrast, interfaces that incorporate true 3D input and output technologies, e.g., six degree of freedom (6DOF) sensors and stereoscopic displays seem more promising, even though the use of advanced interface devices does not guarantee a superior user interface.

We present a system that uses transparent props for two-handed interaction on the Barco BARON [4] Virtual Table (VT), a tabletop VR display based on a workbench metaphor [14]. The hand-held transparent props are a pen and a pad and related to earlier research on the Personal Interaction Panel (PIP) [24], an augmented reality interface. While augmented reality systems use semi-transparent or video-based head-mounted displays to overlay computer graphics onto real-world objects (e.g., [10] or [3]), our system overlays transparent physical props onto the back-projected display of the VT. to achieve a kind of inverse augmented reality, which we call *augmented VR*. The VT thereby provides an enhanced workspace with capable multipurpose tools. As Wloka & Greenfield [27] point out, the tactile feedback that the physical props provide makes the tools feel real.

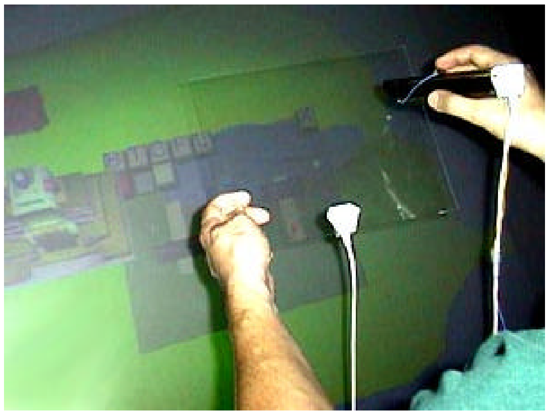


Figure 1: The transparent pen and pad props.

Our system unifies several previously isolated approaches to 3D user interface design, such as two-handed interaction and the use of multiple coordinate systems, but more importantly it allows for the experimentation with the affordances [17] of transparent props that— with the exception of[25]— are generally unexplored. Our interface supports the following important features:

- two-handed interaction
- multi-purpose physical props
- embedding 2D in 3D
- use of multiple coordinate systems (i.e., of the table and the pad)
- transparent tools, especially window-tools and through-the-plane tools.

Each of the listed properties allows the design of distinct forms of interaction. This paper describes our efforts to explore these possibilities of transparent props for 3D interaction. After an overview on related work in Section 2, we describe the system setup used for our experiments in Section 3. We then report on the interaction techniques supported by our transparent props in Section 4. Our ideas are illustrated by examples from a *Virtual Landscaping* application developed to depict the capabilities of

---

[1]Vienna University of Technology, Institute of Computer Graphics, Schönbrunner Strasse 7/A/1, A-1040 Vienna, Austria (email: dieter@cg.tuwien.ac.at)

[2]Fraunhofer Center for Research in Computer Graphics, Inc., 321 S. Main St., Providence, RI 02903, USA (email: mencarna@crcg.edu)

[3]Imagination GmbH, Schönbrunner Strasse 7/A/1, A-1040 Vienna, Austria (email: zsolt@cg.tuwien.ac.at)

our platform. In Section 5 we give some implementation details and finally present results and observations of the system in practice.

2. RELATED WORK

Our approach was originally inspired by the work of Szalavári & Gervautz [24] on the Personal Interaction Panel. This work explored the use of (opaque) pen and pad props in a head-mounted, see-through augmented reality system, called *Studierstube* [21]. Other researchers use pen and pad props, though either in fully immersive or desktop setups: Sachs et al. [19] describe a system for the design of 3D curves and shapes. Angus & Sowizral [2] report on their use of pen and pad props for embedding traditional 2D GUIs in a 3D immersive system. Billinghurst et al. [7] describe 3D Palette, a virtual content creation tool using pen and pad props in a fishtank VR setup.

Several researchers reported on the use of two-handed interaction. For tabletop VR devices, Cutler et al. [9] have developed a two-handed object manipulation framework using two gloves or a glove and a stylus. Other uses of two-handed interaction for object design and manipulation can be found in [16] and [11]. These designs are based upon Guiard's observation of how humans distribute work between the two hands [12].

The window-based tools we have developed are related to the toolglass and magic lenses proposed by Bier et al. [6] and extended to 3D by Viega et al. [26], but their approach has some drawback in terms of generality and is not fully embedded into a VR system. Our window-based tools—having real extension into all 3 dimensions—share the goals of 3D magic lenses, but are based on the more flexible implementation of SEAMS originally developed for navigation of virtual environments [20].

Our work also shares aspect with both the active and passive lens of the metaDESK [25]. The metaDESK passive lens is a transparent prop, but does not use stereoscopic graphics and is not used for general-purpose interaction like the props in our system.

Wloka & Greenfield [27] point out that using tools are equally expressive as using one's hands. They propose the use of a one-handed multi-function tool, the virtual tricorder, which inspired our work as well.

Finally, Pierce et al. [18] report on image-plane interaction techniques for immersive virtual environments and let users interact with 2D projections of 3D objects, an approach related to our through-the-plane metaphor.

3. SYSTEM SETUP

The system we have developed uses the Barco *Baron* Virtual Table as its display device. This device offers a 53"x40" display screen built into a table surface and connects to an SGI Indigo2 Maximum Impact workstation. Together with CrystalEyes shutter glasses from StereoGraphics, a large stereo display of very high brightness and contrast is available.

The transparent props we use are an 8"x10" Plexiglas sheet and a large, pen-shaped, plastic tube (Figures 1,2) which is additionally fitted out with a button. Both props as well as the shutter glasses are equipped with 6DOF trackers (Ascension Flock of Birds) for position and orientation tracking. For details on tracker calibration, refer to Section 5.

Using the information from the trackers, 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 augmented VR, as the physical props and their virtual counterparts have to appear aligned in 3D.

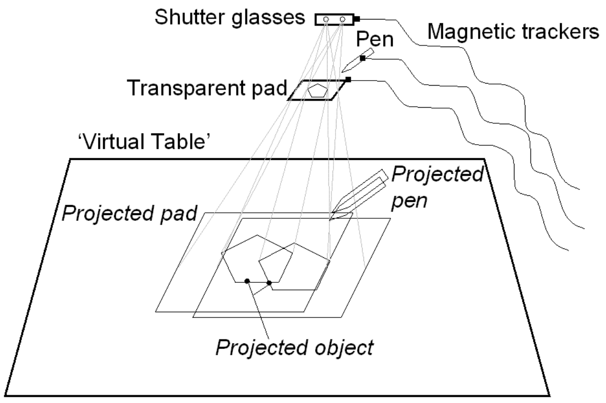


Figure 2: The Virtual Table's display creates the illusion of graphics aligned with the pen and pad.

The material for the pen and pad was also 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. 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. 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.

The pen was chosen to be relatively large to provide room for graphics displayed inside the pen. In that way, the pen also provides visual feedback such as showing the tool is currently associated with. So far, however, we have made only basic use of this capability and have instead focused on the pad as a carrier for the user interface.

4. THE TRANSPARENT PROPS'DESIGN SPACE

The focus of our work is to explore the user-interface and interaction possibilities of the transparent pad as a distinct object. While the two-handed pen-and-pad metaphor is asymmetric [12] and the pad is assigned the more "passive" role (e. g., it is held in the non-dominant hand), it has much more interesting affordances than the pen. Pen and pad have a relationship similar relationship to mouse pointer and window in a conventional desktop system. However, the difference to the desktop is not only that pen and pad operate in 3D, but also that the pad is directly controlled by the user's non-dominant hand and can therefore additionally be used as an active tool.

The pad therefore represents an embedding of 2D in 3D, as already pointed out by Angus & Sowizral [2]. Yet its possibilities extend far beyond that by *combining* several individual metaphors:

- *Tool and object palette*: The pad can carry tools and controls, much like a dialog box works in the desktop world (as e.g. in Smartscene [13]). It can also offer collections of 3D objects to choose from.
- *Window tools*: As the user can see through the pad into the scene, the pad becomes a see-through tool (as e.g. the Virtual Tricorder [27]).
- *Through-the-plane tool*: The user can orient the "window" defined by the pad and then manipulate objects as seen through the pad, i. e. manipulate the 2D projections of objects on the pad.
- *Volumetric manipulation tool*: The pad itself can be used for active object manipulation (as e.g. the WIM [22]) exploiting the fact that the pad has a spatial extent (unlike the point represented by the pen tip).

These options co-exist in the design space of our user interface and together form a very powerful and general framework for 3D interaction. Due to the fact that the physical and geometric properties of the pad are of very basic nature, it is possible to use all the metaphors mentioned above for application tasks without confusing the user. Our transparent props form a two-handed multi-purpose tool in the spirit of Wloka & Greenfield [27].

4.1 Tool and Object Palette

In its basic use, the pad serves as a palette offering various tools and controls. The pad resembles a dialog box in a desktop system by grouping various application controls such as buttons, sliders, dials etc. Since the pad is hand-held, it is always in convenient reach for the user, which is an advantage if working on different areas of the table. It is easy to remember where the controls are, and the pad can even be put aside temporarily without causing confusion.

Controls are manipulated with the pen, which is also used to select tools. The active part of a chosen tool is generally associated with the pen, while the pad acts as a passive counterpart for many tools.

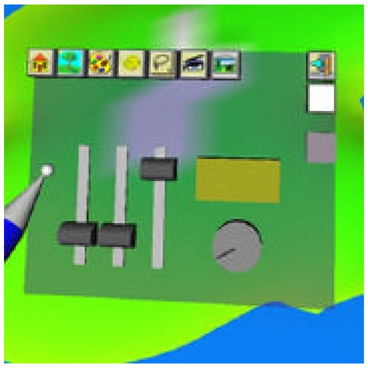


Figure 3: In its basic function, the pad serves as a palette for tools and controls. Shown is an RGB color selection tool.

The basic mode of our sample landscaping application is object placement. The pad serves as an object browser presenting a collection of objects to select from. Objects are then dragged and dropped into the scene via direct 3D manipulation. Additional controls—some implemented as space-saving pop-up button bars—allow to scale, colorize, and delete objects. 2D controls and 3D direct manipulation naturally blend as the pad represents a 2D surface similar to many real-world control containers of other application areas (e. g., a remote control, a radio, a dishwasher's front panel).

Another interesting property is the multiple surfaces of reference with which the user simultaneously interacts, a fact also observed as being beneficial by Ullmer & Ishii [25]. A sample use is the drag and drop operation from the pad to the table space. We make further use of this property with the window and through-the-plane tools.

4.2 Window Tools

Because it is transparent, our pad prop invites users to look through it. Consequently, we chose to experiment with a set of functions which we call window tools. Conceptually, they are very similar to 3D magic lenses introduced by Viega et al. in [26]. However, we have their work in two significant ways: First, the underlying implementation does not have the limitations of the original work (see Section 5) and have real extension into all 3 dimensions. Second, our two-handed interaction allows us to manipulate objects seen *through* the lens instead of just the magic lens itself. Our window tools are therefore more related to the toolglass of Bier et al. [6]. This approach is not unlike that of a watchmaker using a magnifying glass together with other tools—a task that naturally fits into a workbench-like environment.

Instead of manipulating controls and objects on the pad, the user manipulates objects on the table surface *under* the pad, which divides the table space into two 'design' spaces.

In our landscaping application, we have implemented a cable TV tool that provides the user with X-ray vision (Figure 4). The user can look under the surface of the landscape representing an island (using wireframe rendering) and use the pen to lay wire and connect houses to a cable TV network. The X-ray tool is bound to the *backside* of the pad, making use of the pad's two-sided property, thus having the X-ray tool always available. (Which side of the pad the user looks at is easily determined by examining the pad's normal vector).

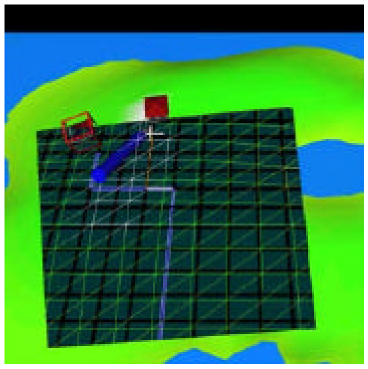


Figure 4: The cable TV routing tool is a special "X-ray" view attached to the back of the pad and allows the placement of wires underneath the island.

While the X-ray tool is an example of a modified view of the environment, a window can also show different content. Windows in the desktop world usually show different content: multiple windows can either be entirely unrelated, or they can show data from different points in space (different viewpoints) or time (different versions). CAD systems normally use four windows with different viewpoints, and text tools like *xdiff* show a side-by-side comparison of different versions of data.

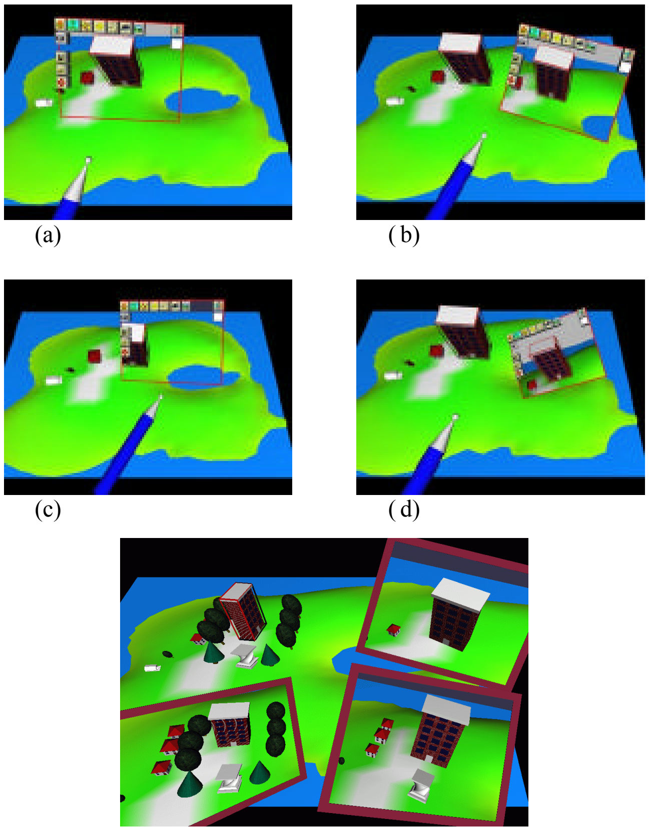

(a) (b) (c) (d) (e)

Figure 5: The snapshot tool allows the user to manage a collection of scenes that are viewed from different perspectives and in different stages of development. Note how the scene in the snapshots is not flat, but a real 3D view (compare a to b and d), how a scene variant is visible as a snapshot for comparison (c), and how multiple snapshots can be kept, floating in windows above the virtual scene (e).

We built this idea into our landscaping application using a *snapshot* facility. In normal mode, the view through the window (pad) is identical to the normal scene. However, a particular view (or more precisely, viewpoint) of the scene can be locked on the pad (Figure 5a). This snapshot not a flat photograph, but a real 3D scene that can be viewed from an arbitrary angle and distance by moving the pad or one's head (compare Figure 5b and d to Figure 5a).

Such a snapshot may be decoupled from the pad and left floating in the scene at any position, and possibly be picked up again later. By strategically placing multiple of such snapshots in the scene, a user can inspect multiple views at once from inside a virtual environment, a strategy equivalent to the aforementioned multiple views of CAD systems.

Changes to the objects are reflected in all views simultaneously. However, if the user indicates so, the scene observed through the window can be isolated from the scene shown on the table and from other windows' scenes; thus multiple individual scenes are seen simultaneously. This feature resembles a multiple document interface from the desktop world, an aspect that to our knowledge has not been explored for VR system so far.

When scenes are isolated in multiple windows, changes to one scene are not reflected in another scene. It is therefore possible to modify the main scene on the VT while the scene in the window remains the same: it becomes a historical reference.

For the landscaping applications, multiple versions of development (or possible design alternatives) can be presented side by side with little effort by the user. This feature is potentially very useful for any kind of 3D-design application. By picking up a floating window that is carrying a particular variant of a scene and unlocking the frozen viewpoint of the window (i.e. the scene through the window is again rendered with the same viewpoint as the scene on the VT), a special kind of portable magic lens for in-place comparison of two variants is created. An example is shown in Figure 5c, where the large building has been deleted in the main scene but is still visible through the window tool.

The possibilities of the snapshot tool are summarized in Figure 6 in the form of a state diagram.

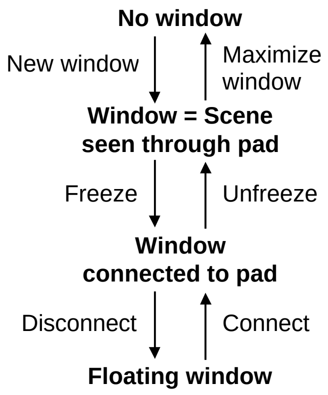


Figure 6: State diagram for managing scenes using the window tools of the landscaping application.

4.3 Through-The-Plane Tool

The look-through affordance of the transparent pad allows the development of yet another class of user interface tools that we call *through-the-plane tools*. They are related to the image plane techniques reported by Pierce et al. [18]. Image plane techniques manipulate 3D objects based on their 2D projection on a plane perpendicular to the line of sight. The pad as a through-the-plane tool differs from this approach in two important aspects:

1. The 2D plane onto which objects are projected is easily manipulated by moving or rotating the pad without the need to move one's point of view.
2. The physical surface of the pad provides a clear definition of the 2D manipulation plane and a tactile surface for making gestures with the pen. (Image plane techniques require a user to make hand gestures in the air without a clearly defined depth of the plane.)

As a consequence of these properties, we have not experienced problems with ambiguities resulting from the stereo projection as reported in [18], although the problem itself remains.

In the landscaping application, we have implemented two tools using the pad as an through-the-plane tool. The first tool is a context sensitive information and manipulation dialog. The user may point the pad into the scene, and the object closest to the tool's center (in the 2D space of the tool) is selected. The object's description is displayed, and context-sensitive controls are displayed on the pad.

In Figure 7, different collections of colorize buttons appear, depending on the type of an object.

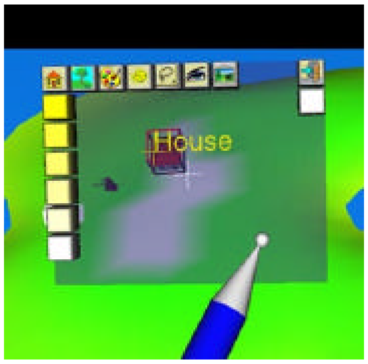


Figure 7: The context sensitive tool uses 2D manipulation through the pad. Depending on the position of the pad, objects in the scene get selected, and context sensitive color controls are offered.

Many desktop applications, such as illustration programs, use context-sensitive menus and toolbars that appear and disappear as different objects are selected. The context-sensitive tool brings these possibilities into a VR system. Note that context-sensitive manipulation requires two steps in a one-handed desktop system: The user selects an object, looks for context-sensitive controls to appear somewhere on the screen, and then manipulates the controls. Although marking menus as proposed by Kurtenbach & Buxton [15] are an already much more effective one-handed interaction, they still require the employment of the user's hand for menu marking and then selection. In contrast, only one two-handed step is required in our system, yet controls still always appear near the selected object. Manipulation of pad and pen can be almost instantaneous and is cognitively more similar to context-sensitive pop-up menus, but without the corresponding disadvantages (e.g., display often obscured by menu, mouse button must be held and cannot be used for interaction).

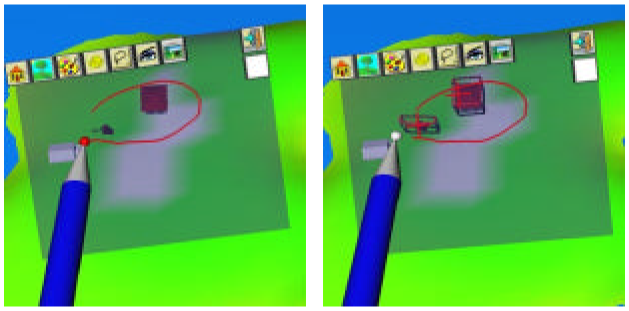


Figure 8: The lasso tools uses the pad as a plane through which objects in the scene are targeted.Instead of selecting objects in the scene, they can then be selected through a 2D circular gesture on the pad. An outline drawn on the pad being held into the scene defines a conical sweep volume that has its tip in the eye point and its contour defined by the gesture. All object contained within this volume are selected (Figures 8,9).

Again, the lasso tool is just one example for a wide design space of tools based on 2D gestures for 3D objects (e.g., objects may be deleted by "crossing them out"). The through-the-plane tool allows us to reuse all the ideas for 2D manipulation of 3D that are known in the desktop world (cf. e. g., draggers and manipulators of Open Inventor [23]). It remains to be verified, however, in which cases this 2D manipulation is more capable than direct 3D manipulation. From our observations we conclude that the power of such 2D gesture tools lies in manipulation at-a-distance, for example when attempting to manipulate objects on one side of the table when standing at the other side.

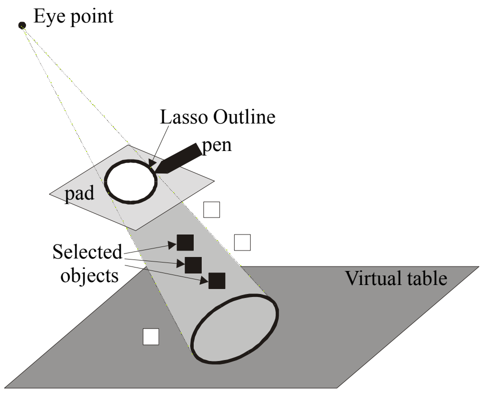


Figure 9: The lasso defines a conical sweep volume to select objects.

4.4 Volumetric Manipulation Tool

Most of the tools we have described so far use the pad to provide the context or frame of reference with the pen (more specifically, the pen tip) being the active part, quite in the spirit of Guiard's observations [12]. However, the pad can be an active (one-handed) tool in its own right.

What sets the pad apart from conventional 3D manipulation tools like a bat, wands, stylus, or buttonball, is its dimension: all these devices have a zero-D (point-shaped) "hot spot" for triggering actions. A laser pointer like tool (which is a popular metaphor for selecting objects at a distance) uses a ray and therefore has a dimension of one. Errors introduced by human inaccuracy make it difficult to perform precise manipulation with tools of essentially no spatial extent, which lack correspondence to real world tools. This is why techniques such as 3D snap-dragging [5] were developed to overcome the mentioned difficulties.

Instead of artificially enhancing the input, we propose to use a tool with a spatial extent, which more naturally resembles real world tools. The 2D surface of the pad can serve such a purpose.

As an example, we have implemented a *fish net* selection tool for the landscaping application. By sweeping the tool through the scene, the user may select objects (Figure 10, top), which are all objects that are intersected with the pad. Since it is undesirable for the user's landscaping efforts to be destroyed as a result of actual objects becoming caught in the fish net, small 3D replicas of the objects are "caught" in the net (or rather, shown on the pad's surface). The replicas are placed in the position on the pad where the object was penetrated, and an "arrange" button aligns the replicas in a regular grid for better overview (Figure 10, bottom).

We have found that sweeping a path with the pad is surprisingly effective for the selection of objects that lie in front of the projection plane of the table, especially when a large number of objects must be selected quickly but selectively. We attribute this ease of usability to the users' real world experience with similar tools.

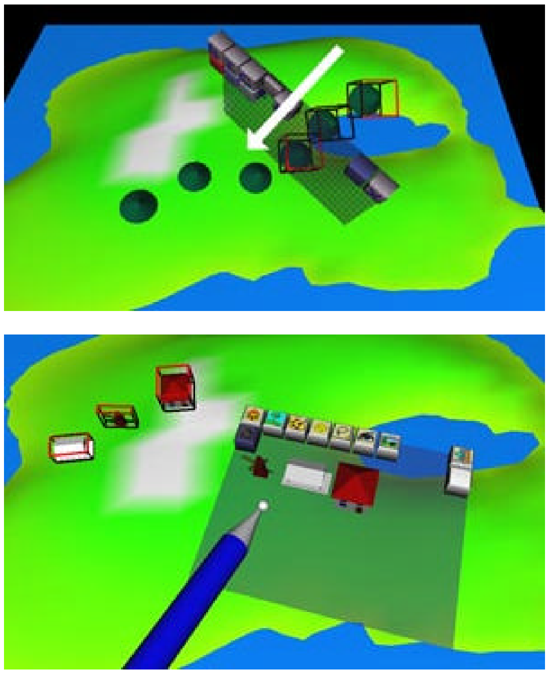


Figure 10: The fish net tool makes use of the pad as a tool with spatial extent. By sweeping it through the scene, objects are selected (top image) and replicas of the objects appear on the pad for further manipulation.

Sometimes it may happen that an object is involuntarily selected together with others. If this occurs, the small replicas on the pad can be discarded by wiping them off the pad, and the corresponding object becomes deselected.

Although we have not yet implemented them, we have imagined several other volumetric manipulation tools that could be used, such as a shovel, a ruler, and a rake. Another possible area of applications are deformation tools for objects made of clay (similar to features found in MultiGen's SmartScene [13]).

5. IMPLEMENTATION

Software architecture. Our system is based on the *Studierstube* [21] software framework. It is realized as a collection of C++ classes extending the Open Inventor toolkit [23]. Open Inventor's rich graphical environment approach allows rapid prototyping of new interaction styles, typically in the form of Open Inventor node kits. Tracker data is delivered to the application via an engine class, which forks a lightweight thread to decouple graphics and I/O. Off-axis stereo rendering on the VT is performed by a special custom viewer class. Open Inventor's event system has been extended to process 3D (i. e., true 6DOF) events, which is necessary for choreographing complex 3D interactions like the ones described in this paper. The .iv file format, which includes our custom classes, allows convenient scripting of most of an application's properties, in particular the scene's geometry. Consequently very little application-specific C++ code—mostly in the form of event callbacks—was necessary.

Calibration. Any system using augmented props requires careful calibration of the trackers to achieve sufficiently precise alignment of real and virtual world, so the user's illusion of augmentation is not destroyed. With the VT this is especially problematic, as it contains metallic parts that interfere with the magnetic field measured by the trackers. To address this problem, we have adopted an approach similar to the one described by Agrawala et al. [1] and Krüger et al. [14]: The space above the table is digitized using the tracker as a probe, with a wooden frame as a reference for correct real-world coordinates. The function represented by the set of samples is then numerically inverted and used at runtime as a look-up table to correct for systematic errors in the measurements.

Window tools: The rendering of window tools differs from the method proposed by Viega et al. [26] in its use of hardware stencil planes. After a preparation step, rendering of the world "behind the window" is performed inside the stencil mask created in the previous step, with a clipping plane coincident with the window polygon. Before rendering of the remaining scene proceeds, the window polygon is rendered again, but only the Z-buffer is modified. This step prevents geometric primitives of the remaining scene from protruding into the window. For a more detailed explanation, see [20].

6. CONCLUSIONS AND FUTURE WORK

We have presented a system that uses transparent props—the pen and pad—for two-handed interaction with the Virtual Table, a desktop VR system. The system exploits the fact that the VT can display 3D graphics aligned with the props, turning them into multi-purpose tools. In this sense, transparent props seem even to be a tool for the guiding person in a Surround-Screen Projection-Based Virtual Reality System (SSVR), who's viewpoint is tracked, and therefore in correct stereoscopic relation to the interface on the panel's surface. We consider such a configuration an interesting next step for our research.

We have explored and prototyped various interaction metaphors, most of which are inspired by the physical properties of the props and analogies to the desktop metaphor. Our experiments have led us to believe that the rich set of user-interface designs developed for the desktop world in the last decade can be transposed to VR systems if proper attention is paid to the requirements of 3D.

Our system was informally tested with several users, most of which had computer (desktop) experience but little experience with VR systems. They generally found our design very appealing and were able to perform simple landscaping tasks after a few minutes of initial instruction. We did not observe any difficulties in understanding the tools. Complaints mainly addressed technical inadequacies like tracker error, lag or frame rate. Fatigue resulting from prolonged use of the props did not seem to be an issue. However, since most test sessions did not last longer than 20 minutes, this usability aspect will require further investigation. One significant disadvantage we found lies in the restriction of the VT to a single head-tracked user, as oftentimes multiple users wanted to use the system concurrently. As a side note, a possible solution to this problem is presented in [1] for two users yet the described approach probably does not scale beyond a few users.

A promising area of future work encompasses the window tools we have discussed in Section 4.2. The snapshot tool built into the landscaping application makes only very basic use of the possibilities of window tools. We observe that there is a trend in computer systems towards "browser" tools that invoke adequate representations for different flavors of multimedia data, and we speculate that windows in the style we have shown may prove to be an adequate metaphor to organize data in a browser for 3D scenes. Furthermore, the windows can also serve as containers for distinct 3D applications, with possibilities such as object drag and drop between them. We also intend to explore the possibilities of creating a *workspace*, the 3D equivalent to a multi-windows desktop.

ACKNOWLEDGMENTS

This work has been sponsored by the Fraunhofer CRCG Student and Scholar Exchange Program (SSEP) and the Austrian Science Foundation (*FWF*) under contract number P-12074-MAT. Special thanks to Michael Gervautz for supporting the way to this research with the PIP, and to Anton Fuhrmann, Markus Krutz, Hermann Wurnig and Andreas Zajic for their contributions to the implementation.

REFERENCES

[1] M. Agrawala, A. Beers, B. Fröhlich, P. Hanrahan, I. McDowall, M. Bolas: The Two-User Responsive Workbench: Support for Collaboration Through Individual Views of a Shared Space. Proceedings of SIGGRAPH, 1997.

[2] I. Angus and H. Sowizral: Embedding the 2D Interaction Metaphor in a Real 3D Virtual Environment. Proceedings SPIE, vol. 2409, pages 282-293, 1995.

[3] M. Bajura, H. Fuchs, and R. Ohbuchi. Merging Virtual Objects with the Real World: Seeing Ultrasound Imaginery within the Patient. Proceedings of SIGGRAPH'92, (2):203-210, 1992.

[4] Barco BARON, URL: http://www.barco.com/projecti/-products/bsp/baron.htm 1997.

[5] E. Bier: Snap-dragging in three dimensions. Proceedings of the 1990 Symposium on Interactive 3D Graphics, pp. 193-203. ACM SIGGRAPH, March 1990.

[6] E. Bier, M. Stone, K. Pier, W. Buxton, and T. DeRose. Toolglass and Magic Lenses: The See-through Interface. Proceedings of SIGGRAPH'93, pages 73-80, 1993.

[7] M. Billinghurst, S. Baldis, L. Matheson, and M. Philips. 3D Palette: A Virtual Reality Content Creation Tool. Proceedings of ACM VRST'97, pages 155-156, 1997.

[8] S. Bryson and C. Levitt: The virtual windtunnel: An environment for the exploration of three-dimensional unsteady flows". Proceedings Visualization'91, pages 17-24, 1991.

[9] L.D. Cutler, B. Fröhlich, and P. Hanrahan: Two-Handed Direct Manipulation on the Responsive Workbench. Proceedings of SIGGRAPH Symposium on Interactive 3D Graphics `97, RI, USA, pages 39-43, 1997.

[10] S. Feiner, B. MacIntyre, and D. Seligmann. Knowledge-Based Augmented Reality. Communications of the ACM, 36(7):53-62, 1993.

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

[12] 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.

[13] Homan. SmartScene: Digital Training - Learn the System by Being Part of the System. Technical report, available from http://www.multigen.com/

[14] W. Krüger, C. Bohn, B. Fröhlich, H. Schüth, W. Strauss, and G. Wesche: The Responsive Workbench: A Virtual Work Environment. IEEE Computer, 28(7):42-48, 1995.

[15] G. Kurtenbach, G. and W. Buxton: User learning and performance with marking menus. Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems (1994), pp. 258-264.

[16] D. Mapes and J. Moshell: A Two-Handed Interface for Object Manipulation in Virtual Environments. Presence, 4(4):403-416, 1995.

[17] D. Norman: The Psychology of Everyday Things. New York, Basic Books, 1988.

[18] J.S. Pierce, A. Forsberg, M. J. Conway, S. Hong, R. Zeleznik, and M.R. Mine: Image Plane Interaction Techniques in 3D Immersive Environments. Proceedings of SIGGRAPH Symposium on Interactive 3D Graphics `97, RI, USA, pages 39-43, 1997.

[19] E. Sachs, A. Roberts, and D. Stoops: 3-Draw: A Tool for Designing 3D Shapes. IEEE Computer Graphics & Applications, pages 18-26, 1991.

[20] D. Schmalstieg, G. Schaufler: Sewing Virtual Worlds Together With SEAMS: A Mechanism to Construct Large Scale Virtual Environments. Technical Report TR-186-2-87-11, Vienna University of Technology, 1998.

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

[22] R. Stoakley, M. J. Conway, and R. Pausch: Virtual Reality on a WIM: Interactive Worlds in Miniature. Proceedings 1995 Conference on Human Factors in Computing Systems (CHI'95), pages 265-272, 1995.

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

[24] Zs. Szalavári and M. Gervautz: The Personal Interaction Panel - A Two Handed Interface for Augmented Reality. Computer Graphics Forum (Proceedings of EUROGRAPHICS'97), 16(3):335-346, 1997.

[25] B. Ullmer and H. Ishii: The metaDESK: Models and Prototypes for Tangible User Interfaces. In Proceedings of ACM UIST'97, Banff, Alberta, Canada, pages 223-232, 1997.

[26] J. Viega, M. Conway, G. Williams, and R. Pausch: 3D Magic Lenses. In Proceedings of ACM UIST'96, pages 51-58. ACM, 1996.

[27] M. Wloka and E. Greenfield: The Virtual Tricoder: A Uniform Interface for Virtual Reality. Proceedings of ACM UIST'95, pages 39-40, 1995.

Bridging Multiple User Interface Dimensions with Augmented Reality

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

Anton Fuhrmann
Research Center for Virtual Reality and Visualization, Vienna, Austria
fuhrmann@vrvis.at

Gerd Hesina
Vienna University of Technology, Austria
hesina@cg.tuwien.ac.at

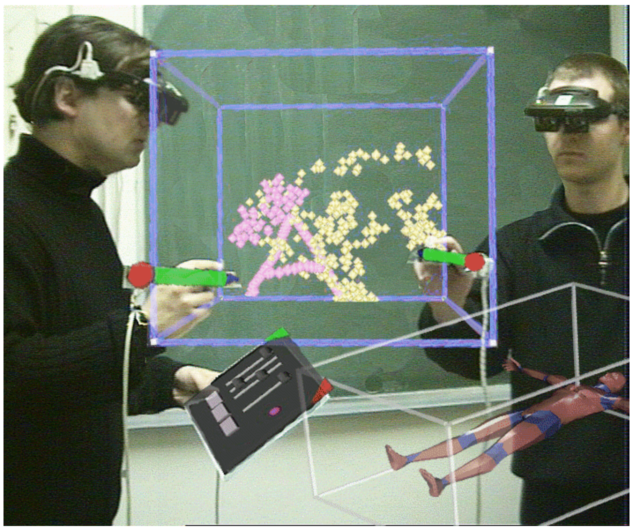


Figure 1: Collaborative work in *Studierstube*: 3D painting application window (with focus, middle) and object viewer window (without focus, lower right)

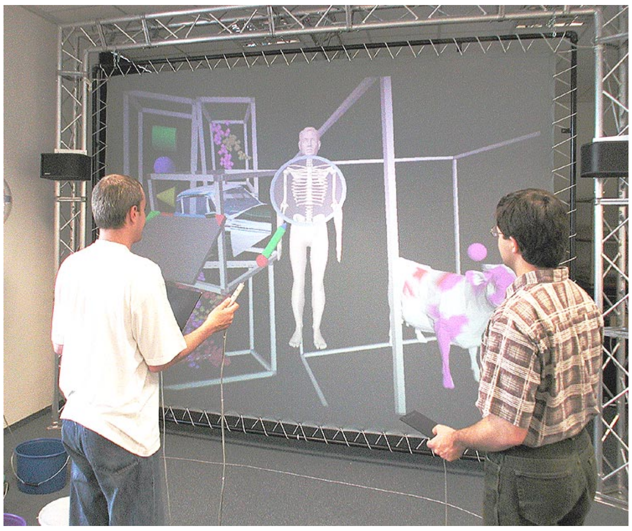


Figure 2: Two *Studierstube* users working jointly on multiple applications in front of a large screen, usually with passive stereo glasses (not shown)

Abstract

Studierstube is an experimental user interface system, which uses collaborative augmented reality to incorporate true 3D interaction into a productivity environment. This concept is extended to bridge multiple user interface dimensions by including multiple users, multiple host platforms, multiple display types, multiple concurrent applications, and a multi-context (i. e., 3D document) interface into a heterogeneous distributed environment. With this architecture, we can explore the user interface design space between pure augmented reality and the popular ubiquitous computing paradigm. We report on our design philosophy centered around the notion of contexts and locales, as well as the underlying software and hardware architecture. Contexts encapsulate a live application together with 3D (visual) and other data, while locales are used to organize geometric reference systems. By separating geometric relationships (locales) from semantic relationships (contexts), we achieve a great amount of flexibility in the configuration of displays. To illustrate our claims, we present several applications including a cinematographic design tool which showcases many features of our system.

1. Introduction

Technical progress in recent years gives reason to believe that virtual reality (VR) has a good potential as a user interface of the future. At the moment, VR applications are usually tailored to the needs of a very specific domain, such as a theme park ride or virtual mock-up. We believe that augmented reality (AR), the less obtrusive cousin of VR, has a better chance to become a viable user interface for everyday productivity applications, where a large variety of tasks has to be covered by a single system. Rather than forcing a user to deal exclusively with a virtual environment, less rigid approaches like UNC's *office of the future* [12] embed VR and AR tools in a conventional work environment.

In a more general sense, this principle is known as *ubiquitous computing* [22], describing a world where computers are embedded in large numbers in our everyday surrounding, allowing constant access to networked resources. Some researcher argue that AR is the opposite of ubiquitous computing, because users carry their computing devices – such as head-mounted (HMDs) displays - to the places they go to rather than expecting the computing devices to be there already.

We believe that these two concepts are just extremes on a scale, and that a lot of useful user interface concepts can be found in between. Our current work on the *Studierstube* project, which started as a pure augmented reality setup [15], focuses on experimenting with the possibilities of new user interfaces that incorporate AR. For efficient experimentation, we have implemented a toolkit that generalizes over multiple user interface dimensions, allowing rapid prototyping of different user interface styles. The *Studierstube* user interface spans the following dimensions:

1.1. Multiple users

The system allows multiple users to collaborate (Figure 1, Figure 2). 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*. However, we also allow multiple users to interface with a single host (e.g. via a large screen display), and a single user to interface with multiple computers at once. On a very fundamental level, this means that we are dealing with a distributed system. It also implies that *multiple types of output devices* such as HMDs, projection-based displays, hand-held displays etc. can be handled and that the system can span *multiple operating systems*.

1.2. Multiple contexts

Contexts are the fundamental units from which the *Studierstube* environment is composed. A context is a union of *data* itself, the data's *representation* and an *application* which operates on the data. Contexts are thus structured along the lines of the model-view-controller (MVC) paradigm known from Smalltalk's windowing system [6]: *Studierstube*'s data, representation, and application correspond to MVC's model, view, and controller, respectively. Not surprisingly, this structure makes it straightforward to generalize established properties of 2D user interfaces to three dimensions.

In other words, a context encapsulates visible and invisible application-specific data together with the responsible application. The notion of an application is therefore completely hidden from the user, in particular, users never have to "start" an application, they simply open a context of a specific type. Compared to the desktop metaphor, this approach is much closer to the concept of an *information appliance*, which is always "on" (compare [2]).

In a conventional desktop system, the data representation of a document is typically a single 2D window. Analogously, in our three-dimensional user interface, we define a context's representation as a three-dimensional structure contained in a certain volume – a *3D-window*. Unlike its 2D counterpart, a context can be shared by any group of users, and even more importantly, can be present in *multiple locales* simultaneously by replication.

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*, a feature which is well established within the desktop metaphor, but rarely implemented in virtual environments. Moreover, *Studierstube* also allows multiple contexts of the same type to co-exist, allowing a single application to work with multiple data sets. In the desktop metaphor, this feature is generally known as a *multiple document interface*. Note that it differs from simply allowing multiple instances of the same application which are unaware of each other. Multiple contexts of the same type are aware of each other can share features and data. For example, consider the shared "slide sorter" from section 5.

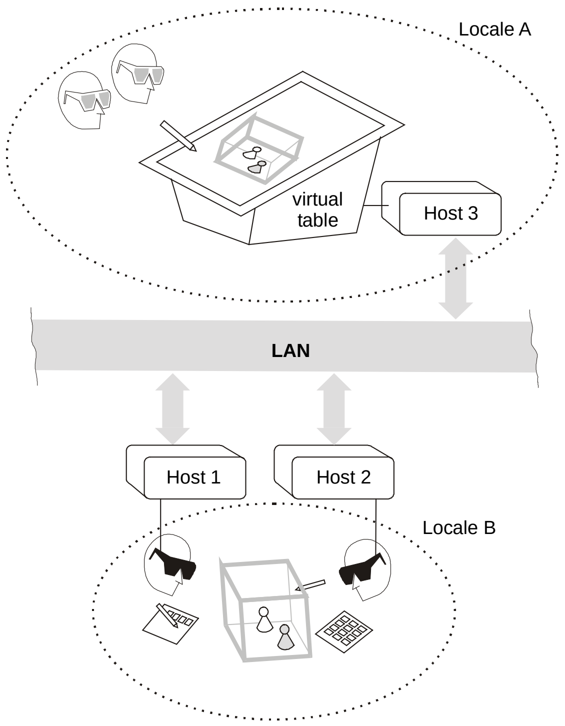


Figure 3: Multiple locales can simultaneously exist in *Studierstube*. They can be used to configure different output devices and to support remote collaboration

1.3. Multiple locales

Locales correspond to coordinate systems in the virtual environment. They usually coincide with physical places (such as a lab or conference room, or parts of rooms), but they can also be portable and associated with a user, or used arbitrarily – we even allow (and use) overlapping locales in the same physical space. We define that every display used in a *Studierstube* environment shows the content of exactly one locale. Every context can (but need not) be replicated in every locale; these replicas will be kept synchronized by *Studierstube's* distribution mechanism.

To understand why the separation of locales and contexts is necessary, consider the following examples:

- Multiple users are working on separate hosts. They can share contexts, but can layout the context representations (3D-windows) arbitrarily according to screen format and personal preferences. This is made possible by defining separate locales, as the position of 3D-windows is not shared across locale boundaries (Figure 3). The hosts can be in separate buildings for remote collaboration, or they can be placed side by side. In the latter case, locales would probably overlap, as users might see several or all screens.
- A user wearing a see-through HMD is looking at a large projection screen *through* the HMD. Both display devices (HMD, projection screen) can be set to use the same locale, so the graphics in a user's HMD may *augment* the projection screen's output. Of course this setup is view-dependent and works for only one user, so alternatively, the projection screen may use a separate locale, and present graphical elements which are complementary to the HMD output.

By separating locales (geometric relationships) from contexts (semantic relationships), we achieve a great amount of flexibility in the configuration of displays. This not only allows to connect multiple *Studierstube* environments over a network for remote collaboration, but also to set up an environment with multiple co-located, i. e., overlapping locales. Consider as a scenario a spacecraft mission control center with dozens of collaborating operators assembled in a large hall. Every involved user will assume a specific role and require specific tools and data sets, while some aspects of the mission will be shared by all users. A naïve approach of embedding all users in a single locale means that users in close proximity can work in a shared virtual space, while other users who desire to participate are too far away to see the data well, and are not within arm's reach for manual interaction. By separating contexts from locales, a remote user can import the context into a separate locale, and interact with it conveniently. While our available resources do not allow us to verify such large-scale interaction, in section 5 we present some results that back up our considerations.

The system presented in this paper must be understood as an experimental platform for exploring the design space that emerges from bridging multiple user interface dimensions. It can neither compete in maturity and usability with the universally adopted desktop metaphor nor with more streamlined, specialized virtual environment solutions (e. g., CAVEs). However, *Studierstube* demonstrates a design approach for next generation user interfaces as well as solutions on how to implement these interfaces.

2. Previous work

Almost a decade ago, Weiser introduced the concept of *ubiquitous computing* as a future paradigm on interaction with computers [22]. In his vision, computers are constantly available in our surrounding 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 head-mounted displays) to enhance a user's perception by overlaying computer generated images onto a user's view of the real-world.

Collaborative augmented reality enhances AR with distributed system support for multiple users with multiple display devices, allowing a co-located joint experience of virtual objects [3, 15]. Some researchers are experimenting with a combination of collaborative AR, ubiquitous computing and other user interface concepts. Prominent examples include EMMIE developed at Columbia University [4, 8], work by Rekimoto [13], and the Tangible Bits Project at MIT [9, 21]. These systems share many aspects with our approach for a collaborative augmented reality system making use of a variety of stationary as well as portable devices.

Working with such a system will inevitably require transfer of data from one computer's domain to another. For that aim, Rekimoto [14] proposes *multi-computer direct manipulation*, i. e. drag and drop across system and display boundaries. To implement this approach, a physical prop (in Rekimoto's case, a pen) is used as a virtual "store" for the data, while in reality the data transfer is carried out via the network using the pen only as a passive locator. Similar transfer functions are available in EMMIE [4]. Such use of passive objects as perceived media containers is also implemented by the Tangible Bits group's mediaBlocks [21].

Other sources of inspiration for multiple dimensions in 3D user interfaces are CRYSTAL [20], which allows concurrent execution of multiple applications in the same three-dimensional workspace, and SPLINE [1], which introduced the concept of multiple *locales* within one large virtual environment. Note that unlike SPLINE, *Studierstube's* allows multiple locales to overlap.

3. Background: *Studierstube*

The original *Studierstube* architecture [15, 18] was a collaborative augmented reality system allowing multiple users to gather in a room and experience the sensation of a shared virtual space that can be populated with three-dimensional data. Head-tracked see-through head-mounted displays (HMDs) allow each user to choose an individual viewpoint while retaining full stereoscopic graphics.

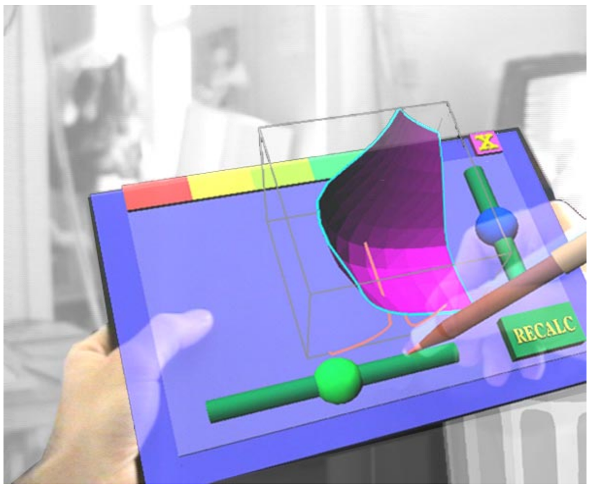


Figure 4: The Personal Interaction Panel combines tactile feedback from physical props with overlaid graphics to form a two-handed general purpose interaction tool.

The personal interaction panel (PIP), a two-handed interface composed of pen and pad, both fitted with magnetic trackers, is used to control the application [19]. It allows the straightforward integration of conventional 2D interface elements like buttons, sliders, dials etc. as well as novel 3D interaction widgets (Figure 4). The haptic feedback from the physical props guides the user when interacting with the PIP, while the overlaid graphics allows the props to be used as multi-function tools. Every application may display its own interface in the form of a PIP ”sheet”, which appears on the PIP when the application is in focus. The pen and pad are our primary interaction devices.

While the original *Studierstube* architecture from [18] incorporated simple distribution mechanisms to provide graphics from multiple host computers and shared data from a separate device (tracker) server, the initial networking approach later turned out to be insufficient for the evolving distribution requirements. An even more limiting factor was that the toolkit allowed to run only a single application and a single context at a time. Our efforts towards a follow-up version resulted in support for projection-based platforms [16] and a toolkit for distributed graphics [7]. This paper presents the results of a two-year long redesign process of *Studierstube* incorporating all these features into a new framework.

4. Implementation

Our software development environment is realized as a collection of C++ classes built on top of the Open Inventor (OIV) toolkit [17]. 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.

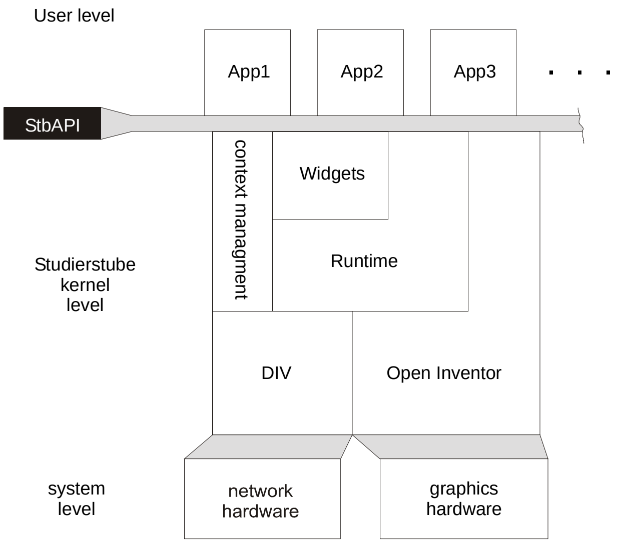


Figure 5: The *Studierstube* software is composed of an interaction toolkit and runtime system. The latter is responsible for managing context and distribution.

This has lead 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 5). 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 4.4). Applications are written and compiled as separate shared objects (.so for IRIX, .dll for Win32), and dynamically loaded into the runtime framework. A safeguard mechanism makes sure only one instance of each application 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 rarely found in virtual environment software.

By using this dynamic loading mechanism, *Studierstube* supports multi-tasking of *different* applications (e.g. a painting application and a 3D modeler), but also multiple concurrent contexts associated with the *same* application (Figure 6). This approach is similar to popular desktop systems such as the *multiple document interface*.

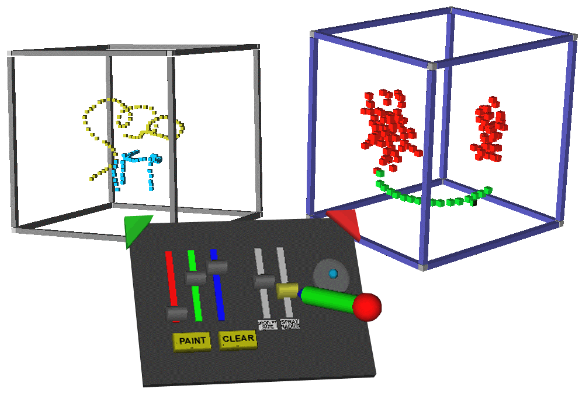


Figure 6: Multiple document interface in 3D – the right window has the user's focus and can be manipulated with the current PIP sheet.

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. A context – represented by its 3D-window - is owned by one user, and subscribed by others. Per default, a context is visible to all users and can be manipulated by any user in the locale.

4.1. 3D-windows

The use of windows as abstraction and interaction metaphor is a long-time convention in 2D GUIs. Its extension to three dimensions seems logical [5, 20] and can be achieved in a straightforward manner: 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 area and also defines its exact boundaries.

A context is normally represented in the scene by a 3D-window, although we allow a context 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 associated "decoration" that visually defines the windows extent and allows it to be manipulated with the pen (move, resize etc). The color of the decoration also indicates whether a window has a user's focus (and hence receives 3D event from that user). Like their 2D counterparts, 3D-windows can be minimized (replaced by a three-dimensional icon to save space in a cluttered display), and maximized (scaled to fill the whole work volume and receive input events exclusively). Typically, multiple context of the same type will maintain structurally similar windows, but this decision is at the discretion of the application programmer.

4.2. 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 geometry (e. g., 2D and 3D icons) that are useful to convey user interface state or merely as decoration.

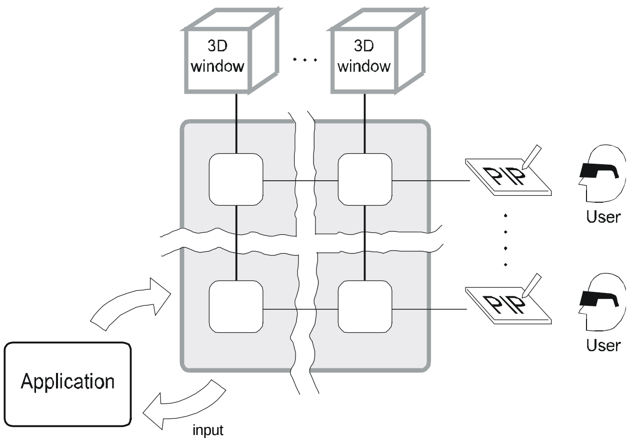


Figure 7: Multiplicity relationships in *Studierstube* - control elements on the PIP are instantiated separately for every (user, 3D-window) pair

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 our artistic spraying application (Figure 6) can be set individually by every user for every context. However, widgets can also be shared by all users, all contexts, or both. Consequently, *Studierstube's* 3D event routing involves a kind of multiplexer between windows and users' PIP sheets (Figure 7).

4.3. Distributed execution

The distribution of *Studierstube* requires that for each replica of a context all graphical and application-specific data is locally available at each host which has a replica.

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.

For that aim, we have created Distributed Open Inventor (DIV) [7] 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*. A scene graph need not be totally replicated – local variations (compare [10]) in the scene graph can be introduced, which is among others useful for fine-tuning low-latency operations such as dragging.

More importantly, local variations allow us to resolve distribution on a *per-context* base. A context is owned by one workstation (called a master context), which will be responsible of processing all relevant interaction on the application, while other workstations (in the same locale and in other locales) may replicate the context (as a slave context).

The roles that contexts may assume (master or slave) affect the status of the context's application part. The context data and its representation (window, PIP sheet etc.) stay synchronized over the whole lifespan of the context for every replica. The application part of a master context is active and modifies context data directly according to the users' input. 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. Note that context replicas can swap roles (e. g., by moving master contexts to achieve load balancing), but at any time there may only be one master copy per replicated context.

The replication on a per context-base provides coarse-grained parallelism. At the same time the programming model stays simple, as the programmer is spared to solve difficult concurrency issues and all relevant input can be processed in a single address space.

Once 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 its directory services are essential. For example, it maintains a list of all active hosts and which contexts they own or subscribe, it gets to decide about 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.

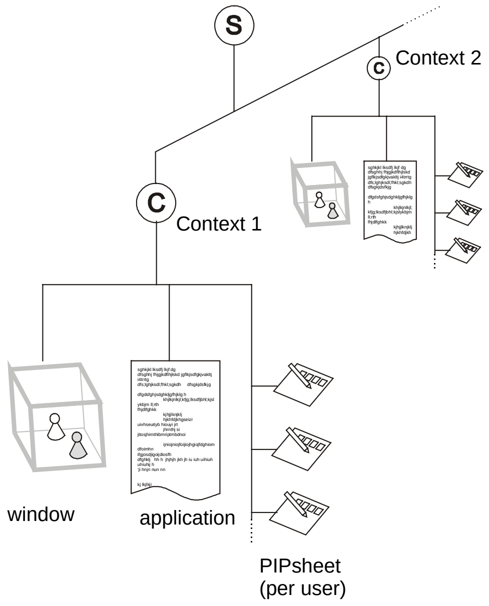


Figure 8: A context is implemented as a node in the scene graph, as are windows and pip sheets. This allows to organize all relevant data in the system in a single hierarchical data structure.

4.4. 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 makes sure the application allows multiple contexts to be created (i. e., offers the equivalent to a multiple document interface), each of which 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 achieve all 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 an array of PIP sheets, which are also special scene graphs. In other words, data, representation, and application are all embedded in a single scene graph (Figure 8), 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 button press or 3D direct manipulation events. Although the programmer has great freedom to use anything that the OIV and *Studierstube* toolkits offer, it is a requirement that any instance data is 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 format 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 both 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 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, our API provides a distributed multiple document interface.

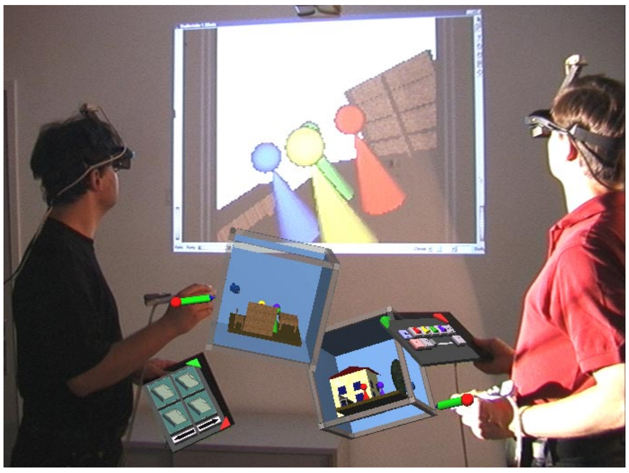


Figure 9: Storyboard application with two users and two contexts as seen from a third "virtual" user used for video documentation. In the background the video projection is visible.

5. Results

To demonstrate our framework, 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, a kind of *world in miniature* [11].

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 9, Figure 10).

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.

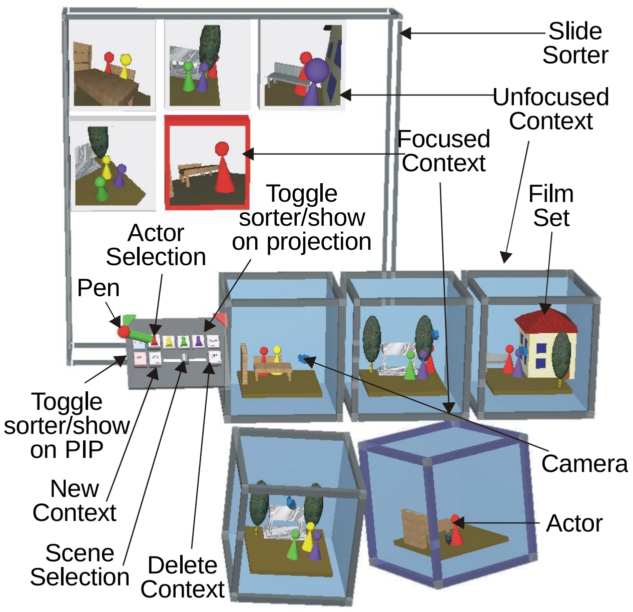


Figure 10: The Storyboarding application allows the 3D placement of actors, props, and cameras. The slide sorter shows a storyboard of all camera "shots"

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.

Using the distributed *Studierstube* framework, we ran the Storyboard application in different configurations.

5.1. Heterogeneous displays

Our first configuration (Figure 9, Figure 11) consisted of three hosts (SGI Indigo2, Intergraph TZ1 Wildcat, SGI O2), two users, and two *locales* (Figure 12). 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 pad each. The Intergraph workstation was driving an LCD video projector to generate a monoscopic image of the projection screen (without viewpoint tracking) on a projection wall. The slider show/sorter 3D-window was hidden from graphics output on the HMDs, so the users could see the result of their manipulation of the miniature scenes on the large bright projection exploiting the see-through capability of the HMDs.

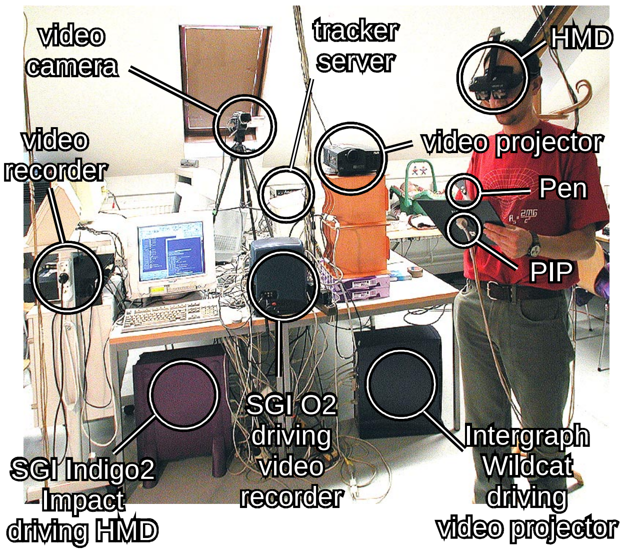

video camera
tracker server
HMD
video recorder
video projector
Pen
PIP
SGI O2 driving video recorder
SGI Indigo2 Impact driving HMD
Intergraph Wildcat driving video projector

Figure 11: Hardware setup for the heterogeneous display experiment

Users were able to perform some private editing on their local contexts, 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 choose 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 for a virtual user, whose position was identical with the physical camera. This feature was used to document the system on video. The configuration used two locales, one shared by the two users and the O2, while a separate locale was used for the Intergraph driving the projection screen (again viewed by a virtual user). The additional video host allowed us to perform live composition of the users' physical and virtual actions on video, while the video projector driving the projection screen could be freely repositioned without affecting the remainder of the system (Figure 12).

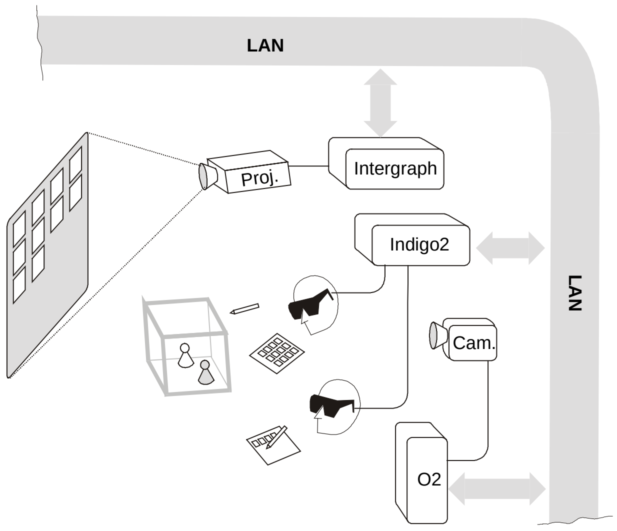


Figure 12: Heterogeneous displays – two users simultaneously see shared graphics (via their see-through HMDs) and a large screen projection

5.2. Symmetric workspace

The second example was intended to show multi-user collaboration in pure augmented reality with multiple hosts. The Storyboarding application was executed in a more conventional augmented reality setup consisting of two hosts (Indigo2, Intergraph), two users, and one *locale* (Figure 13). Both users were wearing HMDs again, but the first user was connected to the Indigo2, while the second user was connected to the Intergraph. In this configuration, the slide show/sorter was included in the graphics shown via the HMD rather than projected by a separate video projector.

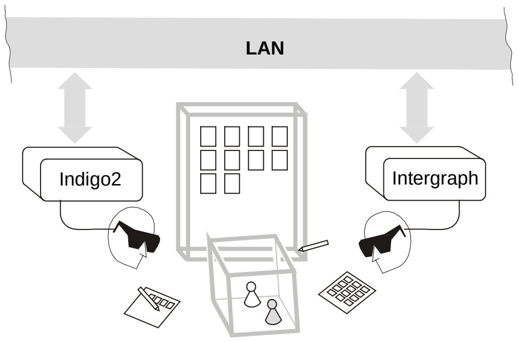


Figure 13: A symmetric workspace configuration uses homogeneous displays (2 HMDs) to present a shared environment to multiple users in one locale

While the obtainable frame rate was significantly higher than for the first configuration, since rendering load for the two users was distributed over two hosts, no high resolution wide field-of-view projection was available for the slide show/sorter. Consequently, only one locale was necessary since users shared the same physical space.

5.3. Remote collaboration

The third example was created to show remote collaboration of multiple users. In this setup, we built a second *Studierstube* environment in the laboratory next door to experiment with the possibilities of remote collaboration. We then let two users collaborate remotely using the Storyboard application.

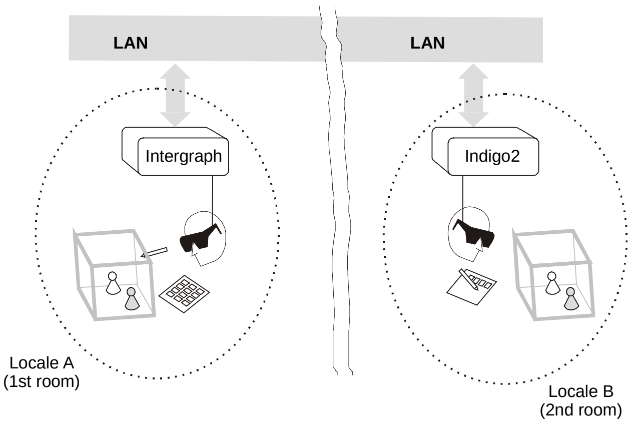


Figure 14: Remote collaboration: Two geographically separated users experience a shared environment

Note that the results are preliminary in the sense that all hosts were connected to the same LAN segment, and network performance is thus not representative of what one would get over a wide area network connection. However, this was not the current focus of investigation.

The system consisted of two hosts (Intergraph in the first laboratory, Indigo2 in the second), two users and two locales (Figure 14). Each user was wearing a HMD connected to the local workstation. In contrast to configuration from section 5.2, two locales were used as the users did not share a physical presence. The sharing of context, but not locale, allowed them to rearrange their personal workspace at their convenience without affecting collaboration.

5.4. Multiple applications

Finally, Figure 1 and Figure 2 show users working with multiple applications such as spraying, painting, and object viewing tools on two possible platforms: HMDs and a large polarized stereo projection wall.

6. Discussion

As observed by Tsao and Lumsden [20], in order 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 re-configured to execute a particular application, i. e., there is a separation of VR system software and application software.

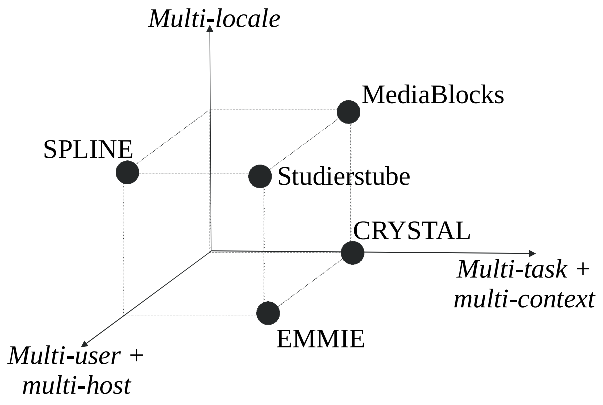


Figure 15: Extended taxonomy for multiple dimensions of user interfaces with some related work (adapted from CRYSTAL).

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 2D 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 2D user interfaces become applicable to 3D, as is evident with *Studierstube’s* PIP, 3D-windows, or 3D event system.

However, the CRYSTAL system from [20] does not incorporate true multi-user operation, and consequently has no need for multiple locales. Extending the taxonomy from CRYSTAL, Figure 15 compares some relevant work. For example, MIT’s mediaBlocks [21] 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 multi-user scenarios were demonstrated.

In contrast, SPLINE [1] is designed towards multi-user interaction. While 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 café, while train rides take place on a train).

The closest relative to our work is Columbia’s EMMIE [4]. Except for explicit support of locales, EMMIE shares many basic intentions with our research, in particular 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 integrating multiple user interface dimensions before a successor to the desktop metaphor can emerge.

7. Conclusions and future work

We have presented *Studierstube*, a prototype user interface that uses collaborative augmented reality to bridge multiple user interface dimensions: Multiple users, context, and locales as well as applications, 3D-windows, hosts, display platforms, and operating systems. *Studierstube* supports collaborative work by coordinating a heterogeneous distributed system based on a distributed shared scene graph and a 3D interaction toolkit. This architecture allows to combine multiple approaches to user interfaces as needed, so that it becomes easy to create a 3D work environment, which can be personalized, but also lends itself to computer supported cooperative work.

Our implementation prototype shows that despite its apparent complexity, such a design approach is principally feasible, although much is left to be desired in terms of quality and maturity of hard- and software. However, addressing issues such as display update rate and tracking accuracy is out of scope of this work.

Our future interest will focus on bringing the element of mobility into the *Studierstube* environment. While the name *Studierstube* ("study room") may be no longer appropriate, we envision a portable 3D information space that allows 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

This project was sponsored by the Austrian Science Fund *FWF* under contract no. P-12074-MAT. Special thanks to Markus Krutz, Rainer Splechtna, Hermann Wurnig, and Andreas Zajic for their contributions to the implementation, to Zsolt Szalavári and Michael Gervautz for inventing the PIP, and to M. Eduard Gröller for his spiritual guidance.

Web information

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

References

1. Barrus, J., R. Waters, R. Anderson. Locales and Beacons: Precise and Efficient Support for Large Multi-User Virtual Environments. *Proc. VRAIS '96*, pp. 204-213, 1996.
2. Billinghurst M., J. Bowskill, M. Jessop, J. Morphett. A Wearable Spatial Conferencing Space, *Proc. ISWC '98*, pp. 76-83, 1998.
3. Billinghurst M., 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.
4. Butz A., T. Höllerer, S. Feiner, B. MacIntyre, C. Beshers. Enveloping Computers and Users in a Collaborative 3D Augmented Reality, *Proc. IWAR '99*, pp. 1999.
5. Feiner S., C. Beshers. Worlds Within Worlds: Metaphors for Exploring N-Dimensional Virtual Worlds, *Proc. UIST '90*, pp. 76-83, 1990.
6. Goldberg A., D. Robson. *Smalltalk-80: The language and its implementation*. Addison-Wesley, Reading MA, 1983.
7. 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.
8. 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.
9. Ishii H., B. Ulmer. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms, *Proc. CHI '97*, pp. 234-241, 1997.
10. MacIntyre B., S. Feiner. A Distributed 3D Graphics Library, *Proc. SIGGRAPH '98*, pp. 361-370, 1998.
11. Pausch R., 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.
12. Raskar R., 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.
13. Rekimoto J. A Multiple Device Approach for Supporting Whiteboard-based Interactions, *Proc. CHI '98*, pp. 344-351, 1998.
14. Rekimoto J. Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments, *Proc. UIST '97*, pp. 31-39, 1997.
15. Schmalstieg D., A. Fuhrmann, Zs. Szalavari, M. Gervautz. Studierstube - Collaborative Augmented Reality, *Proc. Collaborative Virtual Environments '96*, Nottingham, UK, Sep. 1996.
16. 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.
17. Strauss P., R. Carey. An object oriented 3D graphics toolkit, *Proc. SIGGRAPH '92*, pp. 341-347, 1992.
18. Szalavári Zs., 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.
19. Szalavári Zs., M. Gervautz. The Personal Interaction Panel - A Two-Handed Interface for Augmented Reality, *Computer Graphics Forum*, 16(3), pp. 335-346, Sep. 1997.
20. Tsao J., C. Lumsden. CRYSTAL: Building Multicontext Virtual Environments, *Presence*, 6(1), pp. 57-72, 1997.
21. Ullmer B., H. Ishii, D. Glas. mediaBlocks: Physical Containers, Transports, and Controls for Online Media, *Proc. SIGGRAPH '98*, pp. 379-386, July 1998.
22. Weiser M. The Computer for the twenty-first century. *Scientific American*, pp. 94-104, 1991.

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

Gernot Schaufler
gs@graphics.lcs.mit.edu
Massachusetts Institute of Technology
Cambridge, MA

Sewing Worlds Together With SEAMs: A Mechanism to Construct Complex Virtual Environments

Abstract

This paper introduces the Spatially Extended Anchoring Mechanism (SEAM) as a 3-D user-interface metaphor to connect virtual worlds and manage scalability in distributed virtual environments. SEAMs provide a visual and navigable connection between worlds to manage both the complexity of rendering and network communication typically occurring in such environments. In the context of augmented reality, SEAMs can be applied as a 3-D window interface. A rendering algorithm is described which performs well on the graphics accelerators of standard personal computers.

1 Introduction

The idea of using virtual environments (VEs) as a communication medium or as a work environment is found intriguing by many supporters, and it has already been demonstrated that such technology can be successfully used for large audiences (Pausch, Snoddy, Taylor, Watson, & Haseltine, 1996). However, widespread commercial success of VEs is still not evident.

Why is it that the idea of cyberspace as first presented over a decade ago in William Gibson's novel *Neuromancer* (Gibson, 1984) is so enormously appealing, yet we see little manifestation of these ideas in our everyday technological environment? At least a partial answer may be found by comparing a future cyberspace to the most successful networked application of today, the World Wide Web (WWW).

One important property of the WWW that sets it apart from all previous media is the possibility of relatively simple participation by every user. The possibility to communicate with everyone else instead of just consuming the presented content (as with, for example, TV) fuels both individual and commercial interest and leads to the continuing fast growth of the Internet, the largest computer network to date. Distributed content development has the tremendous organizational advantage that no central authority is required for coordination (a requirement known to slow development efforts). This is a key factor in building truly scaleable virtual environments. For virtual environments, this means that users should be able to create 3-D environments and bring them online to allow others to interactively explore their creations and use them as a meeting place.

The document-oriented nature of the WWW makes the use of hyperlinks a

Presence, *Vol. 8, No. 4, August 1999,* 449–461

natural choice for navigation. We are used to the finite extent of documents, and to crossreferences within them that help us to navigate through larger collections of data. In contrast, a virtual environment mimics 3-D space, which is not per se decomposed into individual regions, but continuous. Unfortunately, distributed development requires that individuals concurrently and independently work on different regions of the virtual environment. These regions can be assembled into a continuous whole, but this approach requires a central coordination authority to control who gets to develop which region, which is difficult if not impossible to achieve on a global scale. (The WWW is already troubled by the comparatively simple issue of domain names.)

Alternatively, a VE may be broken down into parts without continuity among them. This has the advantage that unlimited space is available for every part, and is the approach taken by today's VEs on the Internet, based on the Virtual Reality Modeling Language VRML (Hartman & Wernecke, 1996). Traveling from one world to another is done by teleportation—instant transport to another position, which in VMRL is coupled to anchor objects, which allow the user to trigger the teleportation. In our experience, the use of teleportation is often confusing as one often triggers it accidentally and it does not have any correspondence to real-world travel. Bowman, Koller, and Hodges (1997) make the only formal evaluation documented in the literature, and our postulation that teleportation leads to disorientation is confirmed. However, teleportation has the advantage over continuous VEs that large distances can be covered in an instant, a property which we would like to retain.

In this paper, we will present a new mechanism for the construction, organization and navigation of complex virtual environments. The Spatially Extended Anchor Mechanism (SEAM) is a tool for the distributed development of continuous VEs without the need for a central authority. It allows instant travel over large distances without a disruption of continuity (Figure 1). We will point out how it can be used in distributed VEs of different origin and intent, from simple current VRML browsers to environments for large user communities in the sense of NPSNET (Macedonia, Zyda, Pratt, Brutzman, & Barham, 1995). We will also demonstrate how SEAMs are a superior mechanism for building immersive user interfaces based on 3D Magic Lenses (similar in spirit to Viega, Conway, Williams, & Pausch, 1996), which can be used to create another type of complex virtual environment.

2 The Spatially Extended Anchor Mechanism (SEAM)

The SEAM is a new modeling primitive to define the relationship between virtual worlds. A SEAM may be defined as a ''door into another world.'' An example is shown in Figure 1. The name was chosen to indicate that it is an extension of the well-known WWW anchors of VRML and because it expresses the function of geometrically connecting different spaces. The extent of the SEAM is defined by a single polygon, the SEAM polygon, which can be arbitrarily positioned. A complete SEAM definition is composed of a SEAM polygon, a reference (URL) to the world behind the SEAM, and a transformation matrix specifying the geometric relationship between the two worlds.

A user may not only look but also walk through the SEAM to enter another world or reach through the SEAM to manipulate another world. Hence, SEAMs are useful in a number of ways: as a mechanism to visually and spatially concatenate worlds, as a navigation metaphor, and as a 3D user-interface element.

Using SEAMs to connect VEs gives the authors of virtual worlds and the administrators of Internet-based distributed 3-D applications great freedom in the design and organization of their content. It also solves the problem of disorienting navigation via teleportation, because it is possible to look into the adjacent world before entering it and because there are no discontinuities in movement. Yet, SEAMs resemble teleportation in that they allow efficient coverage of large distances. SEAMs allow conventional Euclidean relationships among worlds to be established, but also to connect worlds in ways impossible with Euclidean geometry.

SEAMs improve on the idea of ''magic mirrors'' or ''wormholes'' frequently found in science fiction or fantasy by providing a more continuous transition between

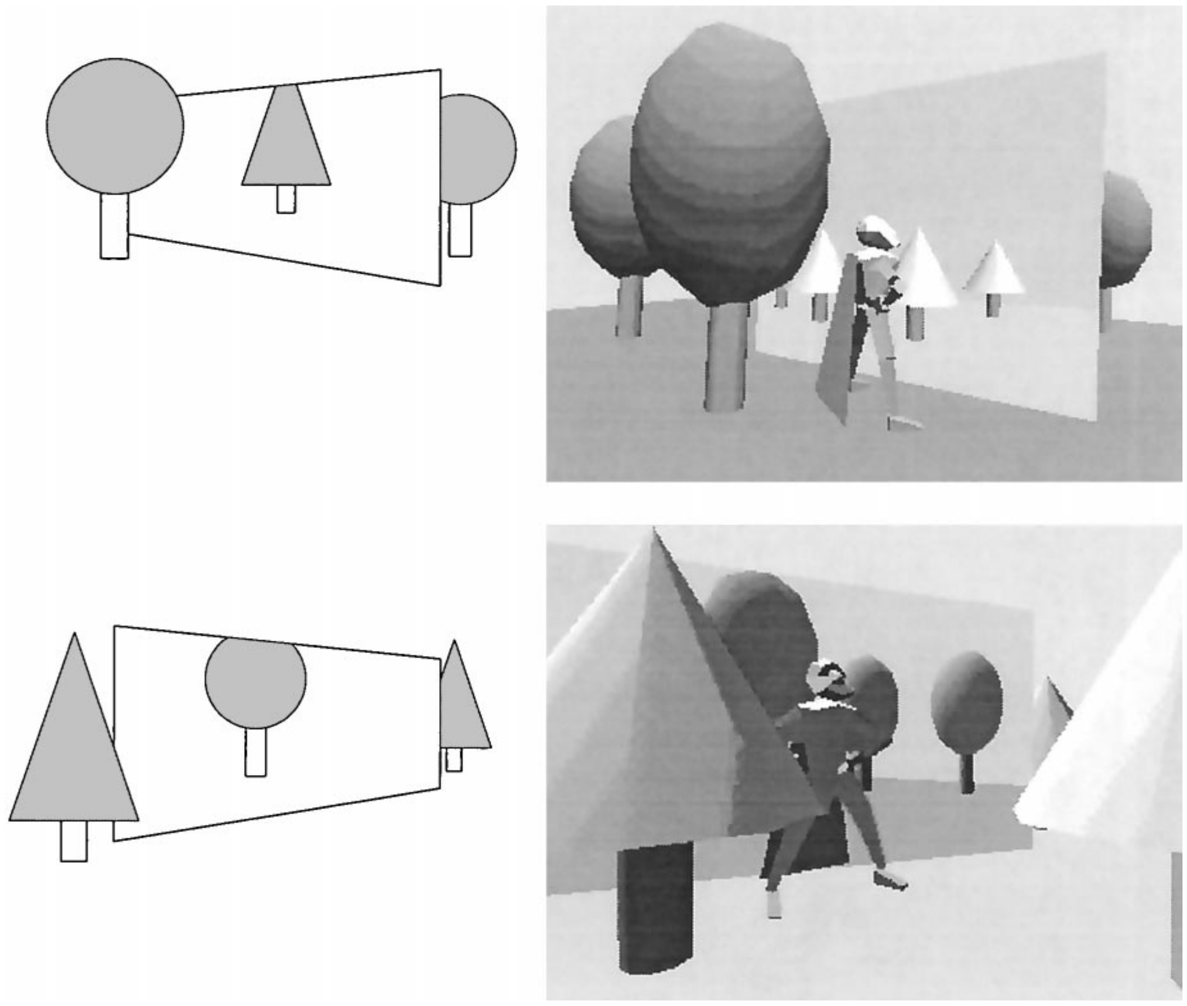


Figure 1. *SEAMs allow one to see and go from one world (round trees) into another (cone trees) and vice versa. Shown are a schematic view (left) and a screen shot (right).*

worlds. Movies like *Mary Poppins, Yellow Submarine* or *Stargate,* books like *Through the Looking Glass* (Carroll, 1872), or TV shows like ''Star Trek: Deep Space 9'' or ''Babylon 5'' (Figure 2) use the disruption in the transition to create tension and drama which is not always intended in a continuous virtual environment.

3 Related Work

Previously proposed VE research systems have aimed at the subdivision rather than the concatenation of virtual worlds: very large VEs are often subdivided into more manageable chunks (called ''regions,'' ''cells,'' or ''locales'') to exploit spatial coherence. This subdivision is used for visibility determination in real-time rendering and for reduction of network communication in distributed VEs. To our knowledge, little attention has been paid to using subdivision for modeling and organization of virtual environments.

Closest to our intentions is Diamond Park / Spline, developed at MERL (Barrus, Waters, & Anderson, 1996). It decomposes the virtual universe into ''locales'' that are associated with separate coordinate systems. Special attention was paid to precision in modeling and simulation. The work also introduced some consider-

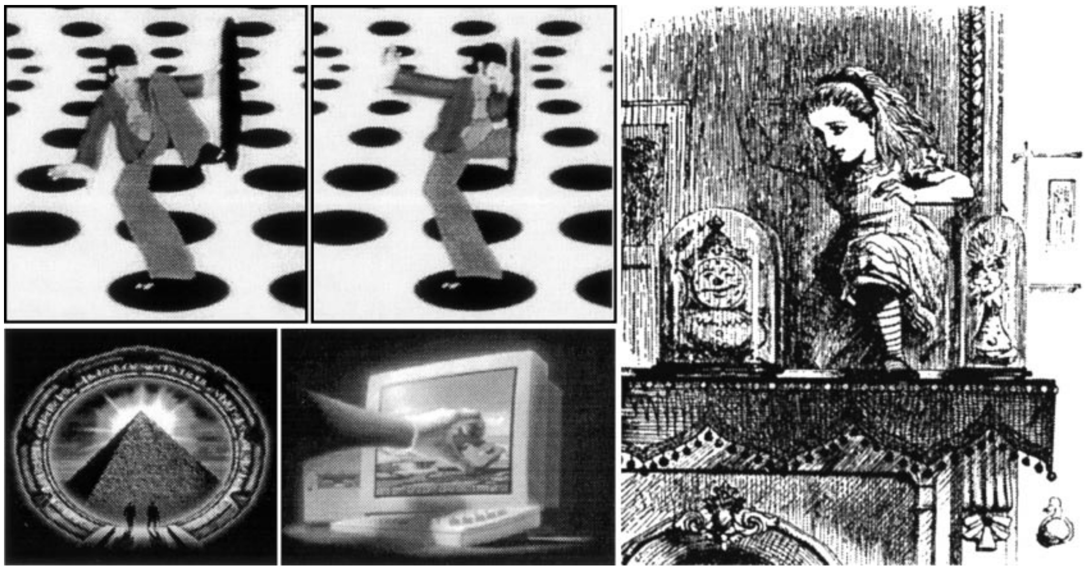


Figure 2. *Examples for connections of different worlds: Yellow Submarine (top left), Stargate (bottom left), advertisement by Superscape, Inc. (bottom middle), Through the Looking Glass (right).*

ations for world modeling and arrangement of locales that are similar to SEAMs, but that avoid the problems of visibility and overlapping worlds by using bent corridors and anterooms.

Visibility preprocessing for real-time rendering was first advocated in (Airey, Rohlf, & Brooks, 1990). A virtual environment is decomposed into cells for which mutual visibility is precomputed. Several approaches for exploiting visibility to accelerate rendering were proposed (for example in the work of Teller and Séquin (1991) and Luebke and Georges (1995).)

A large body of work has been dedicated to the optimization of network communication, which is crucial for large-scale VEs with a large number of participants. Virtual environments are subdivided either regularly based on visibility or into arbitrary regions. To reduce the amount of communication as much as possible, network protocols have been developed based on a multicasting topology such as NPSNET (Macedonia et al., 1995) or DIVE (Carlsson & Hagsand, 1993), a client-server topology such as NetEffect (Das, Singh, Mitchell, Kumar, & McGhee, 1997) or hybrids of both such as RING (Funkhouser, 1996) or Community Place (Lea, Honda, Matsuda, & Matsuda, 1997). Multicasting allows efficient delivery of messages to many participants. However, it loses much of its efficiency when transported over a wide area network, and is not generally supported on today's Internet. Furthermore, maintenance of multicast group membership is costly in terms of performance and only tractable for special types of application (Macedonia et al., 1995). Client-server architectures perform efficient message filtering, but they can suffer from bottlenecks if the server is overloaded. Hybrid topologies can merge the advantages of both, but no single approach has yet been adopted by Internet users as a standard. In Section 6, we will discuss how SEAMs can be a useful tool for implementing distributed VEs that are independent of the networking approach.

4 Rendering SEAMs

In this section, we explain how SEAMs are efficiently rendered using standard hardware acceleration. Technically, a SEAM is a polygon through which a live image of another world is visible. Currently, we employ one polygon per SEAM, but there is no conceptual reason why a SEAM could not be a polygonal mesh or a

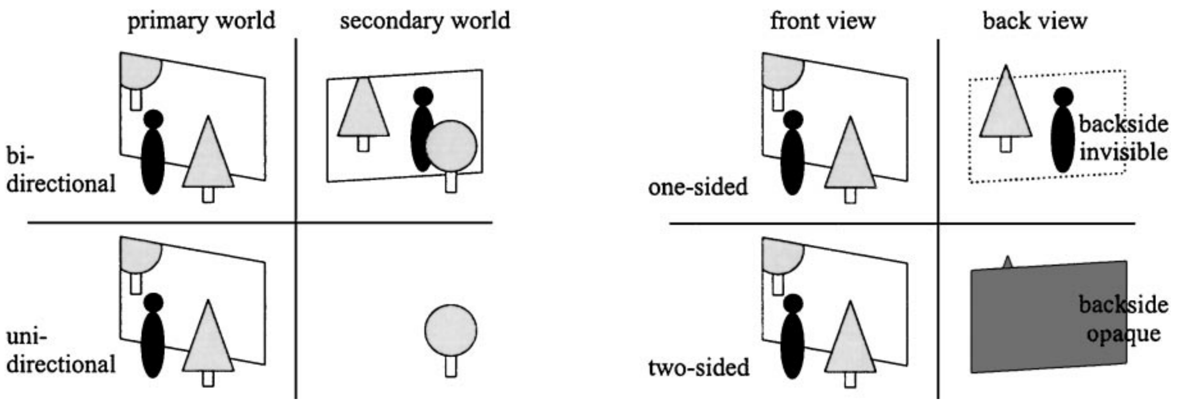


Figure 3. *Different types of SEAMs.*

nonplanar primitive. The polygonal nature of the SEAM, however, allows us to exploit the capabilities of hardware-assisted polygon renderers (Neider, Davis, & Woo, 1993).

4.1 Terminology

To aid in the explanation of the rendering method, we will introduce a few terms. (See also Figure 3.) The world that currently contains the user and the SEAM is called *primary world*, and the world behind the SEAM is called *secondary world*. Any world that can only indirectly be accessed via multiple SEAMs is called a *higher-order world*. The union of all worlds is called the *virtual universe*. SEAMs can be *unidirectional* (there is only a connection from the primary to the secondary world), or *bidirectional* (the connection is two-way). A *one-sided* SEAM can be observed only from the front, and is invisible from the back (in the primary world), whereas a *two-sided* SEAM is visible from the back. (It may show the secondary world, but it may also be rendered opaque, e. g., as a gray polygon.)

4.2 Rendering a Single SEAM

The standard approach to rendering three-dimensional scenes (the primary world) is to render each object in turn, solving the problem of occlusion with a *Z*-buffer. A SEAM object requires us to render a picture of yet another scene (the secondary world) onto the SEAM's surface in a view-dependent manner.

Our approach for rendering a SEAM generates the image of the secondary world at the right position in the image of the primary world. For correct visibility, the rendering of the image of the secondary world must be restricted to the area covered by the visible portion of the SEAM polygon in the image of the primary world (Figure 4). This approach is similar to a method suggested by McReynolds and Blythe (1997).

The mentioned confinement of the secondary world's image to the area of the SEAM polygon is realized using a mask in the ''stencil buffer'' (Neider et al., 1993), which can allow or disallow graphics output on a per-pixel basis. The details of the rendering are as follows (Figure 4):

The geometry is given as a directed acyclic[1] graph (scene graph). The scene graph of the primary world is traversed and rendered. When a SEAM is encountered, the associated polygon is passed to the rendering hardware for scan conversion. For all pixels of the SEAM polygon found to be visible, the mask (stencil buffer) is set to 1, the frame buffer is set to the background color of the secondary world (clear screen), and the *Z*-buffer is

1. May be a directed *cyclic* graph if recursive references are allowed (secondary or higher-order world is the same as primary world).

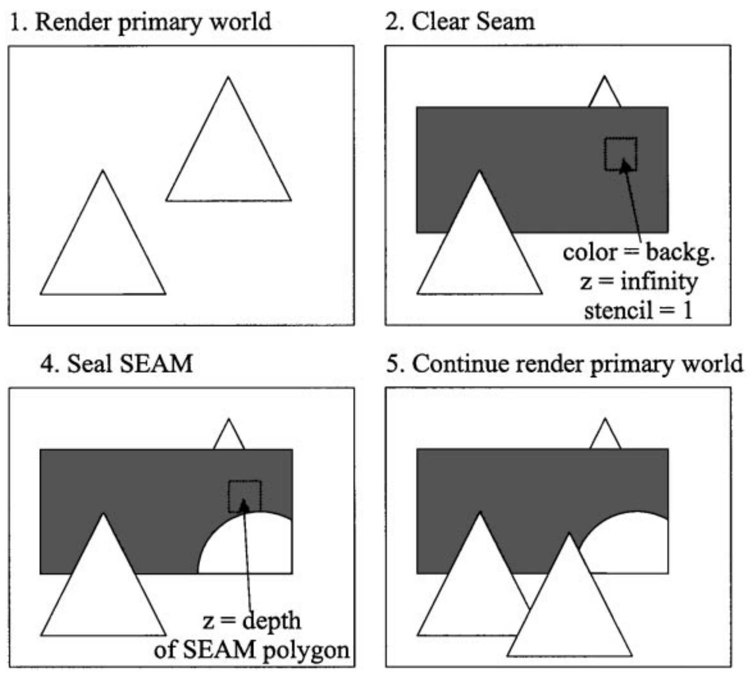


Figure 4. *Process of rendering a SEAM.*

set to infinity (clear *Z*-buffer). Note that these image modifications are carried out only for the visible portion of the SEAM surface and that the second and third steps can be carried out in a single pass.

After this preparation, rendering of the secondary world is performed inside the stencil mask created in the previous step (so that the secondary world is not drawn outside the SEAM area), and with a clipping plane coincident with the SEAM polygon (so that the secondary world does not protrude from the SEAM).

Finally, before rendering of the primary world continues, the SEAM polygon is rendered again, but only the computed depth values are written into the *Z*-buffer. Thereby the SEAM is ''sealed.'' The resulting *Z*-values are all smaller than any *Z*-value of the secondary world. Consequently, no geometric primitive of the primary world located behind the SEAM will overwrite a visible pixel from the secondary world rendering.

4.3 Rendering Nested SEAMS

In a more complex case, a number of virtual worlds is connected using multiple SEAMs (Figure 5).

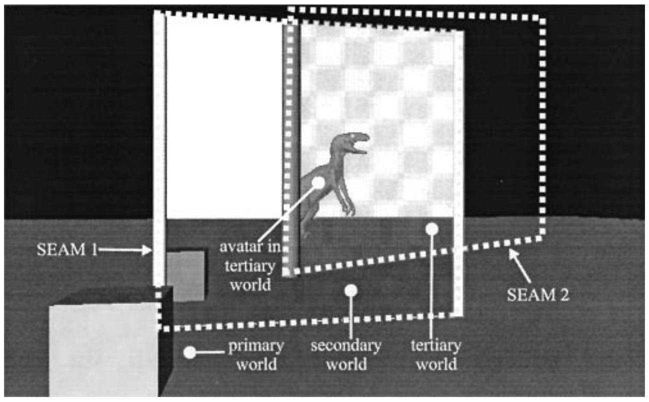


Figure 5. *Nested SEAMs.*

From certain viewpoints, users may be able to see not only a secondary world, but another SEAM leading to a third world, and so forth. Evidently, we need to modify the rendering algorithm to support recursive traversal of worlds during rendering. This is relatively straightforward. Traversal of multiple worlds connected by SEAMs is done in depth-first order, starting from the primary world. For every world, the algorithm proceeds as outlined above, except that the mask for a SEAM is created by incrementing the stencil mask value by one in the process of getting to the SEAM under consideration, thereby progressively reducing the set of pixels that may be affected. The visible area of the SEAM consists of the pixels where the mask value is equal to the recursion depth necessary to reach the SEAM.

An efficient implementation requires that only those worlds of which at least a small portion is visible are processed by the rendering algorithm. Otherwise, a large universe of concatenated worlds would require a potentially infinite number of primitives to be rendered, most of which are not visible at all. The problem is equivalent to the visibility problem for occluded interiors. Potentially visible set (PVS) methods (Airey et al., 1990) decompose a large geometric database into regions with visibility coherence (most of the geometry contained in such a region is visible from most viewpoints inside the region). Fortunately, the difficult part of PVS algorithms—identifying reasonable regions and the portals that connect them—is trivially solved for a virtual universe connected by SEAMs, because regions correspond to worlds and portals correspond to SEAMs, which are both included in the world description.

The simple and efficient algorithm proposed by Luebke and Georges (1995) solves the visibility problem in real time. It projects the portals (SEAMs) into screen space and intersects the screen-aligned bounding boxes (bbox). Typically, only a few SEAMs are visible in sequence, and so the aggregated bbox quickly decreases in size (Figure 6). An empty aggregated bbox indicates that no object can be visible through the considered sequence of SEAMs from the given viewpoint. Further optimization is done by determining the visibility of individual objects against the aggregated bbox. This is useful both for rendering and networking as scene graph traversal and rendering and network communication are only necessary for the visible portions of the universe.

5 SEAMs in Distributed and Multiuser Virtual Environments

SEAMs were conceived to support a virtual universe inhabited by a very large community of users.

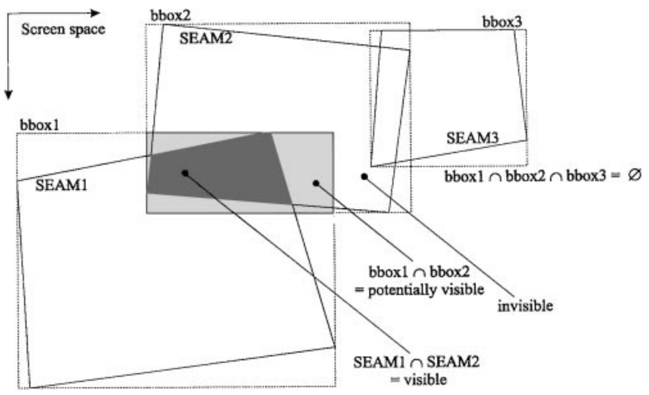


Figure 6. *Visibility algorithm.*

Large-scale multiuser VEs require a careful design of the underlying network architecture. Using today's Internet technology, multiuser VEs are mostly being built with client-server technology. A server may continue to simulate a virtual world, even if no participants are present in it. Each server provides via the Internet a part of the virtual universe (a world). It is responsible for the simulation of this world and can decide on the development of the simulation within this world.

A network architecture for the management of a distributed virtual universe faces a number of problems that may severely compromise performance and scalability. In the following sections, we discuss how to share a virtual universe constructed using SEAMs with Internet users and support live interaction of potentially large audiences.

5.1 Building a Continuous Distributed Universe

Consider the simple case of a single user traveling in a virtual universe composed of individual worlds connected via SEAMs. If multiple users are present in this universe, let us assume for the moment that they cannot perceive each other. Network communication for this configuration is necessary only to transmit the scene descriptions to the user. The user's client manages replicas of the primary world and of those higher-order worlds that can be seen through the SEAMs. This approach leaves the client with the simple task of rendering the worlds for the user. Servers need to consider SEAMs only so far as they have to instruct connected clients of the SEAMs' position and secondary worlds behind them.

Network communication can be reduced by transmitting only those parts of a world that are actually required for rendering (Schmalstieg & Gervautz, 1996). As the transmission of a world description may take some time, clients must anticipate their needs and gather data to have it available in time for rendering. The primary world must be available in any case, but the download of higher-order worlds can be delayed. Only potentially visible worlds need to be considered for downloading, similar to the strategy presented by Funkhouser, Sequin, and Teller (1992).

5.2 Compatibility with Multiuser Network Architectures

In this section, we discuss how multiuser network architectures work together with SEAMs. We first give a review of the approaches to VE networking briefly mentioned in Section 3, and show how they can be used together with SEAMs. We will also explain how multiuser applications can be constructed from these networking approaches.

For true multiuser support, a user must be able to perceive the activities of other users as well as the static world description. Every user is assigned a visual representation (an avatar) in the virtual environment. For correct simulation, users must be aware of each other, which has been called the ''players and ghosts'' approach to simulation (Blau, Hughes, Moshell, & Lisle, 1992). Users have to keep each other informed about their respective actions, in particular changes in position.

In the implementation of multiuser virtual environments, an important issue is scalability. A substantial body of related work has been dedicated to the optimization of network communication. We will discuss one example for each of the relevant approaches.

- Multicasting in NPSNET: NPSNET (Macedonia, 1995) is a system for large-scale military simulation. It uses a hexagonal decomposition of the area. Each hexagonal region is associated with a network multicast group, so that simulation updates (e. g., avatar position) in a particular region are communicated to effected participants only.
- Hybrid client-server / peer-to-peer topology in RING: Funkhouser's RING system (Funkhouser, 1996) is aimed at the simulation of densely occluded environments (e. g., building interiors). It uses a hybrid client-server / peer-to-peer network topology.
- Client-server architectures on the Internet: NetEffect (Das et al., 1997) and Internet game servers (Origin, 1997) are virtual environments aimed specifically at simultaneous participation of thousands of users via the Internet. It is based on a client-server architecture, and uses sophisticated methods such as message filtering and dynamic load balancing in a network of servers to achieve a high degree of scalability.

One may not overlook that the choice for client-server architectures on the Internet is also guided by commercial considerations such as access control and billing. However, currently operational solutions indicate the assumed scalability problem of servers (they may become a bottleneck) does not seem as pressing as it did some years ago.

We stress that SEAMs are compatible with all networking architectures outlined above as long as they subdivide a virtual universe into individual worlds. SEAMs do not resolve the scalability issue itself, but they provide an important building block in the design of large-scale multiuser VEs. Constructing a virtual universe with SEAMs also allows visibility constraints to be exploited (as outlined in Section 4.3) to reduce the amount of necessary communication and hence improve scalability.

In brief, SEAMs extend the general possibilities of a multiuser, multiworld implementation with the following options:

- Non-Euclidean connections between worlds can be created to allow the construction of worlds other than the typical building interiors or dungeons. Arbitrary SEAMs can be used in virtual worlds in the

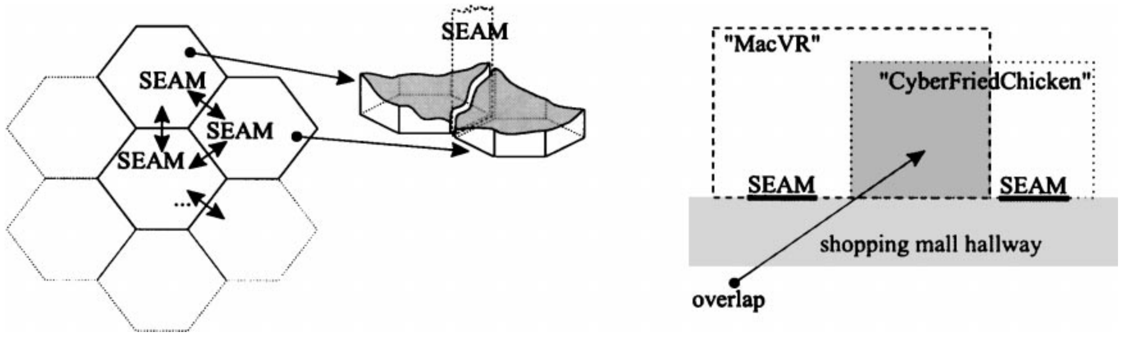


Figure 7. *(left) hexagonal world decomposition; (right) using space in a virtual shopping mall.*

same way anchors are now used in VRML; in fact, a VRML browser may be extended to render anchors as SEAMs rather than static objects. TV screens, billboards, and other artifacts can become doors to other worlds (compare color plate 1).

- Maintenance of the virtual environment is completely distributed. Unlike RING for example, there is no need for a global map of the overall environment. New servers can be added to and removed from the running systems, and SEAMs can be opened into existing worlds. Thus, there is a natural match with the structure of the Internet.

5.3 Applying SEAMs in Multiuser Worlds

SEAMs allow the connection of virtual worlds in almost any desired fashion. However, the most useful applications do not necessarily result from exploiting this freedom to the fullest by building bizarre, geometrically impossible universes. A universe that is geometrically plausible may be easier to comprehend, and users will not lose orientation. Yet employing SEAMs for such a setup allows independent modeling of the individual worlds by different parties, and facilitates load distribution onto different servers.

Good examples for such a natural decomposition into adjacent regions include the rooms of a house, with individual rooms modeled as worlds and with SEAMs as doors. A room is a relatively small unit of simulation and can be simulated with great accuracy. The same principle applies on a larger scale to a virtual city. A frequently mentioned application for VEs that can be constructed much more elegantly with SEAMs is a virtual shopping mall, in which individual vendors offer commerce and clients are invited to browse and make a selection. This type of virtual universe is best presented as geometrically consistent, so that users are not surprised by what they see. Consequently, in such settings, SEAMs should always be two-sided, so that users may switch back and forth between worlds as expected. SEAMs should be fitted into doorways or aligned with architectural openings, so that the user does not even notice the transition from one world to another. Free-standing SEAMs and other supernatural constructs are better avoided. This rule may sometimes intentionally be broken: in the shopping mall example, individual shops may be larger from the inside than from the outside, to provide space for shoppers. If users can access the shop through only one SEAM entrance, they may not even notice the implausibility (Figure 7, right). This idea has also been discussed by Barrus et al. (1996), but their system cannot accommodate neighboring overlapping worlds, which can easily be achieved with SEAMs.

Regular spatial setups like the hexagonal subdivision of NPSNET can be constructed with large bidirectional SEAMs. Every hexagon is a world with six SEAMs at the borders. Care must be taken that the playfield representation (e. g., terrain) is continuous at the borders (Figure 7, left). As users may theoretically see infinitely far over a flat landscape, there must be an artificial horizon, so that users can see through only a few SEAMs, possibly in combination with fog as an artificial limit.

If plausibility is not of importance, SEAMs can be used as improved 3-D hyperlinks. One-sided SEAMs can be used to reach any virtual world. In that way, ‘‘jump points’’ from one world to another can easily be created, e. g., in the form of a room with multiple doors. The advantage over teleportation is that users are able to see into worlds before they enter them, avoiding confusion and allowing inspection of activities. A user may peek into multiple ‘‘chat worlds’’ to find out where his or her friends are. Users can construct their personal ‘‘world access room’’ linked to other worlds in analogy to a WWW bookmark file. Other applications of this type include a traveling brochure where holiday destinations can be experienced not only from pictures but also in 3-D. (See color plate 1.) If one-sided SEAMs are used, the users already present in the secondary world should not be confronted with a new avatar that suddenly pops into the world, but a more appealing presentation (e. g., fade-in) would be preferable.

6 SEAMs as a User-Interface Tool

Besides their application in virtual worlds, SEAMs can be used as a user-interface primitive for immersive or augmented-reality environments. They are the 3-D equivalent to a 2-D window interface, but with the unique property that a user may reach into them to manipulate live 3-D applications. In our augmented-reality environment ‘‘Studierstube’’ (Szalavári, Schmalstieg, Fuhrmann, & Gervautz, 1998), we have implemented an application that arranges SEAMs in the working range of a user wearing a head-mounted display and a 3-D manipulation tool (Szalavári & Gervautz, 1997). (See color plate 4.)

Furthermore, SEAMs can be used as 3-D magic lenses. Magic lenses are unconventional ‘‘see-through’’ user-interface elements that extend the metaphor of a magnifying glass to any sort of useful visual transformation of the data provided by the application. This paradigm was introduced in 2-D by Bier, Stone, Pier, Buxton, & DeRose (1993), and later extended to 3-D magic lenses by Viega et al. (1996). In this section, we show that SEAMs are an equivalent to 3-D magic lenses, but without the shortcomings and limitations of the original implementation mentioned by Viega et al., as detailed below.

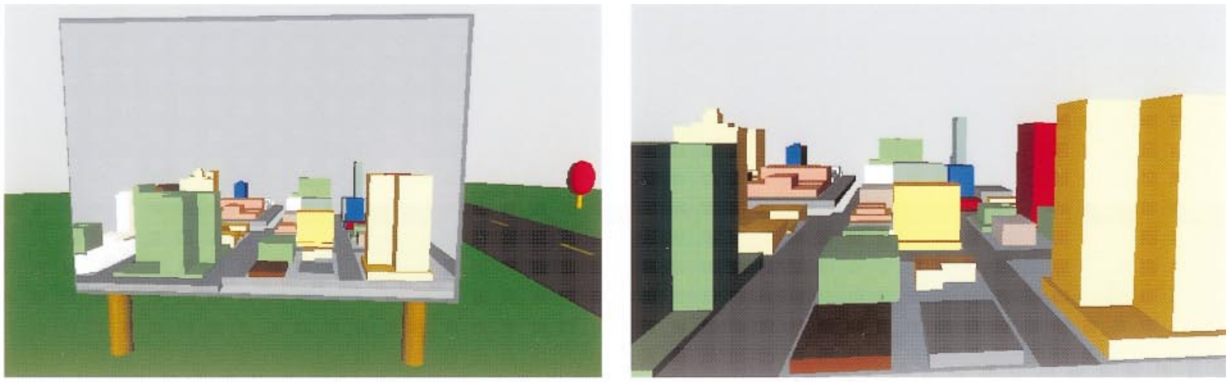
SEAMs are used to display a secondary world somewhere in a primary world. These worlds can have completely different content, but both worlds can also be different representations of the same content. In the latter case, a SEAM is perceived as a lens that modifies the content viewed through it.

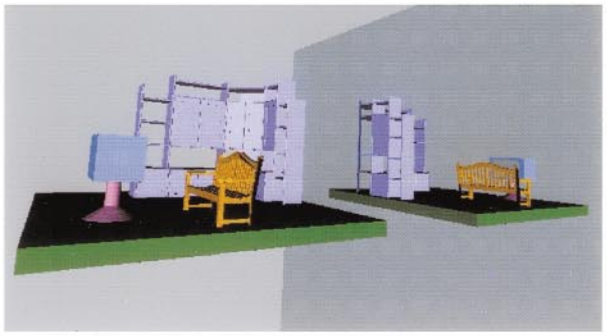
For example, color plate 5 shows a flat X-ray magic lens revealing the skeleton underneath the skin of a virtual human. A volumetric lens is shown in color plate 7, where a magic box is used for focusing: streamlines of a complex dynamical system are shown at two different levels of density; a user-selected focus defined by the extent of a magic box shows higher streamline density than the surrounding (Fuhrmann & Gröller, 1998). The magic box is composed of six SEAMs.

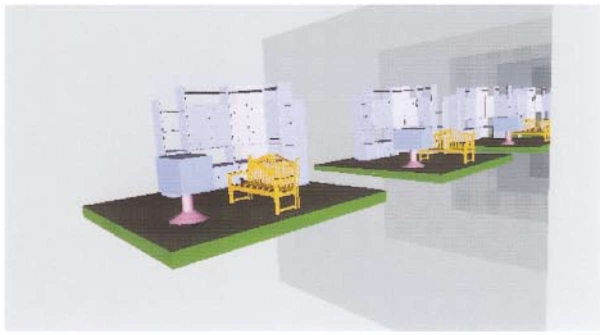
The rendering method for 3-D magic lenses adopted in the original implementation of Viega et al. (1996) was based on hardware clipping planes alone and has several drawbacks that severely affect the generality of the approach:

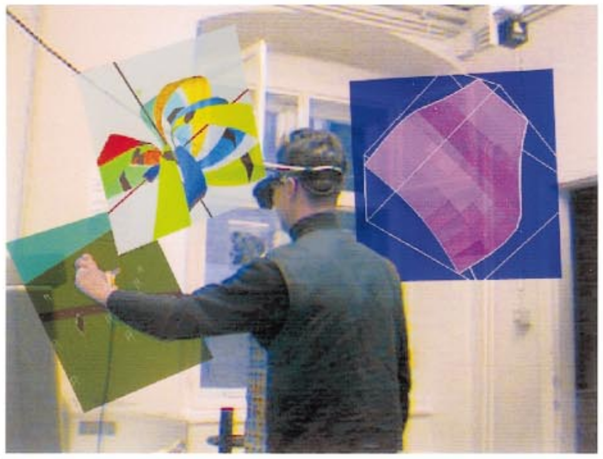
- Trivial rendering is possible only for convex polygons up to six sides. A simple round lens such as depicted in color plate 5 is not possible.
- A separate rendering pass of the same scene is necessary for each side of the lens, which can be a considerable overhead for real-time applications.
- Concave lenses or lenses with more than six sides must be decomposed into convex pieces and require even more rendering passes. The authors report that this solution has not been implemented.

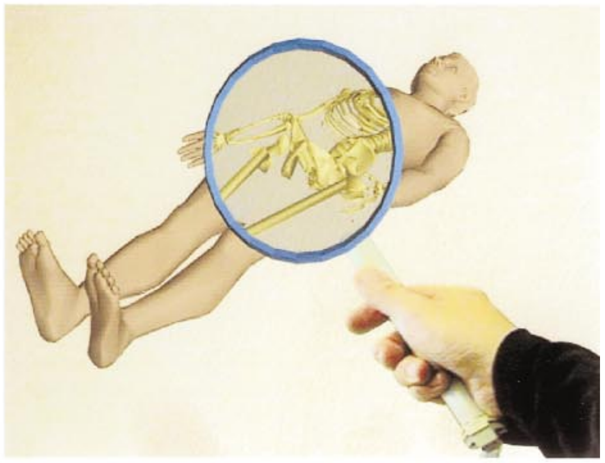
All the mentioned restrictions are easily overcome by using the stencil buffer along with the clipping planes, as our SEAMs implementation does. Only a single rendering pass is necessary, resulting in improved performance. Consequently, it is much easier to develop and use both flat and volumetric magic lenses, as our examples suggest. Ultimately, SEAMs can become the immersive

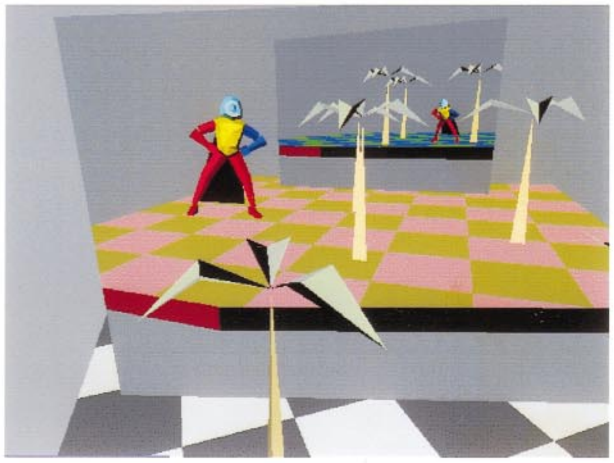

Color plate 1: Going through a billboard into another world


Color plate 2: A SEAM as a mirror


Color plate 3: Infinite worlds with recursive SEAMs


Color plate 4: SEAMs to access 3D augmented reality


Color plate 5: A SEAM as a 3D magic lens for X-raying


Color plate 6: Multi-user VRML browser with SEAMs

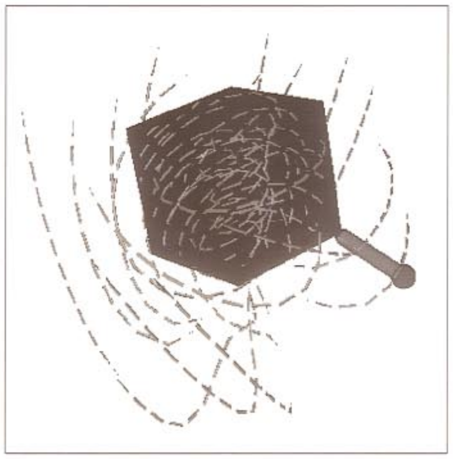

Color plate 7: 3D magic boxes for visualization focusing

Figure 8. *Color plates.*

equivalent to windows in conventional desktop systems, as the experiment depicted in color plate 4 suggests.

7 Implementation and Results

Our SEAMs implementation has been done in C++ and Open Inventor (Strauss & Carey, 1992). The SEAM primitive was embedded into what Open Inventor calls a ''node kit.'' Using Open Inventor software, our applications run on any platform with OpenGL support. As reasonably powerful OpenGL accelerator cards have recently become available at commodity prices to PC users, there is no obstacle to widespread use.

On top of the mentioned software setup, a simple multiuser VE was constructed (color plate 6). Worlds are constructed according to the VRML or Open Inventor file format, but are linked with SEAMs rather than anchors (as mentioned before, anchors can be interpreted and displayed as SEAMs). The result from an informal evaluation conducted using this implementation is that SEAMs are indeed a useful navigation paradigm that is generally preferable to instantaneous teleportation.

Our experimental multiuser support was limited to moving through the virtual universe (walk-through). We have implemented two variants of network support for multiple users: one based on multicasting, and one based on a client-server approach. As expected, multicasting is more efficient in delivering simulation data to the participants for a crowd of avatars that can see each other; message filtering performed by a server delivers better performance for more-complex virtual universes with many worlds, where visibility culling on SEAMs can be employed. We set up a trial using a crowd of computer-controlled participants (''bots'') together with two human users on two workstations connected via a nondedicated 10mbps ethernet and could successfully handle approximately fifty participants with several position updates per second.

8 Conclusions and Future Work

We have introduced a new mechanism for connecting virtual environments: SEAMs, which are essentially doors into another world. The key advantages are

- A technically new—yet culturally familiar—metaphor for navigating in virtual environments;
- A principle that allows to build and organize complex virtual worlds, and link them to other worlds without a centralized organizing authority; and
- Application as 3-D windows and magic lenses to construct user interfaces

Future work will involve creating more applications that make use of SEAMs and to experiment with different policies for the organization of a virtual universe.

Acknowledgments

This work has been supported by the Austrian Science Fund (*FWF*) under project numbers P-11392-MAT and P-12074-MAT. Special thanks to Anton Fuhrmann for his help. Further information and more visual results can be found at http://www.cg.tuwien.ac.at/research/vr/seams/.

References

Airey, J. M., Rohlf, J. H., & Brooks, F., Jr. (1990). Towards image realism with interactive update rates in complex virtual building environments. *Computer Graphics, 24*(2), 41–50.

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

Bier, E., Stone, M., Pier, K., Buxton, W., & DeRose, T. (1993). Toolglass and magic lenses: The see-through interface. *Proceedings of SIGGRAPH'93,* 73–80.

Blau, B., Hughes, C., Moshell, J., & Lisle, C. (1992). Networked virtual environments. *SIGGRAPH Symposium on Interactive 3D Graphics, 25*(2), 157–160.

Bowman, D., Koller, D., & Hodges, L. (1997). Travel in immersive virtual environments: An evaluation of viewpoint motion control techniques. *Proceedings of VRAIS'97*, 45–52.

Carlsson, C., & Hagsand, O. (1993). DIVE—A platform for multi-user virtual environments. *Computers & Graphics, 17*(6), 663–669.

Carroll, L. (first edition: 1872). *Through the Looking Glass.* (New York: MacMillan Publishing).

Das, T. K., Singh, G., Mitchell, A., Kumar, P. S., & McGhee, K. (1997). NetEffect: A network architecture for large-scale multi-user virtual world. *Proceedings of ACM Symposium on Virtual Reality Software and Technology* (*VRST'97*), 157–163.

Fuhrmann, A., & Gröller, E. (1998). Real-Time Techniques for 3D Flow Visualization. *IEEE Visualization '98.*

Funkhouser, T. (1996). ''Network Topologies for Scaleable Multi-User Virtual Environments.'' Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS'96) 222–229.

Funkhouser, T. (1984). Network topologies for scaleable multi-user virtual environments. *Proceedings of VRAIS'96*, 222–229.

Funkhouser, T., Sequin, C., & Teller, S. (1992). Management of large amounts of data in interactive building walkthroughs. *SIGGRAPH Symposium on Interactive 3D Graphics*, 11–20.

Gibson, W. (1984). *Neuromancer.* New York: Berkley Communications Group.

Hartman, J., & Wernecke, J. (1996). *The VRML 2.0 Handbook.* Addison-Wesley.

Lea, R., Honda, Y., Matsuda, K., & Matsuda, S. (1997). Community place: Architecture and performance. *Proceedings of ACM VRML'97*, 41–50.

Luebke, D., & Georges, C. (1995). Portals and mirrors: Simple, fast evaluation of potentially visible sets. *Proceedings SIGGRAPH Symposium on Interactive 3D Graphics*, 105–106.

Macedonia, M., Zyda, M., Pratt, D., Brutzman, D., & Barham, P. (1995). Exploiting reality with multicast groups. *IEEE Computer Graphics and Applications, 15*(3), 38–45.

McReynolds, T., & Blythe, D. (1997). Programming with OpenGL: Advanced rendering. *SIGGRAPH'97* (course notes).

Neider, J., Davis, T., & Woo, M. (1993). *OpenGL—Programming Guide, The Official Guide to Learning OpenGL.* Addison Wesley.

Origin (1997). *Ultima Online.* Online computer game.

Pausch, R., Snoddy, J., Taylor, R., Watson, S., & Haseltine, E. (1996). Disney's Aladdin: First steps toward storytelling in virtual reality. *Proceedings of SIGGRAPH'96*, 193–204.

Schmalstieg, D., & Gervautz, M. (1996). Demand-driven geometry transmission for distributed virtual environments. *Computer Graphics Forum* (*Proc. EUROGRAPHICS '96*) *15*(3), 421–433.

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

Szalavári, Zs., & Gervautz, M. (1997). The personal interaction panel—A two-handed interface for augmented reality. *Computer Graphics Forum (Proc. of EUROGRAPHICS'97, 16*(3), 335–346. Budapest, Hungary.

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

Teller, S., & Séquin, C. (1991). Visibility preprocessing for interactive walkthroughs. *Proceedings of SIGGRAPH'91, 25*(4), 61–69.

Viega, J., Conway, M., Williams, G., & Pausch, R. (1996). 3D magic lenses. *Proceedings of the ACM Symposium on User Interface Software and Technology* (*UIST'96*), 51–58.