Tele-sports and Tele-dance: Full-Body Network Interaction

Benjamin Schaeffer Hank Kaczmarski Lance Chong
Mark Flider Luc Vanier Yu Hasegawa-Johnson

Beckman Institute, University of Illinois at Urbana-Champaign
1-217-333-1527

{schaeffr, hank}@isl.uiuc.edu {l-cong, mflider, vanier}@uiuc.edu
yu@post.harvard.edu

ABSTRACT

Researchers have had great success using motion capture tools for controlling avatars in virtual worlds. Another current of virtual reality research has focused on building collaborative environments connected by networks. The present paper combines these tendencies to describe an open source software system that uses motion capture tools as input devices for real-time collaborative virtual environments. Important applications of our system lie in the realm of simulating interactive, multi-participant physical activities like sport and dance. Several challenges and their respective solutions are outlined. First, we describe the infrastructure necessary to handle full-body articulated avatars as driven by motion capture equipment, including calibration and avatar creation. Next, we outline the PC cluster solution chosen to render our worlds, exploring methods of data sharing and synchronization, both within the PC cluster nodes and between different sites in the distributed system. Finally, virtual sports require physics, and we describe the simulation algorithms used.

Categories and Subject Descriptors

I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism – *virtual reality*

General Terms

Design, Human Factors, Management

Keywords

Immersive Virtual Environment, PC Cluster, networking, motion capture

1. INTRODUCTION

This paper describes experiments in the integration of full-body motion capture technology and systems for experiencing shared virtual worlds, examining the new types of interaction this combination facilitates. The last few years have seen an explosion of high-end commercial motion capture tools, mostly aimed at creating content for the entertainment industry. However, these tools can also power real-time interactive experiences, thus enabling teleimmersion and telecollaboration using a participant's entire body.

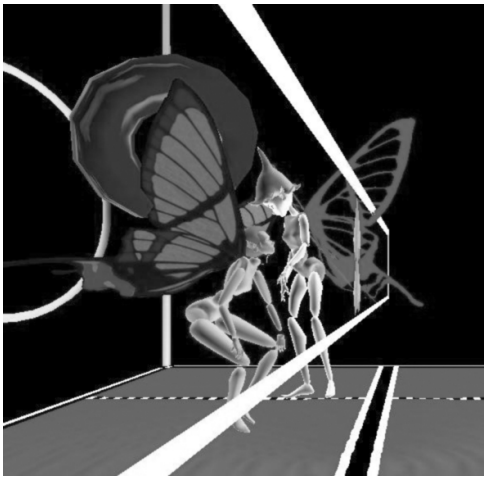

Figure 1. Fairy Sports: Two avatars play a game

Important new types of shared virtual worlds become possible once a participant's entire body is involved. Virtual sports, mimicking the physical engagement of a real sporting event, can be experienced. In addition, collaborative dance, among other performing arts possibilities, can become a reality. With this technology, a sports league with no physical court will become practical. Participants will be able, by changing their avatar, to interact with simulation-rich virtual worlds in ways impossible in the physical world. For instance, a virtual basketball match could

even the odds between short and tall players by letting them drive avatars of equal heights in the virtual arena.

A wide variety of shared virtual worlds exist. Real-time interaction over distance is now common via text-based chatting, first person shooter games, or massively multiplayer online role-playing games. Successful research efforts have also explored telecollaboration very thoroughly, be it in virtual prototyping for engineering [17], collaborative science, as with high-end virtual reality [33], or advanced immersive teleconferencing, as in the National Tele-Immersion Initiative, some of whose research is described in [8]. Other experiments in telecollaboration are also worthy of note, such as EVL's NICE [15], where geographically dispersed children can work together to tend a virtual garden.

However, consumer forms of telepresence, like chat or MMORPG's, rely on mouse and keyboard as input devices, which puts a significant experiential barrier between the user's physical space and the shared virtual space. Common virtual reality interfaces, such as projection-based virtual environments [7], put the user in an artificial space, but, usually, just two or three sensors are used, not nearly enough to map the user's entire body into the virtual world.

The virtual reality community has, however, explored using motion capture technology as an input device to drive avatars in virtual environments. Early work focused on the mechanics of driving the avatar from the motion capture data, examining such methods as filtering noisy input and using physically-based avatar models to shape the data [22]. Related research has created a methodology for controlling an avatar, possibly of very different proportions and features than the controlling actor [29]. Other research efforts have explored using motion capture devices to let a participant receive training in a physical skill, such as Tai Chi [6]. In this particular case, the user wears an HMD and can see his own avatar, as well as the computer-controlled avatar of a Tai Chi master.

In addition, the arts community has embraced motion capture as a tool capable of adding new modalities of expression to human motion. Choreographers can clothe human motion data with interesting geometries to create performances that would be impossible to stage in the physical world. The Ghostcatching installation is a striking early example of this methodology [16]. In addition, the Capacitor arts group has staged pieces that combine computer video generated off-line from motion capture data with a live performer [2]. Finally, a recent production of *The Tempest* used motion capture technology to embody a live performer in a computer-generated avatar, projected on a screen on stage [25]. While these art pieces use motion capture in exciting ways, they do not explore interactivity in a shared space, which only exists mediated through a computer. Such artificial spaces are the subject of the present work.

We describe two experiments that combine full-body motion capture with teleimmersion in shared virtual spaces. Our experiments differ from previous work in several ways. First of all, we investigate having multiple participants in our spaces, which involves merging previous research currents in avatar control and collaborative virtual environments into a single unified software system. Secondly, for reasons of economic practicality, our approach depends heavily on using PC clusters for rendering. This influences the design of the underlying software system, presenting challenges in creating data sharing mechanisms for use between the cluster nodes as well as between remote sites. Furthermore, the large number of computers involved in collaborative virtual environments based on PC clusters requires a developed software infrastructure focused on system management concerns like automatic launching of software components, configuration of those components, and managing the application over its time of use. Finally, a virtual sports arena must include a physical simulation that uses data from the players' bodies to effect objects in the environment. We include such a mechanism in our open source system.

The first experiment, Hummingbird, was an art performance showing collaboration between a live dancer in Los Angeles and a dancer in Illinois who was mapped into their shared world in real-time using motion capture. The second experiment, Fairy Sports, consisted of two performers, each fully motion captured and executing a shared task, in this case passing an oddly-shaped ball back and forth in a virtual arena. For this work, we used an optical tracking system from Motion Analysis and the MotionStar Wireless system form Ascension Technologies.

2. INTERACTIVE TECHNIQUES

Video conferencing is the most common form of telecollaboration. Many tasks, such as document sharing or joint steering of a running scientific simulation or visualization, can be accomplished with video conferencing instead of inside a collaborative virtual environment. For instance, suppose two physically dispersed researchers wish to steer a complex simulation. A GUI window with application steering controls, another window displaying current application state, and a video conferencing link suffice to enable meaningful collaboration. Collaborative virtual environments become more necessary the more physical the interaction between the participants needs to be.

We explored situations in which simple video transmission between sites participating in the shared world would not suffice. Motion capture data has a much lower data rate than video, easily less than 1 Mbps for a single performer. By using it to drive an avatar in a remote virtual world, one can achieve very high quality images with little network usage. Also, since this data is really just raw geometric information, the programmer can easily repurpose it.

For instance, the Hummingbird piece focused on transformation and evolution, something that would have been difficult or impossible to achieve with video. To realize Hummingbird, the performer needed to appear as different creatures, a larvae, a fairy, and a machine, at different times. Because our system focuses on obtaining and transmitting raw geometric information about the participants, these transformations were easily achieved by clothing the performer with different avatars at different points in the piece.

Another limitation of video is that it decreases the ability of participants to share the same virtual space. In Fairy Sports, for example, the performers pushed a squishy torus animated with real-time physics back and forth between one another. This kind of task could not be done with video at all since it directly involves the performers interacting in real-time within an active virtual space. Video does not create the spatial information needed for a simulation.

Figure 2. The immersive VR interface in Fairy Sports

In Fairy Sports, we experimented with different methods of immersing the participant in the shared world. We used a projection-based virtual environment with magnetic tracking and stereo graphics for one of the participants. For the other, who was on our optical motion capture stage, we tried non-stereo methods. We found that one non-stereo view could not give the performer enough information to play the game effectively, but two different views of the shared world enabled participation in the task.

Figure 3. Optical mocap interface for Fairy Sports

Consider the player on the optical motion capture stage. The cameras of the optical motion capture system need significant unobstructed line of sight to the performer to be effective. This condition, when combined with our small 20 foot by 30 foot stage, made an immersive virtual environment utilizing surround screens impractical. Consequently, only a limited, though still significant, proportion of the performer's field of view could be covered by screens. Gaps existed to allow the motion capture cameras to see the performer. As a result, projection from the head position would induce blind spots. Furthermore, without stereo video, the participant has no way to judge distance from himself to the ball in the Fairy Sports game, hindering his ability to hit it and lob it back to his partner. However, multiple views, such as a side view and a behind view, can, in fact, give the performer enough information to control his avatar and play the game.

3. PC CLUSTER SPECIFICS

Both Hummingbird and Fairy Sports use standard PC's for their infrastructure. The PC platform has many desirable characteristics, such as low cost, low weight, and easy serviceability, when compared to proprietary Unix graphics workstations. For Hummingbird, the existence of compact but powerful PC's was important as we had to ship several across country to the performance site. In this experiment, the computers in the distributed system formed an ad hoc cluster connected through the Internet2. Its displays were located at geographically dispersed locations, one in Los Angeles and one in Illinois, and consequently did not have to be precisely synchronized.

On the other hand, in Fairy Sports, we used a 6-sided immersive environment, the Beckman Cube [10], as one of the interfaces. In this case, each wall had its graphics produced by a separate PC, with all six images required to be perfectly synchronized. Some of this synchronization followed from the data sharing mechanisms built into the Syzygy library underlying the experiment, as explained below. However, some hardware synchronization was also needed. Since the Cube uses active stereo, the video signals of the participating PC's need to be genlocked together so that their vertical retraces occur at the same time. Consequently, we use graphics cards capable of accepting an external genlock signal. In this way, active stereo across the cluster display is possible.

4. CONTENT CREATION PIPELINE

The avatars used for this work were constructed from discrete pieces, one piece per bone in the avatar skeleton. This geometry was constructed in a modeling package and exported to OBJ format, with each piece an OBJ group. We represented the avatars' motion with the common htr format, which expresses the skeleton as a hierarchy of bones whose motion is given relative to a base position. An htr file containing only the base position information is created using a Motion Analysis plug-in for Maya. This information then maps each OBJ group in the avatar OBJ to a standard position, with the pivot point at the origin and pointing along the positive y-axis.

The transformed individual pieces are then stored internally in a scene graph, which mirrors the hierarchy of the skeleton. This database has additional structure that allows three aspects of the avatar animation to be independent of one another: the avatar's geometry, the stream of bone motion data, and the precise positioning of markers on the performer. Using this feature, avatars can easily be exchanged between performers or a single performer can take on different avatars dynamically. It even allows one to recover from different marker placement between

sessions. This flexibility comes from associating three additional transform nodes to each bone.

Each piece of bone geometry has 4 transforms associated with it. There are a post-transform node, which attaches to the pre-transform node of the bones' parent, and a transform node, which stores the rotation of the bone relative to its parent, as normally determined by the htr file, and is attached to the post-transform node. There are also a pre-transform node attached to the transform node and a local transform node, which is attached to the pre-transform node and to which the geometry is directly attached (Fig. 4).

The post-transform can be used to change the relative positioning of a limb. The pre-transform can be used to change the scale of the hierarchy going down or the base angle at which the limb protrudes, and the local transform can be used to change the relative orientation or scale of the limb without effecting the hierarchy below it. We used a custom GUI to interactively clothe the performers with their avatar (Fig. 5). The operator manipulates the various bones as above on the GUI's left side while watching data stream in on the right. Note how a poor fit of avatar to performer can be corrected using this tool.

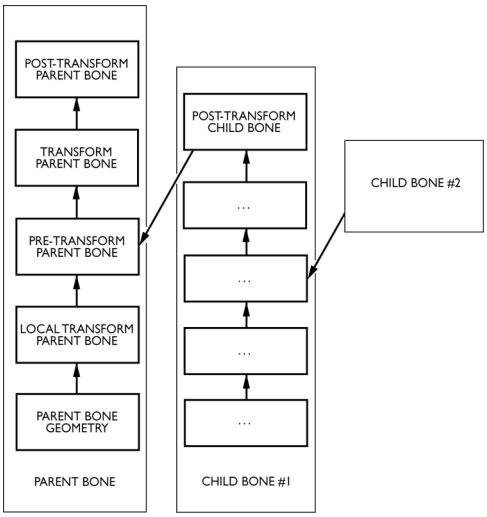

Figure 4. Partial scene graph for the skeleton

As an interesting side effect, the same limb motion data can be used to drive a distorted avatar. Using these methods, the limbs could suddenly appear very large or very small, be placed differently on the body, or even be angled differently than normally (Fig. 5).

5. SOFTWARE ARCHITECTURE

Shared virtual experiences necessarily involve networking software, which presents certain challenges in design and implementation. Interestingly, much of the same infrastructure applies to both cluster-based graphics and shared virtual worlds. In either case, the information necessary to construct the scene must be shared among the computers drawing the scene. A protocol needs to be constructed for this, with a means of encoding messages, and connections between the various components need to be managed and configured. Significant code reuse can occur between these problem domains.

Of course, these software types are not completely identical. Collaborative graphics may need to accommodate situations of much lower bandwidth and much higher latency than cluster-based graphics. And, clearly, cluster-based graphics needs much tighter synchronization, preferably frame-locked, than collaborative graphics.

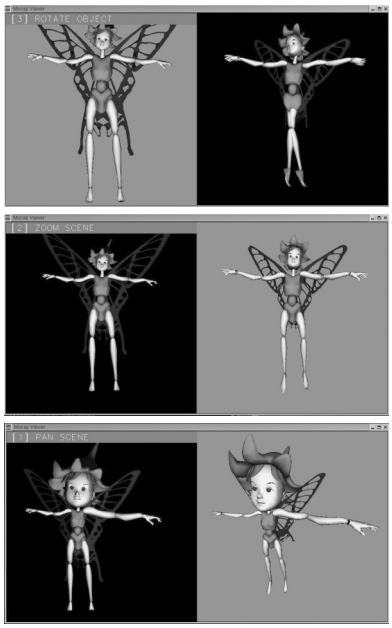

Figure 5. Clothing the performer: Uncalibrated, calibrated, and distorted.

Work on using graphics clusters for visualization and for shared virtual worlds has been intense in recent years. On the graphics cluster side, WireGL [13], Chromium [14], and DGL [19], have been used to display OpenGL applications on tiled walls consisting of dozens of projectors (the goal being very high resolution). On the shared virtual worlds side, Avocado developed the idea of a shared scene graph for sharing virtual worlds between users at multiple sites [31]. Similarly, Repo 3D explored using a scene graph model for distributed graphics [20]. An open source scene graph project, OpenSG, with clustering support, also exists [32].

Furthermore, the various incarnations of the CAVERNsoft project [18][24] have examined the requirements for creating an efficient networking structure for shared virtual worlds, along the lines of a distributed shared memory database with an additional UDP data transfer path. Many other projects for shared virtual worlds exist, notably NPSNET [21][4] and MASSIVE [11].

Syzygy, the software underlying our experiments, differs from these projects in several ways. First of all, it is built to facilitate both cluster-based graphics, where synchronization between the displays is a priority, and teleimmersion, where latencies dictate that displays cannot be tightly synchronized. Second, it includes functionality specifically for facilitating the integration of real-time full-body interaction data. Third, it integrates distributed graphics code, systems management middleware, and input device networking and management into a single coherent code base, reducing overall software complexity.

Syzygy is built in a layered architecture, as described in [28]. At the base layer, is a messaging system that allows the programmer to define custom protocols for communications between the software components. Also at the base layer is a set of network client and server objects that provide connection management functionality along with aid in processing messages. Above this layer, are built the various service-specific objects, like input event processors, input event servers, input event clients, network-aware graphics databases, and systems management components. Syzygy constructs its simulations from these pieces.

Other aspects of Syzygy are similar to well-known software projects. For instance, the VRPN project lets programs connect to their input devices over the network [30]. Syzygy also features a networked input event protocol that lets applications treat peripherals connected to remote computers as local input devices. Furthermore, communication with the various components of the distributed system and the configuration of those components is the sort of middleware task that grid operating systems like Globus [9] and Legion [12] are meant to accomplish. Like these systems, Syzygy has an infrastructure that facilitates launching and configuring distributed applications.

6. SYZYGY: GRAPHICS DATABASE

The visuals for these experiments build on the Syzygy graphics database. This is a hierarchical scene graph with a C++ API and has been described in-depth in [27]. Briefly, the application contains a master copy of the database. A general-purpose rendering program, szgrender, can connect over the network to this copy, and when connection occurs, the database state is transferred atomically. During a given frame, the application's graphics database logs changes that occur, and, when it is time for the next frame, transfers the entire buffer of changes to the slave databases. When the software is operating on a graphics cluster, the consumption of the buffer of database changes and the subsequent display of the new image is performed synchronously across all cluster PC's. Synchronization occurs over the network.

However, this tight synchronization is undesirable for telecollaboration over long distance networks. Specifically, when a slave database is ready to display the next frame, it notifies the master database and waits for the master to send a message telling it to swap buffers. Consequently, for tight database synchronization, network ping time is an upper bound on frame rate. On LANs, with ping times of under a millisecond, this effect is unimportant. However, even a high-performance WAN can see ping times of 40-60 ms on a transcontinental haul. Consequently, to enable collaboration over long distances, the graphics database objects must be able to operate using weaker synchronization schemes. In such cases, the application simply sends a stream of database update buffers to all connected databases.

The graphics database has two other features that prove vital for the work described in this paper. By default, all connected databases are drawn from the same camera position as the master database, with view direction and frustum possibly determined by screen location, as is normal in projection-based virtual reality. However, telecollaboration requires slave databases for different participants to be drawn from different viewpoints, and different views can aid the user in reconstructing depth information and other scene characteristics. Fairy Sports uses this feature, as described below.

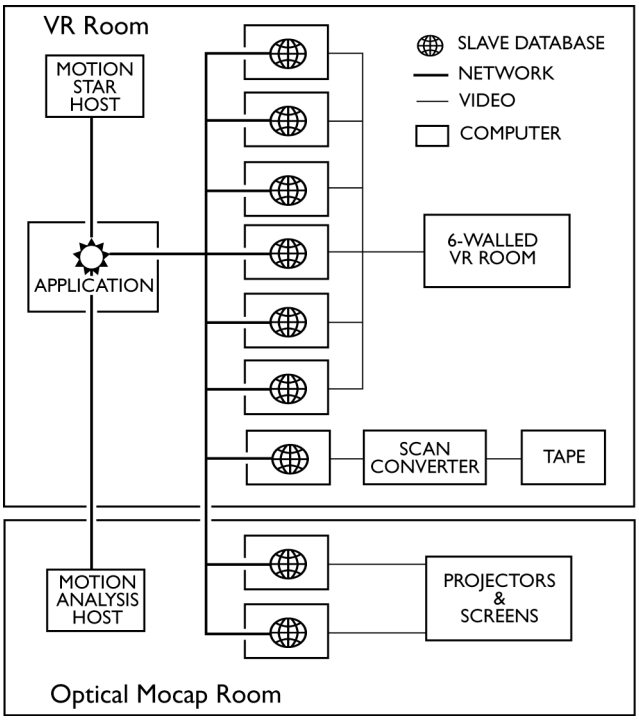

Figure 6. Systems diagram for Fairy Sports

Finally, it is critical for the various components to be able to join and leave the distributed system dynamically. This facilitates experimentation with the distributed system, as viewing nodes can be added to and removed from the cluster while the system is running. The ability to do this while maintaining tight frame-by-frame synchronization is a unique feature of Syzygy [27]. In addition to making experimentation with systems configuration easier, cluster application fault tolerance is increased, a characteristic that becomes more important as the total amount of hardware involved in running the application increases. The systems diagram for Fairy Sports (Fig. 6) shows how complex such clustered applications can become.

7. SYZYGY: DISTRIBUTED SYSTEMS

Experiments in interactive techniques require an infrastructure for handling input devices as well as graphics. In out case, input event

servers put event streams on the network, communicating with the devices themselves via driver objects. Applications use generic input event client objects to connect to the servers and read the event streams. A network-based input event handling framework is essential for the experiments in this paper since too many input devices are involved to connect them to a single computer.

This feature is crucial to the success of projects like Fairy Sports that involve multiple large-scale motion capture systems. For instance, the motion capture device may only provide data output to a particular OS platform but the application may need to run on another OS. In this case, a network protocol for input events allows an application client object to, in a platform independent fashion, receive events from a server object running on an OS platform capable of interfacing to the device. In our case, the Motion Analysis motion capture system only has available an interface to Windows systems but we want to run the Fairy Sports application on Linux.

Another important software feature offered by Syzygy is integrated messaging services. The szgserver program manages the Syzygy distributed system [28]. All components of this system connect to the szgserver, receiving configuration information from it and using it to transfer messages from one component to another. From the command line of any computer in the cluster, one can send a message to any of the running Syzygy programs. In our applications, the operator uses this method to dynamically change camera angles in a running view of an experiment, as is useful for video recording, or in changing application state. For instance, in the Fairy Sports piece, the operator needs to drop the ball from time to time, an action triggered by a Syzygy message.

Furthermore, in a distributed system, components need to be configured. The szgserver maintains a central database of configuration information for the various computers in the cluster. It can also be used to monitor running components, see how they are configured, kill them, and re-launch them. This activity can take place from any network accessible node. Consequently, during the run of a complex application like Fairy Sports, involving over a dozen computers dispersed throughout a large building, a single operator can monitor the health of the system, making changes as needed.

8. PHYSICS

Sports and similar activities depend on physical interactions between objects to operate. Consequently, the Fairy Sports experiment requires a real-time simulation engine to allow the avatars to interact with the ball. We use a simple physics engine, VMAT, an abbreviation for Virtual Materials, for this purpose. The engine uses point masses connected by springs to simulate flexible objects. Polygonal skins supported by these spring networks provide collision detection and response, as well as a graphical representation of the object.

The ball in Fairy Sports is, in fact, a flexible torus. This shape was chosen to allow interesting modalities of interaction, such as an avatar putting its arm through the central hole. Because the torus is flexible instead of rigid, its motion appears rubbery, with the underlying spring network providing the simulated flexibility.

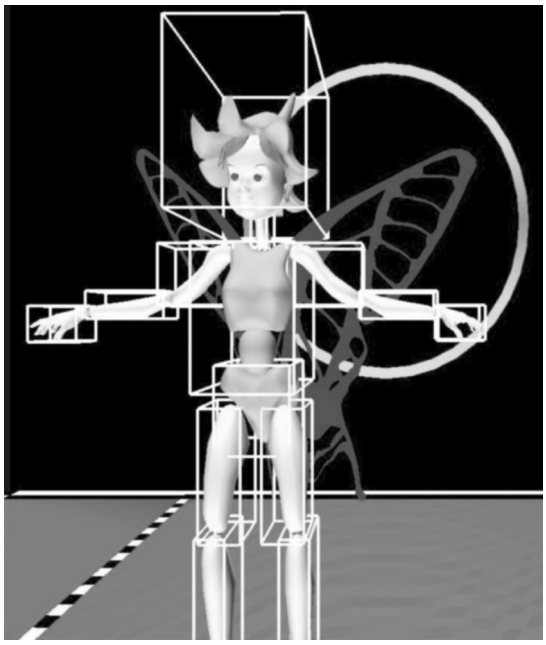

Figure 7. Simplified shadow geometry for the avatar

We now examine how the torus was constructed from springs. First of all, its skin consists of a grid of point masses mapped to its surface. Adjacent point masses are connected via springs. Next, to inhibit buckling of the skin, each point mass has springs connecting it to nonadjacent masses, these being located at distances of both 2 and 3 elements in the skin's grid. To reinforce the object and prevent it from collapsing, a ring of point masses occupies the torus' interior. These interior masses are connected via springs to nearby masses on the skin.

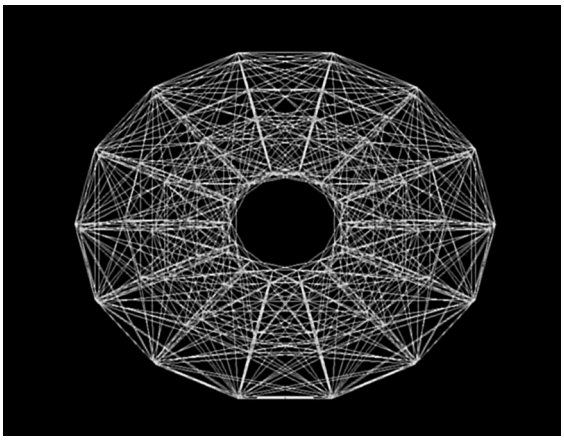

Figure 8. The ball in Fairy Sports: Spring network supporting the torus' skin

The objects in the VMAT engine use polygonal skins to detect collisions with each other. The grid of point masses on the torus' surface is covered, in the obvious way, by triangles. These triangles are both drawn to give a graphical representation of the torus (the Fairy Sports ball) and used by the physics engine to detect collisions with other objects in the simulation. With the

avatars, on the other hand, a simple identification of triangles in the avatars' geometrical representations with collision detecting triangles in the avatars' physics representation is impossible. The avatars are too geometrically complex, containing nearly 10,000 triangles.

Consequently, we do not use the avatars' geometry in the simulation but instead, for each avatar, construct a shadow geometry consisting of axis-aligned bounding boxes attached to each of the avatar's segments. These bounding boxes are driven by the avatar's motion, and the triangles comprising them are used for collision detection with the ball. By making this substitution, we are able to run our simulation in real-time. Other researchers have embedded complex geometry in a simplified physical framework for the purposes of real-time simulation, notably [3].

We now briefly walk through the simulation's inner loop. Each time step of its evolution is divided into two parts. First, the spring network underlying the ball is advanced using an ODE solver. Next, collisions between its point masses and triangles belonging to the avatars' shadow geometry are assessed. To speed up this process, the simulated space is evenly divided into a cubic grid and only point-triangle pairs from nearby cells are tested for collisions. Each triangle has a surface normal associated with it, whose direction defines the inside and outside of the object. If a point mass has passed through the triangle from its outside to its inside, a collision has occurred. In this case, the point mass is moved back along the normal so that it lies on the triangle's surface. Its velocity vector is also reflected along the normal.

We now attempt to give an idea of the simulation's performance. Each avatar has a collision skin built from 240 triangles, as outlined above. The ball is comprised of 126 point masses, 1022 springs, and 224 collision triangles. On a 1 GHz Pentium 3 processor, each animation frame of Fairy Sports is computed in about 24 milliseconds.

Figure 9. Photo from Hummingbird performance

9. HUMMINGBIRD: DETAILS

Our first experiment was an art piece, Hummingbird, performed in October 2002 between a dancer in Illinois and a dancer in Los Angeles. This performance, with its theme of metamorphosis, exploited motion captured data's advantages over video by transforming the performer's rendered avatar, in real-time, from one form to another. It also highlighted another important aspect of motion capture data: it is an excellent way to force high-quality content through a thin pipe. Since the avatar geometry is only transmitted once, and thereafter only the limb transformations need to be transmitted, the bandwidth requirements are significantly under 1 Mbps for a high-resolution animation, obviously far superior to video.

The performance involved a motion-captured dancer in Illinois. The motion capture data was transmitted to a local computer that maintained the master copy of the graphics database and executed the performance logic, which mainly consisted of transitioning the avatar from form to form based on a preset timeline. The master database was displayed on a projection screen in the motion capture room, so that the performer could get feedback regarding her actions in the virtual world.

A computer behind stage in Los Angles ran a slave graphics database that mirrored the database in Illinois, but was not tightly synchronized frame-by-frame to it. This computer projected video onto a transparent screen behind which the live performer in Los Angles danced. To give the performer in Illinois feedback regarding the events on stage, a video camera was set up in LA and streamed video back to Illinois, where it was displayed as a transparent overlay on the animated avatar image.

10. FAIRY SPORTS: DETAILS

Our second experiment with these systems was Fairy Sports. In this piece, two performers try to push a flexible torus, animated by real-time physics, back and forth between one another. The physics is such that any part of the avatar can interact with the object. This simulated world made full use of the advantages in interactivity afforded by using motion capture. The bodies of the participants are projected fully into a shared virtual space, and they are able to manipulate rapidly changing aspects of that space in a cooperative fashion. Video conferencing cannot provide a similar shared experience. An interesting application of this technology would be in integrating multiple synthetic performers into a scene for motion pictures or video games.

For this experiment, we used two motion capture stages. One was a fully immersive virtual reality chamber equipped with a MotionStar wireless system by Ascension Technologies and 10 magnetic sensors. On this stage, the performer was immersed in the simulation using a projection-based virtual environment, as pioneered by EVL [7]. This environment uses active stereo to give a sense of depth, has 6 walls, one of which slides shut so that the user can be fully surrounded (the screens are in a cube configuration), and has its display images drawn from the user's head perspective (Fig. 2).

The other stage used the same Motion Analysis optical tracking system as Hummingbird and Magic Mirror. While we did not have the capability to display stereo video on the optical motion capture stage, we nevertheless attempted to increase the information offered to the performer there by building a new

screen configuration. Two large screens were built, sandwiching the performer in a corridor-like space. The motion capture cameras were rearranged so that they pointed directly into the space from either side of the corridor. Each screen displayed the same image. By mirroring the screens, we allowed the performer to turn around and continue interacting, something that would have been impossible with only a single display.

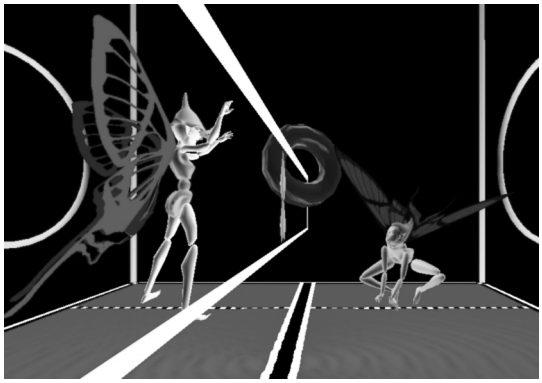

Figure 10. Still from Fairy Sports

We found that a single view was insufficient to give the performer on the optical motion capture stage enough information about the scene. Consequently, we used two computers, each displaying a different view. By using these views together, the performer could reconstruct the scene sufficiently to interact with the other avatar.

In the projection-based virtual environment, much more so than on the optical capture stage, the performer manipulated her avatar as if she was embodied in the space. The mechanics of virtual environments accomplish that for us and, indeed, seem to make the performer embodied in virtual reality more effective at playing the game than the performer on the optical stage.

We now briefly discuss the software architecture for this experiment. The application maintains the master graphics database and runs the simulation of the flexible torus batted back and forth between the performers. The PC on which the application runs does not render graphics. Input client objects embedded in the application connect across the network to input servers that in turn connect to the various motion capture devices, putting their data on the network.

There are, over all, 9 slave databases connected through the building's LAN to the application's master database, each running on a different computer (Fig. 6). Six of these are used to power the projection-based virtual environment, one per wall. Another two are used to provide the views for the performer on the optical motion capture stage. A final computer is used to output video to video tape to provide a record of the performance.

From a control console, the operator can monitor the state of the distributed system. Messages can be sent to the application causing a new to drop from the virtual sky. Camera angles on any of the displaying slave databases can also be changed. This allows, for instance, the operator to dynamically change camera angles in the video tape based upon events in the virtual world.

11. CONCLUSION

By combining work in virtual reality and teleimmersion with motion capture equipment, we are able to present participants with an unprecedented level of interaction with a shared virtual world. Full embodiment in a virtual world enables the completion of tasks requiring whole body presence, like sports or other games. The whole experience is strikingly realistic, both for the participants and for observers watching the virtual world.

12. OBTAINING THE SOFTWARE

The VMAT physics library, the Syzygy clustering software, and the infrastructure for our motion capture and telecollaboration experiments can be freely downloaded from www.isl.uiuc.edu. These libraries and applications are licensed under the GNU LGPL.

13. ACKNOWLEDGEMENTS

The Hummingbird performance was supported by the Beckman Institute of the University of Illinois and by the Kyoto Computer Gakuin. The 6-sided immersive virtual environment mentioned here was supported by NSF Major Research Instrumentation Grant EIA-0079800.

14. REFERENCES

[1] Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., and Cruz-Neira, C. 2001. VR Juggler: a virtual platform for virtual reality application development. In *Proc. IEEE Virtual Reality* 2001, 89-96.

[2] Capacitor. 2002. http://www.capacitor.org.

[3] Capell, S., Green, S., Curless, B., Duchamp, T., Popovic, Z. 2002. Interactive Skeleton-Driven Dynamic Deformations. *ACM Trans. Graphics 21*, 3, 586-593.

[4] Capps, M., McGregor, D., Brutzman, D., and Zyda, M. 2000. NPSNET-V: A New Beginning for Dynamically Extensible Virtual Environments, *IEEE Computer Graphics and Applications*, *20*, 5, 12-15.

[5] Chen, Y., Chen, H., Clark, D., Liu, Z., Wallace, G. and Li., K. 2001. *Software environments for cluster-based display systems*. http://www.cs.princeton.edu/omnimedia/papers/ccgrid.pdf.

[6] Chua, P., Crivella, R., Daly, B., Hu, N., Schaaf, R., Ventura, D., Camill, T., Hodgins, J., and Pausch, R. 2003. Training for Physical Tasks in Virtual Environments, In *Proc. IEEE Virtual Reality 2003*. 87-94.

[7] Cruz-Neira, C., Sandin, D., and DeFanti, T. 1993. Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In *Proc. ACM SIGGRAPH 1993*, ACM Press / ACM SIGGRAPH, 135-142.

[8] Daniilidis, K., Mulligan, J., McKendall, R., Kamberova, G., Schmid, D., and Bajcsy, R. 2000. Real-time 3D Tele-immersion, In *The Confluence of Vision and Graphics*, A. Leonardis et al. (Ed.), Kluwer.

[9] Foster, I. and Kesselman, C. 1997. Globus: a metacomputing infrastructure toolkit, *Intl. J. Supercomputer Applications 11*, 2, 115-128.

[10] Francis, G., Goudeseune, C., Kaczmarski, H., Schaeffer, B., and Sullivan, M. 2003. ALICE on the Eightfold Way: Exploring Curved Spaces in an Enclosed Virtual Reality Theatre, Visualization and Mathematics III, H.-C. Hege and K. Polthier, eds, Springer.

[11] Greenhalgh, C., Purbrick, J., and Snowdon, D. 2000. Inside MASSIVE-3: Flexible Support for Data Consistency and World Structuring, In *Proc. CVE 2000*.

[12] Grimshaw, A. and Wulf, W. 1997. The Legion vision of a worldwide virtual computer, *Comm. ACM 40*, 1, 39-45.

[13] Humphreys, G., Eldridge, M., Buck, I., Stoll, G., Everett, M., and Hanrahan, P. 2001. WireGL: a scalable graphics system for clusters. In *Proc. ACM SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, 129-140.

[14] Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirchner, P., and Klosowski, J. 2002. Chromium: a stream-processing framework for interactive rendering on clusters. *ACM Trans. Graphics 21*, 3, 693-702.

[15] Johnson, A., Roussos, M., Leigh, J., Barnes, C., and Vasilakis. C. 1998. The NICE Project: Learning Together in a Virtual World, In *Proc. IEEE VRAIS*, 176-183.

[16] Jones, B., Kaiser, P., and Eshkar, S. 1999. Ghostcatching. http://www.cooper.edu/art/ghostcatching.

[17] Lehner, V. D. and DeFanti, T. A. 1997. Distributed Virtual Reality: Supporting Remote Collaboration in Vehicle Design. *IEEE Computer Graphics and Applications*, *17*, 2, 13-17.

[18] Leigh, J., Johnson, A., and DeFanti, T. 1997. CAVERN: A Distributed Architecture for Supporting Scalable Persistence and Interoperability in Collaborative Virtual Environments, *Journal of Virtual Reality Research, Development, and Applications*, *2*, 2, 217-237.

[19] Li, K., Chen, H., Chen, Y., Clark, D.W., Cook, P., Damianakis, S., Essl, G., Finkelstein, A., Funkhouser, T., Klein, A., Liu, Z., Praun, E., Samanta, R., Shedd, B., Singh, J.P., Tzanetakis, G., and Zheng, J. 2000. Early experiences and challenges in building and using a scalable display wall system, *IEEE Computer Graphics and Applications 20*, 4, 671-680.

[20] MacIntyre, B. and Feiner, S. 1998. A distributed 3D graphics library. In *Proc. ACM SIGGRAPH 1998*, ACM Press / ACM SIGGRAPH, 361-370.

[21] Macedonia, M., Zyda, M., Pratt, D., Barham, P., and Zeswitz, S. 1994. NPSNET : A Network Software Acrhitecture for Large-Scale Virtual Environments, *Presence*, *3*, 4, 265-287.

[22] Molet, T., Boulic, R., and Thalmann, D. 1996. A Real-Time Anatomical Converter for Human Motion Capture, In *Proc. 7th Eurographics Workshop on Animation and Simulation*, Wein.

[23] Olson, E. 2002. *Cluster Juggler – PC Cluster Virtual Reality*. M.Sc. thesis, Iowa State University.

[24] Park, K., Cho, Y., Krishnaprasad, N., Scharver, C., Lewis, M., Leigh, J., and Johnson, A. 2000. CAVERNsoft G2 : A Toolkit for High Performance Tele-Immersive Collaboration, In *Proc. ACM Symposium on Virtual Reality Software and Technology 2000*, 8-15.

[25] Saltz, D. 2001. The Collaborative Subject: Telerobotic Performance and Identity, *Performance Research*, *6*, 4, 70-83.

[26] van der Schaaf, T., Spoelder, H., Renambot, L., Germans, D., and Bal, H. 2002. Retained mode parallel rendering for scalable tiled displays. In *Proc. Seventh Immersive Projection Technology Symposium*. Orlando.

[27] Schaeffr, B. 2002. Networking and Management Frameworks for Cluster-based Graphics, In *Virtual Environment on a PC Workshop,* Protvino, Russia.

[28] Schaeffer, B. and Goudeseune, C. 2003. Syzygy: Native PC Cluster VR, In *Proc. IEEE Virtual Reality 2003*. 15-22.

[29] Shin, H., Lee, J., Shin, S., and Gleicher, M. 2001. Computer Puppetry: An Importance-Based Approach, *ACM Transactions on Graphics*, *20*, 2, 67-94.

[30] Taylor, R., Hudson, T., Seeger, A., Weber, H., Juliano, J., and Helser, A. 2001. VRPN: a device-independent, network-transparent VR peripheral system, In *Proc. ACM Symposium on Virtual Reality Software and Technology 2001*, 55-61.

[31] Tramberend, H. 1999. Avocado: a distributed virtual reality framework. In *Proc. IEEE Virtual Reality 1999*, 14-21.

[32] Voβ, G., Behr, J., Reiners, D., and Roth, M. 2002. A multi-thread safe foundation for scenegraphs and its extension to clusters. In *Proc. Fourth Eurographics Workshop on Parallel Graphics and Visualization*. Blaubeuren, Germany.

[33] Wheless, G., Lascara, C., Valle-Levinson, A., Brutzman, D., Sherman, W., Hibbard, W., and Paul, B. 1996. Virtual Chesapeake Bay: Interacting with a coupled physical/biological model, *IEEE Computer Graphics and Applications, 16*, 4, 42-43.