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. Connected to a SAF system, BARS technology creates a training system that works in a real location (as compared to a virtual reality simulation) to make simulated forces appear to exist in 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 and his M.S. in Computer Science from the University of North Carolina at Chapel Hill. He works on the Battlefield Augmented Reality System (BARS) and multi-modal virtual reality projects. His research interests include ubiquitous computing, specifically, novel user interfaces 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 a 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 Computer Society, 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 cannot perfectly replicate the sensory experience provided by the real world. Augmented Reality (AR), in which virtual reality techniques are added to the user's real world experience, is a promising alternative. This paper explains one approach to using AR for embedded training and, specifically, how data is distributed and shared.

Research on the Battlefield Augmented Reality System (BARS) (Julier et al. 2000a, Livingston et al. 2002) has focused on the problem of developing information systems able to 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 Naval Research Laboratory and the Embedded Training for Dismounted Soldiers (ETDS) Science and Technology Objective (STO) (Dumanior et al. 2002) at U.S. Army RDECOM, the BARS team is developing an embedded training system for Military Operations in Urban Terrain (MOUT) scenarios. The BARS system for embedded training (BARS-ET) 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, BARS supports a consistent information space. Therefore, data objects tend to be less complicated (for example, they are not articulated) and updates occur less frequently than in all-virtual environments. Furthermore, BARS has to deal with state information from a variety of sources for a variety of purposes. These include information dissemination (distributing the virtual object database), system configuration, and even system control through a "remote control" handheld computer. Furthermore, network connectivity can be poor. Given these constraints, a robust, flexible, and generalized event-based networking infrastructure for data distribution has been developed (Brown et al. 2003). The mechanism builds upon three techniques: distributed databases, pluggable transport protocols, and a high-level management technique known as channels. The mechanism supports the use of "bridge" applications allow BARS to share data with external information systems.

In this paper, the BARS networking system is summarized and its use for embedded training is described.

PROBLEM STATEMENT

The Battlefield Augmented Reality System (BARS) is a collaborative mobile augmented reality system designed to improve the situation awareness and the coordination between a team of mobile users. Improving situation awareness means 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 a prototype wearable system is shown in Figure 1. It consists of a wearable computer, a display and a tracking system. The computer is responsible for generating 3D graphics and spatialized audio in real-time. It can be implemented using an embedded PC with high-end graphics and sound cards or a laptop with built-in high-end graphics. The generated graphics are shown on an optical see-through head mounted display. Prototypes have been developed using both the Sony Glasstron™ and the Microvision Nomad™ retinal displays. The tracking system determines the position and orientation of the user's head. Position is measured using a Global Positioning System (GPS) receiver. Orientation is determined using a solid-state inertial navigation system. Wireless 802.11b networking is used for data distribution and GPS corrections. A camera can be used for tracking and sending video reports to a base station. The user operates the system using a cordless mouse and a wrist keyboard. When a future system based on BARS research is used in real operations, communication will likely happen over the US military's hardened communication systems of that time. However, it is probable that any deployed system will still be vulnerable to connectivity and bandwidth complications in urban areas, and the design reflects that consideration.

Figure 1. The BARS Wearable System

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 shows a view using the system, in which another BARS user is augmented and is following a virtual path.

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

First, some terms will be defined as they are used in this discussion. A *session* consists of one or more *applications*, or program instances, which may exist in any number on one or more machines on a network. Each application 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 applications on the network. The distribution system is responsible for selectively replicating the master database in all applications.

The 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. These components will be described, as will bridge applications, which communicate with outside virtual and augmented reality systems.

Replicated Object Repositories

All of the data for a scenario is stored in an object repository. The 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. This repository is replicated in whole or in part for each application.

When an 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. The 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. An 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 is based on the Java Abstract Window Toolkit event model

(Sun Microsystems 2003), the Reflection Application Programming Interface is leveraged to achieve these steps. Each event type is implemented in its own class. For each event class type, a listener interface is defined. When an object 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 pose based on data gathered from tracking devices. In turn, this method creates an event that encapsulates the change in pose. The event is enqueued at the event dispatcher. The dispatcher sends the event to all listeners, including the initial object itself, as well as other system components (such as the graphics system, which updates the viewpoint). Note that the object's pose 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 sent)—this way, the order of events is preserved. Figure 3 shows the flow of events within an application.

The propagation of events within a single application instance has been described. This event mechanism was extended to allow many separate applications to trade events by creating *Event Transporters*. Event Transporters allow Object and Event Managers in different application instances to send and receive events over Internet Protocol (IP). Figure 4 shows the flow of events between applications. If an event is tagged as distributed, an Event Transporter serializes the event and broadcasts it to other applications. The

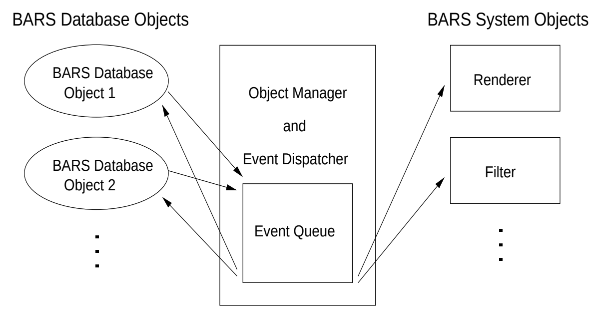

Figure 3. Event distribution within an application. Arrows show event movement.

Event Transporters in remote applications synthesize the event object and dispatch it on those applications' event queues. The system uses several types of transporters based on IP multicast, the Lightweight Reliable Multicast Protocol (LRMP) from INRIA (Liao 1998), and a combination protocol called 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

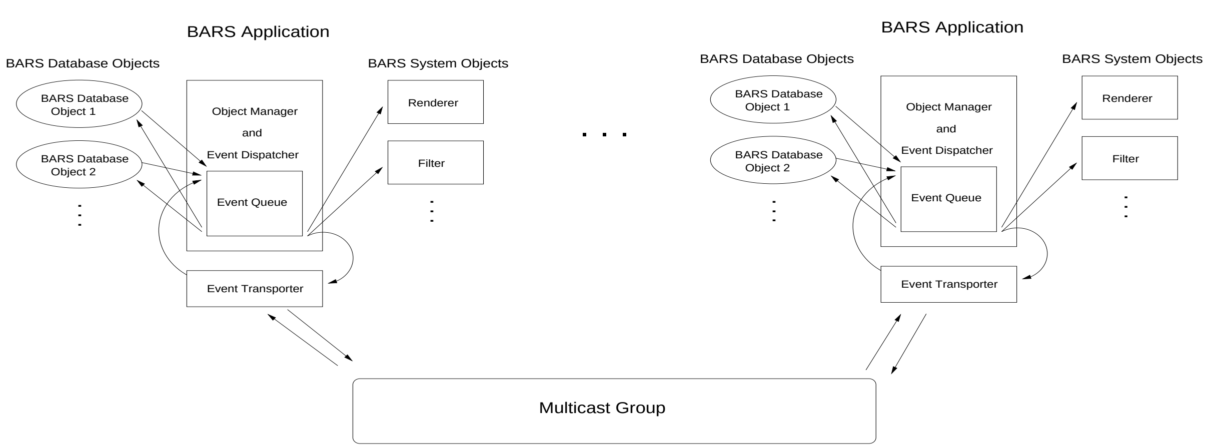

Figure 4. Event distribution between applications. Arrows show event movement.

first on whether the modification is relative to other each one is important, so those are sent reliably. Non-relative changes, such as the constant updates of a user's position, are mostly sent unreliably since if one were missed, the next would overwrite it anyway. Periodically, these non-relative 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 events are not sent reliably unless they are thought to be truly necessary.

Channels

The problem with the event distribution mechanism described above is that all events for all objects would be broadcast to every single application. Creating copies of every object for every user and updating those replicas would swamp the network with information that would be irrelevant for many users. To overcome this problem, the database is only partially replicated in each application instance.

In creating this replication mechanism, the uses of BARS drove the 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 a spatial area-of-interest mechanism was considered. 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, the BARS mobile application includes an overhead map mode in which the user can zoom out to an arbitrary height to observe objects within a huge radius 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.

Another condition is the type of information which is being distributed. Even if some objects are near to a mobile user, they may not be important and might 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. 2000b), a filtering mechanism for mobile augmented reality was described. 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 changes, or not. Relative changes have an ordering and holds replicas of these objects or receives events related to these objects.

Keeping these situations in mind, *channels* have been developed. The term is overloaded in the literature, but in this 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.

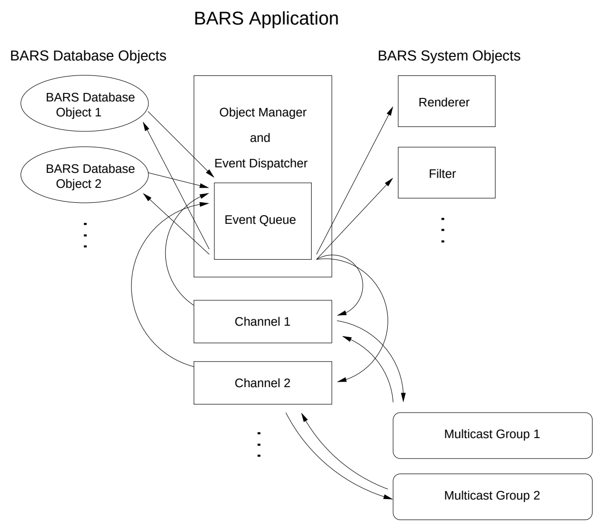

Figure 5. Event distribution using two channels. Arrows show event movement.

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. For example, one interaction module is principally responsible for real-time, interactive geometric construction (Baillot et al. 2000). It allows users to collaboratively place points and build new objects from those points; 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 interaction module lets a user 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 the multimodal example illustrates, some of the BARS applications communicate with other systems. These applications are called *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, those objects can be represented in BARS and vice-versa. Two systems with which BARS 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 discussed how the 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 (BARS-ET) 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 sent through the headphones to replicate the aural cues that the simulated forces would make if they were real. These sounds include footsteps, shouting, 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, which 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 our team has 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.

The problem of occlusion of simulated objects by real objects, more than lighting or model complexity, is the one that would most likely 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 BARS —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.

The occlusion problem is solved by using a model of the training environment. Techniques for creating environmental models for AR have been previously published (Julier et al. 2001). In the AR system for operations, it is known where the user is looking and the system can draw an augmenting model of buildings and features superimposed on the real features. In AR for training, this same model is rendered 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. This solution was introduced for indoor applications by State et al (1996) and applied to outdoor models by Piekarski and Thomas (2002) for use in outdoor AR gaming. Figure 6 shows a sequence of images demonstrating this technique. Figure 6A shows the real-world scene with no augmentation. In figure 6B, the same scene is shown 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.

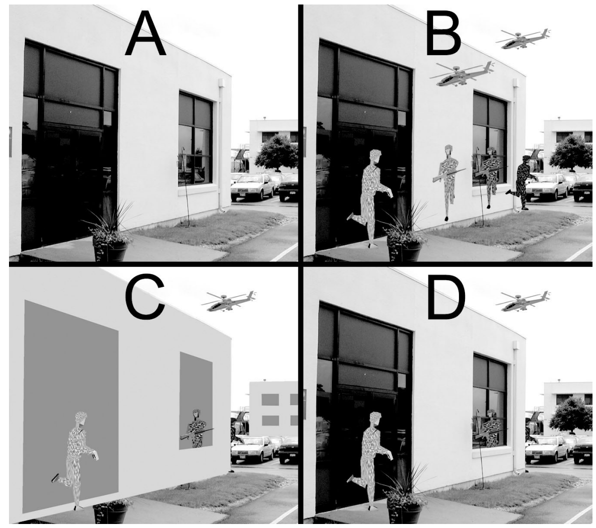

Figure 6. Stages in the development of AR models for embedded training.

Inserting Aural Cues

Since the system already has a 3D world model, and the locations of the user and the simulated forces are known, existing 3D sound libraries are used to provide spatialized audio. Sound streams are simply attached to simulated forces and the audio library is updated 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, the years of work already put into simulating forces for both non-immersive and immersive VR-based training can be leveraged, and the user 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.

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.

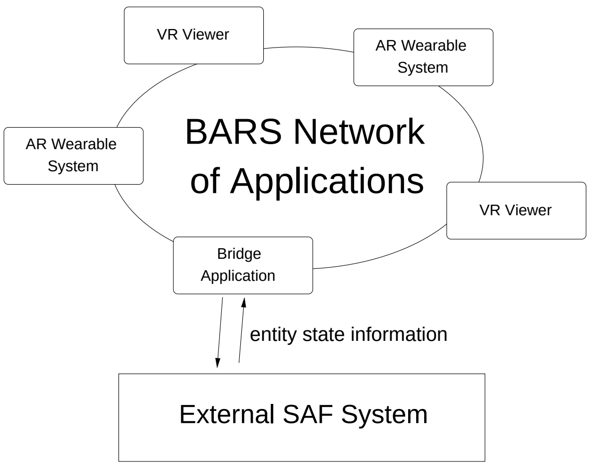

Figure 7. Sharing information between BARS and an external SAF system using a bridge application.

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., 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.*

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, M. Lanzagorta, L. Rosenblum, & D. Brown (2001). Urban Terrain Modeling For Augmented Reality Applications. In M. Abdelgurfi (Ed.), *3D Synthetic Environment Reconstruction* (pp. 118-138), Dordrecht, The Netherlands: Kluwer Academic Publishers.

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

Piekarski, W. & B. H. Thomas (2002). "ARQuake: The Outdoor Augmented Reality Gaming System," *ACM Communications*, Vol. 45, pp. 36-38.

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.*

State, A. M. A. Livingston, G. Hirota, W. F. Garrett, M. C. Whitton, E. D. Pisano MD, and H. Fuchs (1996). "Technologies for Augmented Reality Systems: Realizing Ultrasound-Guided Needle Biopsies," *SIGGRAPH 96 Conference Proceedings*. Aug 1996. pp. 439-446.

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