CIAO: A COLLABORATIVE, IMMERSIVE ARCHITECTURAL LAYOUT SYSTEM

Un-Jae Sung, Jae-Heon Yang, K. Wohn
Department of Computer Science,
KAIST, Taejon, Korea

ABSTRACT

This paper presents the design of a multiuser, large-scale 3-D layout system: *Collaborative Immersive Architectural layout* (CIAO). In contrast to many existing systems that sacrifice responsiveness to maintain consistency, CIAO achieves optimal response and notification time without compromising awareness or consistency. The optimal responsiveness is achieved by a new multicast-based, optimized concurrency control mechanism. Even operations on a group of related objects do not entail any latency for concurrent control. We also present the multiuser interfaces of CIAO that provide some sense of isolation as well as rich awareness.

1. **Introduction**

Virtual environments can provide a user friendly interface to diverse applications such as battlefield simulation, virtual conferencing, and collaborative modeling [1][2][6][8][9][10][12]. Some of these applications, including battlefield simulation, require large scale virtual environments. City planning or the emerging electronic commerce can also benefit from the natural interfaces provided by large scale virtual environments.

Tools for modeling individual object, laying out the modeled objects in the virtual environment, and walking through the environment are a must for building a large scale virtual environment. We present the design of a multiuser, large scale 3-D layout system CIAO (Collaborative Immersive Architectural layOut) in this paper. Because CIAO supports full functionality for layout and walk-through, one can construct virtual environments with CIAO in conjunction with a conventional modeler. Using CIAO, designers can try out what they think, and get realistic feedback instantly. In order to enable the synergy among a group of design participants, CIAO is designed to meet the following requirements:

- Each participant must be aware of other members' presence and their actions. (Awareness)
- User actions should be responded to locally and be notified at remote sites as soon as possible. (Responsiveness)
- Users perceive the shared virtual world in a consistent way, although they can manipulate the shared objects in the virtual environment. (Consistency)

We call the virtual environment that satisfies the three requirements listed above a *collaborative virtual environment*. One of prime difficulties in designing and implementing a collaborative environment is that the responsiveness and consistency requirements often conflict with each other and suggest opposite directions in the design space. Although there exist many virtual environments that addressed awareness and consistency issues, none of them achieve an optimal level of responsiveness. In fact, many systems sacrifice their responsiveness in order to maintain consistency. In this paper we describe how CIAO achieves optimal responsiveness without sacrificing awareness or consistency.

In a collaborative environment, it is often possible for many participants to attempt to manipulate a single shared object almost at the same time. Concurrent actions may result in inconsistent views of an object among a group of users. Without careful coordination, a sequence of concurrent actions would soon confuse the participants by presenting different views of the supposedly same shared world, violating the consistency requirement.

In CoCAD, a collaborative CAD system, a central communication server called the facilitator coordinates all participants' activities [6]. In order to manipulate an object, the participant must send an update request to the facilitator, which ensures the updates arrive at all sites in same order. Because the single server serializes all updates, it is straightforward to maintain consistent views among the users. However, the response time of each action is increased by at least the round trip latency between the participant node and the facilitator. This latency can be several hundred milliseconds in a wide area network. An Internet packet might take a few seconds before delivery due to congestion. The facilitator also tends to become a performance bottleneck when the number of participants increases and further slow down the response.

DIVE (Distributed Interactive Virtual Environment) is a collaborative virtual environment developed at the Swedish Institute of Computer Science. In an early version of DIVE, a node acquired the lock before each manipulation to VE and then sent the update message to its peers[2]. This has the serialization effect that CoCAD achieved by employing the facilitator. It was reported that this approach did not scale to more than ten peers on a local area network, partly because the locking mechanism employed did not scale well [8]. A recent version of DIVE assigns a token to each object that must be obtained before manipulation of the object. Still, the response time is increased by the round trip latency between the participant node requesting a token and the node possessing the token.

CIAO adopted an optimized concurrency control. In other words, the users manipulate objects without any waiting, and also notify other participants of their actions immediately in CIAO. If there are conflicting operations on the same object, the token associated to the object is used to maintain consistency. The users can also manipulate a group of related objects as easily and quickly as they manipulate an individual object. In fact, the concurrency control in CIAO is optimal in the responsiveness standpoint. The interface of CIAO is designed to provide rich awareness and to reduce any confusion that might be caused by concurrent actions from multiple users at the same time. We believe the interface, with the concurrency control, enables close and responsive collaboration among the participants in a large scale layout session.

The main contributions of CIAO are as follows. The first and foremost characteristic of CIAO is that it is very responsive. In fact, most operations in CIAO are completed without being delayed at all by the concurrency control mechanisms. Even operations on a large group of related objects do not entail any latency for concurrency control. This is achieved by combining an efficient implementation of optimized concurrency control with an elegant way of manipulating a group of related objects. The second contribution is in new interfacing techniques that minimize the user confusion/embarrassment that might be caused by concurrent interactions of multiple users. These techniques bring the sense of isolation to a CIAO user when his current task may perceptually interfered by the concurrent actions of other users.

The rest of the paper is organized as follows. In Section 2, we describe the overall design of CIAO that is based on multicast communication. Section 3 presents an efficient implementation of optimized concurrency control that eliminates the delay usually caused by other concurrency control mechanisms. In Section 4, we discuss how we create a group of objects that have a hierarchical relation in CIAO, and how concurrent operations on the grouped objects are completed with optimal response time. We end the paper with concluding remarks in Section 5.

2. Design of CIAO

In this section the overall design of CIAO is discussed briefly. CIAO, as depicted in Figure 1, supports three channels of communication in order to facilitate cooperation. The first channel is a provision for verbal communication, and is implemented by an audio conferencing tool called *vat* [13]. Through this channel, participants can discuss their plans and may try to reach a consensus when an important decision is needed. Avatars provide the second line of communication. An avatar is the object that represents a human participant in the virtual environment [11]. Through avatars participants can identify each other and become aware of other users' locations and interests. A third type of communication is achieved by manipulating the shared objects in the virtual environment. Although one may dispute our view that object manipulation is one form of communication, we take the position that interpersonal communication can be greatly facilitated by an accompanying appropriate manipulation in the virtual environment as well as in the real environment. Our work focuses on this type of communication, which has received little attention in the literature.

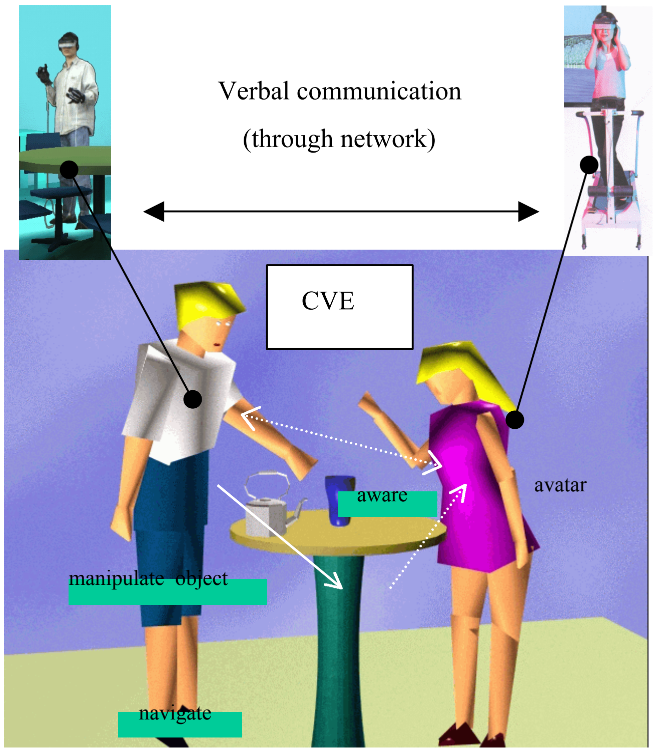

Figure 1: Communication in CIAO

In the remainder of this section, we describe briefly our design of CIAO to support all three types of communication. The communication architecture of CIAO is depicted in Figure 2. A CIAO system consists of participant nodes, a session management server, and object information servers that communicate over the Internet. Participant nodes ask the session management server for the permission to join a CIAO session, and receive the session information such as multicast addresses in use and the locations of 3-D model data in the URL (Unified Resource Locator) format. Actual model data are then obtained from the object information servers. The communication between a participant node and a server relies on the TCP protocol. The steps described so far create a replica of the initial state of the shared virtual environment on every site. Then each participant node updates its own replica using the avatar and shared object state information that comes through the multicast channel. In short, the shared virtual environment is fully replicated at all participating sites for the sake of responsiveness. As we shall discuss in Sections 3 and 4, concurrency control in CIAO is based on multicast and is distributed among participant nodes.

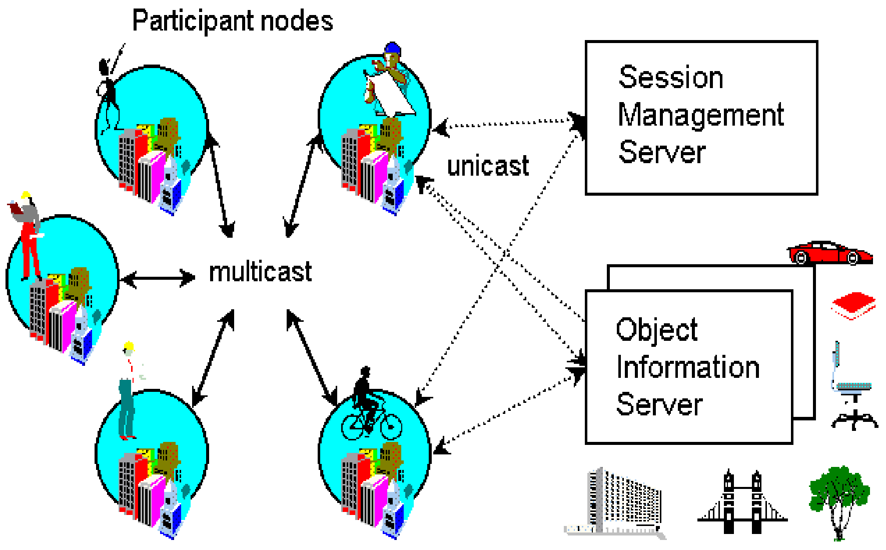

Figure 2: CIAO communication architecture

Modules consisting of a CIAO node are depicted in Figure 3. A participant node performs human input processing, real-time rendering, avatar and task management, and network communication continuously. The user perceives the virtual environment from the sensory data presented by the rendering module, plans, and reacts continuously. The user interface module extracts navigation and manipulation commands from the states of various VR input devices such as stationary bicycles, joysticks, and data gloves. These commands are then passed to the avatar/task manager, which multicasts the commands through the network interface. The navigation commands are also passed from the user interface module to the rendering module in order to speed up the view updates.

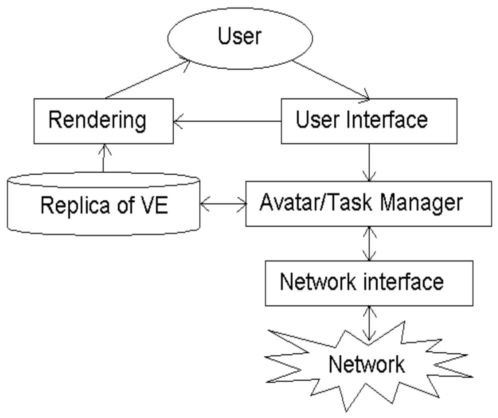

Figure 3: CIAO participant node architecture

The avatar/task manager controls the avatars and modifies the replicated virtual environment. The updates to shared objects and avatars are handled by same module in order to synchronize the avatars' movement with the manipulated objects' movement. The avatar/task manager also coordinates concurrent tasks. The concurrency control for collaborative virtual environment like CIAO has different requirements from traditional non-interactive distributed systems. Therefore instead of simply adopting conventional mechanisms, we designed novel concurrency control methods that minimize the response time and maximize concurrency. The details of our concurrency controls are described in the following sections.

3. Optimized Concurrency Control

In this section, we describe the optimized concurrency control and related interface mechanisms in CIAO. Many existing virtual environments including DIVE and CoCAD adopted pessimistic concurrency control methods. A concurrency control is considered pessimistic if it guarantees that there is no inconsistency caused by concurrent operations at any time. The traditional non-interactive systems like distributed databases employed pessimistic concurrency control because it offered a simple interface to the programs running on these systems. In fact, database systems require that each transaction be isolated from others. Such a strong requirement limits the choice of concurrency control to ones that offer strong guarantees.

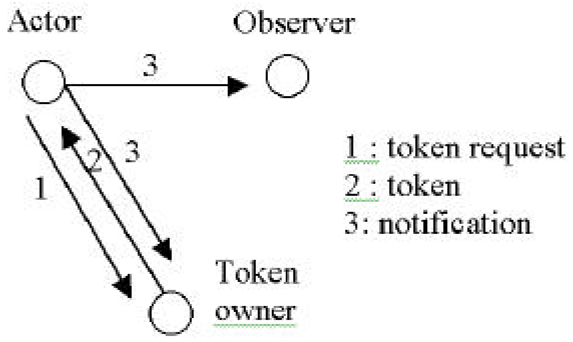

Figure 4: Pessimistic locking in DIVE

A serious drawback of pessimistic approaches is that they are not responsive. They block operations until they can guarantee the operations do not cause any inconsistency in the system. Consider the concurrency control mechanism of DIVE illustrated in Figure 4. For a user to manipulate an object, he must ask for the token associated with the object by sending the token request message to the owner of the token (1). Then the token owner sends the token as a reply (2). After receiving the token, the user can now manipulate the object and notify the other participants of his action (3). Let L denote the average network latency between two participating nodes. The latency L varies from few milliseconds on a LAN to a few hundred milliseconds in WAN. The response time is the delay until the manipulator gets the first feedback of his action. The notification time is the delay until other participants get the first message of the operation. Then DIVE's response time is 2L, and its notification time is 3L. A similar analysis of CoCAD shows that both of its response time and notification time is 2L.

Next we describe the concurrency control mechanism of CIAO. A version number and "has-token" bit are maintained at each replica of an object in CIAO. The create operation of the object initializes its version number to 0. Only the node that created the object sets the has-token bit. In other words, the creator node is the initial owner of the token.

Figure 5 illustrates how concurrency is controlled in CIAO. A participant starts his manipulation of an object without any waiting and multicasts a message containing the object identifier, one plus the version number of its replica of the object, and its action (1). The token owner receives the message and verifies whether the version number in the message is current. If the version number matches the owner's version number, then the owner validates the action by multicasting the message that the ownership is transferred to the node that started the manipulation (2). This validation message cancels other concurrent manipulation of the same object, if any. Then the version of the objects is incremented in all replicas. The token owner may transfer the ownership only once per version, which ensures that there is always only one legal state of an object. All replicas of the object will converge to this legal state unless "critical" messages are lost. If the version numbers fail to match, the token owner gives the manipulator the current state of the object.

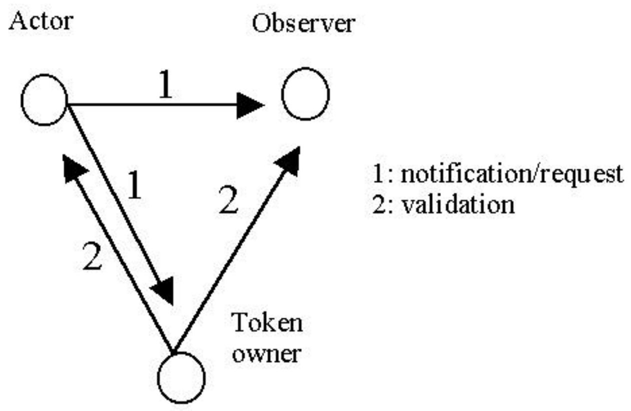

Figure 5: Multicast based semi-optimized locking

Because a participant gets immediate feedback for his action locally, the response time is 0. Other participants are notified of the action after the delay of L. It should be obvious that the response time is optimal. The notification time is also optimal if one cannot depend on prediction techniques. The delay until validation is 2L, which equals to the response time in CoCAD and DIVE.

The concept of optimized concurrency control is not new. There are works in the area of CSCW that investigate the use of optimized concurrency control [3][4][7]. However, we are not aware of any optimized concurrency control mechanisms that exploit the multicasts as in CIAO.

In spite of providing better responsiveness, the use of optimized concurrency control is less than whole-heartedly accepted in conventional CSCW applications. The most important reason is that temporary inconsistencies introduced by "conflicting" operations confuse the users. Because actions are communicated immediately, it is possible that two users carry an object in different directions. Thus it is imperative for a system with optimized concurrency control to make provisions for reducing the user confusion.

We now describe the multi-user interface designed for better awareness and less confusion in case of conflicts. For the sake of simplicity, our description is presented in terms of concurrent move operations. Because create and delete operations are treated as a special case of move operation in CIAO, it is important to note that our discussion below apply to all basic operations.

Some screen shots from a CIAO session are presented in Figure 6. There are two participants who may manipulate the cup on the table as in Figure 6-1. When one of them begins his operation by holding the cup, a translucent clone of the cup is created, leaving the wire-frame of the cup at the original position as in Figure 6-2. The translucent clone gives an immediate feedback to the manipulator along with the hint that it has not been validated yet. Another participant may also reach the cup and create her own clone of the cup. This may happen because of the notification delay between two participants. When the notification message from the other user arrives, two clones will be displayed in Figure 6-3. Finally, when one of these operations is validated, the corresponding clone gets a solid representation and other representations of the cup are cleaned up as in Figure 6-4.

Some analysis of the user interface is in order. As in Figure 6-2, CIAO provides immediate feedback to user actions so that the participants can work with precise visual feedback without waiting. Figure 6-3 illustrates what the users get in the case of a conflict. They see multiple versions of an object, which is avoided in systems with pessimistic concurrency control. We believe, however, that showing multiple clones of an object foster collaboration among the interested participants. After all, multiple copies came into existence because two people have different ideas on a shared work. In such cases, it would be natural and productive for the users to start discussion using the audio channel. Figure 6-4 shows that such temporal inconsistencies are resolved automatically by the concurrency control. We believe that this resolution is rather arbitrary, just as in other systems, and should be regarded as the last resort to keep consistency.

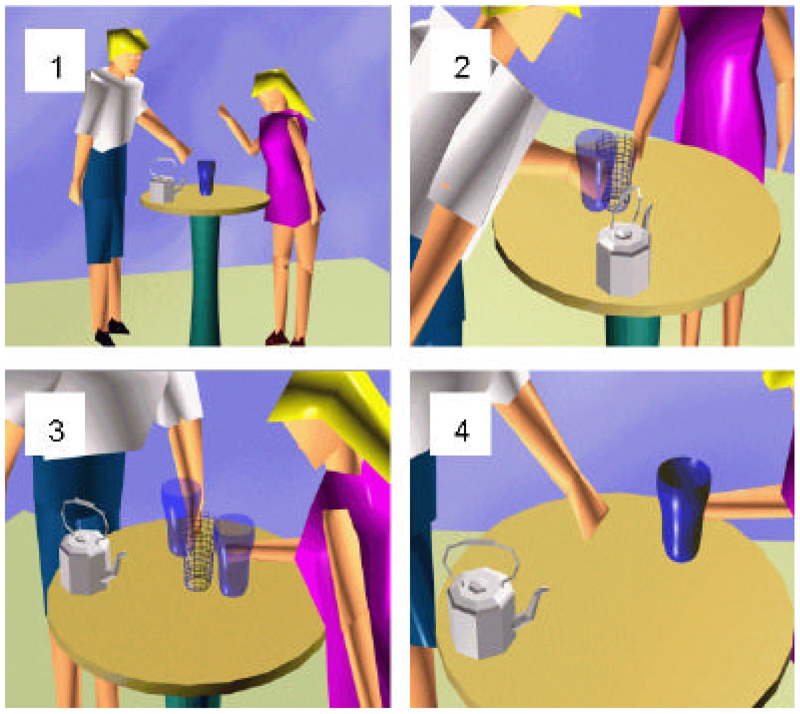

Figure 6: Cloning objects for optimized concurrency control

4. Concurrent Operations on Related Objects

The previous section described how we eliminated the latency usually resulting from concurrency control in case of single object manipulation. In this section, we extend the concurrency control on individual object to that on a group of objects for convenient and efficient works in large-scale layout. In particular, we describe how related objects are created and manipulated with an optimal responsiveness and a high degree of concurrency in CIAO. We begin by describing the hierarchical relation supported in CIAO, and then explain an implementation of the hierarchy of related objects that offers a maximal degree of concurrency between layout operations.

We support a hierarchical structuring of objects for the layout in a virtual environment. The attach operation introduces an asymmetric relation between two objects: one object becomes the parent of the other. The object becoming a child must not have a parent so that a set of related objects maintain a hierarchy. In fact, all objects in a CIAO application form a tree with the root “universe”. Observe that the relationship is similar to “Is-Part-Of” relation found in diverse applications. CIAO interprets the operation on an object as the operation on the subtree of objects whose root is the specified object. For example, CIAO treats a move operation as the update to the spatial relation between the subtree of objects and its immediate ancestor in the tree. Consider the tree of objects depicted in Figure 7. Moving the city hall would change the location relative to the city of the entire subtree of objects including the room, the table, and the cup inside the city hall.

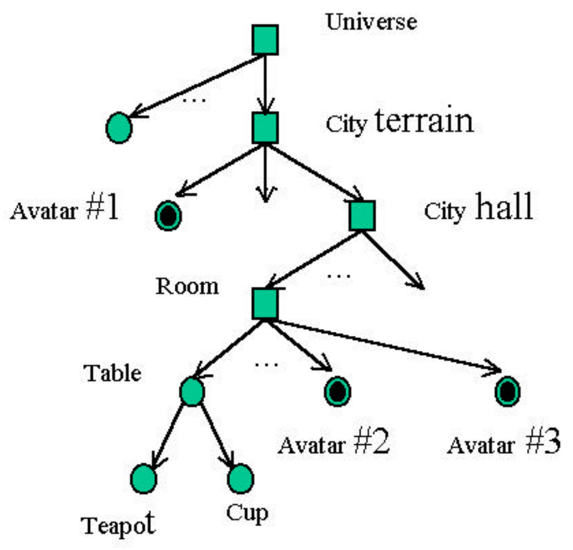

Figure 7. Hierarchical relation in 3-D layout

Next we explain how the hierarchy is implemented in CIAO to enable concurrent manipulations of related objects in the tree. As was explained in Section 3, a token is assigned to each object for the coordination of concurrent operations on the object.

Suppose that many participants are working on the various objects in Figure 7. It is possible that a designer tries to move the city hall and another tries to move the table almost at the same time. In many multi-user applications that employ tokens as fine-grained locks, the former move operations must request the token for the table as well as the token for the city hall, in order to avoid the possible interference from the latter move. This not only reduces the available concurrency, but also multiplies the response time. Moving a large group of objects usually requires sending a large number of lock requests entailing delay proportional to the size of the group of objects, when the granularity of concurrency control is small. Employing coarse-grained locking reduces the overhead, but it severely limits the concurrency. There is a technique called granular locking that achieves a better balance of overhead and concurrency, but obtaining a granular lock, still takes a few latency periods to obtain a simple lock [5].

We adopted a radically different approach in CIAO. An operation on a hierarchical group of objects requests only the token for root object of the subtree. The position of an object in CIAO is represented in coordinates that are relative to its parent's position. Thus when the root node moves relatively to its parent, the whole subtree of objects makes a corresponding move. Observe that with the parent-relative coordinate representation, any two concurrent moves of the objects in a tree are either independent or additive, and hence "commute". Commutative operations have the characteristic that they produce the same result if they are executed in any order. This implies that the results of concurrent operations will converge to the same view, although they may temporarily show different views to different participants. It is important to note that moving a large group of objects is processed in the same way as moving a single object in CIAO, and has the same response time as for moving single object.

Although making concurrent operations commute helps CIAO to maintain consistency, it does cause an interesting problem in multiuser interface. Consider the case of concurrent moves of a table and the building containing it by two different users. The move operations have significant duration, and the participating sites may experience a period that these two operations overlap. In that period the two movements involve the same table. The observation made above that these two movements commute implies that CIAO shows the vector sum of the two partial moves in the period. A remaining problem concerns the position of the avatar that is manipulating the table. If that avatar keeps its current position relative to the universe, it may pop out from the building and lose the grip on the table. This problem is solved naturally if the avatar moves along inside the building. In CIAO, when an avatar moves into a building, it is attached to the building automatically. Afterwards, the user's navigation is interpreted as changing the corresponding avatar's position relative to the building. Moving the avatar out of the building will detach the avatar from the building.

There is a case of concurrent operations that requires a different solution from attaching avatars to a node. Consider the concurrent moves of a table and a cup on the table . Suppose the cup has been attached to the table. What should happen if a user changes the position of the cup on the table when another user pulls the table away? Moving the cup along with the table's movement may pull it out of the avatar's hand holding it. Attaching the avatar to the table in this case is not a solution as natural as in the case involving the building. If the avatar moves along with the table, the corresponding user will experience an unexpected, drastic change of his viewpoint.

The difference between attaching the avatar to the building and attaching it to the table is that the building encloses the avatar while the table does not. So the change of the avatar's view is expected to be much smaller in the former case. We differentiate the former case from the latter by introducing the concept of "enclosure object". The enclosure object is one that is considered to be an effective abstraction of the environment for avatars inside it. A room, a building, and a city block can be designated as the enclosure blocks in CIAO applications.

Two move operations on different objects in CIAO have dependency only if one object is an ancestor of the other. Two cases are handled differently depending on whether there is an enclosure object between the ancestor object and the parent of descendant object included in the hierarchy. In case there is no such enclosure object, we need another mechanism, described below, to keep the view of the user manipulating the descendant object. If there is such an enclosure object, the enclosure object provides proper perceptual isolation to the user manipulating the descendant object. The avatar should have been attached to the lowest of such enclosure objects, and will not experience a drastic change of view. Observe that for any avatar, its lowest enclosing ancestor is a minimal environment for its work.

Figure 8 shows a part of the object hierarchy from Figure 7. It illustrates how a sense of isolation is provided along with awareness information in CIAO. In Figure 8-1 user is about to move the table while another user is holding the cup. When he moves the table, the motion of table is rendered as usual at his site and other observers' sites. The cup is left in the other user's hand, providing an explanation of the motion of the cup shown in Figure 8-2. If the second picture was rendered at her site, then she might lose the references for her action because the table was moving. In CIAO, the objects in the subtree of table except the objects in the subtree of cup is cloned and rendered translucently at original position until she finishes her task at her site (Figure 8-3). We call the translucently rendered objects ghost ancestors. This gives her a sense of isolation from the table's motion. At the same time, she is also informed that the other participant is moving the table so that she can understand that her cup will move to the solid table at new position after her operation. When all tasks complete, the views of users will converge to the last picture (Figure 8-4) that shows the combined result of both operations.

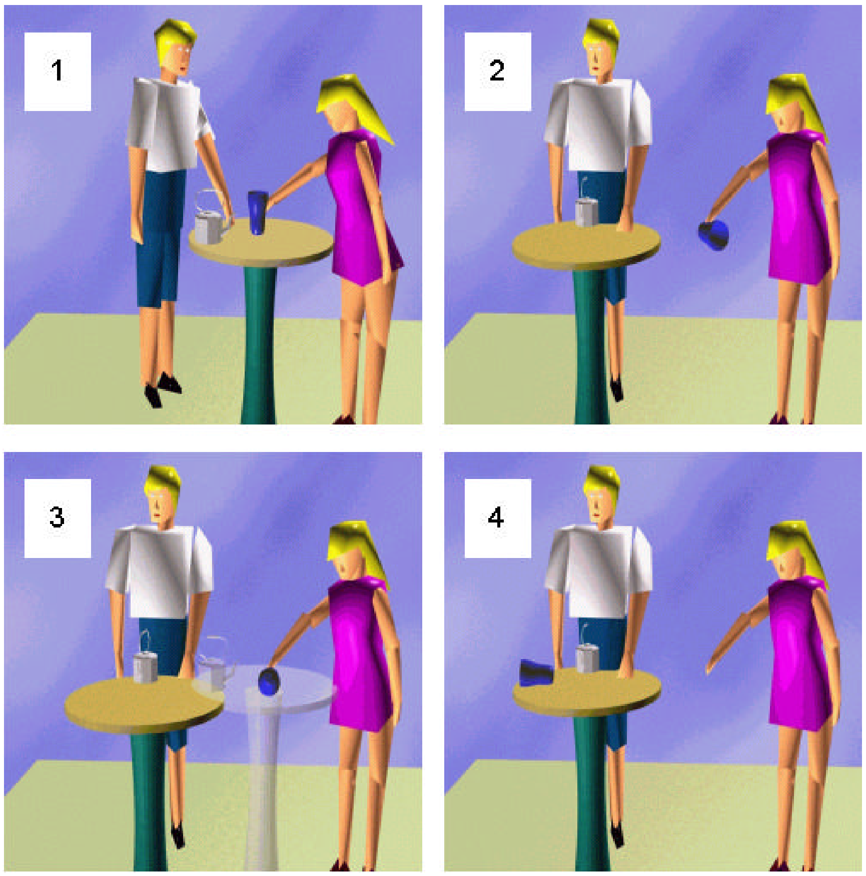

Figure 8: Ghost ancestor display for sense of isolation

An alternative to the interface techniques described above might be to process the movement of the table after the cup is released at her site. This may provide an intuitive explanation for the final result by serializing two concurrent tasks. However, the multi-user interface we proposed in this section has two advantages. First, it provides the awareness information more responsively. Second, it enables a simple and efficient synchronization with other channel such as audio.

5. Concluding Remarks

In this paper we have described how CIAO achieves its responsiveness without sacrificing awareness and consistency in large-scale, collaborative layout environment. We believe that CIAO is the first collaborative virtual environment of which response and notification time is optimal.

The concurrency control in CIAO is, to be more precise, semi-optimized [7]. In other words, a participant may start only one task without validation. This reduces any confusion in the rare case of invalidation due to conflicting operations. Because the operations are validated quickly (within multicast message round trip time, smaller than the duration of usual operations), the human users will not experience any blocking.

CIAO is a successor of CVRAT (Collaborative Virtual Reality Authoring Tool) that has been developed at KAIST during the past five years. We are in the last stage of the implementation of CIAO, and hope to get experimental results over the Internet within a few months.

The objects are structured as a tree in the current version of CIAO, with only one exception for sharing geometric models of objects. Generalizing the structure of the objects to a directed acyclic graph would be interesting future work. It would enable more convenient construction of large-scale environments as well as more efficient sharing of data.

REFERENCES

Barrus, J. W., Waters, R. C., and Anderson, D. B. “Locales: Supporting Large Multiuser Virtual Environments”, IEEE Computer Graphics and Applications, Nov. 1996, pp. 50-57

Carlsson, C., and Hagsand, O. “DIVE - a Multi-user Virtual Reality System”, IEEE VRAIS '93, Sep. 1993, pp. 394-400

Ellis, C. A., and Gibbs, S. J. “Concurrency control in groupware systems”, ACM SIGMOD International Conference on the Management of Data, Jun. 1989, pp. 399-407

Ellis, C. A., Gibbs, S. J., and Rein, G. L. “Groupware some issues and experiences”, CACM, ACM, Vol. 34, No. 1, Jan. 1991, pp. 38-58

Gray, J., and Reuter, A. Transaction Processing, Morgan Kaufmann Publisher, pp.406-411

Gisi, M. A., and Sacchi, C. “Co-CAD: A Collaborative Mechanical CAD System”, PRESENCE, MIT Press, Vol. 3, No. 4, Fall 1994, pp. 341-350

Greenberg, S., and Marwood, D. “Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface”, CSCW 94, ACM, 1994, pp. 207-217

Hagsand, O. “Interactive Multiuser VEs in the DIVE System”, IEEE MultiMedia, Vol. 3, No. 1, 1996, pp. 30-39

Jacobson, V., McCanne, S., Jansen, J., Frederick, R., Bormann, C., Degener, J., Casner, S. and Deering, S. Vat (Visual Audio Tool) manual page. http://www-nrg.ee.lbl.gov/vat/, Manual pages, Feb 1992

Macedonia, M. R., Zyda, M. J., Pratt, D. R., Barham, P. T., and Zeswitz, S. “NPSNET: A Network Software Architecture for Large-scale Virtual Environments”, PRESENCE, MIT Press, Vol. 3, No. 4, fall 1994, pp. 265-287

Macedonia, M. R., Zyda, M. J., Pratt, D. R., Brutzman, D. P., and Barham, P. T. “Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments”, IEEE VRAIS, 1995, pp. 2-10,

Pratt, D. R., Pratt, S. M., Barham, P. T., Barker, R. E., Waldrop, M. S., Ehlert, J. F., and Chrislip, C. A. “Humans in Large-scale, Networked Virtual Environments”, PRESENCE, MIT Press, Vol. 6, No. 5, 1997, pp. 547-564

Singh, G., Serra, L., Png, W., and Ng, H. “Bricknet: A Software Toolkit for Network-based Virtual Worlds”, PRESENCE, MIT Press, Vol. 3, No. 1, Winter 1994, pp. 19-34

Author contact information:

Un-jae Sung, Jae-Heon Yang, K. Wohn
Department of Computer Science, KAIST
373-1 Ku-Sung-Dong, Yu-Sung-Gu
Taejon, KOREA, ZIP 305-701
Phone: +82-42-869-3572/8719
Fax: +82-42-869-3510
Email: ujsung@vr.kaist.ac.kr (or click on mailto:ujsung@vr.kaist.ac.kr while online)