Computing and Informatics, Vol. 24, 2005, 7–29

SUPPORTING A CLOSELY COUPLED TASK BETWEEN A DISTRIBUTED TEAM: USING IMMERSIVE VIRTUAL REALITY TECHNOLOGY

David J. Roberts, Robin Wolff, Oliver Otto

The Centre for Virtual Environments
University of Salford
Manchester, M5 4WT, UK
e-mail: {d.j.roberts, r.wolff, o.otto}@salford.ac.uk

Manuscript received 16 December 2004

Abstract. Collaboration and teamwork is important in many areas of our lives. People come together to share and discuss ideas, split and distribute work or help and support each other. The sharing of information and artefacts is a central part of collaboration. This often involves the manipulation of shared objects, both sequentially as well as concurrently. For coordinating an efficient collaboration, communication between the team members is necessary. This can happen verbally in form of speech or text and non-verbally through gesturing, pointing, gaze or facial expressions and the referencing and manipulation of shared objects. Collaborative Virtual Environments (CVE) allow remote users to come together and interact with each other and virtual objects within a computer simulated environment. Immersive display interfaces, such as a walk-in display (e.g. CAVE), that place a human physically into the synthetic environment, lend themselves well to support a natural manipulation of objects as well a set of natural non-verbal human communication, as they can both capture and display human movement. Communication of tracking data, however, can saturate the network and result in delay or loss of messages vital to the manipulation of shared objects.

This paper investigates the reality of shared object manipulation between remote users collaborating through linked walk-in displays and extends our research in [27]. Various forms of shared interaction are examined through a set of structured sub tasks within a representative construction task. We report on extensive user-trials between three walk-in displays in the UK and Austria linked over the Internet using a CVE, and demonstrate such effects on a naive implementation of a benchmark application, the Gazebo building task. We then present and evaluate application-

level workarounds and conclude by suggesting solutions that may be implemented within next-generation CVE infrastructures.

Keywords: CVE, shared object manipulation, communication, IPT, immersive collaboration

1 INTRODUCTION

Advances in immersive display devices are ensuring their acceptance in industry as well as research. Within an immersive device, a user may walk around an object, move the body and head to examine it from every angle and manipulate it with the hand. Walk-in (CAVE-like) displays increase this naturalness by allowing you to see your own body within the environment.

Many team related tasks in the real world centre around the shared manipulation of objects. A group of remote users can be brought into social proximity to interactively share virtual objects within a Collaborative Virtual Environment (CVE). CVEs are extensively used to support applications as diverse as industrial design review, medical simulations, military training, online games, and social meeting places. A user's real body may be situated within a group of remote users congregated around a shared object by linking walk-in immersive displays through a CVE infrastructure. This allows each team member to use their body within the space to interact with others and virtual objects. The spoken word is supplemented by non-verbal communication in the form of pointing to, manipulating and interacting with the object as well as turning to people, gesturing and other forms of body language. This offers unprecedented naturalness of interaction and remote collaboration.

The actions of a remote user are often reflected through an articulated human-like embodiment, the avatar. Driving an avatar from motion tracking of a user within an immersive display device considerably improves non-verbal communication. The high frequency of tracker updates can, however, saturate the network resulting in delay or loss of other messages describing vital interactions with shared objects, which synchronise or trigger actions and are essential for steering the application. The constraints of network technology and the CVE system can make the experience of shared interaction both disappointing and frustrating.

This paper investigates the reality of shared interaction of common objects between users in distributed immersive walk-in displays and presents solutions to address the effects of remoteness. Various forms of shared object manipulation and human interaction are examined through a benchmark application that employs the structured task of building a Gazebo. We report on a number of trails between three walk-in displays in the UK and Austria. These were linked over the Internet using the DIVE-Spelunk immersive CVE [33]. This paper is an extension to [27].

1.1 Related Work

Various forms of interaction with shared objects have been considered. A simple ball game is using prediction to overcome the effect of network delays in a football game between UK and Germany [23]. The advanced ownership transfer allows instantaneous exchange of a ball between players in competitive scenarios. In IEEE 1516 concurrency control is defined to allow various attributes of a given object to be affected concurrently by distinct users. Roberts et al. describe optimisations above the standard that allow control of an artefact to be passed to a remote user with little or no delay [31]. A hierarchy of three concurrency control mechanisms is presented in [18] to tailor the problem of "surprising" changes during closely coupled collaboration. A virtual tennis game is played between remote sites in Molet et al. and Basdogan et al. investigate the importance of haptic interfaces for collaborative tasks in virtual environments [19, 1]. The authors state that finding a general solution to supporting various collaborative haptic tasks over a network may be "too hard". A distinction is made between concurrent and sequential interaction with shared objects but this is not discussed further. As with Choi et al. [7] a spring model is used to overcome network latencies to support concurrent manipulation of a shared object. Four classes of shared behaviour: autonomous behaviours, synchronised behaviours, independent interactions and shared interaction are introduced by Broll [3]. Levels of cooperation within CVEs have been categorised by a number of research groups in similar ways: Ruddle et al. described the different levels of cooperation as level 1 – *co-existence and shared-perception*; level 2 – *individual modification of the scene*; and level 3 – *simultaneous interactions with an object* [28]. A similar taxonomy was presented for haptic collaboration that describes the respective levels as *static*, *collaborative* and *cooperative* [5]. Our studies provide a more detailed taxonomy of level 3, which will be described later.

The COVEN project [12] undertook network trials of large scale collaborative applications run over the DIVE [6] CVE infrastructure. This produced a detailed analysis of network induced behaviour in CVE applications [13]. DIVE was ported to CAVE-like display systems [33] and consequently an experiment on a non-coupled interaction task with two users in different walk-in displays was found to be very successful [30]. It was shown that closely coupled concurrent interaction with a shared object was not possible with CVE technology in 1995 [2]. However, causal surface manipulation allows two users to carry a shared object while hiding the effects of latency through gradual deformation [29].

Recent work investigated carrying a stretcher by allowing the material to follow the handles [20]. The work concludes that, although the Internet2 [15] has sufficient bandwidth and levels of latency to support joint manipulation of shared objects, the CVE did not adequately address the consistency issues arising from the networks characteristics.

Immersive displays place a user in a spatial social context allowing natural first person observations of remote users interacting with objects. This improves human communication and the work within such an environment [27], and when connected

with other non-immersed users it can be observed that the immersed user adopts a leadership role [32, 34].

1.2 Principles of Distribution within CVEs

A key requirement of Virtual Reality (VR) is the responsiveness of the local system. Delays in representing a perspective change following a head movement are associated with disorientation and feelings of nausea. A CVE system supports a potentially unlimited reality across a number of resource-bounded computers interconnected by a network. The network, however, can induce perceivable delays in updating a distributed simulation. Key goals of a CVE are to maximise responsiveness and scalability while minimising latency. This is achieved through localisation and scaling.

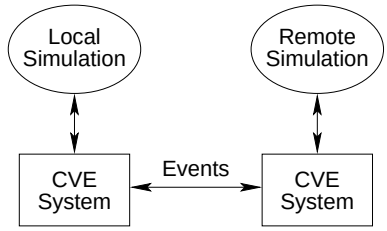

Fig. 1. Localisation through distribution

Localisation is achieved through replicating the environment, including shared information objects and avatars, on each user's machine. Sharing experience requires that replications be kept consistent. This is achieved by sending changes across the network in the form of events (see Figure 1 for an illustration of localisation). Localisation goes further than simply replicating the state of the environment; it also includes the predictable behaviour of objects within it. The organisation and content of a scenegraph is optimised for the rendering of images. Although some systems [17, 35] directly link scenegraph nodes across the network, most systems introduce a second object graph to deal with issues of distribution. Known as the replicated object model, we will from here on refer to it as the replication and to its nodes as objects. Objects contain state information and may link to corresponding objects within the local scenegraph.

A virtual environment is composed of objects, which may be brought to life through their behaviour and interaction. Some objects will be static and have no behaviour. Some will have behaviour driven from the real world, for example users. Alternatively, object behaviour may be procedurally defined in some computer program. In order to make a CVE attractive and productive to use it must support interaction that is sufficiently intuitive, reactive, responsive, detailed and consistent. By replicating object behaviour we reduce dependency on the network and therefore make better use of available bandwidth and increase responsiveness. Early systems

replicated object states, but not their behaviour. Each state change to any object was sent across the network to every replica of that object.

1.3 Natural Communication and Interaction

Immersive environments support social human communication (SHC) in an unprecedented way that draw on universal primitives in the way people understand things, events, relationships and information. We are embodied beings, for which meaning ultimately resides in bodily experiences, which is a virtue difficult to achieve on a desktop and because of this, immersive displays naturally support social communication and interaction. We have evolved to act in the physical world and how we are able to understand abstract information is derived from this capacity. If we design for embodiment and immersion, understanding and communication comes free, which is the first step in creating a system that supports our natural way to socially interact with each other.

Social communication is a dynamic process that has been under investigation for many years by both psychologists and sociologists. It can be categorised into four basic forms: verbal and nonverbal communication and the role of objects and the environment in communication [4, 16]. Those forms of SHC are differently supported by technology allowing us to communicate and interact with others (Table 1).

Forms of SHC	Example	Telephone	Video-conference	Typical CVE	Immersive CVE
verbal	natural speech	natural	natural	natural	natural
non-verbal	gesture, posture and facial expression	not available	natural	unnatural	natural
objects	artefacts of interest, person and non-person related	not available	not shared, natural	shared, unnatural	shared, natural
environ-ment	set the scene for natural collaboration and communication	not available	look into other's	look into shared	physically situated in shared

Table 1. Supporting SHC across distance

A system that does not constrain us or forces us to change our behaviour should be the ultimate objective for creating a CVE that is efficient and socially involving, and the current status of technology leaves us a long way to go before we can achieve this goal.

1.4 Road Map

This paper uses the structured task of building a Gazebo to examine various forms of shared object manipulation between users in distributed walk-in devices. The Gazebo, along with lessons learnt in prototyping, is presented in Section 2. Revisions to the application and CVE, along with a detailed analysis of the effect of network delays, are given in Section 3. Section 4 discusses our findings and Section 5 concludes and suggests how a CVE could be improved to overcome our problems without resorting to application level constraints.

2 THE GAZEBO PROTOTYPE

We have designed the structured task of building a gazebo in order to examine distinct scenarios of sharing the manipulation of an object and as a benchmark for further investigations. This section introduces the original Gazebo, describes how it was tested between the UK and Austria and how results of these led to a rethink.

A Gazebo is a simple structure that is often found at a vantage point or within a garden. The working environment containts materials, tools and users. Wooden beams may be inserted in metal feet and united with metal joiners. Screws fix beams in place and planks may be nailed to beams. Tools are used to drill holes, tighten screws and hammer nails. To complete the Gazebo, tools and materials must be shared in various scenarios of shared object manipulation, distinct in the method of sharing attributes. Scenarios include planning, passing, carrying and assembly (see Table 2). The time taken to complete each scenario is a measure of the success of collaboration.

Scenario	Fig.	Description	Method of sharing
planning	2 a)	discussing how to proceed	referencing objects and environment
passing	2 b)	a tool or material is passed from one user to another	sequential sharing and manipulation of the same object attribute
moving	2 c)	a wooden beam is too heavy to lift alone requiring one user to lift each end	concurrent sharing of object through the same attribute
assembling	2 d)	a wooden beam must be held in place by one user, while another fixes it by drilling a hole and inserting a screw	concurrent sharing of an object through distinct attributes

Table 2. Scenarios of object sharing

These scenarios are a more detailed taxonomy of level 3 of the categorisation of Ruddle et al. [28], which the authors describe as simultaneous interactions with

an object. We extend this with the method of sharing attributes during object manipulation, as summarised in Table 3.

Timing	Manipulated Attributes	
sequentially / concurrently	distinct	same

Table 3. Methods of sharing object attributes

2.1 Building the Gazebo

On logging in, the user is placed in a garden strewn with building materials and tools. Avatars appear, as the rest of the team enter the garden. "Wonder"-stacks keep the building site tidy by creating materials on demand. A user can take material from a nearby stack and start to build the Gazebo. In the real world, constructing a Gazebo on your own is not an easy task. To simulate the real world task we introduced some constraints. The simulation of gravity prohibits leaving materials in thin air and makes some materials too heavy to lift alone. The only task a single person can undertake is to drill holes and fit nails or screws. Moving, positioning and building all require teamwork. For example, one user must hold a joiner in place so that another user can fix it with a screw. In the following sections, we examine the four scenarios of planning, passing, moving and fixing, as summarised in Table 2, in more detail.

2.1.1 Planning and Instructing

The task of building the Gazebo routinely requires communication of the referencing of objects as well as the place within the environment that they are to be taken, (see Figure 2 a). Communication of referencing must reflect nuances of speech and gesture and the interface must not restrict the recipient from capturing these. When using a walk-in display, control of gaze and pointing are driven through a tracking system and the user is surrounded by the display surface to the front, both sides and the floor. The complexity of the task requires the collaborative planning of a number of steps, which may involve several collaborators and objects. A wide field of view and direct control and communication of gaze and pointing should allow efficient referencing, location and identification of each.

2.1.2 Passing a Tool

A hand-held "multi"-tool can be fitted with the necessary attachments for construction. A drill makes holes in wood and metal, a screwdriver tightens screws and a hammer hammers nails. The garden only contains a single tool so that users will need to pass it between each other. Ideally, only one user can hold the tool at a time but can pass it smoothly to another user. Passing the tool (see Figure 2 b)

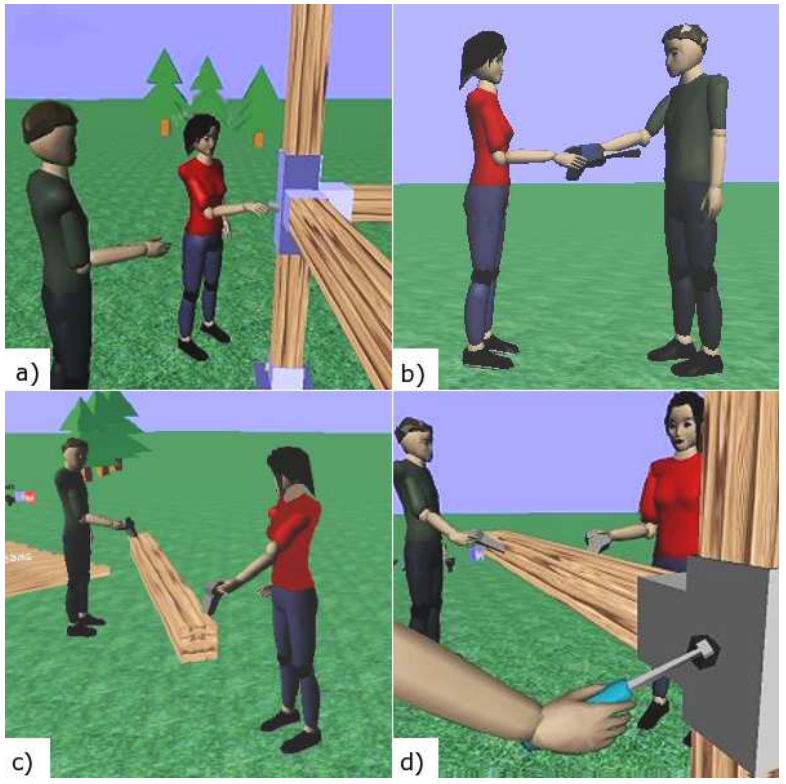

Fig. 2. Collaborative scenarios when building a gazebo. a) Planning and Instructing, b) Passing a Tool, c) Moving a Beam, d) Fixing a Beam

demonstrates sequential manipulation of the movement attribute as well as that of ownership. (With movement attribute we refer to position and orientation, which may also be described by a path communicated between replicas.)

2.1.3 Moving a Beam

A wooden beam is artificially made too heavy to lift alone requiring one user to lift each end (see Figure 2 c). This demonstrates the concurrent manipulation of the movement attribute of the beam. Ideally, when two users attempt to drag the beam in opposing directions, it should move to a mean position between them.

2.1.4 Fixing a Beam

Beams can be united with a metal joiner and screws. A joiner may be attached to a beam by drilling a hole through both and fixing with a screw. A second beam can then be fitted into that joiner in a similar manner. One person must hold a beam

while it is attached to prevent it falling (see Figure 2 d). This demonstrates that one user is able to affect the attribute for fixing while another affects those of movement; in other words, the concurrent sharing of distinct attributes.

2.2 Application Design

The Gazebo application was developed to work over the DIVE CVE [6]. DIVE was chosen because firstly, it is a well-established and widely accepted CVE tested, and secondly because of the ease of application development. The immersive extension Spelunk [33] allowed us to link various combinations of walk-in displays and desktop systems.

The application is implemented as a set of interactive objects. Three classes of objects were needed: avatars, materials and tools. Each object has a graphical representation and all have scripted behaviours that match their purpose. In DIVE, all objects are structured hierarchically in a distributed database. Their current state is represented by attributes, which may be brought to life by user-defined object behaviour scripts. All behaviour scripts are reactive and triggered by specific DIVE events. These are update messages, generated by the CVE system to update replicated versions of the distributed virtual environment. DIVE supports several event types. These include object transformation events, such as movement or rotation; object interaction events, such as grasp, release or select events; object collisions; and changes to object-specific properties and flags. Most functionality of the Virtual Gazebo is triggered by collisions of material and tool objects. For example, when a drill tool is held closely to a material object so that they collide, the resulting collision event, generated by the system, would trigger a procedure in the material object's behaviour script to increment a "hole-counter" property.

2.2.1 Object Behaviour

One behaviour that all objects implement is reacting to gravity. For simplicity, the objects just move back to ground level when released, rather than following a path with increasing velocity. There is no collision detection during the fall. In some cases, reaction to gravity is deactivated. This happens when the object is a material that has been fixed to another material, which is internally signalled by the fixed-flag of that object. In the case of a nail or screw that has been stuck into a material part, the stuck flag would deactivate gravity of this object as well. A simplified script for the gravity behaviour would look like this:

```
React to gravity {
 If (NOT stuck OR fixed) {
  Calculate distance to ground
  Move by this distance
}}
```

Avatars are used to represent a user inside a virtual world. The immersive extension of DIVE, Spelunk [33], gives support for lifelike avatars, by which the tracking data of the display is used to animate the body parts. Most immersive displays support tracking of the head and both or at least one hand. This, in combination with a simple inverse kinematics algorithm within Spelunk, allows the representation of arm-articulation for gesturing or pointing, as well as gaze or nodding with the head. For the distribution of those movements the DIVE internal distribution-layer is extended to permit multi-transform messages. This allows sending multiple transformations, for example that of an arm movement, in a single message. We have used both rigid and jointed body avatars in our tests. The rigid body shows no movement except locomotion of the entire body. The second one supports articulated head and arm articulation. Head and hand movement is controlled by live tracking data from the user within an immersive walk-in display.

Materials are the building blocks of the benchmark application. They consist of planks, beams, and joiners to connect beams, nails and screws. Tools, like hammer, drill and screwdriver, are used to interact with the materials. Planks and joiners can be moved by one person, whereas beams have to be lifted by at least two people. This simulated weight is achieved by counting the number of acting users in the beam's script. If less than two users are grasping the beam, it just pretends to be released and moves back to ground. If more than two users are grasping, it follows the hand movements of the users. Here, a convergence of the beam's position between the hands is expected. A simplified script would look like this:

```
Grasp {
 Increment acting users
}
Move {
 If (Acting users > 1)
  React to gravity
 Else
  Align and update
}
```

Planks, beams and joiners are fixed by simply letting them not react to gravity anymore when released. Before fixing a plank, a nail has to be inserted and hit with the hammer, whereas for fixing beams with joiners a screw and the screwdriver are used. Tool objects interact directly with behaviour scripts of nails and screws. When these collide, the tool sends a signal to the nail or screw's behaviour script that tells them to perform a fix-procedure. Nails and screws, in turn, interact directly with the other material parts when they fix them. Nails can always be inserted into a plank. As soon as they collide, the stuck flag of the nail is set and its gravity behaviour is deactivated. A screw, however, requires a hole. This is realised by a holes-counter in material parts that increases on collision with a drill tool. If the screw collides with a beam or joiner with a hole, then it will fit in place. Once in place it be can tightened with a screwdriver.

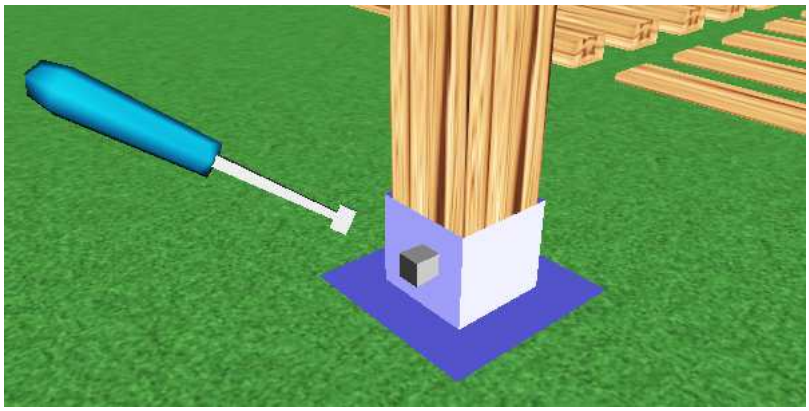

Fig. 3. In the centre: a screw intersecting a beam and a joiner. On the right: a screwdriver about to collide with the screw

Screws may fix more than one material object at once if they are inserted in overlapping objects (Figure 3). However, until the parts are not fixed they may still move. For instance, the supporting person may wobble the hand that holds a beam so that the beam is not positioned exactly inside the joiner. We want to fix only those parts that intersect with the screw. For intersection testing we need to know which objects are involved. So, when nails or screws collide with material parts, they keep those object names in a list. On a fix command, initiated by a colliding tool, this list is then used to test for intersections with once collided objects. A simplified behaviour script for a screw looks like:

```
Check collision{
 If (colliding with material AND
   spare holes in collided object)
  Add to intersection list;
 }
```

A better understanding of the application may be gained through Table 4, which details the various object attributes that define the shared behaviours of our objects. These are added to the default attributes such as position, rotation, parenting and graspable.

DIVE provides loose consistency in which events are sent via fast but unreliable multicast messages, relying on eventual convergence of the replicas. The DIVE/Tcl scripts are distributed and executed once on each remote node, ensuring an identical initial state. However, when DIVE events occur, an event notification will run on that node only where it occurred. The distribution layer of DIVE is responsible for delivering this event to all other nodes so that they update their simulation. Distribution of a shared environment introduces the possibility of inconsistency, caused by latency and message loss. Inconsistencies between remote replications of

Attribute	Purpose	Effect of critical divergence
all objects		
"falling"	applies gravity to an object	would stay in the air when released
"users"	count number of acting users to simulate mass	would not be movable
all materials		
"fixed"	fix two materials together by disabling further manipulation and gravity	would still fall down or be still graspable although fixed
beam, plank and all joiners		
"holes"	count number of holes	screws could not be inserted if hole did not appear
screw and nail		
"coll_list"	remember collided objects for intersection testing when fixing parts	would not fix some parts
"stuck"	signalise state for fixing parts and impact of gravity	would fall down and not stay in or allow fixing a part
power tool		
"curr_tool"	handle to currently active attachment	would apply incorrect attachment

Table 4. Shared attributes and the effect of divergence

attributes may lead to divergent behaviour of a shared object, creating confusion between users.

Most functionality in the Gazebo is based on collisions. It has been tried to optimise the object scripts so that they apply incoming events only, when they make any sense. For example, a screw would fix parts only, when it collides with a screwdriver. Several tasks require a certain order of events to occur. For example, a beam must first have had contact with a drill to increase its holes counter, before a screw can be inserted and then tightened with a screwdriver. When the collision signals do not arrive in order, the logic of the behaviour scripts would "not be matched" and the current task cannot be finished without repetition.

Visual feedback is given for all events that change the state of the object. This was sometimes implemented by flashing objects to signalise that an action has taken place. If responsiveness of the CVE system is low, it is expected that the human can adapt. In the real world, tasks are synchronised via natural communication or repeat task. Both natural communication between users and visual feedback helped to synchronise structured tasks. This way, synchronisation between the tasks is supported by the application's semantic.

2.3 Experimentation

The prototype Gazebo application has been tested between walk-in display devices across Europe. The majority of trials were undertaken between walk-in displays at the University of Reading (UK) and Johannes Kepler University Linz (Austria). On two occasions, these were joined by another at University College London (UK) and more recently from the University of Salford. Further desktop users often joined from Reading and Linz. Spelunk, an immersive extension to the DIVE CVE [33], was used to link the walk-in displays. Here, we present findings of the first application prototype which was regularly tested between sites over a three week period.

DIVE uses multicast, which is used extensively by many CVE systems to increase scalability of group communication. Although multicast works within a local area network, it is usually necessary to tunnel multicast packets between local area networks, particularly when they are separated across the Internet. DIVE proxy servers [11] were used to tunnel packets between local area networks at each site. Audio communication was supported through the UCL Robust Audio Tool (RAT) [14].

2.4 Results

Using the prototype Gazebo, each user was able to interact with objects successfully and it was generally easy to interpret what remote users where doing, especially with the support of audio communication. This reinforces the findings of other work, like [13]. The actions and gestures of tracked users were much easier to understand than those of desktop counterparts.

Two problems, however, severely hampered collaboration around shared objects: Firstly, the (although lose) ownership mechanism in DIVE made it difficult for two users to carry a beam concurrently. Secondly, many important interactions with shared objects were not being reflected remotely, such as creating objects or grasping parts. With these problems it was very difficult to build the Gazebo. We lightened the beam so that one user could lift it and undertook user trials to see what could be achieved. Users in a link-up between the three walk-in displays achieved what resembled a sloppily constructed corral or sheep pen. A series of later test between Reading and Linz with users of various experiences did not improve upon this. In addition we had communication problems, as it was not always clear if an action such as drilling a hole was successful due to the lack of feedback on the remote side.

An investigation was undertaken into the loss of remote representation of interactions with objects. The effects of a remote user's interaction with an object were seldom presented. This was most apparent with the following interactions: creating a material from a "multi"-stack; picking; passing; drilling holes; inserting nails or screws and switching tool attachments. Unlike the above user-to-object interactions, movement was always represented remotely. The primary difference between the two is the frequency and importance of updates. Our avatar's movement was represented by a continuous stream of position and orientation events. The effect of

losing movement events in transit is an increase in the jerkiness of avatar movement, something an observer can cope with. In contrast, the effect on an object of user interactions is communicated by a short burst of events, that if lost will result in a lack of remote representation.

We undertook extensive tests to verify a hypothesis of event loss and why this should be a particular problem for shared manipulation between walk-in displays [36]. The movement of avatars, materials and tools all increased during shared manipulations, causing bursts of events at exactly the time when reliability and low latency were needed. These bursts were evident in latencies rising to several seconds for scenarios such as fixing beams with a joiner. We found that the problem did not arise when representing a desktop user interacting with an object. The avatar used to represent a desktop user is simpler than that used for the user of a walk-in display. We tried a simpler avatar to represent the immersed display user and found this to solve the problem. The new avatar had less moving parts and thus produced less network traffic to update. Although this avatar solved one problem, its simplicity made human-like, non-verbal communication much harder.

A follow-up investigation revealed that with every tracking update the whole avatar body is updated and then all those updates are sent over the networks integrated in multi-transform messages. However, this produces considerably more movement events as the simple transfer of the head and hand position. The later and a remote calculation of the complete avatar may reduced accuracy, needs more computational power on remote sites, but would drastically reduce the event traffic created by the immersive user. Unfortunately, the current implementation of Spelunk within DIVE makes it difficult to disentangle the source code and to implement the last suggestion.

DIVE incorporates an optional reliable message service, Scalable Reliable Multicast (SRM) [10]. When enabled, SRM ensures all messages from that user's device are delivered. Enabling SRM, while using the more complex avatar, ensured the representation of the effect of remote interactions with an object. The drawback of using SRM was a lag of greater than a second in the representation of the actions of a remote user, including movement and interaction.

3 IMPROVED GAZEBO

The earlier trials showed that the implementation of the Gazebo prototype application lacks heterogeneous mechanisms for concurrent sharing of objects. The reason for this can be found in the ownership transfer when grasping an object. This is to avoid consistency problems in interactive environments. Both, DIVE and Spelunk, implement this in particular ways. Spelunk implements manipulation of a selected object through a change in parenting within the scene graph. The selected object, in this case the beam, is detached from its parent and re-attached as a child of the user's virtual hand. The virtual hand follows tracking information from the wand held by the user. Therefore, a script managing the beam's height according

to gravity would have no effect, as the position is always overwritten by data from the wand. This has also the implication that users in two linked walk-in displays cannot manipulate a shared object at the same time. DIVE simply adds translation data, derived from translations of the mouse, to the transformation matrix of an object. Many users can, however, manipulate an object concurrently. An object that is being carried can be manipulated with respect to the carrier. To overcome this problem, intermediate carrying objects have been implemented to support cooperative manipulation of the beam. Now, two carrying tools attract the beam to align its position and orientation between them. With this solution, hierarchy changes affect the tool instead of the beam, enabling concurrent manipulation of the latter.

The second fundamental problem we experienced during our tests was the loss or disorder of critical messages that disturb the consistent state of the environment. The effects of a remote user's interaction with an object were seldom presented. These occurred mostly at hierarchy changes, e.g. when creating new objects at runtime or switching the attachments of a multi tool. An investigation was undertaken into the loss of remote representation of interactions with objects. The main reason for message loss was found to be an overwhelming frequency of movement messages for the joint avatar generated by the tracking system in a walk-in display. Our avatar's movement was represented by a continuous stream of position and orientation events. The effect of losing movement events in transit is an increase in the jerkiness of avatar movement, something an observer can cope with. In contrast, the effect of user interactions on an object is communicated by a short burst of events, that if lost will result in a lack of remote representation.

Solutions for this would be firstly, reducing the tracking rate and secondly, to simulate the arm-articulation at client side based on the head and hand position, rather than transmitting the movement of each single limb. Both solutions are accessible by an interface of Spelunk. Additionally, the amount of communication between the behaviour scripts could be reduced within the application. For instance, update messages that originated from carrying tools to affect the beam's alignment, were first sent at tracker update rate, which was about 10 times per second. These have been reduced to a single delivery each second, reducing considerably the jerkiness of beam movement. To overcome problems of critical event loss the application was constrained to avoid vital, infrequent, events. "Wonder"-stacks were replaced by stocked material stores. The universal "multi"-tool was replaced by separate tools to avoid the dynamic switching of state.

For supporting the synchronisation of interactions and communication, a feedback in form of visual clues was implemented. This feedback was a colour-change of the drill when drilling a hole as well as adding an additional transparent coloured aura around one end of a beam as soon as it has collided with a carry tool.

To overcome the problem of the high amount of event occurrences, introduced by the tracking system, we enhanced our DIVE version with an event filter based on the magnitude of movement allowed to generate an event only of movements above 1 cm. We found that this is a good compromise between sufficient detail to support understandable non verbal communication and sufficient synchronisation to

	Prototype		Revised	
Attribute	Trigger event	Typical occurrences per scenario	Trigger event	Typical occurrences per scenario
"falling"	move/release	>100	grasp/release	1-5
"users"	grasp/release	1-5	not used	
"fixed"	collision	1	collision	1
"holes"	collision	1-10	collision	1-10
"coll_list"	collision	1-5	collision	1-5
"stuck"	released	1	released	1
"curr_tool"	select	2-3	not used	

Table 5. Frequency of attribute modifications

achieve shared object manipulation. Many events below this seemed to be caused by tracking jitter rather than real user movement. This reduced the frequency of events and allowed us to use our more human-like avatar. Finally, the complexity of application level scripts was reduced to minimise the frequency of events. Table 5 compares expected event occurrences between the prototype and revised gazebo for various attributes, as detailed in Table 4.

3.1 Experimentation

Again we attempted to build the Gazebo in a number of linkups between the three sites, tuning event communication to gain acceptable levels of latency and reliability. In earlier experiments we compared the usability of the application while enabling and disabling reliable multicast for all events [24]. Investigation of the DIVE source code unearthed a way to map reliability to three categories of events: movement, geometry and general (everything else). Figure 4 shows the result of a successful collaboration of about 30 minutes.

3.2 Results

In order to obtain a workable level of reliability, while three users shared the manipulation of objects, it was necessary to reduce the rate of sending of avatar movement to the network to 5 Hz while maintaining 10 Hz for the shared objects. For example, this was found sufficient when three users fixed a beam, one holding the beam, another drilling the hole and a third inserting a screw. In order to gauge latency between displays we undertook a wave test. A user at UCL moved his hand up, down, left then right, speaking the movements as he did them. At a 5 Hz update rate, these movements were reflected in Reading before the spoken word, suggesting that the CVE had less latency than the audio tool. Network latencies between Reading and UCL typically vary between 15 and 25 ms. The reduced update rate of avatars resulted in less natural movement making it harder to interpret their actions.

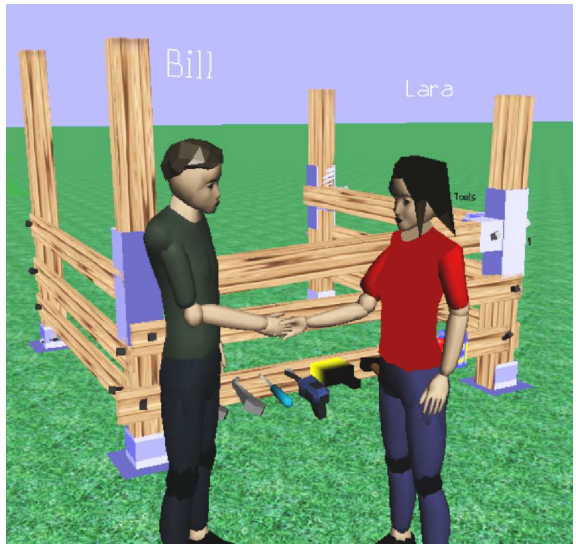

Fig. 4. Result of a 30 min Gazebo session

Objects still became un-graspable after they had been picked by another user but far less frequently than in the original Gazebo. The reliability of infrequent state changes such as drilling holes and inserting screws was also increased. Infrequent loss of such changes was often overcome through teamwork. For example, if the creation of a hole is not reproduced at all sides, users can report this and ask for another hole to be drilled. The introduction of a carrying tools enabled joint manipulation of beams. Human communication and visual clues helped synchronise lifting of the beam and choosing a direction in which to carry it. Although latency was not apparent in avatar movement, remote manipulation of the beam was often delayed by up to one seconds. This resulted in wild beam movement not unlike that of a rodeo horse. The fact that only one object was affected suggests a backlog of interpreted script events as opposed to filling of a receive buffer, which would have effected all. A change of the tracker rate limit from 100 ms to 200 ms seemed to solve the delay problem.

Enabling reliable multicast for all events solved the problem of event loss but brought latency up into the order of seconds, even with avatar movement update was set to 5 Hz. This suggests that SRM is not appropriate to our test conditions. In the current DIVE version SRM cannot be applied to selected event types only. Enabling reliability for only vital messages could solve problems of critical event loss without incurring excessive latency. For example, movement events could be send unreliable since they will be updated frequently.

4 DISCUSSION

We have found that multiple tracked users sharing the manipulation of common objects can lead to unacceptable latency and level of reliability but have shown how this may be overcome through careful tailoring of the application and configuration of event communication within the CVE. The Gazebo benchmark application was constrained to avoid vital changes to shared objects that might easily be lost, for example, minimising the dynamic creation of objects and tools with various attachments were replaced with separate tools for different jobs. Carrying objects addressed problems of joint ownership in terms of hierarchy and movement. Reducing the event communication rate from 10 Hz to 5 Hz reduced latency close to the level of perception and provided acceptable reliability. Although problems of remoteness can be tackled in the application, it would preferable to solve problems within the CVE itself, setting the application programmer free of concerns of the network.

The latency we have experienced is orders of magnitude grater than that of the network and comes from events being received faster than they can be processed. The problem of event loss may be related to this, and arised from overflow of receive buffers. Movement events generated from tracking are highly frequent whereas those describing vital object manipulations, such as a pick, come in short bursts. It appears that the latter are being lost by being overwritten by the former. This problem is exacerbated by a bucket algorithm within DIVE that throws away events when too many are received. Although these problems arise from receiving tracking generated events from many users, they cannot be addressed with traditional scalability mechanism such as awareness management, level of detail and augmentation. These address scalability of large groups of users with subjective views of the environment. Here we have a small group sharing the same view. Mechanisms within the network level offer more appropriate answers. Categories of events, for example, may be mapped to various qualities of service for delivery. CAVERNsoft [17], a network framework for walk-in displays, and PING [21] organise these mappings in channels, each of which has its own send and receive buffer. CAVERNsoft, unlike DIVE, relies on the application programmer determining event categories. Typically one channel is dedicated for movement and another for everything else. Thus, restricting a channel to low frequency events may decrease their latency.

The problem, however, does not stop there. Many events are causally dependent on others sent down a separate channel. For example, many CVEs communicate movement relative to an object's parent, such events become invalid when the object is picked up and those generated after the pick must not be delivered until after the pick event. For example, in a networked ball game [26] a delayed ball movement coming after the ball had been caught by a player resulted in the ball appearing under the ground and becoming stuck there. Neither DIVE nor CAVERNsoft can cope with this type of divergence. This problem was addressed in PaRADE [25], which made a distinction between causally supportive and relative events, sending the former reliably and ensuring any event was only enacted if causally supported.

In addition to a causal time stamp, all events were stamped with natural (wall clock) time, allowing superseded events to be discarded, thus reducing latency.

The results above show that traditional CVEs are no longer suitable to work with modern technology, such as walk-in displays [9] and the existing CVE requirements [6, 8, 22, 37] need to be extended. CVEs have worked well as a scalable multi-user environment between desktops, but the use of immersive displays with tracking systems requires a rethink in the design of CVEs. It may be argued that the scale of interaction events has not changed, but it is clear that of movement events have done so by a large amount and this is largely to increase the natural interaction and non-verbal communication. We postulate that extending the PaRADE approach to time management through the use of channels would provide a level of CVE adaptation capable of supporting applications like Gazebo without the need for the application programmer to worry about delay and event loss. Such a system could be improved through an application interface and language that supports hints and reflection. Hints may allow the application or application programmer to suggest event categories, defining acceptable latency and reliability of each as well as causal relationships between them. Reflection allows the application's behaviour to adapt to available qualities of service in terms of reliability, ordering and latency. A similar philosophy was taken in the design of PING, which unfortunately was not completed. We have addressed those issues in our latest CVE prototype ICE (Immersive Collaborative Environment), which implements a user interface, rendering capabilities, avatar support, and simple simulation control. ICE includes a system that allows the dynamic re-configuration of the event-handling infrastructure called FLOW. The basic idea behind FLOW is to provide a number of customised event-handling pipes, which process events according to a particular consistency strategy. The strategy can be re-configured on a per-event basis at run-time.

5 CONCLUSION

CVEs have been routinely used for linking desktop display systems over a decade. The Gazebo experiment has demonstrated that users, sharing the manipulation of objects, can adapt to the limited effects of remoteness between networked walk-in displays. Limiting these effects, however, required considerable effort in application development and deployment. Although many CVEs provide mechanisms for dealing with the effects of remoteness, these are barely sufficient for such linkups and require a combination of application constraints and workarounds as well as fine tuning of event communication. We concur with earlier work [32] that it is easier to collaborate with a remote user when their avatar is driven by tracking data. Walk-in and other immersive displays are different because the users are tracked and the communication of tracked human movement is data-intensive. This problem is exacerbated by the very different data requirements of shared object manipulation, where occasional vital events must be sent reliably and in order, often coincident with bursts of non-vital movement events.

We conclude that a CVE does not yet exist which is capable of supporting applications like the Gazebo across walk-in displays, without unnaturally constraining the application and laboriously tuning event passing. Combining best practice from CVEs such as DIVE, PaRADE and PING could overcome the critical problems of remoteness, such as minimising the effect of latency while providing sufficient levels of reliability and causality. Network latencies will always be perceptible for some forms of shared manipulation. Application workarounds such as intermediate objects, prediction [25], or causal surface manipulation [29] can help to reduce the effect of latencies on shared manipulation.

Acknowledgement

We wish to thank Christoph Anthes for helping with developmental trials between the UK and Austria and Dieter Kranzlmüller and Vassil Alexandrov for facilitating collaboration between the University of Reading and the Johannes Kepler Universität Linz. We also wish to thank Anthony Steed for facilitating collaboration to UCL London.

REFERENCES

[1] Basdogan, C.—Ho, C.-H.—Srinivasan, M.—Slater, M.: An Experimental Study on the Role of Touch in Shared Virtual Environments. ACM Transactions on Computer Human Interaction, Vol. 7, Issue 4, 2000, pp. 443-460.

[2] Broll, W.: Interacting in Distributed Collaborative Virtual Environments. In Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS'95), Research Triangle Park, North Carolina, 1995, pp. 148–155.

[3] Broll, W.: Populating the Internet: Supporting Multiple Users and Shared Applications with VRML. In Proceedings of Second Symposium on Virtual Reality Modelling Language (VRML'97), Monterey, California, 1997, pp. 33–40.

[4] Burgoon, M.—Hunsaker, F. G.—Dawson, E. J.: Human Communication, 3rd ed., London, SAGE Publications, 1994.

[5] Buttolo, P.—Oboe, R.—Hannaford, B.: Architectures for Shared Haptic Virtual Environments. Computers and Graphics, Vol. 21, Issue 4, 1997, pp. 421–429.

[6] Carlsson, C.—Hagsand, O.: DIVE – A Platform for Multi-User Virtual Environments. Computers & Graphics, Vol. 17, Issue 6, 1993, pp. 663–669.

[7] Choi, H.—Choi, B.—Ryew, S.: HapticDisplay in Virtual Collaborative Shared by Multiple Users. Proceedings of IEEE International workshop on Robot and Human Communication, 1997, pp. 478–483.

[8] Churchill, E. F.—Snowdon, D.—Munro, A. J.: Collaborative Virtual Environments. Digital Places and Spaces for Interaction. Springer Verlag, London 2001.

[9] Cruz-Neira, C.—Sandin, D.—DeFanti, T.—Kenyon, R.—Hart, J.: The CAVE: Audio Visual Experience Automatic Virtual Environment. Communications of the ACM, Vol. 35, Issue 6, 1992, pp. 65–72.

[10] Floyd, S.—Jacobson, V.—McCanne, S.—Liu, C.-G.—Zhang, L.: A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing. In Proceedings of Applications, Technologies, Architectures, and Protocols for Computer Communication (ACM SIGCOMM '95), Cambridge, Massachusetts, 1995, pp. 342–356.

[11] Frècon, E.—Greenhalgh, C.—Stenius, M.: The DiveBone – An Application-Level Network Architecture for Internet-Based CVEs. In Proceedings of VRST '99 – Symposium on Virtual Reality Software and Technology 1999, London, UK, 1999, pp. 58–65.

[12] Frècon, E.—Smith, G.—Steed, A.—Stenius, M.—Stahl, O.: An Overview of the COVEN Platform. Presence: Teleoperators & Virtual Environments, Vol. 10, Issue 1, 2001, pp. 109–127.

[13] Greenhalgh, C. M.—Bullock, A.—Frècon, E.—Lloyd, D.—Steed, A.: Making Networked Virtual Environments Work. Presence: Teleoperators and Virtual Environments, Vol. 10, Issue 2, 2001, pp. 142–159.

[14] Hardman, V.—Sasse, A.—Handley, M.—Watson, A.: Reliable Audio for Use over the Internet. In Proceedings of INET '95, Honolulu, Hawaii, 1995, pp. 171–178.

[15] Internet2, `http://www.internet2.edu/`, 1996.

[16] Knapp, M. L.: Nonverbal Communication in Human Interaction, 2nd ed.: Holt, Rinehart and Winston, 1978.

[17] Leigh, J.—Johnson, A. E.—Park, K. S.—Cho, Y. J.—Scharver, C.—Krishnaprasad, N. K.—Lewis, M. J.: CAVERNsoft G2: A Toolkit for High Performance Tele-Immersive Collaboration. In Proceedings of Symposium on Virtual Reality Software and Technology, Seoul, Korea, 2000, pp. 22-25.

[18] Linebarger, J. M.—Kessler, D.: Concurrency Control Mechanisms for Closely Coupled Collaboration in Multithreaded Peer-to-Peer Virtual Environments. In Presence: Teleoperators & Virtual Environments, Vol. 13, Issue 3, June 2004, pp. 296–314.

[19] Molet, T.—Aubel, A.—Capin, T.—Carion, S.—Lee, E.—Magnenat-Thalmann, N.—Noser, H.—Pandzic, I.—Sannier, G.—Thalmann, D.: Anyone for Tennis? In Presence: Teleoperators & Virtual Environments, Vol. 8, Issue 2, 1999, pp. 140–156.

[20] Mortensen, J.—Vinagayamoorthy, V.—Slater, M.—Steed, A.—Lok, B.—Whitton, M. C.: Collaboration in Tele-Immersive Environments. Presented at the Eighth Eurographics Workshop on Virtual Environments, Barcelona, 2002.

[21] Ping Project, `http://www.pingproject.org/`, 2001.

[22] Prasolova-Førland, E.—Divitini, M.: Supporting Social Awareness: Requirements for Educational CVE. Presented at the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT '03), Athens, Greece, 2003.

[23] Roberts, D.—Strassner, J.—Worthington, B. G.—Sharkey, P.: Influence of the Supporting Protocol on the Latencies Induced by Concurrency Control within a Large Scale Multi User Distributed Virtual Reality System. In Proceedings of International Conference on Virtual Worlds and Simulation (VWSIM), SCS Western Multi-conference '99, San Francisco, CA, 1999, pp. 70–75.

[24] Roberts, D. J.—Wolff, R.—Otto, O.—Kranzlmüller, D.—Anthes, C.—Steed, A.: Supporting Social Human Communication between Distributed Walk-in Displays. In Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST '04), Hong Kong, 2004, pp. 81–88.

[25] Roberts, D. J.: A Predictive Real Time Architecture for Multi-User, Distributed, Virtual Reality. PhD, Library of the University of Reading, 1996.

[26] Roberts, D. J.—Strassner, J.—Worthington, B. G.—Sharkey, P.: Influence of the Supporting Protocol on the Latencies Induced by Concurrency Control within a Large Scale Multi User Distributed Virtual Reality System. In Proceedings of International Conference on Virtual Worlds and Simulation (VWSIM), SCS Western Multi-conference '99, San Francisco, CA, 1999, pp. 70–75.

[27] Roberts, D. J.—Wolff, R.—Otto, O.: Pushmepullyou: The Reality of Interaction with Shared Objects in Networked Walk-in Displays. In Proceedings of the ISCA 17th International Conference on Parallel and Distributed Computing Systems (PDCS '04), San Francisco, CA, USA, 2004, pp. 470–477.

[28] Ruddle, R. A.—Savage, J. C.–Jones, D. M.: Symmetric and Asymmetric Action Integration During Cooperative Object Manipulation in Virtual Environments. ACM Transactions on Computer-Human Interaction (ToCHI), Vol. 9, Issue 4, 2002, pp. 285–308.

[29] Ryan, M. D.—Sharkey, P. M.: Distortion in Distributed Virtual Reality. In Proceedings of First International Conference on Virtual Worlds, Paris, 1998, pp. 42–48.

[30] Schroeder, R.—Steed, A.—Axelsson, A.—Heldal, I.—Abelin, A.—Widestom, J.—Nilson, A.—Slater, M.: Collaborating in Networked Immersive Spaces: As Good as Being There Together? In Computers and Graphics, Vol. 25, 2001, pp. 781–788.

[31] Roberts, D. J.—Lake, T.—Sharkey, P.: Optimising Exchange of Attribute Ownership in the DMSO RTI. In Proceedings of the Spring Simulation Interoperability Workshop SIW '98, Orlando, Fl., 1998, pp. 379–386.

[32] Slater, M.—Sadagic, A.—Usoh, M.—Schroeder, R.: Small Group Behaviour in a Virtual and Real Environment: A Comparative Study. In Presence: Teleoperators and Virtual Environments, Vol. 9, Issue 1, 2000, pp. 37–51.

[33] Steed, A.—Mortensen, J.—Frècon, E.: Spelunking: Experiences using the DIVE System on CAVE-Like Platforms. In B. Frohlicj, J. Deisinger, and H.-J. Bullinger (Eds): Immersive Projection Technologies and Virtual Environments. Springer-Verlag, Wien 2001, pp. 153–164.

[34] Steed, A.—Slater, M.—Sadagic, A.—Tromp, J.—Bullock, A.: Leadership and Collaboration in Virtual Environments. In Proceedings of IEEE Virtual Reality (VR '99), 1999, pp. 112–115.

[35] Tramberend, H.: Avocado: A Distributed Virtual Reality Framework. In Proceedings of IEEE Virtual Reality (VR '99), Houston, Texas, 1999, pp. 14.–21.

[36] Wolff, R.—Roberts, D.—Otto, O.: A Study of Event Traffic During the Shared Manipulation of Objects Within Collaborative Virtual Environments. Presence: Teleoperators & Virtual Environments, Vol. 13, Issue 3, June 2004, pp. 251–262.

[37] Zhao, H.—Georganas, N. D.: Collaborative Virtual Environments: Managing the Shared Spaces. Networking and Information Systems Journal, Vol. 3, Issue 2, 2001, pp. 225–247.

David J. Roberts is a Reader at the University of Salford where he directs research and teaching at the National Industrial Centre for Virtual Environments. His ultimate research aim is to develop technology that can reduce the need to travel for face-to-face meetings in the support of team work.

Robin Wolff is currently Ph.D. student at the Centre for Virtual Environments at the University of Salford, UK. His research interests are in system design of collaborative virtual environments, with focus on shared object manipulation, as well as human-to-human interaction and communication through interconnected immersive environments.

Oliver Otto is currently a Ph.D. student at the Centre for Virtual Environments (NICVE) at the University of Salford, UK. His research concentrates on social human communication during close coupled collaboration, including shared object manipulation and inter-human behaviour within a virtual environment.