

A Prototype Virtual Emergency Operations Center using a Collaborative Virtual Environment

Timothy E. Wright
University of Notre Dame
twright@nd.edu

Gregory Madey
University of Notre Dame
gmadey@nd.edu

ABSTRACT

In the realm of emergency operations, planning and training is a critical ingredient for success. The use of virtual environments can offer a convenient means of practicing and simulating activities in an emergency operations center (EOC). Although many virtual environments strive to offer realism in their simulations of weather, population, and incident happenings, they often fall short in terms of collaboration among simulation participants: unless participants are at the same physical location, their ability to see and interact with one and other is limited. Moreover, interactivity that is possible may not be truly synchronous (e.g., network lag can cause activities to happen out of order). These are compelling drawbacks to computer-based EOC simulators/trainers, since collaboration is a cornerstone for successful EOC teams. To address these problems, we present the virtual EOC. Our prototype aims to provide a collaborative virtual environment that enables interactivity among participants while executing synchronous, script-driven tests and simulations.

Keywords

Virtual reality, desktop virtual reality, collaborative virtual environment, emergency management training, emergency operations center, simulation, Croquet.

INTRODUCTION

In the United States, the 2002 National Research Council (NRC) report on technology and terrorism clearly notes the importance of modeling and simulation techniques for disaster planning and training (NRC, 2002). Subsequent to this, in early 2003, a U.S. Homeland Security Presidential Directive was issued creating the National Incident Management System (NIMS). The primary function of NIMS is to provide a framework that can be leveraged by government, private-sector, and other organizations to “to prepare for, prevent, respond to, recover from, and mitigate the effects of incidents regardless of cause, size, location, or complexity” (Department of Homeland Security, 2007). Throughout NIMS, training is noted as an essential ingredient of preparedness, skills development, and skills maintenance. Of course, the U.S. is hardly alone in the ambition to leverage technology for disaster management: subsequent to the Indian Ocean tsunamis in 2004, the Tsunami Evaluation Coalition (TEC) recommended the use of open source software to help prepare for disaster response (Currión, et al., 2007). Following the guidance of both the 2002 NRC report and TEC, one method for emergency operations centers (EOCs) to address their training needs is to employ virtual environments for testing and simulating emergency scenarios.

Within the context of EOC training, virtual environments offer a way to model incidents of different types and severity levels while testing EOC team members’ abilities to respond. Though realism and completeness vary from one virtual environment to the next, the use of computer-simulated emergencies can be an efficient and expeditious means of training (Chakrabarty and Mendonca, 2006; Hendela, et al., 2006). For example, scenarios can be electronically created, saved, edited, and re-used. Also, a similar amount of effort is required whether running drills for an individual or a group of people. Finally, many virtual environments include substantial capabilities regarding the simulation of weather, populations, and events such as fires, floods, chemical spills, etc. Despite all of these relevant characteristics and features, we believe that there is a significant limitation regarding current emergency management virtual environments: collaboration.

Most commercially developed EOC virtual environments are geared toward the participation of centrally located team members. Products such as Alion Science’s *Emergency Command System™ Training & Exercising Tool (ECS)*, BreakAway’s *Incident Commander™*, ETC Simulations’ *Advanced Disaster Management Simulator*

Command (ADMS-COMMAND™), and MASA-SCI's *Simulation for Wargaming Operation Research and Doctrine—Critical Infrastructure* (SWORD-CI™) all include robust virtual environment and simulation mechanisms. However, they are also best suited for individuals or groups of participants operating in a local area network (LAN) context. The reasons for this are time and synchronicity. In a stand-alone (single participant) situation there are no timing or synchronicity issues to consider: one's desktop is always in lock step with itself. For groups of participants, however, unless everyone is in the same geographic location sharing the same LAN (so that timing issues are negligible if not imperceptible), experiences with a virtual environment can include network latencies and disruptions. In discussing the potential use of virtual reality game engines for incident management training, Jain and McLean (2005) capture this issue succinctly by noting that:

The primary challenge is the development of mechanisms for communications and time synchronization between and among simulation modules and game modules. Major issues associated with distributed multiplayer games are how and when players receive information on fellow players' actions. Time lags may occur between when a player initiates an action and when other players see the action. This latency causes problems in the execution of distributed games.

Because collaboration is such an important component on EOC teams, we contend that timing and synchronization are essential ingredients for any virtual environment used in training and simulation. This is not to diminish the other roles played by the virtual environment. However, timing and synchronization problems can limit the potential of training and simulations for geographically dispersed participants. In turn, this reduces convenience and cost effectiveness, since collaboration may suffer unless resources are expended to bring all participants to the same physical location.

We believe that a virtual EOC (vEOC) environment predicated on synchronous collaboration as well as useful testing and simulation is readily achievable. Following a review and risk assessment of many popular and capable open source tools for building collaborative virtual environments (CVEs), we have settled on the Croquet Software Development Kit to implement an initial prototype of a vEOC. This prototype represents a step in the initial phase of Project Ensayo, an endeavor to construct a large-scale system to research, simulate, and train for emergency incident management (Becerra-Fernández, et al., 2008). Written in Squeak, a derivative of Smalltalk-80, Croquet was selected for several reasons, including its built-in collaboration, timing, and synchronization mechanisms; OpenGL 3D graphics; support for avatars; text messaging communication; and platform independence. The vEOC prototype includes only very basic, but important, features: fully synchronized collaboration (through Croquet) among distributed participants, an Extensible Markup Language (XML) interface to control the vEOC during tests and simulations, and a three dimensional virtual environment with programmable video and still image displays.

The remainder of this paper is structured as follows: the next section summarizes related works; *Croquet and TeaTime* provides further background and details about Croquet, its core architecture, TeaTime, and how this relates to the prototype vEOC; *Tour of the vEOC* introduces the prototype vEOC including its 3D room (modeled after the Miami-Dade EOC); *Croquet Risks* discusses some of the drawbacks discovered while using Croquet; and *Conclusion and Future Work* ends the paper with summary remarks and a brief discussion of what lies ahead for the vEOC.

RELATED WORK

A variety of different commercial products exists in the area of emergency management training and simulation. Although not an exhaustive list, the four offerings noted earlier, ECS, Incident Commander, ADMS-COMMAND, and SWORD-CI, represent a range of approaches in virtual environment technology. All of these products have several common, basic goals: realistic simulations, engaging scenarios that can be customized, single- and multi-participant support, geographic information system (GIS) functionality, and, of course, an overall effective training and simulation experience. Differences are largely manifested in the degree of immersion (the sense a participant has of being in the virtual environment) and proximity to the simulated incidents. For example, Incident Commander offers a 2D “shadow box” display and abstract user interface, while ADMS-COMMAND provides a 3D experience that is on-the-ground and in first-person. Meanwhile, ECS and SWORD-CI integrate several components such as 3D visualizations, training and simulation workflow mechanisms, features to manage and choreograph scenarios, and, in the case of ECS, interweaving real-world elements (e.g., faxes, emails, video news clips) into the training or simulation.

An interesting, recent trend is that of adapting pre-existing game engines to the purposes of emergency management simulation. One study on homeland security training and simulation software notes that cooperative team training for “geographically dispersed organizations” may benefit from CVE systems found in online games such as the Sims Online and EverQuest (Smith, 2003). McGrath and McGrath point out that established game technology can help make immersive virtual environments more affordable and available for emergency management training and simulation endeavors (2005). To this end, they profile two relevant projects developed with the commercial Unreal Tournament game engine: Unreal Triage (a first-responder arriving at the scene of an airline accident) and Unreal Tunnel (remote EOC interactions with responders to a highway traffic tunnel incident). Also, Jain and McLean devise an architecture for integrating game engines with the needs of emergency management training (Jain and McLean, 2005), and then construct a prototype illustrating this architecture. Their prototype, which executes a dirty-bomb scenario, incorporates a number of simulation modules (plume, crowd, traffic, healthcare, and transportation) and two gaming modules (triage [based on Unreal Triage] and incident management strategy) for incident response and management training.

Jain and McLean (2005) as well as McGrath and McGrath (2005) point out that one of the drawbacks to re-purposing game engines is the potential for software license costs. Other issues identified by Jain and McLean (2006) include, but are not limited to, difficulties integrating disparate simulation modules, enabling communications between these modules and the game engine, and, of course, timing and synchronization among training and simulation participants.

Other efforts in the area of emergency management simulation and training have emphasized the simulation aspect. For example, Loper and Presnell use complex adaptive systems and agent-based models to simulate a state-level EOC’s behavior and information flows (2005). Along similar lines, Pollak, Falash, Ingraham, and Gottesman employ discrete event simulation models to investigate activities during an anthrax attack scenario. Specifically, they model an EOC, hospital, medical care distribution center, and civilian population. In building their simulation, an operational analysis framework is devised to integrate the various models, providing a solution to some of the issues (noted above) raised by Jain and McLean (Pollak, et al, 2004). A different framework approach, suggested by Bowers and Prochnow, leverages simulations developed at the Department of Defense together with the IEEE’s High Level Architecture (HLA) (2003). Here, a combination of the Joint Theater Level Simulation (JTLS—an operational-level warfare simulator) and the Joint Conflict and Tactical Simulation (JCATS—an interactive, entity-level simulator), along with HLA to integrate these simulations, is offered as a possible framework to meet the needs of emergency response simulation and training.

CROQUET AND TEATIME

The vEOC prototype establishes a basic distributed architecture within the open source Croquet environment. Croquet is designed for “creating and deploying deeply collaborative multi-user online applications on multiple operating systems and devices” (Croquet Consortium, 2007a). Croquet employs its TeaTime architecture as the means for communication and synchronization of local and remote system objects (Croquet Consortium, 2007b). Implemented as a peer-to-peer network, TeaTime enables objects to share information and interact through simple message passing. By maintaining a universal time that is strictly coordinated among all peers, TeaTime enforces synchronicity and, through a two-phase commit process, atomicity in the objects and activities that peers host. This means that vEOC participants, regardless of their proximity to one another, will experience their virtual environment in a coherent and simultaneous manner. The imperative goal of establishing the vEOC prototype within Croquet, then, is to derive fundamental EOC training and simulation capabilities that are collaborative and synchronous. Specifically, these capabilities include: a basic means of scripting and testing scenarios; support for local and remote, simultaneous users; and support for a desktop virtual reality (desktop-VR) interface including avatars and some form of participant communication.

A simple network diagram of the prototype vEOC system is provided in Figure 1. Here, we see that all participants connect to and replicate an initial Croquet “island” (a collection of objects) containing the vEOC system; from then on, these participants and the initial island are synchronized peers of one another who communicate through message passing. For security purposes, a virtual private network (VPN) tunnel is used to protect remote accesses to the prototype vEOC’s LAN.

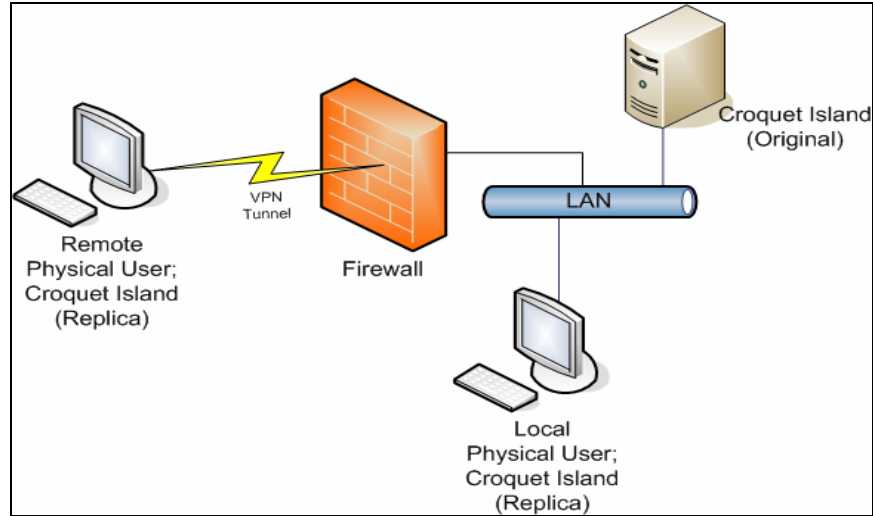


Figure 1: vEOC Architecture under Croquet

TeaTime's peer-to-peer network approach also brings efficiency and resiliency to the prototype vEOC. Efficiency is gained as a result of replicating (at peer sites) not only the island containing vEOC components, but the computations as well. This may not seem intuitive at first; by having "smart" copies of the initial island on all peers, small messages describing state and activities are transmitted during a training/simulation session, instead of shuffling around large collections of data. This also means that high-latency participants will not slow down other peers. All peers operate against a coordinated time system, which maintains strict synchronicity. Slower peers will end up experiencing sluggish performance as they work to stay in synch, but all replicated islands will remain identical. Resiliency is attained through Croquet objects tracking state information to enable recovery from system faults. This is a product of the two-phase commit protocol that Croquet uses to ensure atomicity of behaviors among peers.

To better explain Croquet's TeaTime architecture, figures 2, 3, and 4 depict a simplified example of a participant connecting to an island of objects.¹ First, Host 2 notifies Host 1 that it intends to join Host 1's island (Figure 2); Host 1 responds by replicating its island (objects and state) on Host 2 (Figure 3). Later, when object B in Host 2's island replica changes state (e.g., due to user interaction), a message is communicated back to Host 1 to maintain synchronicity between the island copies (Figure 4). Because Host 1 is the originator of the island, it also includes a Croquet mechanism known as a *router* (not pictured in the figures). The router is responsible for managing the clocks of the initial and all other island replicas. Whenever a message is transmitted from one replica to another (e.g., from Host 2 to Host 1), that message must travel through the router to be time-stamped. It is by way of this process that synchronicity and atomicity are enabled. Also, not all object state changes need to be communicated among all island replicas. This is the case, for example, when rendering images to a participant's screen. In summary, each Croquet island replica is identical and operates from a known starting point with the same inputs; thus, all replicas are deterministically the same; through simple message passing, replicas maintain their identical nature over time. It is this property that enables Croquet to deliver a truly synchronous experience, making it well suited to collaboration among distributed participants.

¹ A great level of detail is being glossed over, here, in favor of brevity and succinctness. The interested reader should refer to the Developer's Guide located on the OpenCroquet.org website.

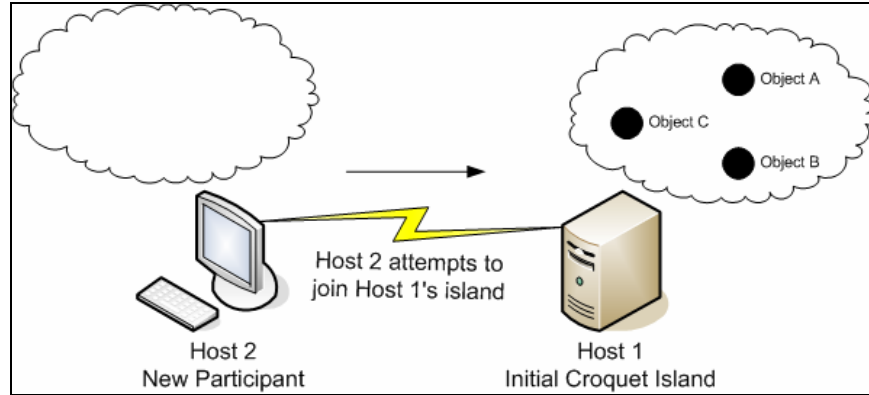


Figure 2: Joining a Croquet Island

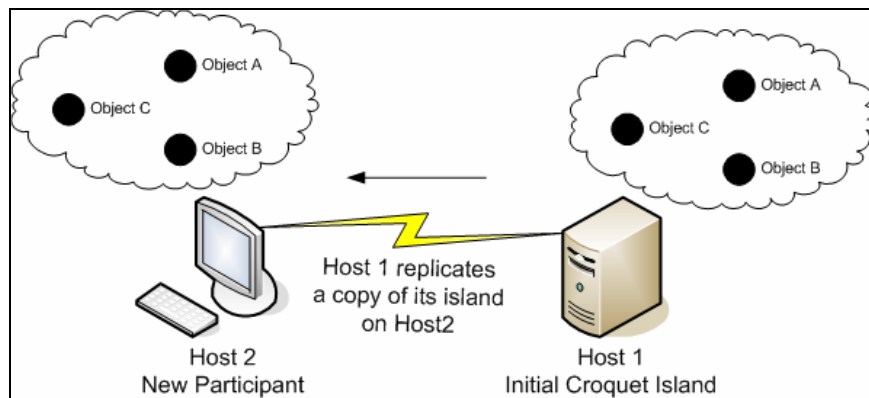


Figure 3: Island Replication

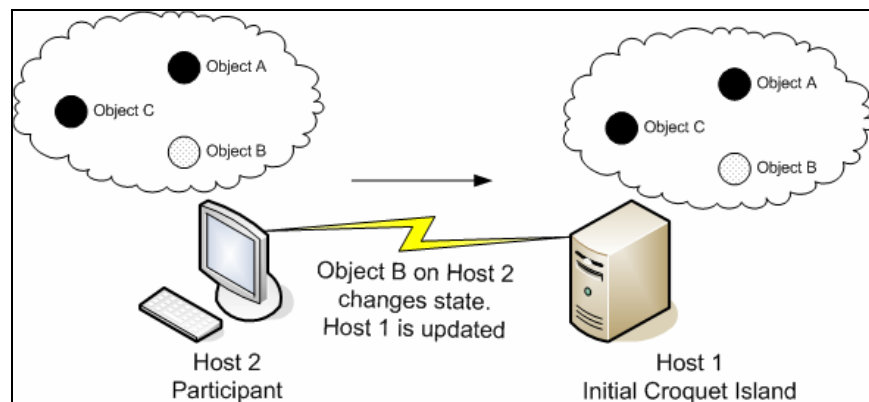


Figure 4: Maintaining Synchronicity between Island Replicas

Finally, the desktop-VR interface provided by Croquet makes interactions within the prototype vEOC more intuitive and natural. The use of VR in this fashion can offer advantages in terms of user interfaces, training, and

collaboration (Franco, J. F., et al 2006; Macredie, R., et al., 1996; Otto, O., et al 2006). Again, the idea is to enhance collaborative characteristics of the vEOC environment, thereby creating a better training/simulation experience.

TOUR OF THE VEOC

The prototype vEOC, which leverages Croquet's KidsFirst Application Toolkit (KAT) demo, may be broken down into three major components: fully synchronized mechanisms for collaboration (already built into Croquet), an XML interface to uniformly specify tests and simulations, and a three dimensional virtual environment with video and still image displays that can be programmed through the XML interface.

Collaboration Mechanisms

We have already reviewed Croquet's inherent synchronization capabilities provided by the TeaTime architecture. Nevertheless, this is such an important topic that it is worth reiterating briefly. Croquet's guarantee of synchronicity and atomicity among participants begins by ensuring that each participant's copy of the virtual environment is identical. Next, Croquet enforces a universal clock for participants by rigidly controlling and time-stamping all messages they pass. Such communications take place at various moments, including whenever an object in one participant's virtual environment changes state. Under this circumstance an update message is broadcast to all other participants, since their environments also contain the same object. The implication is that if all participants' environments are the same and begin execution from a known, synchronized state, then they can maintain their synchronicity through peer-to-peer communications that are regulated by Croquet's universal clock. Atomicity becomes a by-product of this system, implemented as a two-phase commit process such that a change is aborted for all island replicas if it fails to be applied in any one of them.

Croquet offers several other collaborative tools, two of which, text messaging and text whiteboards, are used by the prototype vEOC. Before looking at these further, however, it is important to note that Croquet's 3D virtual environment itself (based on OpenGL) is a collaboration mechanism by which participants' avatars can see and interact with one and other. Figure 5 shows us two avatars standing in the vEOC's main operations area. Because Croquet does not currently offer collision detection, it is possible for avatars to wander through the walls of the vEOC building and end up "outside."



Figure 5: Avatars within the prototype vEOC virtual environment.

Text messaging within Croquet is similar to other CVEs: participants may direct simple messages to one and other or to groups. Figure 6 shows the same two participants from Figure 5 engaged in a brief discussion; note that both participants joined the vEOC environment under the same, general moniker of "everyone."

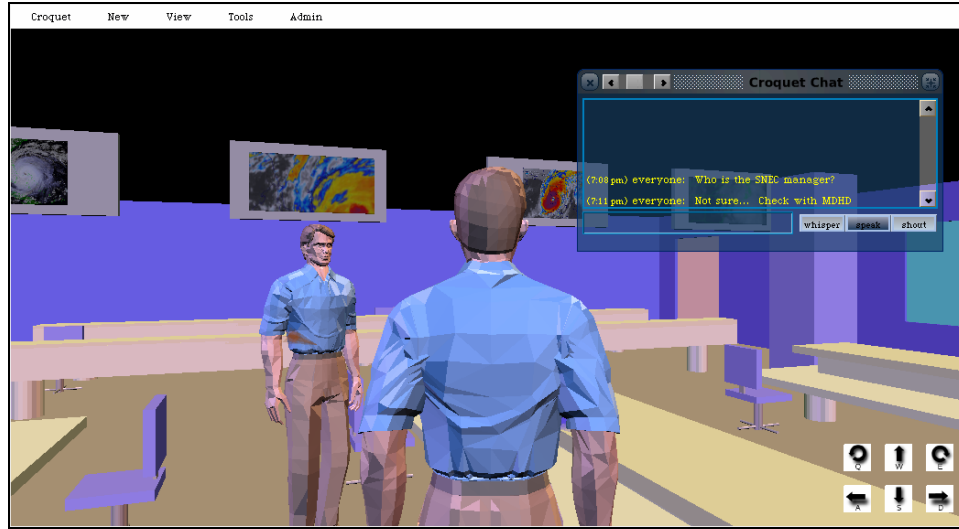


Figure 6: Participants in the vEOC using text chat.

The text whiteboard offers one convenient way for participants in the prototype vEOC to collaboratively track information. To use the whiteboard, a participant merely clicks on it and enters/edits text. Text may also be copied from programs outside of Croquet and pasted directly onto the whiteboard. Figure 7 demonstrates the text whiteboard in action (from the first-person perspective of one of the avatars).

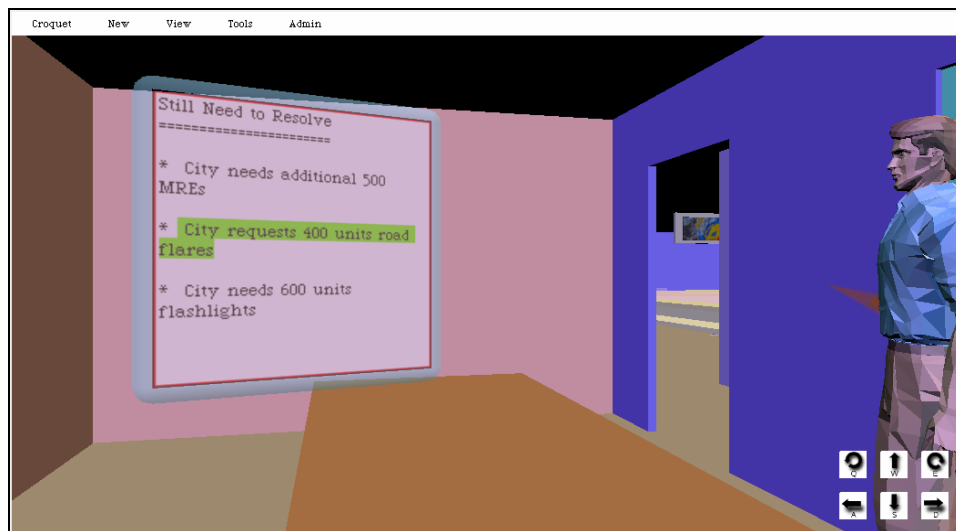


Figure 7: The Croquet text whiteboard, showing highlighted text.

The XML Interface

We elected to use XML as a means of describing vEOC tests and simulations in an accurate and uniform way, as well as to make these descriptions portable. At the present time, Croquet supports document type definitions (DTDs), but not XML schemas. Despite the limitations of DTDs (e.g., they are not written in XML and can specify only text data types for content [Sebesta, 2008]), we find the use of a vEOC DTD helpful in guiding the construction of test/simulation scripts. Also, the XML interface acts as a form of abstraction for the entire vEOC environment:

in future iterations of the prototype vEOC, an external game engine may feed a continuous, dynamic XML stream into Croquet.² In this fashion, simulations and tests that take advantage of external resources, such as agent-based modeling, could be implemented. Currently, however, the prototype vEOC follows a static, XML script to carry out tests and simulations.

The basic DTD is patterned after the injection script format used by the Miami-Dade EOC.³ Under this format, individual script items are viewed as messages, although a message may be information, an activity, or a state of some sort. Primary script fields include *time* (offset from the start of a test), *injection number* (unique integer denoting the sequence in which scripted items take place), *sender/receiver* of a message, *type of message* (e.g., information, resource request, incident), *message text*, and *expected action*. We have extended this format to accommodate timed displays of MPEG videos and GIF/JPEG images (see the following sub-section). The DTD is provided in Figure 8, while a sample XML script follows in Figure 9.

```
<!-- Document Type Defintion for vEOC Simulation Scripts -->
<!ELEMENT simulationScript ((msgRecord|rsrcRecord)+)>
<!ELEMENT msgRecord (time,sender,receiver,message,action,explanObj)>
<!ELEMENT rsrcRecord (time,device,file)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT sender (agency+)>
<!ELEMENT receiver (agency+)>
<!ELEMENT agency (#PCDATA)>
<!ELEMENT message (#PCDATA)>
<!ELEMENT action (#PCDATA)>
<!ELEMENT explanObj (#PCDATA)>
<!ELEMENT file (#PCDATA)>
<!ELEMENT device (#PCDATA)>

<!-- A message record includes a type attribute -->
<!ATTLIST msgRecord type
(infoRequest|infoUpdate|actionRequest|incident|resourceRequest) "infoUpdate">

<!-- Time consists of offset (from some starting point) and injection number attributes -->
<!ATTLIST time offset CDATA #REQUIRED>
<!ATTLIST time injectNum CDATA #REQUIRED>

<!-- A device consists of type (video or photo display) and ID number attributes -->
<!ATTLIST device type (video|photo) "video">
<!ATTLIST device devID CDATA #REQUIRED>
```

Figure 8: The vEOC simulation script DTD

² It should be noted that going outside of the Croquet environment for test or simulation directives creates synchronicity challenges. For example, a dynamic XML feed may experience network problems, such that one participant can access the feed while another cannot. This situation could disrupt Croquet's synchronicity and atomicity characteristics.

³ Injections are simply pieces of narrative that describe a test or simulation's activities and state. Inject messages, then, are provided to vEOC participants as a means of describing events that unfold.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulationScript SYSTEM "vEOC-Sims.dtd">
<simulationScript>
  <rsrcRecord>
    <time offset="5" injectNum="1"/>
    <device type="video" devID="1"/>
    <file>Content/vEOC/Movies/hurr-bertha-vid.mpg</file>
  </rsrcRecord>

  <msgRecord type="infoUpdate">
    <time offset="10" injectNum="9"/>
    <sender>
      <agency>Miami Beach EOC</agency>
    </sender>
    <receiver>
      <agency>Miami Beach Divisional</agency>
    </receiver>
    <message>City Mayor and commissioners want to confirm when tropical-storm-
force winds will arrive and which shelters will be open.</message>
    <action>Access IAP or contact Municipal Branch Director for
information.</action>
    <explainObj>4</explainObj>
  </msgRecord>
</simulationScript>

```

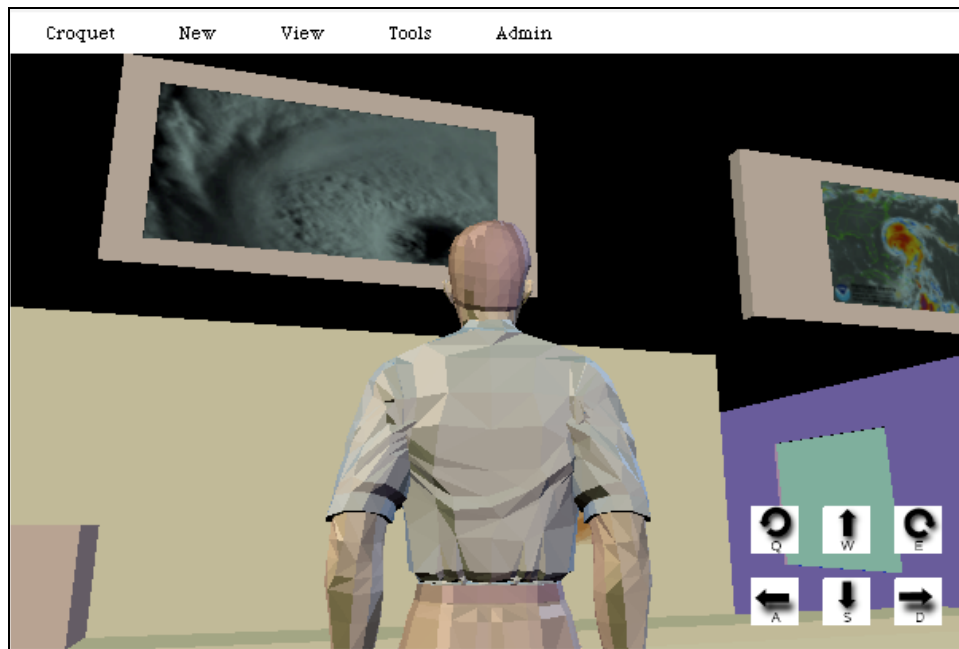
Figure 9: A sample vEOC XML test/simulation script.

When the prototype vEOC is initially executed, it reads an XML test/simulation script. As the script's directives are processed, video and image resources are loaded and stored until they are needed for display later. Similarly, test and simulation message injects are built and set aside for use at the appropriate times. In the current prototype vEOC, message injects are displayed outside of the 3D window (but within the Croquet environment); presently, there is no input solicited from a participant in response to a message. This is, of course, not desirable, and is intended as a simple placeholder for future functionality. Ultimately, participants must be able to react to a given message and have their response critiqued. Figure 10 shows an example message display.



Figure 10: Example message display.**Video and Photo Displays**

As already seen in the various figures preceding this sub-section, the prototype vEOC is equipped with video and image displays that hang down from the ceiling. During observations of a Miami-Dade EOC test activation in May 2007, it was observed that such displays were used to purvey a variety of information (e.g., news programs, weather and geographical information system (GIS) maps, and slide shows containing EOC team updates). We decided to incorporate eight of these displays (four MPEG displays and four JPEG/GIF displays) into the prototype vEOC. Each display is programmable via the XML test/simulation script facility: specific videos and still images can be loaded at the start of a test or simulation and then rendered at specific times and on specific displays. While the still images remain static (until they are replaced with different images), the MPEG videos may be started, paused, and re-started by participants clicking on a given display; when a video plays through to the end, it automatically rewinds to its beginning. Figure 11 shows an avatar looking up at the MPEG of a hurricane as seen by a weather satellite.

**Figure 11: Looking up at an MPEG display.****CROQUET RISKS**

Croquet's strong emphasis on collaboration and its OpenGL-based 3D environment make it a good choice for implementing a vEOC trainer/simulator. However, the use of Croquet is not without costs. Croquet offers an array of useful resources through its web site (www.opencroquet.org) and mailing lists (*croquet-user* and *croquet-dev*). Also, it comes out-of-the-box with several demonstration CVEs—the prototype vEOC was, in fact, built on top of one of these, which provide both example code and a starting place for budding Croquet developers. Unfortunately, the demos are poorly commented (or not commented at all, in some cases), and often have overlapping functionality that is not uniformly implemented. As soon as one steps off the path embodied by these demos, one must spend a significant amount of time unraveling and learning about the complex Croquet hierarchy of Squeak objects. This is often necessary because official Croquet documentation, though extensive, is still incomplete. Of course one can, and often must, appeal to the *croquet-dev* mailing list for expert help. Such help is offered on a volunteer basis, however, and may require some time to materialize. Other Croquet weaknesses include very limited support for

avatars with humanoid movements (e.g., walking, running, jumping), and incomplete support for content built in the Virtual Reality Modeling Language (VRML)—there is currently no support for Extensible 3D (X3D), VRML's successor. Avatar limitations can directly constrain the collaboration experience in Croquet, since avatars that are less realistic diminish the sense of immersion. Also, non-VRML/X3D content may be less portable and require learning complicated 3D graphics software to build.

Perhaps the most serious risk manifested in Croquet is the difficulty of using resources outside the TeaTime architecture. Synchronicity can be broken if, for example, a resource is available to one island replica but not another. This is significant to the endeavors of the vEOC since the use of existing, external simulation and database tools may be desirable. For example, Fiedrich and Burghardt (2007) note how agent-based simulation (ABS) may be used to create and analyze complex disaster scenarios, and as a means of decision support during a disaster response. Having to build agents within Croquet, as opposed to leveraging pre-existing ABS technology, may be tedious, costly, and contrary to good software engineering practice. In fact, this issue may be compelling enough to warrant consideration of CVE options other than Croquet (e.g., the Java-based Wonderland system). Weaker synchronicity/atomicity could be tolerated in exchange for greater integration and support of outside resources.

It is important to note that Croquet is billed as a software development kit, not a polished system for the masses. Understanding Smalltalk is a requirement for doing anything meaningful with Croquet; this alone can be a challenge for those raised on a diet of languages that are not truly object-oriented.

CONCLUSIONS AND FUTURE WORK

Within the realm of emergency management training and simulation, there is a growing precedent to leverage collaborative virtual environments. Such environments enable the creation, customization, and re-use of scenarios, as well as the ability to support multiple participants. This can be a potent tool for EOC teams that look to collaborative training as a critical ingredient of success. However, many virtual environments are aimed at participation on a LAN and may not support synchronized and atomic activities among distributed participants. For EOC training and simulation, this can have a negative impact on collaboration. In response to this problem we have built a prototype virtual EOC based upon the Croquet software development kit. Our prototype takes advantage of Croquet's TeaTime architecture, which guarantees synchronicity and atomicity among all participants—regardless of their proximity to each other. Moreover, the prototype vEOC makes use of additional Croquet collaboration tools such as text chat, text whiteboards, and general 3D capabilities. We have also added an XML interface as a convenient, uniform way to accurately describe tests and simulations. Finally, within the vEOC we have implemented video and still image displays that are programmable via the XML interface.

As a first prototype, our current vEOC implementation demonstrates only the most rudimentary capabilities required to test and simulate EOC scenarios. Clearly, this is a long way from being a useful training tool. For example, the interaction between participants and scenario messages still needs to be implemented: presently, messages are displayed to participants at times that are defined in the XML test/simulation script, but there is no response or evaluation mechanism. There are also no specific roles for participants to play during a test or simulation. Presumably, when roles are put into effect, messages can be directed to specific participants as opposed to everyone. Other important EOC functions are missing at the moment, too; for example there is no GIS capability. Finally, administrative and account management tools need to be devised, in particular with regard to creating and editing the vEOC test/simulation scripts, as well as setting up role-based participant accounts.

ACKNOWLEDGMENTS

The authors wish to acknowledge the support of the National Science Foundation under grant CNS050348. Also, the authors acknowledge the collaboration and support of Professor Irma Becerra-Fernández, Florida International University, Frank J. Reddish, Emergency Management Coordinator, and C. Douglas Bass, Director, the latter two of the Miami-Dade Office of Emergency Management and Homeland Security.

REFERENCES

1. Becerra-Fernández, I., Madey, G., Prietula, M., Rodríguez, D., Valerdi, R., and Wright, T. (2008) Design and development of a virtual emergency operations center for disaster management research, training, and discovery, *Hawaii International Conference on System Sciences (HICSS-41)*, Waikoloa, Hawaii.

2. Bowers, F. A. III and Prochnow, D. L. (2003) Simulation for emergency response: JTLS-JCATS federation support of emergency response training, *Proceedings of the 35th Winter Simulation Conference*, New Orleans, Louisiana, 1052-1060.
3. Chakrabarty, M. M. and Mendonca, D. (2005) Design considerations for information systems to support critical infrastructure management, *Proceedings of the 2nd International ISCRAM Conference*, 13-18.
4. Croquet Consortium (2007a) About the Technology, Found on the Web at [www.opencroquet.org/index.php/About the Technology](http://www.opencroquet.org/index.php/About_the_Technology)
5. Croquet Consortium (2007b) System Overview, Found on the Web at [www.opencroquet.org/index.php/System Overview#TeaTime Architecture](http://www.opencroquet.org/index.php/System_Overview#TeaTime_Architecture)
6. Currión, P., Silva, C. d., and Van de Walle, B. (2007) Open source software for disaster management. *Commun. ACM* 50, 3, Mar., 61-65.
7. Department of Homeland Security (DHS), Federal Emergency Management Agency (FEMA) (2007) *National Incident Management System—FEMA 501/Draft August 2007*, 3.
8. Fiedrich, F. and Burghardt, P. (2007) Agent-based systems for disaster management. *Commun. ACM* 50, 3, Mar., 41-42.
9. Franco, J. F., da Cruz, S. R., and de Deas Lopes, R. (2006) Computer graphics, interactive technologies and collaborative learning synergy supporting individuals' skills development, *International Conference on Computer Graphics and Interactive Techniques*, ACM SIGGRAPH 2006 Educators program, New York, NY, 42.
10. Hendela, A. H., Turo, M., Yao, X., Hiltz, R., and Chumer, M. (2006) Virtual emergency preparedness gaming: A follow-up study, *Proceedings of the 3rd International ISCRAM Conference*, 450-459.
11. Jain, S. and McLean, C. R. (2005) Integrated gaming and simulation architecture for incident management training, *Proceedings of the 37th Winter Simulation Conference*, Orlando, Florida, 904-913.
12. Jain, S. and McLean, C. R. (2006) A Concept prototype for integrated gaming and simulation for incident management, *Proceedings of the 38th Winter Simulation Conference*, Monterey, California, 493-500.
13. Loper, M. L. and Presnell, B. (2005) Modeling an emergency operations center with agents, *Proceedings of the 2005 Winter Simulation Conference*, Orlando, Florida, 895-903.
14. Macredie, R., Taylor, S. J. E., Yu, X., and Keeble, R. (1996) Virtual reality and simulation: an overview, *Proceedings of the 28th Winter Simulation Conference*, Coronado, California, 669-674.
15. McGrath, D. and McGrath S. P. (2005) Simulation and network-centric emergency response, *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, Orlando, Florida.
16. National Research Council (NRC) (2002) *Making the Nation Safer: The Role of Science and Technology in Countering Terrorism*, National Academy Press, Washington, D.C., 316.
17. Otto, O., Roberts, D. and Wolff, R. (2006) A review on effective closely-coupled collaboration using immersive CVE's, *Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and its Applications*, New York, NY, 145-154.
18. Pollak, E., Falash, M., Ingraham, L., and Gottesman, V. (2004) Operational analysis framework for emergency operations center preparedness training, *Proceedings of the 36th Winter Simulation Conference*, Washington, D.C., 839-848.
19. Sebesta, R. W. (2008) *Programming the World Wide Web, 4th Edition*, Pearson Addison Wesley, Boston, 272-289.
20. Smith, R. (2003) The Application of existing simulation systems to emerging homeland security training needs, *Proceedings of the European Simulation Interoperability Workshop*, Stockholm, Sweden.