An Event-Based Data Distribution Mechanism for Collaborative Mobile Augmented Reality and Virtual Environments

Dennis Brown, Simon Julier, Yohan Baillot
ITT Advanced Engineering & Sciences
2560 Huntington Ave
Alexandria, VA 22303
{dbrown,julier,baillot}@ait.nrl.navy.mil

Mark A. Livingston
Naval Research Laboratory
4555 Overlook Ave SW
Washington, DC 20375
markl@ait.nrl.navy.mil

Abstract

The full power of mobile augmented and virtual reality systems is realized when these systems are connected to one another, to immersive virtual environments, and to remote information servers. Connections are usually made through wireless networks. However, wireless networks cannot guarantee connectivity and their bandwidth can be highly constrained. In this paper 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. We demonstrate the mechanism in the Battlefield Augmented Reality System (BARS) situation awareness system, which is composed of several mobile augmented reality systems, immersive and desktop-based virtual reality systems, a 2D map-based multi-modal system, handheld PCs, and other sources of information.

1. Introduction

As computers have become smaller and more powerful, the concept of portable, high performance system computer system for augmented and virtual reality has become feasible. State-of-the-art (2002) laptops boast processor, memory, disk, and graphics hardware that rival supercomputers from only five years ago. The full power of these systems is realized when these systems are networked with one another and with remote information servers. For example, a user walking down a street might be able to see the locations of other users, up-to-date information about prices in shop windows, and even email [4]. In our research on the Battlefield Augmented Reality System (BARS) [9] [11], we are focused on the problem of developing information systems able to provide users with "situation awareness" — data about their environment and its contents.

The problem of distributing data to such users is somewhat unusual. Most research into distributed virtual or augmented reality focuses on the support of common, consistent virtual worlds, typically maintained by a small number of users. However, the objective of BARS is to support a consistent information space. Therefore, 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, such as the obvious task of distributing the virtual object database, as well as more general uses such as "remote control" of applications. 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 general 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.

The structure of this paper is as follows. The problem statement and a survey of existing literature is given in Section 2. Section 3 describes the event distribution system. Conclusions and future work are discussed in Section 4.

2. 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. By situation awareness, we mean that each user obtains a better understanding of the environment. 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 (AR) makes distribution of this type of information important for mobile AR systems.

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 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 handles position tracking while an inertial device handles orientation tracking. A camera can be used for tracking or for sending video reports to a base station. Wireless 802.11b networking is used for data distribution and GPS corrections[1]. The BARS mobile user sees graphics on or next to the real objects they are intended to augment, as well as status information such as a compass and messages from other users. Figure 2 shows an output from the system — the parking lot is augmented to highlight the neighboring building and a fellow BARS user.

Figure 1. The BARS Wearable

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

- The objective is to provide information, not a consistent virtual world. The environment can be viewed to be populated by a set of objects which are self-contained entities and other types of discrete data. Each object can be relatively simple. It is not necessary to transmit complicated geometric objects or behaviors (such as animation). The latency in the update of an object or an entity is a secondary consideration.

[1]When some future version of BARS is used in real operations, communication would likely happen over the US military's communications system of that time, its construction probably influenced by the current research described by North et.al. [13]. However, it is likely that any deployed system will still have problems with connectivity and bandwidth in built-up areas.

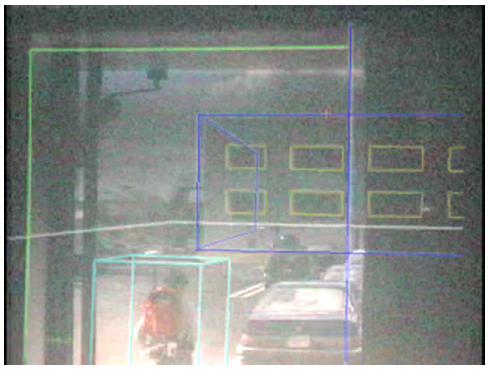

Figure 2. A Sample Augmentation

- Data distribution between users can be heterogeneous. Different users might carry out different tasks and thus have different information requirements.
- The distribution system should facilitate collaboration between different users. In addition to environmental data, the distribution system must support the propagation of meta-data such as task assignments, objectives, and 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 moves from location to location, reception strength will vary and the bandwidth can be limited and can be time varying (depending on signal strength and number of users).

Given its potential importance, a great deal of research has been carried out into distributed systems for virtual and augmented reality.

The Naval Postgraduate School has been working on large-scale distributed virtual environments for over a decade, their first version of NPSNET being shown in 1991. The data distribution in the current version, NPSNET V [3], uses replicated data and the model-view-controller paradigm to maintain it. Users of the system see a view of the entities (models) drawn by a rendering system. These entities are modified by the protocols (controllers) that translate network messages into state changes. New protocols can be loaded at run-time to extend the behaviors of entities. The packets are sent via multicast, and the system allows many protocols to share a single multicast socket. Network traffic is controlled with an area-of-interest manager.

The Distributed Interactive Virtual Environment (DIVE) system [5] is designed to scale to many participants while maintaining a high level of interactivity at each site. It is based on a peer-to-peer architecture that uses reliable multicast to maintain a database that is replicated at each peer. Using a heirarchical partitioning of the database, only part of the database may be replicated at some peers. To maintain highly interactive rates, some copies of the database may be updated faster than others, but there are mechanisms to ensure equality over time.

MASSIVE-2 [6] uses the concept of a local object, or artifact, and remote views of that artifact. These artifacts are paged in and out of remote applications. A spatial model of interaction is used in that a remote application will have a focus that defines a subgroup of the artifacts to page in and out, instead of copying the entire database. Artifact state changes are distributed using reliable multicast.

Not all work has been solely for immersive virtual environments. Some collaborative AR systems have been built, typically for small groups of co-located users. One example is Shared Space system [2] for computer-supported collaborative work using AR. Typically, users are operating closely with one another and latency requirements are extremely high. Another system designed for augmented and virtual reality is COTERIE [12], which provides useful semantics through its notions of Distributed Shared Memory and Callback Objects. Distributed Shared Memory allows many applications to shared the same data objects, while Callback Objects allow applications to be notified of changes to those data objects — an event-like mechanism. The Studierstube system [16] extends the concept through its use of an extensible scenegraph of application objects that is distributed to all users.

3. The BARS Event Distribution System

The BARS distribution system is based entirely on the concept of *events*. Events are used both to instantiate objects (in effect, transmit a view of a database between systems), to send information to update preexisting 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: event transporters, replicated object repositories, and communication channels. We will describe these components, as well as the use of bridge applications to communicate with outside virtual and augmented reality systems, and some other uses of the event distribution system.

3.1. 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.

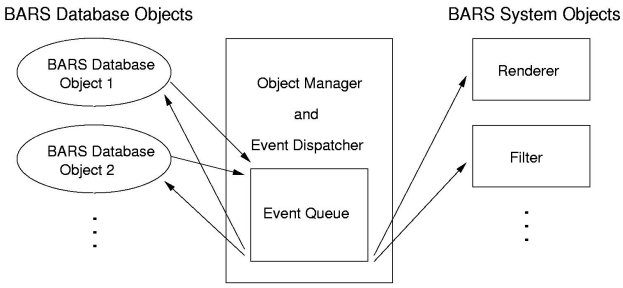

Figure 3. Event distribution within an application: Arrows show event movement.

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 its roots in the Java AWT event model [17] we leverage the Reflection API to achieve these steps. The type of each event is specified by its class. For each event type, a listener is defined as well. When a listener is registered with the object and event manager, its interface is queries 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 an application. A thread handling the position and orientation trackers will update the user's position by calling a method in the user object to set its attitude based on data gathered from the trackers. 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 soon processed. The dispatcher sends the event to all listeners, including the object itself, as well as the graphics system (to update the viewpoint) and the filter (to update what objects are being drawn), 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 retained by going through the event queue in the dispatcher. Figure 3 shows the flow of events within an application.

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 application instances to send events

to and receive events from each other using 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 [10], 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 section). Because of the connectionless nature of IP multicast, the distribution is robust in that the network connection can come and go and the user application still functions, 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 change 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.

3.2. 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 for users and entities, and objects created as communications, such as reports of enemy locations, routes for users to follow, and digital ink. This database is replicated in whole or in part for every BARS 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. 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. Therefore, since the user will have some form of the 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 very few limitations on the base users. Their applications run on stationary VR systems such as a desktop computers, 3D workbenches, and immersive VR rooms. Using the same distribution system, they will have high levels of detail and interaction by taking advantage of the better bandwidth for replicating more objects and seeing change events at a higher frequency.

3.3. Channels

If simply implemented as described above, the event distribution mechanism will send all events to every application, which for example would replicate the entire database inside every application. As demonstrated by the works of others described earlier, creating copies of every object for every user and updating those replicas would soon swamp the network with information many users would not care about anyway. So, like those systems, we have devised a way of partially replicating the database to minimize the amount of unwanted data distributed to each application.

In creating this replication mechanism, we first looked at the uses of BARS that should 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 looked at 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 and would want to see objects within a possibly huge radius around the current position. However, it seems that there would be few situations in which a mobile user would care about objects very far away, so for most situations, a simple area-of-interest mechanism could work.

However, another condition is the type of information being distributed. Even if some objects are near to a mobile user, they may have no importance and could be distracting. Or, they 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 [8], we described a filtering mechanism for mobile augmented reality. This filtering mechanism works on the object repository within an application. It is sufficient for its original goal of not showing users objects in which they have no interest in order to reduce display clutter. However, it simply hides objects from the user — it does not actually control whether or not the application holds replicas of these objects, or whether or not the

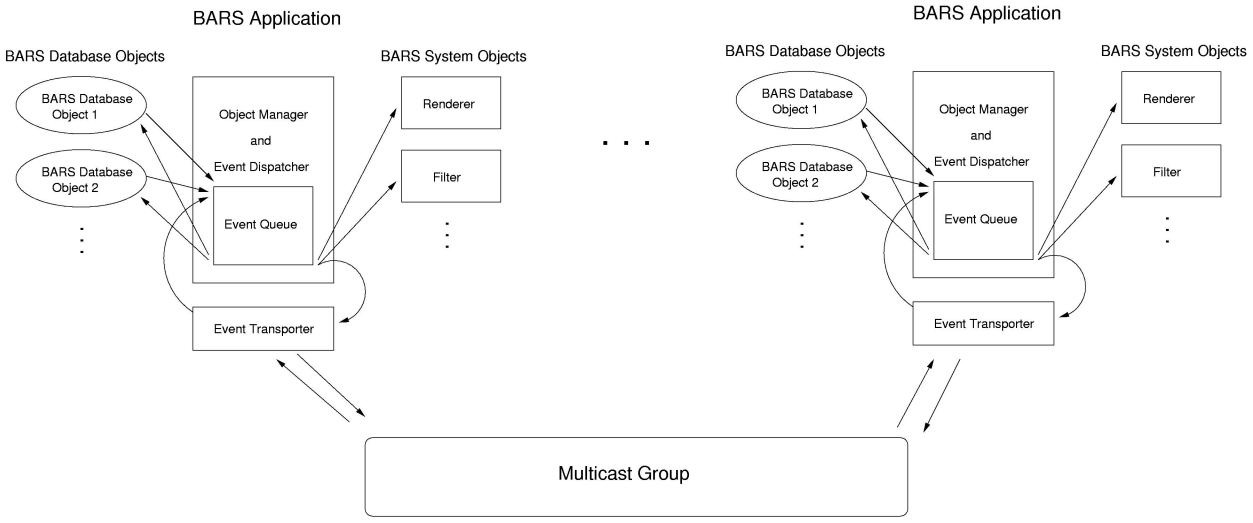

Figure 4. Event distribution between applications: Arrows show event movement.

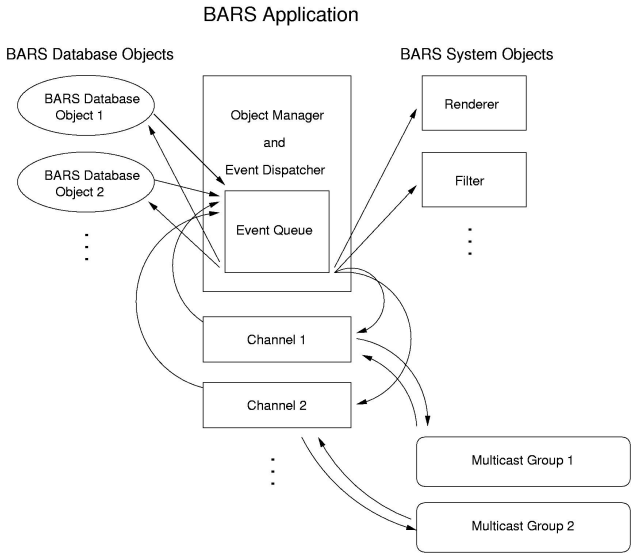

Figure 5. Application joined to two channels: Arrows show event movement.

application still receives events related to these objects.

Keeping these situations in mind, we have developed *channels*. The term has many uses in many systems, 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 ones, until all available multicast groups are used up. Figure 5 shows a single application using two channels.

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 would replicate most objects nearby, the hazardous objects channel could cover a larger area, but only include those hazards. BARS also has several interaction modules that produce many objects as they are used. One interactor is for construction in which users can collaborate to place points and build objects from those points [1]; in this case, the intermediate points would be in a channel only joined by the constructing users so that 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 user drawing the ink and the application to interface with the multimodal system would place the ink in a separate channel so that other users would not see these sketches out of context.

3.4. Bridges

As we alluded to in the previous section with the multimodal interaction example, some of our applications communicate with other systems. We call these applications *bridges*. These applications join both the BARS distribution system and an outside system. They translate object creation and changes between BARS and outside systems. By maintaining maps between BARS objects and these outside objects, we can represent the outside objects in BARS and vice-versa. Two systems with which we can communicate are the Columbia Mobile Augmented Reality System [7] and the Oregon Graduate Institute's Quickset multimodal interface [14].

We also mentioned that a TCP/IP transporter is avail-

able in the BARS distribution system. Although the system is designed primarily for multicast, sometimes we need to connect to networks that block multicast. In this case we have a simple TCP/IP bridge application that joins local multicast groups and creates socket connections to other BARS applications through TCP/IP event transporters. This can be a bit of a bottleneck, but we can create one TCP/IP bridge per outside application if necessary, and effective creation of channels can help limit the amount of data transferred.

3.5. Other Uses

The final aspect of the BARS event distribution system we will discuss is its flexibility. Although the system was designed to carry information about objects in the database, at its heart, it is still a distributed event system. We use the same base event types and transporters for other purposes. One set of events are ”meta-events” to configure the event transporters; a similar set of events configures channels. Another set of events is used to distribute our user interface to remote systems — for example, when a new user is trying out the BARS mobile system and is unfamiliar with the user interface, we can control its user interface from another computer (including handheld computers and PDAs) by sending and receiving events that are handled by the user interface system.

4. Conclusions and Future Work

We have presented an event-based data distribution system that we have fully implemented for BARS, our mobile augmented reality system, that addresses some requirements we encounter that developers of networked virtual environments do not. Some of these are working on unreliable network connections, handling many users through its channels, working well in both high-bandwidth (base VE) and low-bandwidth (mobile AR) applications, communicating with outside systems through its bridges, and being flexible enough for non-database data distribution.

For future work, we need to develop the channel concept further. Specifically, we would like to develop intelligent algorithms to automatically create channels and join applications to those channels. These algorithms would take into account such factors as database object types, the user's task, location, and short-term interests, and network conditions. We also will continue creating bridge applications to widely-used military information systems.

The second area we shall consider is the problem of “partitioned” data networks [15]. These arise if a mobile user (or set of mobile users) are out of contact for prolonged periods of time. During this time, multiple state updates have occurred and it is necessary to synchronize the systems again.

Finally, we would like to implement some performance evaluation code and run tests to compare against other distribution systems. While we've found the performance to be sufficient for our needs, we have not actually quantified it to compared against other systems.

References

[1] Y. Baillot, D. Brown, and S. Julier, “Physical Model Authoring Using Mobile Computers”, in *Proceedings of the 2001 International Symposium on Wearable Computers*, Zurich, October 2001.

[2] M. Billinghurst, S. Baldis, E. Miller, and S. Weghorst, “Shared Space: Collaborative Information Spaces,” in *Proceedings of HCI International 1997*, pp. 7-10.

[3] M. Capps, D. McGregor, D. Brutzman, and M. Zyda, “NPSNET-V: A New Beginning for Dynamically Extensible Virtual Environments,” in *IEEE Computer Graphics & Applications*, September/October 2000.

[4] S. K. Feiner, “Augmented Reality: A New Way of Seeing”, in *S*cientific American, Vol. 286, No. 4, April 2002, pp. 48–55.

[5] E. Frécon and M. Stenuis, “DIVE: A scaleable network architecture for distributed virtual environments,” in *Distributed Systems Engineering Journal (special issue on Distributed Virtual Environments)*, Vol. 5, No. 3, Sept. 1998, pp. 91–100.

[6] C. Greenhalgh, “Dynamic, embodied multicast groups in MASSIVE-2,” Technical Report NOTTCS-TR-96-8, Department of Computer Science, The University of Nottingham, UK, 1996.

[7] T. Höllerer, S. Feiner, T. Terauchi, G. Rashid, and D. Hallaway, “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.

[8] S. Julier, M. Lanzagorta, S. Sestito, L. Rosenblum, T. Höllerer, and S. Feiner, “Information Filtering for Mobile Augmented Reality,” in *Proceedings of the IEEE 2000 International Symposium on Augmented Reality*, Germany, October 2000.

[9] S. Julier, Y. Baillot, M. Lanzagorta, D. Brown, and L. Rosenblum, “BARS: Battlefield Augmented Reality System,” NATO Information Systems Technology Panel Symposium on New Information Processing Techniques for Military Systems, Istanbul, Turkey, October 2000.

[10] T. Liao, “Light-weight Reliable Multicast Protocol Specification,” Internet-Draft http://webcanal.inria.fr/lrmp/draft-liao-lrmp-00.txt, October 13, 1998.

[11] M.A. Livingston, L. Rosenblum, S. Julier, D. Brown, Y. Baillot, J. Swan, J. Gabbard, and D. Hix, “An Augmented Reality System for Military Operations in Urban Terrain,” Proceedings of the Interservice/Industry Training, Simulation and Education Conference 2002, Orlando, December 2002.

[12] B. MacIntyre and S. Feiner, “Language-level support for exploratory programming of distributed virtual environments,” in *Proceedings of UIST ’96 (ACM Symposium on User Interface Software and Technology)*, Seattle, November 1996.

[13] R. North, D. Bryan, and D. Baker, “Wireless Networked Radios: Comparison of Military, Commercial, and R&D Protocols,” in *Proceedings of the 2nd Annuel UCSD Conference on Wireless Communications*, San Diego, February 1999.

[14] J. Pittman, I. Smith, P. Cohen, S. Oviatt, and T. Yang, “Quickset: A multimodal interface for millitary simulations,” in *Proceedings of the 6th Conference on Computer-Generated Forces and Behavioral Representation*, University of Central Florida, 1996, pp. 217-224.

[15] N. Reijers, R. Cunningham, R. Meier, B. Hughes, G. Gaertner, and V. Cahill, “Using Group Communication to Support Mobile Augmented Reality Applications,” in *Proceedings of the 5th IEEE International Symposium on Object-oriented Real-time Distributed Computing (ISORC 2002)*, Crystal City, VA, 2002.

[16] G. Reitmayr and D. Schmalstieg, “Mobile Collaborative Augmented Reality,” in *Proceedings of the IEEE 2001 International Symposium on Augmented Reality*, New York, October 2001.

[17] Sun Microsystems, Inc., Java API Documentation, http://java.sun.com/docs