Data Distribution for Mobile Augmented Reality in Simulation and Training

Dennis Brown[1] dbrown@ait.nrl.navy.mil
Yohan Baillot[2] baillot@ait.nrl.navy.mil
Simon J. Julier[2] julier@ait.nrl.navy.mil
David Armoza[1] armoza@ait.nrl.navy.mil
Joshua J. Eliason[3] jelias1@evl.uic.edu
Mark A. Livingston[1] markl@ait.nrl.navy.mil
Lawrence J. Rosenblum[1] rosenblum@ait.nrl.navy.mil
Pat Garrity[4] Pat_Garrity@peostri.army.mil

[1]Advanced Information Technology, Naval Research Laboratory, Washington, DC 20375
[2]ITT Advanced Engineering and Sciences, Alexandria, VA 22303
[3]Electronic Visualization Laboratory, University of Illinois at Chicago, Chiacgo, IL
[4]U.S. Army Research, Development, and Engineering Command, Orlando, FL

ABSTRACT

The Battlefield Augmented Reality System (BARS) is a mobile augmented reality system that displays head-up battlefield intelligence information to a dismounted warrior. BARS consists of a wearable computer, a wireless network, and a tracked see-through Head Mounted Display (HMD). The computer generates graphics that, from the user's perspective, appear to exist in the surrounding environment. For example, a building could be augmented to show its name, a plan of its interior, icons to represent reported hazard locations, and the names of adjacent streets.

The full power of mobile augmented reality systems is realized when these systems are connected to one another, to immersive virtual environments, and to remote information servers. These connections are made through wireless devices that cannot guarantee connectivity and may have highly constrained bandwidth. Based on these constraints, we present a robust event-based data distribution mechanism for mobile augmented reality and virtual environments. It is based on replicated databases, pluggable networking protocols, and communication channels.

For use in simulation and training exercises, we have been working with U.S. Army RDECOM to create an interface between this data distribution mechanism and a Semi-Automated Forces (SAF) system. With this interface, the BARS user appears as a dismounted warrior in the SAF system—the BARS user's position and orientation are fed to the SAF system, and the state from the SAF system is sent back to the BARS user's display. With this SAF interface, BARS becomes a training system that works in a real location (as compared to a virtual reality simulation) to make simulated forces appear to existin and interact with the real world.

ABOUT THE AUTHORS

DENNIS BROWN is a Computer Scientist at the Naval Research Laboratory. He received his B.A. in Computer Science from Rice University in 1996 and his M.S. in Computer Science from the University of North Carolina at Chapel Hill in 1998. He works on the Battlefield Augmented Reality System (BARS) and multi-modal virtual reality projects. His research interests include ubiquitous computing and data distribution. He is a member of IEEE.

YOHAN BAILLOT is a computer and electrical engineer of ITT Industries at the Naval Research Laboratory. He received an M.S. in electrical engineering in 1996 from ISIM, France, and an M.S. in computer science in 1999 from the University of Central Florida. His research interests are in computer graphics, 3D displays, tracking, vision, mobile augmented reality and wearable computers. Baillot is a member of the IEEE Computer Society.

SIMON J. JULIER is a Research Scientist for ITT Industries at the Naval Research Laboratory. He received a D.Phil. from the Robotics Research Group, Oxford University, UK. He is a technical lead on the Battlefield Augmented Reality System (BARS) project. His research interests include mobile augmented reality and large-scale distributed data fusion.

DAVID ARMOZA is a Computer Scientist at the Naval Research Laboratory, where he works in the area of Distributed Simulation. His current research involves use of the US Navy's Joint Semi-Automated Forces (JSAF) simulation system and distributing stand-alone tools with DMSO's High Level Architecture (HLA). He received a BS in Computer Science from the University of Maryland, and a MS in Computer Science from The Johns Hopkins University.

JOSHUA J. ELIASON is a graduate student at the Electronic Visualization Laboratory, University of Illinois at Chicago. He received a BFA in 1999 from the University of Wisconsin-Madison. He is an intern on the Battlefield Augmented Reality System (BARS) project at the Naval Research Laboratory. His research interests include human factors in augmented and virtual reality, computer graphics, animation, and film production.

MARK A. LIVINGSTON is a Research Scientist in the Virtual Reality Laboratory at the Naval Research Laboratory, where he works on the Battlefield Augmented Reality System (BARS). He received his Ph.D. from the University of North Carolina at Chapel Hill, where he helped develop a clinical augmented reality system for both ultrasound-guided and laparoscopic surgical procedures, focusing on tracking subsystems. His current research focuses on vision-based tracking algorithms and on user perception in augmented reality systems. Livingston is a member of IEEE, ACM, and SIGGRAPH, and is a member of the VR2004 conference committee.

LAWRENCE J. ROSENBLUM is Director of VR Systems and Research at the Naval Research Laboratory (NRL) and Program Officer for Visualization and Computer Graphics at the Office of Naval Research (ONR). Rosenblum received his Ph.D. in mathematics from The Ohio State University. He is on the Editorial Board of IEEE CG&A and J. Virtual Reality and the Advisory Board of the IEEE Transactions on Visualization and Computer Graphics. He was the elected Chairman of the IEEE Technical Committee on Computer Graphics from 1994-1996 and is currently a TC Director. He is a founder and steering committee member of the IEEE Visualization and IEEE VR Conference Series. Elected a Senior Member of the IEEE in 1994, Rosenblum is also a member of the IEEE Computer Society, ACM, SIGGRAPH, and the AGU.

PAT GARRITY is a principal investigator at U.S. Army Research, Development, and Engineering Command (RDECOM), Simulation Technology Center. He currently works in the Dismounted Embedded Training Technologies (DEST) enterprise area conducting R&D in the area of dismounted soldier embedded training & simulation. Prior to his involvement with tech base division at RDECOM, he worked as the Project Director for the Advanced Concepts Research Tools (ACRT) program in PM STI at STRICOM. His current interests include Human-In-The-Loop (HITL) networked simulators, virtual and augmented reality, and embedded training applications. He earned his B.S. in Computer Engineering from the University of South Florida in 1985 and his M.S. in Simulation Systems from the University of Central Florida in 1994.

Data Distribution for Mobile Augmented Reality in Simulation and Training

Dennis Brown[1] dbrown@ait.nrl.navy.mil
Yohan Baillot[2] baillot@ait.nrl.navy.mil
Simon J. Julier[2] julier@ait.nrl.navy.mil
David Armoza[1] armoza@ait.nrl.navy.mil
Joshua J. Eliason[3] jelias1@evl.uic.edu
Mark A. Livingston[1] markl@ait.nrl.navy.mil
Lawrence J. Rosenblum[1] rosenblum@ait.nrl.navy.mil
Pat Garrity[4] Pat_Garrity@peostri.army.mil

[1]Advanced Information Technology, Naval Research Laboratory, Washington, DC 20375
[2]ITT Advanced Engineering and Sciences, Alexandria, VA 22303
[3]Electronic Visualization Laboratory, University of Illinois at Chicago, Chiacgo, IL
[4]U.S. Army Research, Development, and Engineering Command, Orlando, FL

INTRODUCTION

Distributed virtual reality (VR) technology is used in many immersive training and simulation environments, and there is ongoing research and development in improving the fidelity of these simulators (Stytz, 1996). However, current technology still does not perfectly replicate the sensory experience provided by the real world. Augmented reality, in which virtual reality techniques are added to the user's real world experience, is a promising alternative. Here we will explain our approach to using augmented reality for embedded training, specifically, how data is distributed and shared.

In our research on the Battlefield Augmented Reality System (BARS) (Julier et. al. 2000, Livingston et. al. 2002), we have focused on the problem of developing information systems able tothat provide users with "situation awareness"—data about the environment and its contents. The centerpiece of BARS is a mobile augmented reality system that displays head-up battlefield intelligence information to a dismounted warrior. It consists of a wearable computer, a wireless network, and a tracked see-through Head Mounted Display (HMD). The computer generates graphics that, from the user's perspective, appear to exist in the surrounding environment. For example, a building could be augmented to show its name, a plan of its interior, icons to represent reported hazard locations, and the names of adjacent streets.

In an effort sponsored by the Embedded Training for Dismounted Soldiers (ETDS) Science and Technology Objective (STO) (Dumanior et. al. 2002) at U.S. Army RDECOM and the Naval Research Laboratory, we are developing an embedded training system for Military Operations in Urban Terrain (MOUT) scenarios using BARS. The BARS system for embedded training allows the user to train in the real world with real and simulated forces. It combines the fidelity of a real MOUT training environment with the convenience of simulated forces.

Similar but distinct efforts at using AR for embedded training are currently underway within the same STO, including MARCETE (Kirkley et. al. 2002), which places an emphasis on working with SCORM datasets, and VICTER (Barham et. al. 2002), which fits within the limitations of the current Land Warrior system (Natick Soldier Center 2001).

In its capacity as a situation awareness tool, the BARS supports a consistent information space. Therefore, data objects tend to be less complicated and updates occur less frequently than in virtual environments. Furthermore, it requires a unified architecture that allows transport of general state information that can be used for many purposes. The obvious task is distributing the virtual object database, but there are more general uses such as ``remote control" of applications (using the event system to control the user interface of a remote application). This system must also handle the poor network connectivity that can sometimes be encountered in military operations. Given these parameters, we have developed a robust, flexible, and generalized event-based networking infrastructure for data distribution. The mechanism builds upon three techniques: distributed databases, pluggable transport protocols, and a high-level management technique known as channels. Additionally, interfaces called "bridge" applications allow BARS to share data with external information systems.

In a previous paper, we described the BARS distribution system and various applications of it

(Brown et. al. 2003). In this paper, we briefly summarize again the BARS networking system, and then explain how it is used for embedded training.

PROBLEM STATEMENT

The Battlefield Augmented Reality System is a collaborative mobile augmented reality system designed to improve the situation awareness and the coordination between a team of mobile users. By improving situation awareness, we mean that each user obtains a better understanding of the environment through enhanced sensory perception. The types of data include the names of buildings, routes, objectives, and the locations of other users. While short-range radio communications can accomplish much of this, the passive and natural display paradigm of augmented reality makes the internalization of the information by an individual faster and easier.

The hardware of one of our prototype systems is shown in Figure 1. It consists of a mobile computer, either an embedded PC with a high-end graphics card or a laptop with high-end graphics built in. A see-through SVGA display is worn by the user—we have built systems with the Sony Glasstron™ and with the Microvision Nomad™ retinal display. The system supports spatialized audio through software, using commodity sound cards and headphones. The user operates the system using a cordless mouse and a wrist keyboard. A Global Positioning System (GPS) receiver provides position tracking while an inertial device handles orientation tracking. A camera can be used for tracking and sending video reports to a base station. Wireless 802.11b networking is used for data distribution and GPS corrections. We predict that when a future system based on our BARS research is used in real operations, communication will happen over the US military's hardened communication systems of that time. However, it is likely that any deployed system will still be vulnerable to connectivity and bandwidth complications in urban areas, and our design reflects that consideration.

The BARS mobile user sees computer graphics superimposed on or next to the real objects they are intended to augment, in addition to status information such as compass direction and messages from other users. Figure 2 a view using the system—the parking lot is augmented to highlight the neighboring building and a fellow BARS user.

Figure 1. The BARS Wearable System

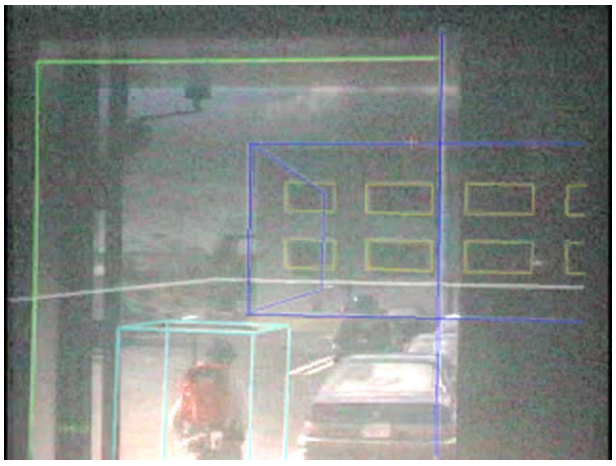

Figure 2. Sample view through BARS display, showing building information and the location of another BARS user.

This application introduces a number of characteristics that impact the distribution of information and events between users:

- The objective is to provide relevant information, not a consistent virtual world. The BARS environment is populated by a set of objects that are self-contained entities and other types of discrete data. Each object can be relatively simple, representing a building type and location, an avatar to symbolize another user, the location of a hazard, and so on. It is not necessary to transmit complicated geometric objects or behaviors—only semantic information. The latency in the update of an object or an entity is a secondary consideration.
- Data distribution between users can be heterogeneous. Different users might perform

different tasks and thus have different information requirements.

- The distribution system should facilitate collaboration between users. In addition to environmental data, the distribution system must support the propagation of meta-data such as task assignments, objectives, and personalized messages.
- Users should have the ability to create reports and update entities in the database. For example, a user might observe that an environmental feature (such as a vehicle) is not where the database indicates it should be. The user should have the ability to move the object to its correct location.
- Network connectivity is unreliable. As a user traverses a terrain, reception strength and bandwidth may vary.

BARS EVENT DISTRIBUTION SYSTEM

A BARS session consists of one or more BARS applications, or program instances. Each BARS application minimally uses a core set of libraries to maintain a local database of objects and communicate over a network. Applications may also include modules to read data from sensors, draw the augmented display, and perform other tasks, depending on the purpose of the application. The local database is a copy of a master database that is shared between all BARS applications on the network. The BARS distribution system is responsible for selectively replicating the master database in all applications.

The BARS distribution system is based entirely on the concept of *events*. Events are used to instantiate objects (in effect, transmit a view of a database between systems), update existing objects, and to provide other non-database status information such as a new objective for an individual user.

The event distribution system is based on three components: replicated object repositories, event transporters, and communication channels. We will describe these components along with bridge applications, which communicate with outside virtual and augmented reality systems.

Replicated Object Repositories

An object repository inside a BARS application holds the data for that application. This data consists of the mostly static models of the physical surroundings (buildings, streets, points of interest, etc), dynamic avatars that represent users and other entities, and objects created to communicate ideas, such as reports of enemy locations, routes for users to follow, and digital ink. The database is replicated in whole or in part for every BARS application.

When a BARS application starts, it loads an initial set of objects from a number of sources, including data files, other applications already running on the network, and files specified on the command line. This initial set of objects typically consists of street labels, landmarks, building information, and other terrain-like information, as well as an initial set of objectives, routes, and phase markers for the current task. Since a BARS user is initially given a database to start, and everything else in the wearable BARS system is self-contained, the user will have a working AR system even if all network connectivity is lost during an operation.

Although network limitations may hamper wireless communications for the mobile users, there are few limitations on the base users. Base users are those that use stationary systems and are not mobile, such as users at fixed command centers. Their applications run on stationary VR systems such as a desktop computers, 3D workbenches, and immersive VR rooms. Using the same distribution system, they can have high levels of detail and interaction by taking advantage of the increased bandwidth for replicating more objects and seeing change events at a higher frequency.

Event Transportation

The heart of the event transportation system is the *Object and Event Manager*. The Object and Event Manager is responsible for dispatching events within an application and distributing those events to remote applications.

When the Object and Event Manager receives an event, it places it on an asynchronous event queue. The event dispatching thread delivers the event to all the listeners that are subscribed to receive the specified event type. The event dispatching mechanism maintains two sets of data—the set of valid event/listener pairs, and the set of listeners registered for each event type. Because the event system has is based on the Java Abstract Window Toolkit event model (Sun Microsystems 2003) we leverage the Reflection Application Programming Interface to achieve these steps. The type of each event is specified by its class. For each event type, a listener is defined. When a listener is registered with the object

and event manager, its interface is queried and it is registered to receive all event types for which its interface is compatible. Therefore, it is possible to dynamically extend the set of event and listener types at runtime.

The following is an example of the life of an event within a BARS application that tracks a user's position. The user's position is updated by calling a method in the user object to set its attitude based on data gathered from tracking devices. This method in turn creates an event encapsulating the change to the attitude and sends it to the dispatcher. The event arrives in the dispatcher's incoming queue and is processed. The dispatcher sends the event to all listeners, including the initial object itself, in addition to the graphics system (to update the viewpoint) and the filter (to update what objects are rendered), and any other registered listeners. Note that the object's attitude isn't set until it receives the event back from the dispatcher (the alternative is to set the position at the same time the event is set)—this way, the order of events is preserved. Figure 3 shows the flow of events within an application.

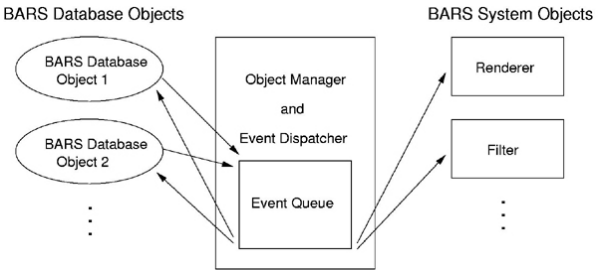

Figure 3. Event distribution within an application. Arrows show event movement.

We have described how events propagate within a single application instance. We extended this event mechanism to allow many separate BARS applications to trade events by creating *Event Transporters* that allow Object and Event Managers in different BARS application instances to send and receive events over Internet Protocol (IP). If an event is tagged as distributed, an Event Transporter serializes the event and broadcasts it to other applications. The Event Transporters in remote applications synthesize the event object and dispatch it on those applications' event queues. Our system uses several types of transporters based on IP multicast, the Lightweight Reliable Multicast Protocol (LRMP) from INRIA (Liao 1998), and a combination protocol we call the Selectively Unreliable Multicast Protocol (SUMP) that combines IP multicast and LRMP. Typically, application instances use SUMP on the local network. To communicate outside of the local network (where multicast is typically filtered out) a TCP/IP transporter and bridge are used (described later in the Bridges subsection). Because of the connectionless nature of IP multicast, the distribution is robust in that the network connection can be unreliable and the user application will still function, although without network updates at some times.

As events are created, they are tagged ``reliable" or ``unreliable" designating how they should be sent. Object creation and deletion are always sent reliably. Object changes are sent reliably or unreliably based first on whether the modification is relative or absolute. Relative changes have an ordering and each one is important, so those are sent reliably. Absolute changes, such as the constant updates of a user position, are mostly sent unreliably since if one is missed, the next would overwrite it anyway—periodically these changes are sent reliably. This policy makes the assumption that the implementation of IP networking in a real operation may drop IP packets often, making reliable multicast expensive, and so we don't send events reliably unless we think it is truly necessary. Figure 4 shows the flow of events between applications.

Channels

If simply implemented as described above, the event distribution mechanism will send all events to every application, which would replicate the entire database inside every application instance. Creating copies of every object for every user and updating those replicas would swamp the network with information many users would not care about anyway. To avoid this situation, we have devised an approach of partially replicating the database to minimize the amount of unwanted data distributed to each application instance.

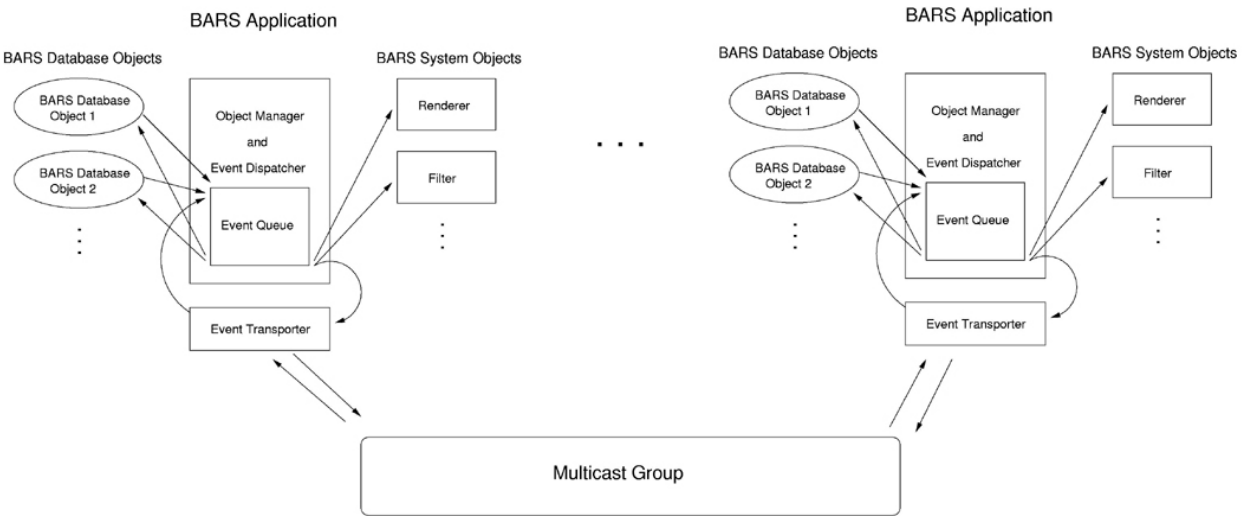

Figure 4. Event distribution between applications. Arrows show event movement.

In creating this replication mechanism, we first looked at the uses of BARS that drive our policies. One condition to consider is that a mobile user can only see so much and deal with information in a relatively small radius, so we considered a spatial area-of-interest mechanism. It is *not* necessarily the case that a mobile user only cares about objects that can be seen from his or her current position in the real world; for example, our mobile application includes an overhead map mode in which the user can zoom out to an arbitrary height to observe objectspossibly hugeradius around the current position. However, it seems that there would be few situations in which a mobile user would request for objects farther away, at the horizon for example, so for most situations, a simple area-of-interest mechanism is reasonable.

However, another condition is the type of information being distributed. Even if some objects are near to a mobile user, they may have importance and only cause distraction. Alternatively, the objects may indeed be too far away to be seen, but very important, such as with possible sniper locations. For these cases, a simple area-of-interest mechanism isn't sufficient. In an earlier paper (Julier et. al. 2000), we described a filtering mechanism for mobile augmented reality. This filtering mechanism operates on the local object database within an application instance. It does not show users objects in which they have no interest in order to reduce display clutter. In practice, it simply hides objects from the user—it does not actually control whether or not the application instance holds replicas of these objects or receives events related to these objects.

Keeping these situations in mind, we have developed *channels*. The term is overloaded in the literature, but in our system, a channel is a set of related objects. It is implemented as an instance of an event transporter and a multicast group designated for that transporter. An application can join an arbitrary number of channels and create new channels, until all available multicast groups are allocated. Figure 5 shows a single application using two channels.

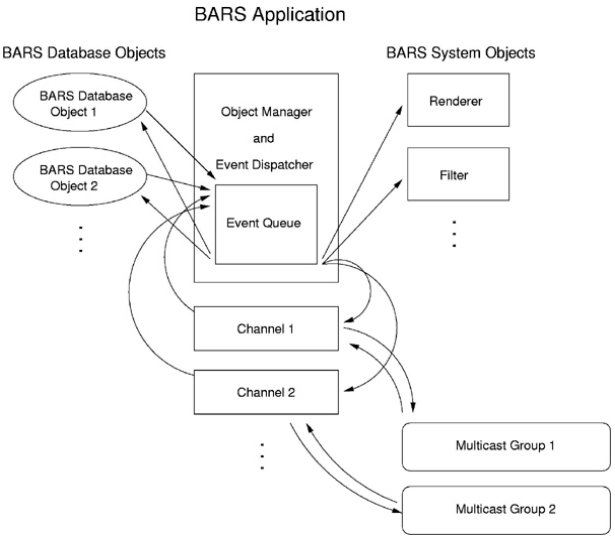

Figure 5. Event distribution using two channels. Arrows show event movement.

One example of a channel is a set of objects in a certain spatial area. As users move from location to location, they can join and leave channels based on spatial areas. Another example is the set of hazardous objects; while

in the previous example, the application instance would replicate only objects nearby, the hazardous objects channel could cover a larger area, but only include those hazards. Also, BARS incorporates several interaction modules that produce subsequent objects. One such interactor module is principally responsible for real-time, interactive geometric construction. It allows users to collaboratively place points and build new objects from thosepoints; in this case, the intermediate points would not be visible to other users because they are placed in a channel only joined by the constructing users. Other users would only see the final objects. Another interactor allows a user to draw digital ink for interpretation by a multimodal interaction system—this ink is turned into new objects or user interface commands. In this case, the application instance of the user drawing the ink would be placed in a separate channel, joined by the application to interface with the multimodal system. The ink is placed in this channel so that other users would not see these sketches out of context.

Bridges

As we alluded to in the previous section with the multimodal interaction example, some of our BARS applications communicate with other systems. We call these applications *bridges*. Bridges join both the BARS distribution system and an external system. They translate object creation and change events between BARS and external systems. By maintaining tables linking BARS objects and these external objects, we can represent those objects in BARS and vice-versa. Two systems with which we can communicate are the Columbia Mobile Augmented Reality System (Höllerer et. al. 1999) and the Oregon Graduate Institute's Quickset multimodal interface (Pittman et. al. 1996).

USING BARS FOR EMBEDDED TRAINING

We have shown how BARS wearable systems and their operators can communicate with each other, with central data repositories, and with external information systems. Although BARS was originally designed for providing situation awareness during operations, its components can be reused for training in real environments by augmenting the real world with simulated forces and other factors.

BARS for embedded MOUT training works as follows:

- Simulated forces are rendered on the display, so as the user looks around the real MOUT facility, forces appear to exist in the real world (within current graphics limitations) even though they do not truly exist. At the same time, fellow real trainees remain visible.
- Spatialized audio is piped through the headphones to replicate the aural cues that the simulated forces would make if they were real. These sounds may include footsteps, helicopters, and so on. Since the sound is spatialized, the user can determine the location of the simulated force by listening, like in the real world.
- Interaction with the simulated forces is very limited at this time. Real and virtual forces can shoot at each other.
- Simulated forces are controlled through various means and are distributed to the trainees using the BARS distribution system.

There are several technical challenges to this task, even with all of the work already completed for BARS, that will be explained further.

Rendering Simulated Forces Realistically

The simulated forces need to appear on the user's display to give the illusion that they exist in the real world. There are several inherent problems: model fidelity, lighting to match the real environment, and occlusion by real objects.

Model fidelity is controlled by the modeler and is limited by the power of the machine running the application. Although models that can be rendered in real time still look computer generated, just like in VR-based simulations, the limited AR model representation capabilities are adequately realistic for embedded simulation and training. AR actually has an advantage over VR with respect to rendering; the AR graphics system does not need to draw an entire virtual world, only the augmented forces, so they could potentially be more detailed than those in VR-based simulations.

Lighting the rendered forces is a problem we have not approached yet. This task would require knowing the lighting conditions of the real environment in which the model would appear, and changing the renderer's light model to match. Another limitation is the display itself, as it is very sensitive to outside light, and even if the image is rendered with perfect lighting, it still might not appear correctly on the display.

Occlusion of simulated objects by real objects is a problem we have tackled, as we feel that this problem, more than lighting or model complexity, is the one that would ruin the immersion of training using AR.

Imagine using an AR training system and seeing a simulated force, which is supposed to be behind a building, rendered in front of the building. This property is actually a feature of AR used for operations—it gives the user a way to see through walls. However, today's dismounted warriors cannot see through walls, and so in the AR-based trainer, they should not see simulated forces that should be occluded by real objects.

We solve this problem by using a model of the training environment. In using our AR system for operations, we know where the user is looking and can draw an augmenting model of buildings and features superimposed on the real features. In AR for training, we simply render this same model in flat black. On the computer display, these black features will occlude the parts of the simulated forces the user should not see. However, since black is the "see through" color on the AR display, the user will still see the real world, along with the correct non-occluded parts of the simulated forces. Figure 6 shows a sequence of images demonstrating this technique. Figure 6A shows the real-world scene with no augmentation. In figure 6B, we show the same scene but with simulated forces simply drawn over the scene at their locations in the world—there is no occlusion. It is hard to tell if all of the forces are intended to be in front of the building, or if they are just drawn there due to limitations of the system. Figure 6C shows the simulated forces occluded by a gray model, however, the model also occludes some of the real world. Finally, figure 6D shows the scene rendered using a black model, which serves two purposes: it occludes the simulated forces properly and, since it is the "see through" color, allows the user to see the real world instead of the gray model.

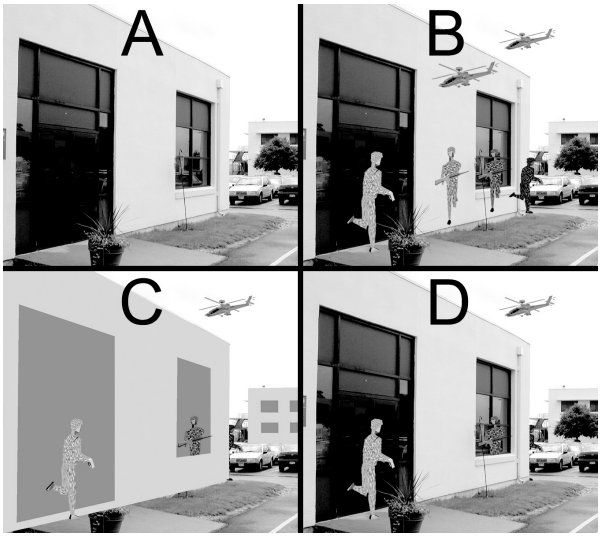

Figure 6. Stages in the development of AR for embedded training.

Inserting Aural Cues

Since we already have a 3D world model, and we know the locations of the user and the simulated forces, we can use existing 3D sound libraries to provide spatialized audio. We simply attach sound streams to simulated forces and update the audio library with the positions of those forces and with the user's listening attitude. Open-air headphones naturally "mix" the sounds of the real world with the computer-generated sounds.

Interacting With Simulated Forces

The simulated forces can be controlled in several ways including simple animation scripts. However, the animations are not reactive and tend to create a simple "shooting gallery" type of simulation. They can also be controlled by users of immersive VR simulations that participate on the same network as the AR user. Finally, they can be controlled through Semi-Automated Force (SAF) systems.

BARS communicates with outside information systems using bridge applications, as described in the previous section. By creating a bridge application between BARS and a SAF system, we can leverage the years of work already put into simulating forces for both non-immersive and immersive VR-based training, and interact with those forces in a real training environment.

Figure 7 shows a set of BARS applications for an embedded training scenario: two trainees using wearable systems, a trainee using an immersive VR system, an observer using a VR system, and a bridge synchronizing the entities in BARS and a connected SAF system. The bridge converts SAF entities into BARS entities and vice-versa. It keeps those entities updated on each side of the bridge as they change by converting BARS events into DIS or HLA packets and vice-versa. The bridge is not a simple filter for converting these events; it must maintain internal state information in order to convert the events and packets properly. In addition to sharing entity information, the

system allows BARS users to engage the simulated forces and allows the simulated forces to retaliate.

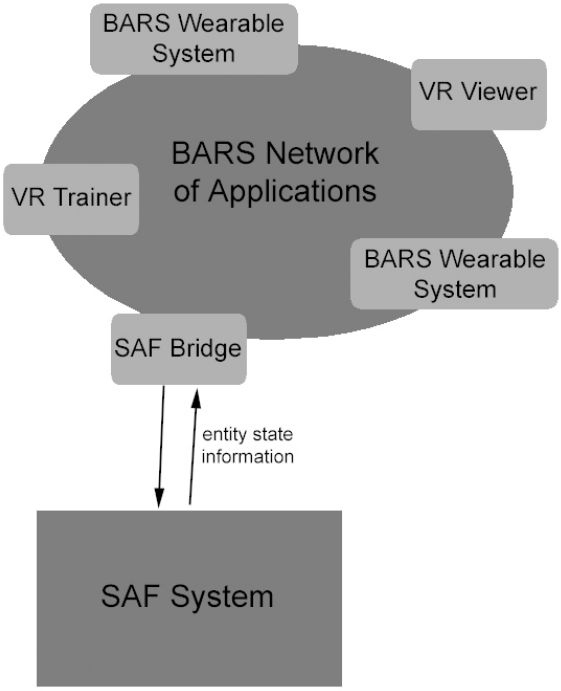

Figure 7. Sharing information between BARS and a SAF system using a bridge application.

FUTURE WORK

We plan to pursue refinement of the BARS-SAF interface to exploit advanced SAF functionality such as DISAF human articulation; rather than static human models that are positioned similar to toy soldiers.

We also anticipate exploring a mixture of AR techniques that we developed for situation awareness in real operations with the AR techniques for training. While this would no longer mimic the real world (since, for example, we can't really see through walls), we hope that in the future, real operations will use BARS or one of its descendents. By inserting the enhanced AR capabilities into the training system, we can test which capabilities are most useful and refine them in a controlled environment.

REFERENCES

Barham, P., B. Plamondon, P. Dumanoir, & P. Garrity (2002). "VICTER: An Embedded Virtual Simulation System for Land Warrior." *Proceedings of the 23rd Army Science Conference, Orlando, FL, USA.*

Brown, D., Y. Baillot, S.J. Julier, & M.A. Livingston (2003). "An Event-Based Data Distribution Mechanism for Collaborative Mobile Augmented Reality and Virtual Environments," *Proceedings of the 2003 IEEE Virtual Reality Conference, Los Angeles, CA, USA.*

Dumanoir, P., P. Garrity, V. Lowe, & B. Witmer (2002). "Embedded Training for Dismounted Soldiers (ETDS)," *Proceedings of the 2002 Interservice/Industry Training, Simulation, and Education Conference, Orlando, FL, USA.*

Höllerer, T., S. Feiner, T. Terauchi, G. Rashid, & D. Hallaway (1999). "Exploring MARS: Developing Indoor and Outdoor User Interfaces to a Mobile Augmented Reality System," in *Computers and Graphics* 23 (6), Elsevier Publishers, Dec 1999, pp. 779-785.

Julier, S., M. Lanzagorta, S. Sestito, L. Rosenblum, T. Höllerer, & S. Feiner (2000). "Information Filtering for Mobile Augmented Reality," *Proceedings of the 2000 IEEE International Symposium on Augmented Reality*, Germany.

Julier, S., Y. Baillot, D. Brown, & L. Rosenblum (2000). "BARS: Battlefield Augmented Reality System," *NATO Symposium on Information Processing Techniques for Military Systems, October 2000, Istanbul, Turkey.*

Kirkley, S., J. Kirkley, S.C. Borland, T. Waite, P. Dumanior, P. Garrity, & B. Witmer (2002). "Embedded Training with Mobile AR," *Proceedings of the 23rd Army Science Conference, Orlando, FL, USA.*

Liao, T (1998). *Light-weight Reliable Multicast Protocol*. Internet Draft retrieved June 9, 2003 from http://webcanal.inria.fr/lrmp/draft-liao-lrmp-00.txt

Livingston, M.A., L.J. Rosenblum, S.J. Julier, D. Brown, Y. Baillot, J.E. Swan II, J.L. Gabbard, & D. Hix (2002). "An Augmented Reality System for Military Operations in Urban Terrain," *Proceedings of the 2002 Interservice/Industry Training, Simulation, and Education Conference, Orlando, FL, USA.*

Natick Soldier Center (2001). *Operational Requirements Document for Land Warrior*. Retrieved June 6, 2003, from

http://www.natick.army.mil/soldier/WSIT/LW_ORD.PDF

Pittman, J., I. Smith, P. Cohen, S. Oviatt, & T. Yang (1996). "Quickset: A multimodal interface for military simulations," *Proceedings of the 6th Conference on Computer-Generated Forces and Behavioral Representation, Orlando, FL, USA.*

Stytz, M.R. (1996). "Distributed Virtual Environments," *IEEE Computer Graphics And Applications*, May 1996, pp. 19-31.

Sun Microsystems, Inc. (2003). *Java API Documentation*. Retrieved June 9, 2003 from http://java.sun.com/docs