Distributed Augmented Reality for Collaborative Design Applications

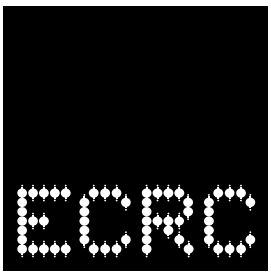
Klaus H. Ahlers, André Kramer,
David E. Breen, Pierre-Yves Chevalier, Chris Crampton,
Eric Rose, Mihran Tuceryan, Ross T. Whitaker,
Douglas Greer

ECRC-95-03

Technical report ECRC-95-03
(also published in Eurographics '95 Proceedings, Maastricht, NL, 1995)

Distributed Augmented Reality for Collaborative Design Applications

Klaus H. Ahlers, André Kramer,
David E. Breen, Pierre-Yves Chevalier, Chris Crampton,
Eric Rose, Mihran Tuceryan, Ross T. Whitaker,
Douglas Greer



European Computer-Industry Research Centre GmbH (Forschungszentrum)
Arabellastraße 17
81925 Munich
Germany
Tel. +49 89 9 26 99-0
Fax. +49 89 9 26 99-170

For more information please contact : Klaus H. Ahlers (colas@ecrc.de)
André Kramer (akramer@ecrc.de)

Abstract

This paper presents a system for constructing collaborative design applications based on distributed augmented reality. Augmented reality interfaces are a natural method for presenting computer-based design by merging graphics with a view of the real world. Distribution enables users at remote sites to collaborate on design tasks. The users interactively control their local view, try out design options, and communicate design proposals. They share virtual graphical objects that substitute for real objects which are not yet physically created or are not yet placed into the real design environment.

We describe the underlying augmented reality system and in particular how it has been extended in order to support multi-user collaboration. The construction of distributed augmented reality applications is made easier by a separation of interface, interaction and distribution issues. An interior design application is used as an example to demonstrate the advantages of our approach.

1 Introduction

The User Interaction and Visualization group at ECRC is currently investigating augmented reality (AR) techniques for a range of applications. In augmented reality, information that enhances or augments the real world is provided by the computer and incorporated into the reality of the user. This is in contrast to virtual reality, where the user is completely immersed in the world of the computer. With AR, the user interacts with the real world in a natural way, simultaneously using the computer to explore related information and to interact with virtual objects. Our work in this area is embedded in the context of a larger project to develop a general-purpose augmented reality platform.

Augmented reality can apply to the visualization of products in marketing, manufacturing, and design. The evaluation of a product design usually involves a demonstration of the designer's work. If the design addresses items that are intended to become visible parts of the real world, producing this visualization can be a lengthy and challenging task. The visualization should convincingly demonstrate that the design fits both spatially and aesthetically into its environment. To convey this information one could build physical scale models or mock-ups. A more recent approach involves computer models created through computer-aided design. While they can be produced in a shorter time, they cannot always show the design in its intended context. Virtual reality techniques put the design into a *virtual* context and allow one to explore it in the form of a "walk through" by using an immersive interface.

Figure 1.1: Trivial example of augmented reality: virtual juggling in a real room.

Augmented reality is a more natural and effective means to exhibit a design in its real-world context. A user is present in the real environment, or receives a view from a video camera on location. The user's view is then merged with a graphical representation of the design, either by using a see-through display or by combining video with graphics (Figure 1.1). This clearly presents the

relationships between the design objects and the real world. Using AR devices, users can interact with the design objects in a natural way, perceive and comprehend the design features easily, and judge the visual impact of the finished design. Compared with virtual reality, AR also has the advantage of not requiring an explicit and realistic rendering of the surroundings. It should be noted though that a convincing interaction between real and virtual objects requires detailed modeling beforehand.

A design process typically involves more than one person. In many cases there will be a team of designers participating in the design, along with the client. A design review draws in a possibly large number of people to inspect and discuss the result. The communication and cooperation of these people can be supported directly if we allow for multiple users in a design visualization application. The users have different goals, depending on their role in the process. Designers want to modify objects in a design space, simultaneously if possible. Participants in the design review want to view and browse the design independently. Part of the communication between these users is expressed in the visualization and the modification in real time.

In this paper, we focus on distribution and interaction aspects of our AR system. We investigate the main issues that arise from distributing an AR application between multiple users, and propose clear architectural solutions to the problem of sharing logical application objects over a network. Within each interface, there has to be a separation between a representation of logical objects and the shared objects themselves. The sharing on the logical level has to be maintained between distributed interfaces, which requires synchronization and consistency mechanisms for communication. In order to make the development of distributed AR applications easier, we implemented broadcasting and group communication services in a distributed programming environment. Our goal is to offer good control mechanisms and flexible support for interaction and collaboration tasks in distributed AR applications. To illustrate our approach, we present an application for collaborative interior design. The scenario for the application assumes users in different roles (e.g. client, colleague, consultant, designer) and different locations working together on the layout of a room.

2 Previous Work

Nakamae et al. [17] have experimented with image-based augmented reality for an architectural application. Today there are several research groups exploring augmented reality for a variety of interactive applications. Feiner et al. [13] have developed a knowledge-based AR system for maintenance and repair instruction. Lorensen et al. [14] have focused on AR for medical applications. At Boeing [21], AR is being developed to assist in manufacturing processes. Milgram et al. [16] have explored AR for enhancing tele-robotic interactions.

Cooperative design based on distributed realistic graphics has also been of interest to a number of researchers, both in the area of computer-supported cooperative work (CSCW) and graphics visualization. Shu and Flowers [20] have studied user interface issues in the context of 3D computer-aided design. Bentley et al. [4] have explored system architectures for the construction of cooperative multi-user interfaces. Shastra [2] is a collaborative environment for scientific design and manipulation applications. Fahlen et al. [12] have discussed issues pertaining to visualization, awareness, and interaction in the context of a distributed environment called DIVE. DIVE [7] is a VR toolkit for multi-user applications sharing synthetic worlds.

3 Augmented Reality System

The cooperative application described in this paper is implemented on top of a general-purpose augmented reality system [1]. The system currently supports a form of AR that combines a video signal from a standard video camera with computer-generated graphics. During the development of this system we focused on the issues of tracking, calibration, and user interaction. The core of the system is an interactive 3D computer graphics system, providing methods for representing and viewing 3D geometric models. Geometric primitives are organized hierarchically to produce complex 3D models. The camera model includes position and orientation and intrinsic camera parameters such as aspect ratio, focal length, and clipping information.

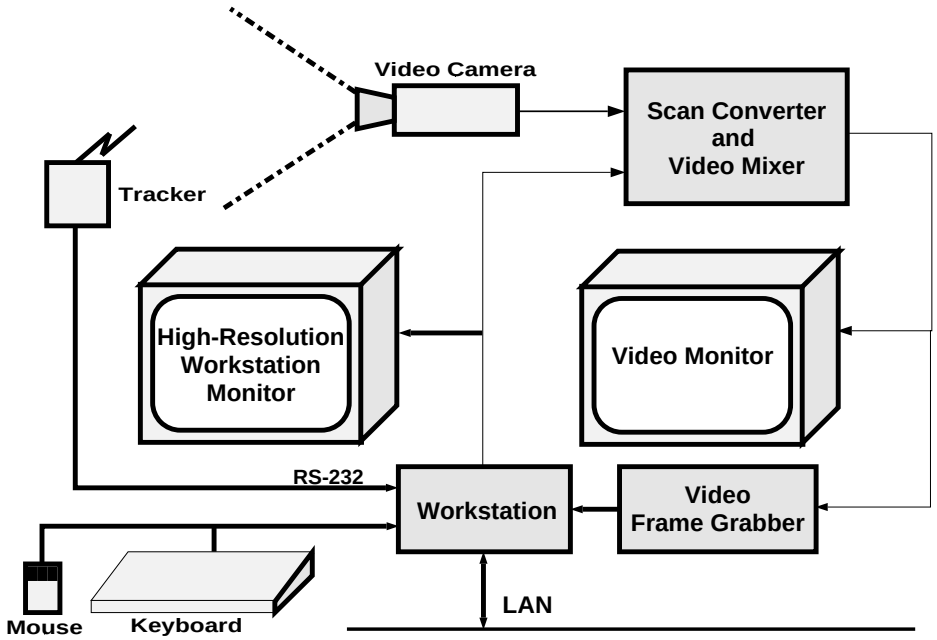


Figure 3.1: System hardware configuration

The hardware configuration is illustrated in Figure 3.1. The graphical image is generated by the workstation and displayed on the workstation's high

resolution monitor. The scan converter takes the relevant portion of the graphical image and converts it to standard video resolution and format. The scan converter also mixes the generated video signal with the video signal from the camera, thus merging live video and real-time graphics. We use luminance keying to allow the video signal to pass through any areas of the graphics that are black. The result is displayed on a standard video monitor.

A magnetic tracker provides six-degrees-of-freedom input. Receivers are attached to the camera and to a pointing device and provide the workstation continually with values for position and orientation. The pointer, and the workstation keyboard and mouse are used in the user interface, for setup and calibration, as well as for manipulation of application objects. A frame grabber acquires images from the camera. It is used during the initial calibration procedure as part of the interactive process that determines the optical characteristics of the camera.

4 Distributing Augmented Reality

User interfaces for AR applications are not as well explored as those of the traditional desktop model of computer interaction. Real world objects and interactions play a role in AR interfaces, as well as the computer-generated presentations. Distribution of AR has to address the special nature of these interfaces. The problem is difficult enough if the users are all present in the same "reality", and only the computer-based interface is distributed. If this is not the case, then there must be a way to share the real world between the users. The interface of a remote user becomes similar to a virtual reality with an immersive reconstruction of the real world. Ideally, the remote interface would be indistinguishable from the local AR environment. With current VR technology this cannot be achieved, so distributed AR applications have to find solutions that offer an acceptable degree of both realism and immersion.

For example, the interior design application shares the physical reality by distributing video frames and camera parameters to the remote sites. The user on location relies on video and graphics for the AR effect, and the remote user has a very similar interface with a video monitor showing the room to be furnished. There can be more than one camera at the "real" site. In the current implementation, remote users are sharing one camera, without having direct control of camera position and orientation.

4.1 Architecture for Distributed Visualization

The global architecture of the AR system determines many aspects of the implementation of interface distribution. A *centralized* architecture is easier to build, especially when an underlying networked window management provides

the distribution. However, need for rapid feedback in the user interface makes *replicated* control much more attractive for AR. The display management is local and each user is running an exact copy of the application. Bentley et al.[4] classify the different architectures and point out advantages and disadvantages. With replication, it is easier to maintain local, customized presentations for the shared "logical" objects. The separation of logical application objects from presentation objects is a prerequisite for *object-level sharing* or *loose coupling* of interfaces, so that each user can view and manipulate objects independently. The drawback is that a replicated architecture makes the management of shared information more complicated. Simultaneous updates that conflict with each other can lead to inconsistent interfaces.

The infrastructure of the system described in this paper is built in the form of a replicated architecture. This section stresses the logical separation between the shared data and the views in each interface, and explains how this concept is implemented in order to allow users to work with different presentations of the same information, and to interact with shared information simultaneously. A distributed environment (described in Section 5) provides the mechanisms needed to support the management of multiple user interfaces in real-time.

4.2 Model-View Paradigm

The world of virtual objects in an AR application incorporates the information that makes sense at the application level and is relevant to the users. Conceptually, this is a shared data base of logical objects, the "model". Because of the replicated architecture, the model is available as a copy in each instance of the application. The structure and type of the model is dependent on the application. In the interior design example we deal with a model that stores geometric data for a set of furniture. Each model object represents a piece of furniture, and maintains geometric transformations and visual attributes relevant to the object. The model objects are organized hierarchically, so that it is possible to select and interact with groups of furniture.

An interactive representation of the model in the user interface is called a "view". Views are created based on a specific interpretation of the model information. The interpretation is determined by the type of view, the type of model data, and the context of the interface. As an example, consider the furniture model of the interior design application and two views, a graphics rendering of the furniture and a browser for the items in the model. The rendering creates geometric primitives appropriate for the type of furniture, and geometric transformation and attributes are used directly to customize the presentation. On the other hand the browser creates a list of labels taken from the model objects and ignores all geometric information. Both views have presentation parameters that are not bound by the model interpretation and can be used for local customization of the interface.

The interpretation is not so straightforward for non-geometric model information. Highlighting a view object, for example, can be used to indicate a current selection in the model, but different views are likely to use different methods to show the highlight. This type of feedback becomes more complicated in multi-user applications with local selections at each site. In a distributed application, information about the state and actions of remote users becomes part of the model. It is important to give each user an awareness of who is participating and what other participants are doing. The interior design application makes use of an object browser that is capable of showing remote selections. In general, it can be a challenging task for the interface builder to find a concrete visualization for some abstract structure or behavior in the model.

The model-view mechanism is well known in the area of user interface construction, and used to implement a separation between interface and application functionality. The importance of separability for modularity and independent development of interface components has been recognized [9] even in non-distributed environments, where it is considered good software design. Yet separability becomes an indispensable architectural feature for distributed interfaces, at least for those with a need for object-level sharing. By using the model-view paradigm, we achieve the necessary independence between the conceptual objects of the global model and the objects and manipulations of a particular interactive view.

4.3 User Interaction and Model Update

Views provide the context for user interaction. During an interaction sequence views generate the feedback that is necessary for the user to understand the effect of his/her actions. Although the manipulation is directed towards view objects, conceptually the user is working at the model level and is assuming that the interactions effect changes in the model. After recognizing the modification of the representation, the view therefore has to update the model. If there is more than one view connected to the model, then the model in turn will inform the other views about the update (Figure 4.1). In the interior design application for example, furniture is selected by either selecting a browser item or picking the furniture directly using one of the graphics views. In both cases the model is informed about the selection, and it notifies the other views, who update their visual presentation.

The possible burst of update messages during an interaction sequence is curbed in several places. First, the view can decide that the update is local only and will not be sent to the model. This can be the case for local device feedback, generated as a transient part of the interaction. Then the model decides if the update is relevant on the logical, i.e. model level. Sometimes an update was triggered by a change in the representation that has no effect on the model data. A technical detail is the prevention of loops in the update

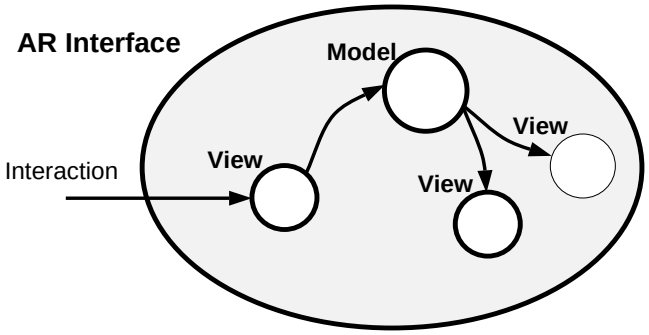


Figure 4.1: Update paths between model and views

scheme. The model, for example, updates views as a privileged agent, so that the view modification does not start a new round of updates.

Updating the model from a view reverses the data translation between model and view. Again there is the distinction between updates that directly modify data stored in the model and updates that need further interpretation. In the interior design application, moving furniture is translated directly into a change of the transformation information of the corresponding model object. Other interaction sequences or input commands will execute more complicated update methods in the model, as in the case of object selection, or object insertion or deletion.

For the sake of efficiency, there are no constraints on most updates. For some types of updates, it is unavoidable that the model or other views enforce a constraint that could not be checked while the user interaction took place. If this is the case, then the model will refuse the update, either on its own or after it failed to reach an agreement with the other views. The interior design demonstration makes use of this two-tiered update scheme for object selection. Any view can veto the selection or de-selection of model objects by another view. This means a user selection can fail if another view already "locks" the selection. Within one instance of the interface, i.e. one user, there is currently only one selection at a time, so there is no use for this locking feature. In the application as a whole, each distributed interface can have its own selection. All other users have to be prevented from selecting the same object in order to avoid conflicting updates.

5 Support for Distribution

Support for distribution in the AR system is provided via a separate "distributed application environment" which is based on the Facile distributed language. Facile is a strongly typed functional language (developed at ECRC[11]) that extends the functional paradigm with concurrency and communication. The Facile environment allows distributed application issues, such as concurrency control and dynamic joining, to be addressed separately from the rest of the

application by the developer.

The Facile environment includes group communication mechanisms and conferencing services that facilitate the construction of the distributed parts of shared, group-based AR applications. One of the aims of the environment is to enable developers who are not experts in distributed systems to construct a shared AR application. The environment provides more than simply communications, as would be provided by a message bus [8], for example, between the distributed components. In fact, the environment may maintain its own representation of parts of the global state of the application based on the history of interactions that are forwarded from the various connected user interfaces.

The construction of applications is made easier by isolating an application's distribution concerns into a separate component (the conferencing component). This is especially true if an environment is available where the programmer can use generic services including lock-based concurrency control and conferencing functions. Special features can be implemented without much effort, such as the transparent initialization of models as they connect to a distributed session.

5.1 External Views

The connection between the replicated models and the distributed application environment is established by a special kind of view at each site, an "external" view (Figure 5.1). Models join shared sessions by connecting their external views to the conferencing component and then exchange model updates. External views do not manage an interactive presentation in the interface, but act as a gateway to the distributed environment.

There are two main advantages to this approach. On the one hand, there is no extra code required to communicate model changes to the outside world. External views participate in the normal update mechanism of the interface just like any other view. In particular, there is no difference between the execution of the model-view updates in a stand-alone interface or in a distributed application. On the other hand, the implementation of the rest of the interface remains independent of the communication requirements, since the external view encapsulates all the details of setting up the connection and translating updates into messages.

The conferencing component maintains a representation of the global state of the application (and thus the history of the session) by monitoring the inter-model exchanges. This state is used to bring a model joining the on-going distributed session up-to-date, by generating an initial sequence of event notifications from the shared state.

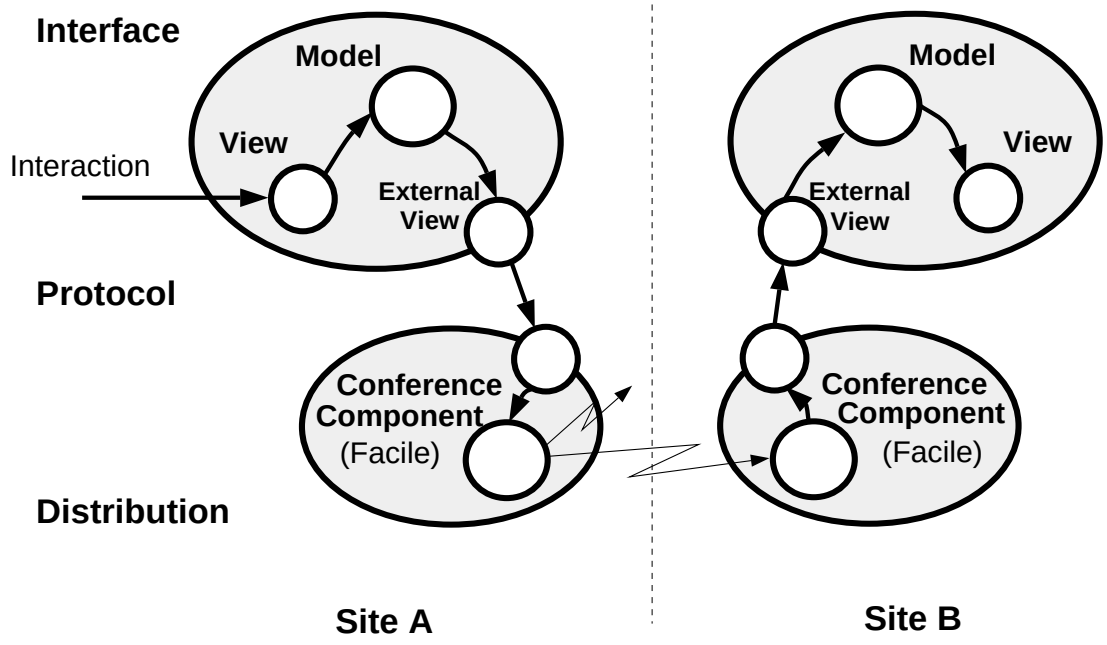


Figure 5.1: Update paths in distributed application

For example, in the interior design application, the shared state consists of transformation and attribute values for each model object, as well as the current camera orientation. The current set of model objects is constructed transparently to the individual models by the conferencing component from the global sequence of model insertion and deletions. This provides an example of the separation of model level interaction and distribution.

5.2 Reliable Broadcast and Group Support

The Facile environment provides an atomic reliable broadcast protocol, which facilitates the replicated maintenance of the application's global state using the state machine approach [19]. In order to change the application's global state, a request message is submitted to the broadcast service which then schedules the request and delivers it to each attached (remote) receiver in the same order. If more than one request is submitted concurrently by different senders then the broadcast service will impose an arbitrary global order on them. The request is acted upon at each of the receivers to deterministically change the state at each participant. This style of computation has the advantages of a centralized approach while providing some of the benefits of a distributed implementation, such as tolerance of failures of participants [10]. A further advantage of broadcast-based systems is that they naturally support the reactive synchronous type of collaboration that we believe to be typical of shared AR applications, which normally require close synchronization with the real-world.

This type of broadcast service is provided by several distributed platforms including Isis [5] and ANSA [3], as well as by Facile. It may be used in the

implementation of distributed services, such as locking, where it is used to globally schedule certain events, submitted to it by external views, by imposing a global serialization on the events. The Facile environment also provides a group abstraction which allows the transmission of ordered broadcasts between group members, which are processes that may join (and later leave) multiple application level groups. A group also provides its members with notifications of the joining and leaving of other members, possibly due to a processor crash.

Applications may use groups to implement services such as audio conferencing, floor control and locking. For example, locks held by a model that is quitting a shared session are dropped by the lock service and other models are notified via their external views.

5.3 External View Communications

Communication between external views and the conferencing component is based on a semi-automated message passing facility. External views forward requests to the local component via inter-process remote procedure calls that may block until the request is serviced, while the conferencing component forwards asynchronous event notifications to attached external viewers.

The local conferencing component broadcasts requests to other remote components. Once a request is received from the broadcast service by a conferencing component, it is locally acted upon and an event notification is forwarded to the relevant local external view. The requesting external view may also be blocked on the outcome of the request which is now returned to that view. The above communication mechanism connects the event loop-based AR environment with a more general environment that explicitly supports concurrency via processes and message-based communication.

5.4 Camera Communications

Besides transmitting camera parameters, shared AR applications need the ability to transmit actual images or video to other remote interfaces. When multiple cameras are used by an application or when image transfer is switched or not continuous, then the transfers must be coordinated.

In the interior design application, a site initiates a camera transfer by submitting a request to the conferencing component which schedules the request and notifies other attached AR systems. The notification carries the current camera transformation and causes a direct communication channel to be established between the sending site and each receiver using a reliable multi-cast transport protocol (if such a channel was not already established). This channel allows high resolution images to be transferred directly between AR systems, which are the real source and sink end-points for the transfer, without further

intervention from the conferencing component. This transport utilizes IP multi-cast[15] which maps to efficient hardware supported multi-cast on networks such as Ethernet.

5.5 Object Creation, Selection and Locking

Models maintain a hierarchal tree-based representation of design objects and use the local multi-phase model-view protocol (described in section 4) to create, insert and delete, and manipulate design objects. External views participate in the protocol and impose global consistency constraints by interacting with the distributed application component, for example, in order to obtain a unique global identifier for an object when it is created.

Insertion of objects into the model's tree must be authorized by all its views. An external view requests a lock from the distributed component before allowing an insertion. In fact, the lock logically locks not only one design object but a sub-tree based on the path from the root of the model's tree to the object, thus protecting against concurrent manipulations of enclosing or component objects. A special type of lock based on path comparisons implements this type of mutual exclusion in the conferencing component. Other global lock types are also available. All lock types transparently handle failures of lock holders and give notifications for locks already held to new participants. Path based locks are used by external views in connection with object selection for user interaction. Interaction events pass through the external view on the way to remote participants. If a selected object has not been manipulated within a certain time span, the external view locally initiates a lock release. This prevents monopolization of selections by individual participants.

6 Application Scenario

Several examples in this paper have already referred to the interior design application which was created to demonstrate distributed interfaces in the context of AR. The scenario for the application involves a customer who intends to order furniture to decorate a room. After setting up the AR equipment, the customer is presented with a computer display that has three different views of the furniture that will be selected for the room (Figure 6.1).

On the left hand side there is an object browser which will list the furniture loaded from the database. Above it, there is a small view with an orthographic projection of the room. This view gives an indication where the furniture is positioned relative to the floor plan. This provides a better understanding of the layout of the furniture, and is convenient for interactive positioning of furniture items with a 2D input device like the mouse (sliding furniture around on the floor is a 2D task). The third and largest view is the presentation of the

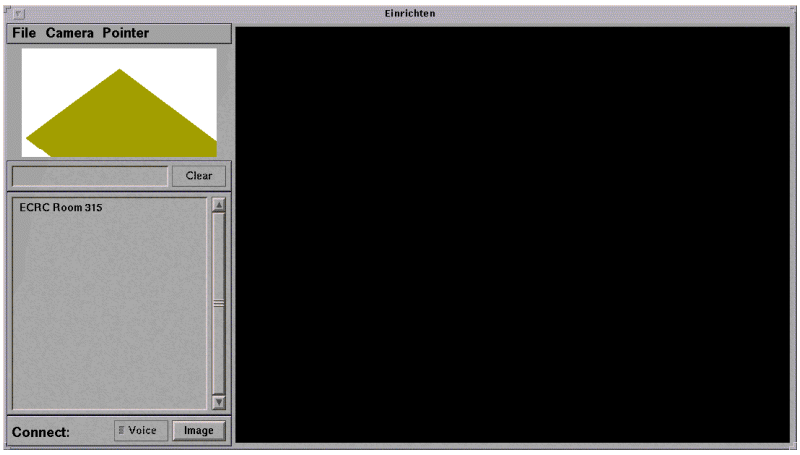


Figure 6.1: Initial user interface for interior design application

furniture in the AR interface. After calibration the camera model used for rendering the furniture corresponds to the camera generating the video input. The video monitor will show the same view after all black parts in the image have been replaced by the video information. This view is also interactive and provides feedback for direct 3D manipulation of the furniture.

After starting the interface, the customer now contacts an interior designer who is in his office at a different location. The designer joins the session and uses the same AR system to visualize the room and judge the effect of different possible furniture selections. The designer chooses furniture from an on-line database catalogue which understands selection criteria like type of furniture and color, or manufacturer and price. The 3D rendering of the furniture appears on the monitor together with the live view of the room, which changes whenever the camera moves to look at the furnished room from a changing point of view.

Both users can now add, delete, and rearrange furniture until they come to an agreement on the design. Figure 6.2 shows a snapshot of the cooperative editing. The furniture browser shows the different selections; the local user is currently working with a chair, whereas the remote user has selected a desk. They simultaneously move the furniture in real-time. Locally, a bounding box around the selected piece of furniture provides feedback for the manipulation. Remote updates cause the furniture to move as if by magic. A voice channel is provided so that users can notify each other of their intentions.

During the planning stage, the customer can also consult with friends or colleagues at other sites. They again run an instance of the same interface and join the on-going session. They too can view and manipulate the same set of furniture. All changes are seen instantaneously by all of the users, and the distributed locking mechanism ensures that a piece of furniture is moved by only one user at a time. Figure 6.3 shows a simple example of what the result of such a design session could be. Finally, the furniture selection will be

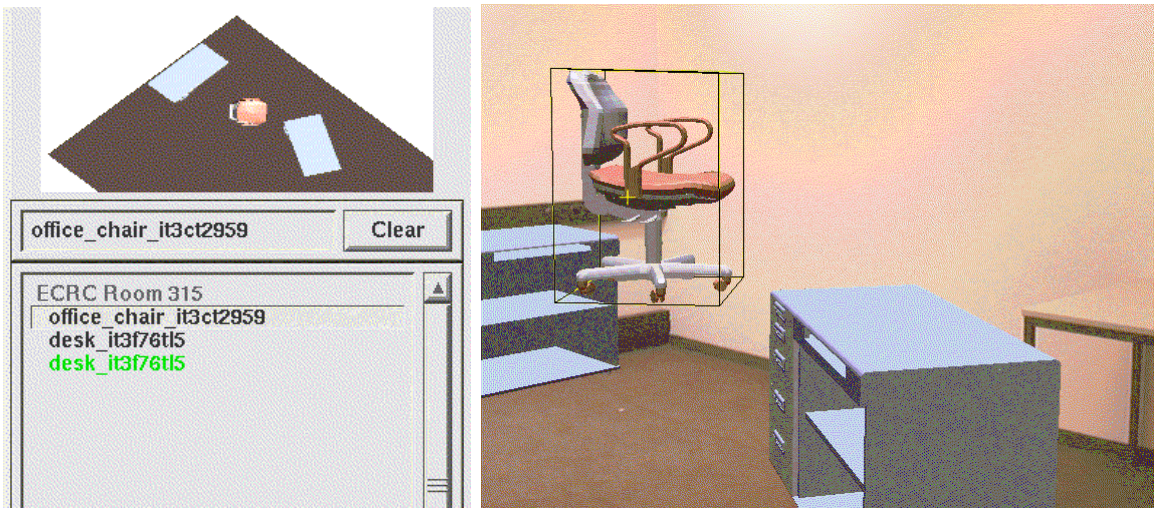


Figure 6.2: Local user lifts chair while remote user rolls desk into room.

recorded and can be used to fill out an order form.

7 Future Work

In the near future we intend to improve the interior design demonstration in various ways. The user interface and interaction methods need enhancement to make manipulation of graphical objects easier, and to convey more awareness of the collaboration. An important step is the inclusion of real-time video (currently remote sites receive static images of the scene). Fine tuning of the application scenario will lead to a more fluid integration of user tasks at different stages in the application, including system initialization, cooperative design, furniture selection, and even preparation and issuing of a purchasing order.

We plan to expand the current conferencing component with additional generic support, and to make it available to other non-cooperative demonstrations that were implemented using the underlying AR system described in this paper. An example is a "mechanical repair" application [18], that uses AR to present information about an automobile engine to a mechanic. A multi-user interface to this application would enable the mechanic to consult with a remote expert about peculiarities of the engine and the job at hand.

Our basic research and future development of the AR system will bring more fundamental changes to the AR applications. We are investigating visual calibration and tracking techniques to make these applications more robust and easier to set up. Research leading to automatic (geometric) model generation will allow us to construct an accurate model of the actual scene. We are

Figure 6.3: Real room before (small video image) and after furnishing (augmented reality)

already using this model information for object occlusion and for geometric constraint handling. Graphical objects can move behind real ones or stop when colliding with walls and floors. Such an approach could, for instance, ensure that furniture sits squarely on the floor [6]. There are other techniques, like modeling the lighting of the real scene and using the same illumination during rendering of the graphical objects, which will bring us closer to the ultimate goal of giving more realism to the visual presentation.

8 Conclusion

Distributed augmented reality is a promising concept for a new class of practical applications. Combining AR technology with techniques from the CSCW area and distributed systems support is a challenging task and requires the integration of various hardware and software solutions. Early benefits of our approach are apparent in the interior design application, such as the advantages of separation of functionality between the view and the model, and the ease of session management provided by the distribution level. The group

communication software ensures data consistency for the shared logical model, and offers building blocks for cooperation and awareness in the interface. The demonstration shows that the AR software platform, combined with advanced computing and communication hardware, is capable of producing powerful results for the end user.

9 Acknowledgments

The interior design demonstration was greatly improved by an interface to a data base of furniture provided by Philippe Bonnet and Stephane Bressan at ECRC.
The work presented in this paper is partially supported by ESPRIT Basic Research Action "Confer" 6564. ECRC GmbH is financially supported by Bull SA, ICL PLC, and Siemens AG.

Bibliography

[1] K. Ahlers, D. Breen, C. Crampton, E. Rose, M. Tuceryan, R. Whitaker, and D. Greer. An augmented vision system for industrial applications. In *SPIE Photonics for Industrial Applications Conference Proceedings*, October 1994.

[2] V. Anupam, C. Bajaj, D. Schikore, and M. Schikore. Distributed and collaborative visualization. *IEEE Computer*, pages 37–43, July 1994.

[3] Architecture Projects Management. *A model for interface groups.* APM Limited, Poseidon House, Cambridge CB3 ORD, United Kingdom, February 1993.

[4] R. Bentley, T. Rodden, and I. Sommerville. Architectural support for cooperative multiuser interfaces. *IEEE Computer*, pages 37–46, May 1994.

[5] K. P. Birman and T. A. Joseph. Exploiting virtual synchrony in distributed systems. In *Proceedings of the 11th ACM Symposium on Operating Systems Principles*, pages 123–138, Austin TX (USA), November 1987. ACM.

[6] D. E. Breen, E. Rose, and R. T. Whitaker. Interactive occlusion and collision of real and virtual objects in augmented reality. Technical Report ECRC-95-02, European Computer-Industry Research Centre GmbH, Munich, Germany, 1995.

[7] C. Carlson and O. Hagsand. Dive - a platform for multi-user virtual environments. *Computers & Graphics*, 17(6):663–669, 1993.

[8] A. Carroll. *ConversationBuilder: a collaborative erector set.* PhD thesis, Department of Computer Science, University of Illinois, 1993.

[9] E. A. Edmonds, editor. *The Separable User Interface.* Academic Press, London, UK, 1992.

[10] C.A. Ellis, S.J. Gibbs, and G.L. Rein. Groupware: some issues and experiences. *Communications of the ACM*, 34(1), January 1991.

[11] B. Thomsen et al. Facile Antigua Release Programming Guide. Technical Report ECRC-93-20, European Computer-Industry Research Centre GmbH, December 1993.

[12] L. E. Fahlen, C. G. Brown, O. Stahl, and C. Carlson. A space based model for user interaction in shared synthetic environments. In *INTERCHI '93 Conference Proceedings*, pages 43–48, April 1993.

[13] S. Feiner, B. Macintyre, and D. Seligmann. Knowledge-based augmented reality. *Communications of the ACM*, 36(7):53–62, July 1993.

[14] W. Lorensen, H. Cline, C. Nafis, R. Kikinis, D. Altobelli, and L. Gleason. Enhancing reality in the operating room. In *Visualization '93 Conference Proceedings*, pages 410–415, Los Alamitos, CA, October 1993. IEEE Computer Society Press.

[15] M. R. Macedonia and D. P. Brutzman. MBone provides audio and video across the Internet. *IEEE Computer*, 27(4), April 1994.

[16] P. Milgram, S. Shumin, D. Drascic, and J. Grodski. Applications of augmented reality for human-robot communication. In *International Conference on Intelligent Robots and Systems Proceedings*, pages 1467–1472, Yokohama, Japan, July 1993.

[17] E. Nakamae, K. Harada, T. Ishizaki, and T. Nishita. A montage method: The overlaying of the computer generated images onto a background photograph. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 207–214, August 1986.

[18] E. Rose, D. Breen, K.H. Ahlers, C. Crampton, M. Tuceryan, R. Whitaker, and D. Greer. Annotating real-world objects using augmented reality. Technical Report ECRC-94-41, European Computer-Industry Research Centre GmbH, 1994. To appear in: Proceedings of Computer Graphics International '95, Leeds, UK.

[19] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: a tutorial. *ACM Computing Surveys*, 22(4), December 1990.

[20] L. Shu and W. Flowers. Teledesign: Groupware user experiments in three-dimensional computer-aided design. *Collaborative Computing*, 1:1–14, 1994.

[21] D. Sims. New realities in aircraft design and manufacture. *IEEE Computer Graphics and Applications*, 14(2):91, March 1994.