Building a Mobile Augmented Reality System for Embedded Training: Lessons Learned

Dennis G. Brown[1] dbrown@ait.nrl.navy.mil
Yohan Baillot[2] baillot@ait.nrl.navy.mil
Simon J. Julier[2] julier@ait.nrl.navy.mil
Paul Maassel[3] maassel@reallaer.com
David Armoza[1] armoza@ait.nrl.navy.mil
Mark A. Livingston[1] markl@ait.nrl.navy.mil
Lawrence J. Rosenblum[1] rosenblum@ait.nrl.navy.mil

[1]Advanced Information Technology, Naval Research Laboratory, Washington, DC 20375
[2]ITT Advanced Engineering and Sciences, Alexandria, VA 22303
[3]ReallaeR, LLC, Port Republic, MD 20676

ABSTRACT

Mobile augmented reality (AR) is a method for providing a "head up display" to individual dismounted users. A user wears a miniaturized computer system, tracking sensors, and a see-through graphics display. The system superimposes three-dimensional spatially registered graphics and sounds onto the user's perception of the real world. Because information can be presented in a head up and hands free way, it has the potential to revolutionize the way in which information is presented to individuals.

A mobile AR system can insert friendly, neutral, and enemy computer-generated forces (CGFs) into the real world for training and mission rehearsal applications. The CGFs are drawn realistically and properly occluded with respect to the real world. The behaviors of the CGFs are generated from two Semi-Automated Forces (SAF) systems: JointSAF and OneSAF. The AR user appears as an individual combatant entity in the SAF system. The AR user's position and orientation are fed to the SAF system, and the state of the SAF entities is reflected in the AR display. The SAF entities react to the AR user just as they do any other individual combatant entity, and the AR user interacts with the CGFs in real time.

In this paper, we document the development of a prototype mobile AR system for embedded training and its usage in MOUT-like situations. We discuss the tradeoffs of the components of the hardware (tracking technologies, display technologies, computing technologies) and the software (networking, SAF systems, CGF generation, model construction), and we describe the lessons that have been learned from implementing several scenarios.

ABOUT THE AUTHORS

Dennis G. 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.

Paul Maassel has provided systems engineering support for modeling, simulation, and virtual world construction for the past fifteen years. Mr. Maassel was a civil servant at the Naval Aviation Maintenance Office, Naval Air Test Center, and Naval Air Warfare Center Aircraft Division where he took delivery of first beta release of ModSAF for the U.S. government. He served as Systems Engineer on a number of major M&S programs including Synthetic Theater of War (STOW) and Joint Countermine Operational Simulation (JCOS). Mr. Maassel currently manages Reallaer, LLC, a small business working to develop practical augmented reality systems for training and operations.

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.

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.

Building a Mobile Augmented Reality System for Embedded Training: Lessons Learned

Dennis G. Brown[1] dbrown@ait.nrl.navy.mil
Yohan Baillot[2] baillot@ait.nrl.navy.mil
Simon J. Julier[2] julier@ait.nrl.navy.mil
Paul Maassel[3] maassel@reallaer.com
David Armoza[1] armoza@ait.nrl.navy.mil
Mark A. Livingston[1] markl@ait.nrl.navy.mil
Lawrence J. Rosenblum[1] rosenblum@ait.nrl.navy.mil

[1]Advanced Information Technology, Naval Research Laboratory, Washington, DC 20375
[2]ITT Advanced Engineering and Sciences, Alexandria, VA 22303
[3]ReallaeR, LLC, Port Republic, MD 20676

INTRODUCTION

Modern wars are more often fought in cities than in open battlefields, and warfighter training has been updated to reflect this change. Military Operations in Urban Terrain (MOUT) training is an important component of a warfighter's initial and continued development. Much of this training occurs in purpose-built MOUT facilities, using simulated ammunition and half the team acting as the opposing forces (OPFOR). As an alternative, virtual reality (VR) training systems for MOUT operations are improving. Both of those training modes have several drawbacks. The MOUT facility training provides the trainee with a real-world experience, but there are manpower issues (must schedule two teams, or split one team so that half plays OPFOR), the exercise is not completely repeatable, and there are issues with the simulated munitions such as setup, injuries, and cleanup. In contrast, the VR training provides a safe, controlled, and repeatable training scenario, but it deprives the trainee of many real-world cues that are not yet simulated, requires special equipment that is not easily moved for the most immersive simulations, and does not allow completely realistic navigation through the environment.

In an effort to create a training method that combines the control and repeatability of VR with the authenticity of the real world, we have researched and developed a prototype of an embedded training system that uses augmented reality (AR). Augmented reality technology allows computer-generated information to be projected (in a sense) into the real world. For training, animated three-dimensional computer-generated forces are inserted into the environment. The AR training system moves the repeatability and control of a VR system into a real-world training environment.

A system developed at the Naval Research Laboratory is the Battlefield Augmented Reality System-Embedded Trainer (BARS-ET). BARS-ET is based on the components developed in the BARS program (Julier et al. 2000). Assuming that future warfighters will have equipment capable of providing augmented reality—wearable computers with head-mounted displays, such as the proposed Future Force Warrior system (Natick Soldier Center 2004)—it is prudent to take advantage of those resources to provide embedded training. This technology allows warfighters to truly train as they fight.

Other groups have considered the use of AR for embedded training. MARCETE (Kirkley et al. 2002) places an emphasis on working with SCORM datasets to provide distance education. VICTER (Barham et al. 2002) was built to fit within the limitations of the current Land Warrior system (Natick Soldier Center 2001), replacing pieces of that system as necessary.

In this paper, we will describe the research and development process for building BARS-ET. Since there are several components to this system that are all interrelated, we are presenting this work in the form of lessons learned rather than a piece-wise description of the system.

LESSON 1: Build upon a solid Mobile AR platform

Intended for enhancing situation awareness in urban operations, BARS is a man-portable system that makes computer-generated graphics and sounds appear to exist in the real world. The user dons a backpack-based apparatus consisting of a wearable computer, a see-through head-mounted display, tracking devices, and a wireless network module. Location-specific

situation awareness information, such as the positions of friendly forces hidden by a wall, may be displayed so that it appears in its real-world position, no matter how the user moves around. It is also possible to augment the view of a building to show its name, a plan of its interior, icons to represent reported hazard locations, and/or the names of adjacent streets.

The centerpiece of the BARS project is the capability to display head-up battlefield intelligence information to a dismounted warrior, similar to the head-up display (HUD) systems designed for fighter pilot cockpits. The system consists of a wearable computer (PC-compatible), wireless network support (802.11b), and a tracked see-through Head Mounted Display (HMD) (Sony Glasstron, Microvision Nomad, or Trivisio). Three-dimensional (3D) data about the environment is collected (through surveying, sensors, or reports by other users) and made available to the system. By using a Global Positioning System (GPS) unit and an inertial orientation tracker (such as the Intersense InertiaCube) it is possible to know where the user is located and the direction in which he is looking. Figure 1 shows the BARS wearable system. Based on this data, the desired 3D data is rendered to appear as if it were in the real world.

Figure 1. The BARS Wearable System

Running a training session indoors requires special tracking considerations. As the tracking system on the backpack is GPS-based, it only works outdoors. For indoor demonstrations, a different tracking system is required. Magnetism-based indoor trackers have not proven to have the accuracy needed for AR in our experience, mainly due to their susceptibility to distortion. So, we use ultrasonic- or vision-based trackers, which require installation on site and careful surveying—similar actions would be necessary for running a session inside MOUT facility structures. BARS supports a wide variety of trackers, so no software changes were necessary.

The BARS system software is a multi-tiered custom system written in Java and C++ developed on Linux and Microsoft Windows. It supports many types of commercially available trackers and cameras. The graphics display use OpenGL and OpenSceneGraph. There is a distributed shared database that each user can access using a wireless network, so that BARS users can share data in real time. The software system is currently used to conduct research and development for AR interaction, including speech, gestures, information representation, and so on.

The driving problem for building BARS is to enhance situation awareness in a head-up and hands-free manner. This information is projected in 3D, spatially registered in the real world. Some examples of this data include street names, routes, trails, hazard locations, friendly positions, and so on. This information is purposely made to stand out from the environment, even going as far as sampling the background view and drawing the data in contrasting colors. Although the data is spatially registered, it should not "blend in" to the environment.

Compared to providing situation awareness data, rendering synthetic forces in BARS has additional needs because even though the basic AR functionality is the same, the paradigms required to solve these problems are very different. In the situation awareness mode, BARS adds information to the real-world view that the user would not normally see. It is necessary for this information to stand out and appear artificial. In the embedded training mode, BARS inserts cues into the real world view that, ideally, the user could not distinguish from reality. For example, a team of trainees in a Military Operations for Urban Terrain (MOUT) training facility could work together against a an enemy force—some forces are real, some are virtual, and the blending would be seamless.

By starting with BARS to build our AR training system, we already have a stable platform with tested tracking, networking, and head-mounted display components. Several steps were performed to create BARS-ET using the BARS components. Animated computer-generated forces (CGFs) appear on the display, properly registered and occluded in the real world. The CGF behaviors are controlled by a Semi-Automated Forces (SAF) system, which leverages existing work in simulated forces and provides a well-understood method of creating and controlling training scenarios. Additionally, a weapon tracker was added, so that the system knows where the user is aiming and firing.

LESSON 2: Use a see-through display that can occlude the real world

The type of head-mounted display used can make or break the illusion of computer-generated forces existing in the real world. If the graphics do not occlude the real world, they do not appear solid or realistic. Instead, the graphics, translucent on a non-occluding display, take on a ghostly and unrealistic appearance. Figure 2 shows two similar scenes, one through a non-occluding display, and one through an occluding display. Notice how the avatar is washed out in bright light in the non-occluding display.

Figure 2. Non-occluding and occluding displays

There are two fundamentally different technologies used for AR graphic displays: optical see-through and video see-through. The optical see-through displays use mirrors to optically combine the computer display, generated with LCD, LCOS, or lasers, with the view of the real world. The video see-through display uses a camera in front of the user's eye and a video overlay component to combine the computer graphics and the camera feed. The optical display has the advantage of resolution—it does not alter what the user would normally see (except minor dimming), while the video display is limited to video resolutions, which are well below what the human eye can perceive, and gives a pixilated image. It is also hard to match the display's brightness and contrast to match the real world, as the camera reacts differently than the human eye to changes in lighting. On the other hand, no optical display yet occludes the real world—the user can always see through the computer graphics, while the video display allows the computer graphics to block out the real world and appear more realistic. The video-based display also has some minor lag, in which the display shows an image a few milliseconds later than an unadorned user would perceive it. This effect is noticeable but not particularly disturbing.

For mobile AR applications for tactical situations (not training), the optical display, even with its faults, is better because the user's view of the real world is not degraded and the ghostly appearance of tactical information does not detract from the utility of that information. For this embedded training application, however, the benefits of the video display's true occlusion outweigh the drawback of decreased resolution, and so it is our choice for now—when an optical see-through display becomes available with true occlusion capabilities, it will be the best choice.

LESSON 3: Create an accurate occlusion model

In BARS-ET, the user's viewpoint in the real world is measured using the tracking system. At the same time, the simulated forces exist in the computer's 3D virtual world. The user's viewpoint in the real world is translated to a camera position in the virtual world and that world is rendered on the user's display. Assuming the system has been properly calibrated, the virtual forces are overlaid at the correct locations in the real world. However, this does not yet fully solve the problem of integrating the virtual forces into the real world. Imagine using BARS-ET and seeing a simulated force, which is supposed to be behind a building, rendered in front of the building. This effect would ruin the illusion that the simulated force exists in the real environment. Yet, if the system worked simply as described above, that behavior would result. The system needs some understanding of the static structures within the training environment. An occlusion model solves this problem.

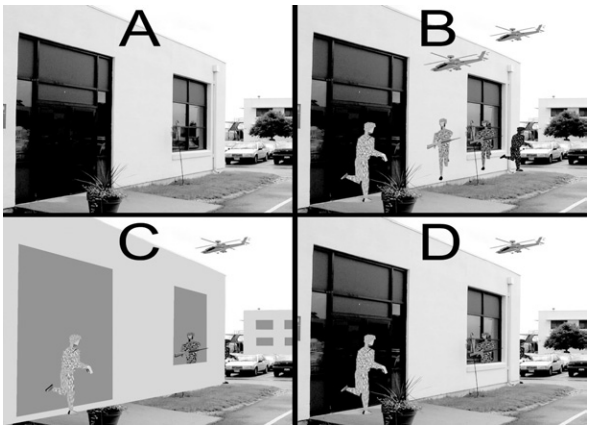

Figure 3. Stages in the development of AR models for embedded training.

Figure 3 shows a sequence of images demonstrating the need for and construction of an occlusion model. Figure 3A shows the real-world scene with no augmentation. In figure 3B, 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 3C shows the simulated forces occluded by a gray model, however,

the model also occludes some of the real world. Finally, figure 3D shows the scene rendered using a black model, which serves two purposes. First, the flat black polygons occlude the simulated forces properly, just like the gray model did. Second, since black is the "see through" color for the display, the user sees the real world behind the occlusion model. 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.

To build the occlusion model, every static structure in the training environment must be carefully surveyed and replicated using a 3D modeling application or the model-building facilities within BARS. Techniques for creating environmental models for AR have been previously published (Julier et al. 2001). It is very important for the model to be accurate, because if it is wrong, strange effects happen, such as avatars appearing behind solid walls. Similar effects occur if the tracking system is not calibrated properly or if the tracker is inaccurate, but those errors are largely unavoidable with current technology—however, there is no reason to use a poorly constructed model to further compound those errors.

LESSON 4: Use realistic CGF behaviors via Semi-Automated Forces

Many hours have been put into the various Semi-Automated Forces (SAF) systems available for simulating training scenarios. Modular Semi-Automated Forces (ModSAF) (Ceranowicz 1994) was an early version used extensively in the training community, and has spawned two current successors: OTBSAF and JointSAF. By creating an interface between these two SAF systems and BARS-ET, this work can be leveraged for interactive training in real-world environments.

Before describing the SAF interfaces, some alternatives for controlling the simulated forces will be recognized. One method supported in BARS is a simple scripting language in which forces can follow predetermined paths and do not react to the environment. This method is unacceptable for reactive training scenarios. Another method is to integrate BARS-ET with a game engine. The use of game technology in military trainers is gaining wider acceptance (Capps, McDowell, & Zyda 2001). However, these game-based systems do not yet have the defense user base and database library available to SAF. Many potential users of BARS-ET already know how to use a SAF system and therefore require no extra education to set up complex scenarios for the AR training system. One advantage of game engines, however, is that they are designed for smooth real-time interaction and animation, whereas SAF systems may not guarantee that smoothness. A solution to this problem is described later in this section.

The BARS architecture is built around an event-based shared distributed database. Each BARS user has a local copy of the environmental database and network events are used to synchronize changes between users. The data distribution system is described in more detail in a previous paper (Brown et al 2003). The way BARS-ET connects to a SAF system is through a bridge application. This application implements both the BARS and SAF network protocols and translates events between the systems. Figure 4 illustrates this concept.

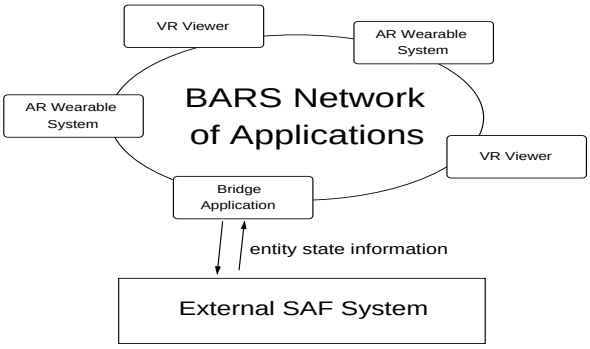

Figure 4. Sharing information between BARS and an external SAF system using a bridge application.

Although the next two subsections give many details on the implementation of the OTBSAF and JSAF bridge applications., the bottom line is that each system provides a useful behavior system for driving the simulated forces. Each SAF has new support for dismounted infantry that can handle fine-grained position and heading changes for the forces as well as pose changes (standing, running, lying prone, shooting, etc). Thus, with the right translations and interpolations of the SAF updates, BARS-ET can show convincingly animated and reactive simulated forces.

4.1 OTBSAF

OTBSAF, the OneSAF Testbed Baseline SAF, is as the name suggests, a testbed for the OneSAF system being built by the US Army. It is a branch of the older ModSAF system. It primarily uses the Distributed Interactive Simulation (DIS) protocol to communicate between instances of OTBSAF applications.

DIS is an Institute of Electrical and Electronics Engineers standard (IEEE 1995) that started out as the communications backbone for the SIMulation

NETwork (SIMNET) community. The primary means of communication is through the transmission of Protocol Data Units (PDU), a bit-encoded packet that carries entity state and event information. Each individual event or entity update can produce a PDU. OTBSAF uses the DIS protocol to distribute entity information over Internet Protocol (IP). OTBSAF also has an implementation of the High Level Architecture, however it was a limited implementation as the time of the development of our prototype, so DIS was used to connect OTBSAF with BARS.

The OTBSAF bridge application implements the BARS networking as well as reading and writing DIS PDUs. Because most of BARS is written in Java, the DIS-Java-VRML package from the Naval Postgraduate School (Naval Postgraduate School 2000) was integrated to handle the DIS receiving and transmitting duties.

One fundamental mismatch in the way the BARS distribution system and DIS work is how entities are distributed and updated. One issue is terminology: BARS uses "objects" and DIS uses "entities" to mean the same type of data, so these two terms will be used interchangeably in the remaining discussion. In BARS, there is a "create object" event that notifies recipients that a new object has been created and specifies the initial state of the object. Then, as the object changes, "change object" events are sent to change the object's state on remote machines. These events typically only indicate the change of a single aspect of the object, for example, its position but not its color, size, etc. Finally, a "destroy object" event designates when an object should be removed from the scenario. All recipients assume an object remains in the scenario until receiving one of these events.

In DIS, the entity state PDU (ESPDU) carries all information about an entity's state. If only one aspect of an entity changes, for example its position but not its marking, orientation, etc., this information is sent in an ESPDU that still contains all other state information for the entity. This design provides a high degree of redundancy over unreliable network transport, but it differs from BARS in three important ways: First, a remote application does not need to receive an explicit "create" PDU for an entity—if it receives an ESPDU for an unknown entity, that PDU contains enough data to go ahead and instantiate the entity. Second, the ESPDU doesn't indicate what parameter of an entity has changed since the last state update. Finally, ESPDUs are sent regularly for each entity to keep them alive—if a remote system doesn't receive an update for an entity after some timeout period, it can assume that the entity can be removed from the scenario. If this decision is wrong, the entity can be completely re-instantiated once another ESPDU is received for that entity.

The main goal of the OTBSAF bridge application is to maintain a one-to-one mapping of DIS entities to BARS objects, including any state changes to those entities that can be translated between the two systems. First, a lookup table was created to map BARS objects to DIS entities based on their ID numbers. When a new object from either BARS or OTBSAF is discovered by the bridge application, its counterpart in the other domain is created, and the IDs are put into this table. This process happens when a BARS object creation event is seen, or when a DIS ESPDU for an unknown object is received.

In maintaining this one-to-one mapping, the bridge application must also translate object changes. When a BARS object change event is received, a new ESPDU is sent out containing a translation of this change, for the corresponding DIS entity. An ESPDU containing only the information for this change cannot be sent, as the other fields would be blank and cause errors in the scenario. So, a complete ESPDU must be sent, but with this new change reflected in it. In order to construct this ESPDU, a second table is maintained, mapping a DIS entity ID to the last ESPDU received for that entity. Instead of completely reconstructing the new ESPDU from the BARS object properties, the most recently received ESPDU is copied and the new information filled in. This process ensures that any state information in the DIS entity that wasn't translated to the BARS object is maintained between updates. The complimentary process of mapping DIS entity changes to BARS object changes is simpler. When a new ESPDU is received, then for each parameter that can be translated, the parameter's value is checked for any change, and if it has changed, a new BARS change event is sent.

Another important aspect of maintaining the one-to-one mapping is dealing with destroyed entities. When a BARS object is destroyed, the bridge just stops sending ESPDUs for that entity, and after a certain timeout period, remote applications will remove that object. When a DIS entity is destroyed, no explicit event is sent. Thus, the bridge application must periodically check all of the objects in the map and check when the last ESPDU was received. If this time exceeds the timeout period, the corresponding BARS object is explicitly killed. Another side effect of the DIS timeout model is that BARS objects must be kept alive in OTBSAF. Since BARS only sends events when the

database changes, relatively static entities are killed by OTBSAF because no updates are received. So, periodically, the bridge goes through the entity map and sends an ESPDU for each BARS object to keep the corresponding DIS entities alive.

One final important issue in maintaining the one-to-one mapping is dealing with dead reckoning. The designers of DIS realized that sending a PDU for every position or orientation update for each entity does not scale and quickly uses up network capacity. In order to prevent flooding the network with too many updates, they used dead reckoning to have each remote machine update an entity in between receiving ESPDUs. The originating machine also maintains a model of the process, and when it sees that the dead reckoned values on the remote machines should exceed some threshold, or some amount of time has elapsed, then a new ESPDU is send to correct the entity state on the remote machines. Unfortunately, BARS, not designed for simulation, does not have any native notion of dead reckoning, and the bridge had to fill the gap. The bridge does a naïve version of dead reckoning to maintain the corresponding BARS object between ESPDU updates for a DIS entity. Going the other way, dead reckoning just isn't supported yet—DIS entities created in response to BARS objects have empty dead reckoning fields and do not change between ESPDU updates. They do, however, change as often as the BARS object changes.

Once the process for maintaining the one-to-one mapping between DIS entities and BARS objects was in place, the actual semantic translation of the entity types and parameter values had to be tackled. For the translations of entity types, BARS needed some work. It already supported some object types that were direct matches to the OTBSAF entity types. New object types were added for the entity types it didn't support. Only a handful of entity types are supported, including humans of various types (friendly, enemy, neutral) and common vehicles. A third-party library was inserted into BARS to provide the human animation and some vehicle models.

The translation of entity parameters was less straightforward. BARS uses a simple coordinate system based in meters in three dimensions from an arbitrary origin, while OTBSAF uses the Global Coordinate System (GCS), which is latitude, longitude, and altitude. The conversion between the two systems uses a third-party library to translate GCS into Universal Transverse Mercator (UTM), and since UTM uses meters in three dimensions from a grid point on the globe, a simple offset correction yields the BARS coordinate. The reverse of this process converts BARS coordinates into GCS. For other parameter types, the authors' best judgement was used to map the meaning of the ESPDU bit fields into values supported by BARS, for example, the pose of an individual combatant (stand, run, kneel, etc). Unfortunately, these conversions are hard-coded, and because the meanings of the ESPDU bit fields vary based on the application sending the PDUs, these conversions will need to be changed should we use another DIS-based SAF system or even s newer revision of OTBSAF.

Finally, the SAF system had to be notified when the BARS user was firing. The tracked weapon uses a simple momentary contact button that the user presses to indicate a firing. When the button is pressed, BARS collects the tracker data, creates a very simple ballistic model, and sends an event to the bridge application. The bridge then creates a fire PDU to send to OTBSAF. In addition, the bridge receives fire PDUs from OTBSAF and can indicate when the user is hit by the synthetic forces.

Although OTBSAF and JSAF both grew out of ModSAF, they've taken different paths in their support of DIS and its proposed replacement, the High Level Architecture (HLA) (Institute of Electrical and Electronics Engineers 2000). Next we will describe how BARS was connected to JSAF using HLA.

4.2 JSAF

JSAF is a collection of libraries and programs that are oriented toward real-time large-scale distributed simulation. JSAF is actively used as a tool that validates the applicability of integrating transition technologies into the modern warfighter's inventory of capabilities and tools. Its current development is sponsored by US Joint Forces Command, Joint Experimentation Directorate (J9), and has been integral in the US Navy's Fleet Battle Experiments. JSAF primarily uses the HLA as its communication layer.

The High Level Architecture (HLA) is an IEEE standard that was developed as a means by which the Modeling and Simulation (M&S) community could share a common architecture for distributed modeling and simulation. There are three underlying portions of this framework; the rules, the federation interface specification, and the object model template. The HLA federation interface defines the common framework for the interconnection of interacting simulations, and is of particular interest to our understanding JSAF. The HLA Runtime Infrastructure (RTI) and a set of services implement this interface. The RTI and these services

allow interacting simulations to efficiently exchange information in a coordinated fashion, when they participate in a distributed federation.

For any object or interaction, JSAF determines the locality of related entities for this communication and sends the necessary information via the RTI if they are known to be upon a different processor. For example, if a DI fired upon another DI on the same JSAF process, the simulation would handle this interaction internally. But if the second DI was known to be in another federate (in our case a BARS bridge), then the interaction would go out via the RTI. The only limitation to remember is that JSAF must publish this interaction (a default setting). To complete this communication, the federate expecting to receive this interaction must correspondingly subscribe to this interaction.

The JSAF bridge application implements the BARS networking paradigm as well as implementing an RTI interface to communicate with JSAF. JSAF has a set of libraries supporting the Agile FOM Interface (AFI). By using this interface, JSAF creates a mapping of its internal data representations into an external representation. This mapping is necessary for JSAF to participate in different federations without modification. The mappings are stored in special files, called reader files that are interpreted at run-time. The unexpected advantage to this approach is that these libraries can be used by other applications, such as our SAF Bridge, to create a pseudo-JSAF federate. By including a subset of the JSAF support libraries, it is possible to create SAF representations of BARS objects in the bridge. This has many advantages:

- The transparent communication between the bridge and JSAF by RTI object updates and interactions (i.e., calls and formatting issues handled by the internal JSAF libraries).
- Using the same terrain databases and JSAF's terrain libraries to ease position translations between BARS and JSAF.
- Leverage physical models in JSAF to handle weapon behavior (ballistics, damage, etc.).

In BARS and JSAF there are corresponding "events" that relate to the creation, change and destruction of an entity. In BARS, these specific events trigger the dissemination of the salient object state changes to other applications, such as the SAF Bridge. When it receives this information, it is necessary to translate the data into the appropriate object updates such that it may be sent via the RTI to JSAF. BARS typically updates positions at a much higher rate than a system such as JSAF desires. So we track the updates for our BARS user in a lookup table, and use the simulation time libraries we inherit from the JSAF interface code to limit the update rate to a more reasonable one (1 Hz). An exception to this rule is when there are significant orientation and movement changes in the BARS user (i.e., begin walking, change facing, etc.). Location coordinates and heading data are converted between BARS and JSAF using the same methods described previously for OTBSAF.

JSAF has a corresponding mechanism for object updates. The SAF Bridge catches incoming updates at the rate they arrive and store the information in another lookup table (STL map). In this case, we also store a pointer to the JSAF platform object that relates to the object update. This allows the SAF Bridge application to use the built-in dead reckoning code from JSAF to interpolate the current position of a moving entity without having to receive constant updates. The dead-reckoning algorithm cuts down on network traffic. In essence, each entity would have a set frequency to update its position. Remote machines would calculate a new position based on dead reckoning. The originating machine on the other hand would simultaneously calculate its ground truth position and its new dead-reckoning position, and if they differed by some delta, a new update would be broadcast to the network. This aspect of dead reckoning has a side effect due to the normal effects of network latency. For example, the SAF Bridge sends position updates to BARS at 10 Hz and a new JSAF update arrives indicating that at some point in the past the entity being tracked had turned. This leads to a visual artifact in the BARS environment display when the SAF entity "jumps" to its new location.

In a similar fashion, the interaction between an "armed" BARS user and JSAF DIs requires additional management. The SAF system needed to be informed whenever the BARS user was firing. The tracked weapon uses a simple momentary contact button that the user presses to indicate a firing. The tracker data is used to call the JSAF provided ballistics library and determine if any of the synthetic forces have been hit. If the target is hit, a "fire" interaction is sent by the RTI to JSAF. Once received, JSAF can compute the damage and if necessary change the status of the SAF entity (damaged, dead, etc.). By using the corresponding libraries inherited from JSAF, the BARS user can also be targeted and damaged by SAF entities. We have yet to implement a mechanism to indicate incoming weapon fire and damage to the BARS user.

LESSON 5: Incorporate other high-impact but low-cost realism enhancements

Stimulating as many senses as possible is believed by many to provide a more realistic and effective virtual training environment—providing tactile, aural, and visual cues enhance a user's feeling of presence and memory of the virtual experience. Much work has been performed on stimulating the various senses in virtual and augmented reality—but most of this work is developed on and for desktop workstations. Wearable computers, while becoming ever more powerful, still lag behind high-end workstations in performance and available resources. Additionally, some of the supporting equipment for these effects is not practically man-portable. However, there are two low-impact (with respect to development time and product weight) enhancements that offer a high payoff: spatialized audio and animated humans.

Spatialized audio enhances the training experience by giving the user more information about the environment (footsteps behind the trainee, for example) and making the experience more realistic and memorable (Dinh et al. 1999). Additionally, the implementation requires only a software library and a set of headphones. To render the graphical display, the system must keep track of the user's attitude in the virtual world along with the locations of the simulated forces. Since this virtual world information is available, it is not a great leap to support spatialized audio. Sounds can be attached to objects (example: helicopter) or specific events (example: gunfire) in the virtual world. A 3D sound API is updated continuously with the positions of the user and simulated forces. BARS-ET supports the Virtual Audio Server (VAS) (Fouad, Ballas, & Brock 2000) and Microsoft's DirectX (Microsoft 2004). The API takes simple monophonic sound files and renders them in the user's headphones so that they sound like they have distinct positions in the real world. The trainee can hear simulated forces come up from behind and can hear simulated bullets flying by.

The first implementation of BARS-ET used static VRML models for the computer-generated forces, and seeing the static models slide around the environment was not convincing to the first users of the system. Adding realistically animated humans to the system was another low-impact improvement that paid off well. In this case, only a third-party software library was added. The DI-Guy animation system (Boston Dynamics 2004) was integrated into the BARS-ET graphics renderer. Combined with the occlusion model, the forces realistically emerge from buildings and walk around corners.

LESSON 6: Coordinating data sets can be hard

The general mechanism for using BARS-ET for training, in conjunction with a SAF system, has been described. However, in order to use BARS-ET with a SAF system in a meaningful way, they must both use the same terrain database, converted into their respective formats. BARS-ET uses a custom database format but can also load VRML models. Both OTBSAF and JSAF use the Compact Terrain Database (CTDB) format to store terrain information for a scenario, and for building structures, the Multi-Elevation Structure (MES) format is used. Unfortunately, the two SAFs currently use different revisions of CTDB, and although databases can be converted between the revisions, the conversion process isn't perfect.

To synchronize the databases between SAF and BARS-ET, one or more conversions must be made. The easiest conversion path is to start with the CTDB/MES files needed for a scenario—luckily, many MOUT facilities have been modeled already and the data is available through the proper channels. If this data isn't available, the implementers have a long surveying task ahead. This data can be converted to a 3D solid model format, which can then be changed to a flat black occlusion model through human intervention. The model is then exported to VRML to create a file that BARS-ET can read. Assuming the original model was surveyed carefully and the conversions worked well, the the model in BARS will match the model in SAF, which matches what is in the real world. That condition is necessary for the training system to work.

Once the terrain is available to the SAF system, a scenario can be created using the features in that system. In the SAF system, the BARS-ET user shows up as just another DI entity, so scenarios can be created to involve the BARS-ET user just like any other human-controlled SAF entity. Additionally, these scenarios can be saved and repeated as necessary during a training exercise.

LESSON 7: Test and Validate

As with any system, test and validation are important pieces of the development cycle. During construction of BARS-ET we have talked with many subject matter experts to help guide some of the user-oriented decisions we had to make. We have had many people try the system on-site at NRL and have used their

feedback to refine the system. In addition, we transported the system to I/ITSEC 2003 for anyone attending the show to try out. Such an event made the weak points of BARS-ET very obvious.

One weak point of BARS-ET is the time and effort required to set up the system for each venue. The training environment must be modeled to build the occlusion model and SAF database. Some venues already have SAF databases, however, the level of accuracy and detail may not be enough for the occlusion model. A model of a training environment, when viewed in a VR application, may appear to be a perfect match. However, when that model is overlaid on the real world using an AR system, many imperfections become apparent. If the model is too inaccurate, the location must be surveyed to build the occlusion model. BARS can connect with commercial surveying equipment, such as the Leica TotalStation, to interactively build the model quickly and accurately

Tracking is another consideration during system set up. If the training environment is indoors, then an indoor tracking system has to be installed. Current tracking systems that are accurate enough to work in AR typically require careful surveying of the tracker components (beacons or fiducial markers). However, with careful planning, this is a one-time cost for each training venue.

The weapon aiming accuracy does not even approach that of a real weapon. This deficiency results from fundamental limitations in the accuracy of the tracking system. Our solution is to draw a weapon avatar on the display that lines up with the real tracked weapon, again, within the limitations of the tracking system. The user thus aims the virtual weapon at the virtual forces instead of the real weapon. This is effective, except when the weapon is not in the user's display. The weapon can still be aimed, but is subject to tracker error. Instead of actively tracking the weapon, a future version of BARS-ET could integrate a passive laser-based fire detector to more accurately register where the trainee fired. The importance of the virtual weapon aiming accuracy is under consideration, as well as possible negative training effects.

Some users could not adjust for the deficiencies of the video-based display. As mentioned previously, those effects are lower resolution, slight lag when moving quickly, and a difference in brightness and contrast compared to the real world source. These effects distorted the view of the real world enough that the users felt they were in a completely virtual world, which works against the purpose of an AR training system. Most users did not experience this problem.

User reaction to the training system was generally positive. Many users who have been through MOUT training liked the concept and the initial implementation.

Figure 5. A conference attendee tries the system.

CONCLUSIONS AND FUTURE WORK

We designed a system that can help trainees in situations requiring engagement between individual combatants, such as those in MOUT scenarios. By using mobile AR, synthetic forces are inserted and engaged realistically in the real world. A connection to a SAF system allows the synthetic forces to behave intelligently and gives trainers a familiar interface with which to control the scenario. This system gives the trainee the benefits of both live training and of having synthetic actors for a predictable, repeatable scenario.

Although the basic pieces are in place to use mobile AR for embedded training, there is still much work to be done. We have in mind several improvements as future work. These improvements would yield a more effective system:

- Implement a method to convert BARS terrain models into the CTDB format used by the SAF systems, thereby allowing the original site model to be built using the model construction facilities in BARS (currently, conversion is only possible in the opposite direction, from CTDB to BARS).
- Make the synthetic forces look more realistic in the AR display. The forces are currently drawn without respect to environmental conditions, shadows, or any occluding items that are not already in the occlusion model.
- Increase the accuracy of the weapon tracking system. The current tracking methods are

accurate enough for measuring the user's viewpoint, but even slight errors in tracking the weapon will greatly reduce the accuracy of the user's aim.

- Test the system at an actual MOUT facility.
- Draw upon virtual reality-based user tests to develop a method of testing training effectiveness using BARS-ET and use that method to run user studies to validate the system.

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.*

Boston Dynamics Inc. (2004) *DI Guy*. Retreived June 23, 2004 from http://www.bdi.com/content/sec.php?section=diguy

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.*

Capps, M., McDowell, P., & Zyda, M. (2001). "A Future for Entertainment-Defense Research Collaboration." *IEEE Computer Graphics and Applications*, Jan-Feb 2001.

Ceranowicz, A (1994). "Modular Semi-Automated Forces." *Proceedings of the 1994 Winter Simulation Conference*, pp. 755-761.

Dinh, H. Q., Walker, N., Song, C., Kobayashi, A. & Hodges, L. F. "Evaluating the Importance of Multi-sensory Input on Memory and the Sense of Presence in Virtual Environments", *Proceedings IEEE Virtual Reality*, Houston, Texas, March 13-17, 1999

Fouad, H., Ballas, J. A. & Brock, D. (2000). "An Extensible Toolkit for Creating Virtual Sonic Environments." *Proceedings of International Conference on Auditory Displays 2000, Atlanta, Georgia, USA.*

Institute of Electrical and Electronics Engineers (1995). *International Standard, ANSI/IEEE Standard 1278.1-1995, IEEE Standard for Distributed Interactive Simulation--Application Protocols.*

Institute of Electrical and Electronics Engineers (2000). *International Standard, ANSI/IEEE Standard 1561.1-2000, Standard for Modeling and Simulation High Level Architecture, Federate Interface Specification*.

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., 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.*

Microsoft Corporation (2004). *DirectSound*. Retrieved June 23, 2004 from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/directX/htm/directsound.asp

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

Natick Soldier Center (2004). *Future Force Warrior.* Retrieved June 24, 2004, from http://www.natick.army.mil/soldier/WSIT/

Naval Postgraduate School (2000). *DIS-Java-VRML Working Group.* Retrieved June 23, 2004 from http://web.nps.navy.mil/~brutzman/vrtp/dis-java-vrml

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

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.