MirrorWorlds: Virtual Worlds for Team Training

Dana Moore
Raytheon BBN Technologies
Rosslyn, VA
damoore@bbn.com

Michael Thome
Raytheon BBN Technologies
Cambridge, MA
mthome@bbn.com

ABSTRACT

Training and simulation innovators in the DoD are beginning to explore the substitution of Virtual Worlds (VWs) into the space traditionally occupied by large-scale trainers and simulators. Moreover, some services (notable the U. S. Air Force) are beginning to look toward VWs for operational use. VWs are significantly cheaper to author and create content for; easier to proliferate to a wider audience; and cheaper to maintain and support both in terms of hardware and software. This has positive implications not only for team training, mission rehearsal, weapon system-specific training, and after action review, but also for mission operations and control.

One area where VWs are in their infancy is in their ability to link to live data feeds, affect events in the real world, and allow rich bi-directional interaction with external resources. The ability to incorporate dynamic simulation elements such as sensors (real and simulated), semantic tagging of objects, and convincing virtual role players will greatly enhance the richness and utility of military training research and engineering offerings to their customers in both kinetic and non-kinetic domains.

In this paper we describe our recent and ongoing work in developing a general approach to incorporating externally driven behaviors and events into virtual worlds to give VW authors access to a much richer set of simulation elements and our progress in incorporating sensing (simulated and real) into a generalized architecture and reference toolkit. This work—coupled with the existing Second Life affordances for voice communications, rich media inclusion, rapid content creation, server intermediation, content distribution and replication, content scripting, and creation of virtual role players—offers the promise of tactical training that can be extended in both scale and scope and offered on increasingly distributed, lightweight platforms., and may offer new capabilities for mission rehearsal, after-action review, and tactical operations.

ABOUT THE AUTHORS

Dana Moore (*ElectricSheep Expedition* in Second Life) is a Lead Scientist with BBN Technologies in Arlington, Virginia and is a Principal Investigator for the ONR LTSN Program leading an effort to create Misson-Aware Sensor Fields. Dana holds an MSc. from the University of Maryland and is a leading authority on Second Life scripting, co-authoring and Scripting Your World: The Official Guide to Second Life Scripting, and author of several books on peer-to-peer (P2P) and collaborative computing. Prior to joining BBN, Dana was Chief Scientist for Roku Technologies, designing an extensive commercial P2P system for 3COM, and Hewlett-Packard. The system, based on automated understanding of contextual meta-data resulted in several U.S. technology patents.

Michael Thome (*Vex Streeter* in Second Life) has been a computer scientist with BBN Technologies for over 20 years, and currently specializes in virtual worlds and scalable computing technologies including parallel, distributed, and agent-based computing, most recently in the air transportation planning and global logistics problem domains. Michael has degrees in Cognitive Science from the University of Rochester and Boston University, with specialties in computational models of human neurological systems. He has been active in Second Life-based software development since 2006 and is co-author of Scripting Your World: The Official Guide to Second Life Scripting.

MirrorWorlds: Virtual Worlds for Team Training

Dana Moore
Raytheon BBN Technologies
Rosslyn, VA
damoore@bbn.com

Michael Thome
Raytheon BBN Technologies
Cambridge, MA
mthome@bbn.com

Introduction

Virtual worlds that mirror actual tactical areas, (which we call "MirrorWorlds" to differentiate them fantasy worlds such as found in games such as "World of Warcraft®) have attracted the attention of and are now employed by the military in numerous ways. Typical corporate activities such as acquainting new civilian and uniform personnel of policies or facility capabilities (often referred to as "onboarding") are quite common. Procedural training has been demonstrated to be an effective component of the Live-Virtual-Constructive training mix.

The Air Force Education and Training Command for example have gone so far as to assert (*ND2010*) that virtual worlds represent a consolidated approach for recruiting, training, and operations. Given the nature of the Air Force command and control system, this approach seems a natural fit for this service whose kinetic operations are largely carried out electronically: A recruit can be trained to guide drones to training targets for example, and that skill set directly transferable to active operations; the command and control loop can be directly plugged into the fabric of the virtual world environment.

Consider this in contrast to the services with a more kinetic operational profile, those such as the Marines and Army with a more "boots on the ground" modus operandi. Is it possible that training and then operating via a virtual world might also offer tactical advantage? This is the question our team wished to address in designing a virtual world based framework. We asked ourselves whether it might be possible, as a design and research goal, to build a training environment that would energize team operations in the same way that networked massively multi-player or MMO games can, but also transcend first- and third-person shooters wherein it's not possible to stray outside the designer's script for encounter and interaction. As a response to these design and game goals as further elaborated in the next section, we designed and implemented a counter-insurgency trainer (*Openmap08*) using the Second Life (SL) VW as the implementation platform; this paper documents the design, architecture, implementation and some results and observations based on user testing.

A Motivating Problem

A common theme that emerges both in the enterprise and in the military is the acknowledgment that teamwork amongst peer groups, while always important, has become even more critical. Much training has traditionally concentrated on optimization of individual competency and performance, and it can often be done with live equipment or in the physical world, but as pointed out elsewhere (*Process09*)

"*Live training has always been the method of choice for training soldiers. As the lethality, expense, and complexity of modern weapon systems has increased and training budgets have tightened, live training is no longer sufficient as the sole training method (ATSC 99).*"

We might add that live training apart from being expensive and insufficient for team and coordinated group training, is difficult to evaluate. Hot washes and group after-action discussions are almost always highly subjective and it it difficult to develop good measures of performance or effectiveness that address how the team performs *as a team*. This is in contrast to individual training on specific equipment where quantitative measures are often sufficient.

Nowhere is team performance more critical than in urban operations or UrbanOps. In physical world UrbanOps, there may be multiple squads operating in parallel in an area of interest, tiering up human gathered and sensor derived intelligence to the company level. A comprehensive common operational picture (*COP06*) is difficult to derive in such a context using traditional flat GIS style interfaces and the advantage often goes to the adversary.

Figure 1: Scene from "Chicken Chase" Counter Insurgency trainer

Design Desiderata

In our concise description of the experimental goals for our successful submission to the federal Virtual Worlds Challenge (*FVWC09*), we explained it this way:

"Intelligence analysts at the Battalion/ Company level require training to master the sorting and collating of information from multiple sources, and the distilling and sharing of their knowledge and inferences out to platoon and squad level. The game ("Chicken Chase") provides a multiplayer game environment in the Second Life universe where players can exercise all of these skills. The game challenges individuals to work as a team to monitor an insurgent-ridden area, observe the location of IEDs, and beat their opponents to retrieving them. The chickens are difficult to distinguish from one another; and there are many of them. There is some intelligence to their behavior, but it is difficult to decipher. The sensors provide a continuous stream of information, but there are both too many and not enough"

This was written in advance of the actual coding and implementation, but clearly defined in our minds a set of necessary design and game elements described next.

Design Decisions and Game Elements

In response to a perceived need to examine virtual worlds as a way to improve multi-participant, multi-echelon training for urban operations, we created a virtual environment situated in, and mirroring the conduct of urban operations.

We wondered whether creating an interface that added the third dimension might enhance prospects for portraying sensed and human gathered intelligence and aid understanding. This became our first design objective.

Secondly, we wanted to explore how difficult it might be to incorporate social networking capabilities into operations and how much they might affect team effectiveness. Finally, we wanted to understand whether it would be possible to enhance in a novice population's understanding and use of the multiple capabilities inherent in a prepared UrbanOps environment. If this could show positive results, then it might auger well for proposing this MirrorWorlds approach to DoD innovators such as RDECOM/STTC or PEO-STRI.

In reviewing these objectives, we began to develop a catalog of the design elements we would have to incorporate:

- Peer (e.g., Squad to Squad) team communications, both in world and interoperating with modern social networking fabrics that US forces might use in the field (e.g., Twitter).
- Cross-echelon (e.g., Squad to Company) communications, as above.
- Representation of insurgents and the threats posed that might typify those found in urban operations; to entice non military users and novice players, we decided to make the insurgent population as superficially innocuous as possible, thus we embodied them as chickens which lead to titling the game and environment "*Chicken Chase*".
- Believable behaviors exhibited by the insurgent population. As in a real UrbanOps environment, behaviors would entail normal day to day activities typical for the population intermixed with hidden and nefarious agendas.
- Compelling threats. In this case we decided to make detection of egg laying a surrogate for placement of improvised explosive devices (IEDs)
- Fungible surrogates for measures of performance or real measures of mission success. In this case, individuals and team competed for prizes and high scores.
- Live sensors embodied virtually, emulating real traditional sensors traditionally placed by others (e.g., Sensor Control and Management Platoons), We also needed to find a way for sensor detects in the real world to affect game play.
- Virtual sensors that team can place themselves, thus emulating a new generation of "organic" sensors, those which are carried with and can be emplaced by the squad member operating in situ (*LTSN, 2007*)
- Conflicting information disinformation

Platform Justification: Benefits of Second Life

In reviewing these objectives, we began to understand the advantages of an open unscripted virtual world platform (versus a first- or third-person shooter platform such as VBS-2 or RealWorld) to incorporate dynamically created content and participant free play.

Second Life® (*SL*) is well known as a social MMO support grid, on which it has been possible to build multiple simulations, from dance clubs to green 3D virtual tours of locales world wide. We determined that all known virtual environment platforms and games generally support a sense of shared mission context and a measurable "place proprioception" (sense of virtually physical location, orientation, movement, and structure). However, we believe that, owing to its well-designed framework, supported by VOIP, rich media (e.g. video-in), and many other attributes, SL was far superior to other frameworks as implementation platform.

Additionally, and more importantly in achieving the design desiderata in emulating a tactical domain, we determined that SL was a highly adaptable platform which demonstrates efficacy in supporting multiple user communities and use cases. Important in achieving our design goals were:

1. Real-time distance/distributed collaboration, model building, analytical work production
2. Support for the concept of an actual group or team.

3. Training and simulation: Short-term Tactical; Analytical; Procedural and policy; Leadership; Medium intensity long duration training
4. Social Contexts: ideation and brain surfing in non challenging environment
5. Intelligence operations: Active operations, COUNTERPROINTEL, recruitment,
6. Command Operations Centers (COCs): real AOI sensor augmented mission C2; enhanced remote interfaces.

In addition, SL is one of a very few environments that supports in world "art path" and content creation. It is quite possible to watch skyscrapers and Afghan villages rise *ex nihilo* into full-blown realizations of areas of interest. Further, there is built in support for:

- Inter-object communication (in this case, between in-world sensors and team HUDs; insurgent (chicken-to-chicken) intelligence sharing,
- Real-time physics,
- Communications with external services (in this case Twitter and the Web). This includes web services proxying real sensors (in this case, certain elements of game play are determined by ambient weather conditions in the real world) and video feeds. It also includes communications with external knowledge stores (in this case MySQL and Parliament, a semantic database).

All of these platform capabilities were useful in creating *Chicken Chase, a Game of Counter-insurgency* to life, in a single remarkably short development spiral.

Game Concepts

The supporting web site and in game materials dispensed by the in world HUD dispenser explained Chicken Chase as a collaborative team-based multiplayer game of stalking and puzzling that combined both real world and virtual world (Second Life) elements and data/information streams (twitter, webcams).

Modeling Opposing Forces

It was explained that participants must explore an urban setting in Second Life and observe the behavior of a number of "paranoid hens" who make their way through the streets and buildings, occasionally laying golden eggs in the darker corners of the city. Here, we hoped to give participants a feel for what it's like to operate in an alien context, where the opposing force is essentially unknowable. Chickens, it was hoped would be a suitable surrogate for an inscrutable alien populace whose goals and objectives could be observed and catalogued, but never really known, much like current operational constraints. There was some debate in the design phase about whether to implement interrogation; in the interest of fielding a working test bed for our fundamental goals in a short development spiral, the idea was rejected.

IED Surrogates

Participants learned that the game objective was for participants to accumulate as many eggs as possible for the team. Eggs could be obtained by touching them in the virtual environment. Some eggs contained virtual prizes as added incentive. It was also explained that like real eggs, the game virtual eggs had a shelf life, albeit somewhat shorter than in the real world – eggs disappeared after some period of time. The teaching objective of the eggs was to help participants think about

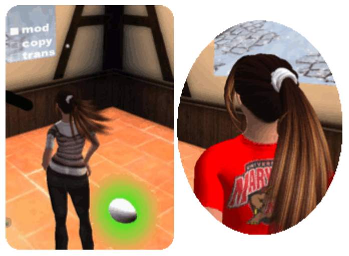


Figure 2: (a) Discovering IED surrogate, (b) wearing prize

how best to collaboratively detect items of interest in a specific environmental context.

Figure 2 shows a player finding an egg, and then donning their prize.

Sensors

The game attempted to give users a feel for working with sensors in situ. Although traditional doctrine says that sensors are placed by and for the benefit the company intelligence function, attitudes are changing such that organic sensors (i.e., travel with and emplaced by operating squads) for blue force protection or situation awareness are being seriously considered.

We devised a virtual sensor that corresponded in many ways to real sensors. For example, only a portion of a player's sensors can be powered at any given time, and sensors automatically shut themselves down after fifteen minutes of continuous operation, and disappear after one hour. This was an attempt to emulate sensor duty cycle, and educate users on the limits of networked sensing. In the game, users could select to receive Twitter notifications (tweets) when sensors shut down just as they received tweets from sensors about interesting nearby chicken events. The choice of which sensors should be active could be controlled in-world or through Twitter or the web. Sensor placement is manipulated only within the virtual world.

We also incorporated real world sensors, using the CitySense (*CitySense* 2007) array as surrogate for sensors placed by SCAMP or other Company level assets. The CitySense array is deployed all around Cambridge, MA and detects primarily atmospheric health in the urban domain. The detection interval is on the order of a few seconds, which was judged as sufficiently dynamic to affect game behavior. These sensors actually correlated to insurgent behavior. If the real world weather tended toward windy or rainy, the chickens uniformly would flock or defer egg laying activities (the surrogate for IED emplacement). We attempted in this way to help users become attuned to social (sometimes called "atmospheric") changes in the overall population throughout the AOI.

We suggested some elements of game tactics, such as, *"One approach is simply to chase the chickens around, but they're very touchy, so this doesn't generally work well. Because the hens are so wily and elusive, each user can employ some number of chicken-sensors that are capable of reporting in Second Life, through Twitter or the Web when a chicken has passed by, or when a nearby chicken has laid an egg. Naturally, chickens aren't afraid of sensors the way they're afraid of people."*

Note Taking, Spreading (Dis)information

We also designed in a capability for participants to "mark up" the virtual world with graffiti applied to buildings that can be seen by their teammates – notes about what they've seen or sensed at a particular location. These graffiti could be placed anywhere in-world directly by the user, or indirectly through Twitter or the web at any of the user's sensor locations.

Individual and Team Exploration

As participants explore the virtual world and the communication channels between virtual-world sensors, Twitter, and the web, it is hoped that they will notice other connections between the virtual world and the real world: weather, traffic, live video streams, and chicken behavior. When it rains in Cambridge, it rains in the virtual world. Outside temperature and wind conditions in Cambridge have effects on where chickens go and the places where they are most likely to lay eggs. A BBN employee entering or exiting the main corporate campus (on a scrubbed-but-live video feed viewable in the virtual world or on the web) may cause a chicken to emerge from a particular building in the virtual world *N* (ten) seconds later.

The ultimate goal of the game from the players' standpoint is to collaborate with teammates (both in the virtual world and electronically through the Web) to combine their knowledge and sensor data to accumulate as many eggs as possible for the team. Team scores are updated in real time to the game site (*Score*) as game play continues over long periods of time. One of the advantages of an open game environment such as SL is that well designed games can continue to unfold in real time regardless of the episodic involvement of human players.; this is in contrast to typical game environments that exist only as long as the player's engagement.

Implementation

The game implementation used a combination of external web servers or proxies and the internal Second Life scripting language (Linden Scripting Language or LSL). LSL is an event driven language (see *LSLBook08, LSLBook09*) that enables proactive (scheduled or asynchronous) and reactive (to user interaction, other objects, external events) automation; physics emulation; environmental controls (both virtual and acting as control surfaces for physical world systems); Integration of multimedia, and communications with the outside world. It is uniquely suited to the demands of a dynamic MMO environment and was a critical success factor in our ability to create the environment. An LSL script encodes a Finite State Machine (*LSLWIKI*), with callbacks for the range of possible triggering events. Exhibit 1 shows the simplest possible script. Exhibit 2 shows a fully functional script as used in the game.

This simplest of scripts illustrates some elements of script structure. First of all, scripting in SL is done in the Linden Scripting Language, usually referred to as LSL. It has syntax similar to the common C or Java programming languages, and is event-driven, meaning that the flow of the program is determined by events such as receiving messages, collisions with other objects, or user actions.

```
default
{
    state_entry()
    {
        llSay(0, "Hello, Avatar!");
    }

    touch_start(integer total_number)
    {
        llSay(0, "Touched.");
    }

}
```

Exhibit 1: The LSL "Hello World" Script

LSL has an explicit state model, and models scripts as finite state machines, meaning that different classes of behaviors can be captured in separate states, and there are explicit transitions between the states. At a bare minimum, a script must contain the `default` *state*, which must define at least one *event handler*. Scripts may contain additional states, each of which must define at least one event handler. Scripts may also contain user-defined functions and global variables.

LSL has some unusual built-in data types, such as vectors and quaternions, as well as a wide variety of functions for manipulating the simulation of the physical world, for interacting with player avatars, and for communicating with the real world beyond SL. Multiple scripts are often used in a single in-world entity (e.g., the insurgents in the game have eight separate scripts representing eight finite state machines or FSMs). Although scripts in the same entity may interact, each runs its own event triggers and handlers. Performance characteristics for event handling are well known and documented by Linden Labs (*LSLWIKI*).

External services were written in Python and offered a typical Web 2.0 web services API as CGI scripts. Because LSL was limited in its ability to absorb large amounts of HTML outputs, proxies were used to facilitate communication between the outside world and the game.

Game Environment

The Game environment was created on a portion of a standard Second Life region and consisted of a typically urban cityscape (shown in Figure 1 above) used to represent the area of interest (AOI). Informational billboards explained game play so that no external game manual was necessary. In addition, a complete set of game instructions, hints, and insights were maintained online (*Openmap08*) No access restrictions were applied as we sought to encourage as many teams as possible to compete. An urban environment was laid down hand using standard "in world" construction tools from both custom and off the shelf parts such as buildings and vehicles. All controllers and HUDs span of control encompassed the entire 65,536 square meters of the region. The insurgents' range of communication encompassed the same area, thus emulating the real world ability to use mobile communications. For example, one chicken might report that it had detected a blue force member (i.e., player) in a specific area and cause changes in game play

Game Controller

The game controller is implemented as an in-world object, managing communications between all game entities including insurgents and human users as well as the virtual sensors they place in the game space, acting essentially as a communications router. The controller connects to the external game data base via outbound HTTP connection (using a fixed URL), and with internal objects (players' HUDs, etc) via normal in-game broadcast messaging.

The controller's responsibilities include checking to assure that the maximum number of sensors allowed per team has not been exceeded, "rezzing" or instantiating a requested sensor, telling the sensor to move to the requesting player's geo-coordinates, and setting up a specific channel for future communications between player control objects (e.g. the user's game HUD) and the sensor. Figure 3 depicts a player placing a virtual sensor.

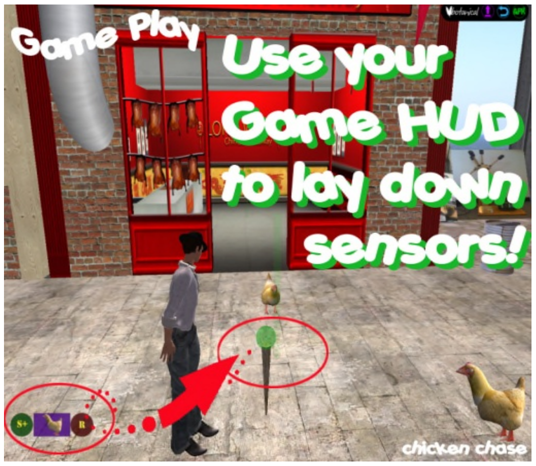


Figure 3: Player deploying virtual sensor

Game HUD

Users were issued a game HUD that allowed placement of virtual sensors, reporting of status of all the sensors, and other team affiliation activities. Figure 3 above and Figure 4 shows the game HUD.

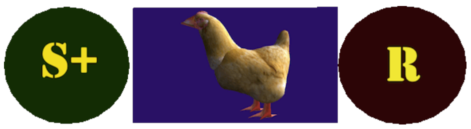
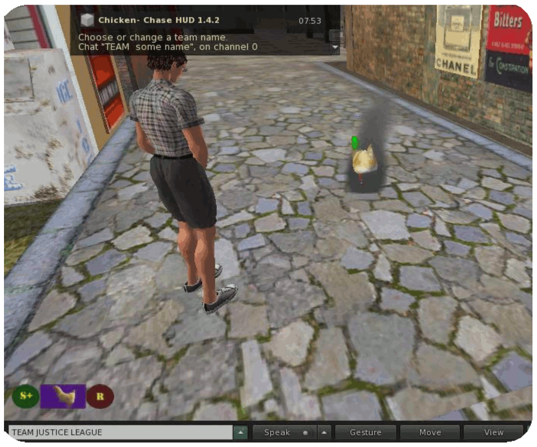


Figure 4: Game Control Head Up Display (HUD)

The "S+" button allowed a user to lay down a virtual sensor, the Chicken icon, to create a team, and the "R" button reports the status of all sensors laid down by the team.

In Figure 5 below, a player has clicked on the HUD Chicken icon and is being asked to reply in normal text chat with a team name, which he has done.



Chicken- Chase HUD 1.4.2 07:53
Choose or change a team name.
Chat "TEAM some name", on channel 0

TEAM JUSTICE LEAGUE

Figure 5: Player sets up team name

SL makes provisions for HUD creation relatively the same as any other scripted object. When a HUD is in the inventory of a player, one clicks on it and selects "attach to HUD" position. As seen in Figure 5 above, the user has chosen screen lower left as the HUD position.
The HUD uses an outbound HTTP communications channel to create the team name on the game site (*Score09*) and to add to the score when IEDS (eggs) are discovered and touched (as surrogate for deactivation).

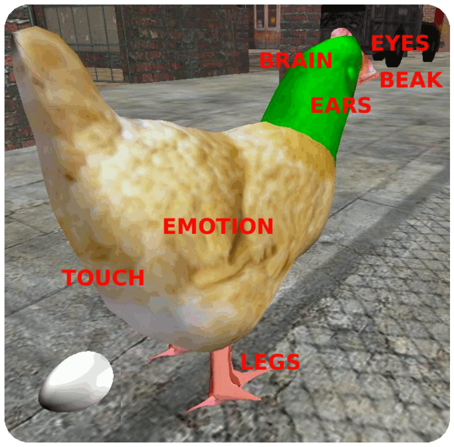


Figure 6: Actions controlled by insurgent AI

Scripted Object Rezzer

A single object handles the creation of all in-world scripted game objects: chickens and eggs through sensors and graffiti objects. It creates instances of (or *rezzes*) the various objects required for game play. Some objects such as chickens are instantiated according to a simple timer. Others are created on depend as a result of messaging from the game controller. All objects are created at a central point near the rezzer and then moved to their target location before relinquishing control to the scripts in each type of object.

Insurgents

Insurgents were modeled as chickens who appear at random and follow what seem to human observers a random path walking through the game environment and doing normal "chicken" activities. Independent state machines handle the various functions shown in Figure 6.

The artificial intelligence (AI) necessary to run a chicken was all written in LSL, but in future versions where human-like actors are anticipated, a combination of server based AI and in game will be used.

The **Brain** module handles navigation through the game space and coordinates all other chicken actions. Although the circumnavigation of the game space may appear random to a casual human game participant, the insurgents are "wire guided", their path through the game space by following a paths defined by a map of the AOI. The brain then chooses between options at each waypoint based on simple heuristics. Each waypoint may be labeled as a simple transit point, a feeding area where the chicken will linger for a while, a nesting zone where it might lay an egg or a death trap such as the neighborhood restaurant.

The map is acquired via HTTP at chicken instantiation time and the chicken will traverse the map until it expires.

The **Legs** module handles locomotion through the map, taking instructions from the brain as to the next goal and autonomously handling collisions with players, steps and the like.

The **Eyes**, **Ears**, and **Touch** functions sense player avatars, signaling the brain so that it can react via behavioral changes. The **Emotion** function causes the chicken's internal state to be visible to the observer, emitting a trail of smoke when "suspicious" of players in the area, or more rarely to explode.

Finally, the **Beak** module both emits clucking sounds audible to players, and sends messages to the Twitter gateway, informing subscribers to the "Chicken Feed" what each insurgent is up to.

Sensors (virtual)
The virtual sensors specifically detected passing chickens and their activities in the area, thus emulating a combination of IR tripwire sensors and imagers. An accommodation was made in the design of the sensors to be smart enough to ignore proximate blue forces in but detect indigenous populations.

Sensors (physical)
Physical sensors were mediated via a Python proxy. Each in-world virtual sensor representation polled the real sensor at 30-second intervals. This pull notification is probably non-optimal, but was sufficient for the small number of real sensors used experimentally. A likely improvement can be gained by using an in-world HTTP Server to act as a push notification proxy for all the virtual instances of real sensors in the Second Life region. Owing to the CitySense architecture, which does not do data push; an external Python proxy was still required.

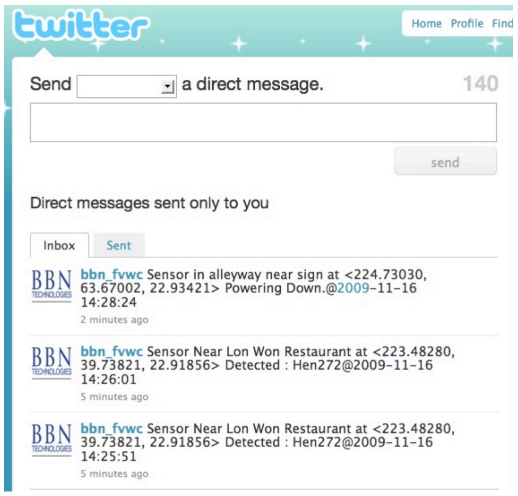
Twitter
An external Python proxy was used as the target of "tweets" outbound from sensors or insurgents within the game. The proxy enabled user registrations, direct message, and private message send-and-receive via XML over HTTP, typical of Web 2.0 AJAX applications.

Thus a typical outbound message from LSL would resemble Exhibit 2 (which also serves

```
tweet(string action,list params) {
 string x = "http://<server>/cgi/TwitProx.cgi?action=" + action;
 integer i;
 integer l = llGetListLength(params);
 for (i=0; i<l; i+=2) {
   x += "&"+llList2String(params,i)+"="+llEscapeURL(llList2String(params,(i + 1));
 }
 llHTTPRequest(x,[],"");
}
default {
  touch_start(integer n) {
    tweet("register",["user","USR","password","PASS"]);
    tweet("post",["to","bbn_fvwc","message","cluck-cluck"]);
    tweet("directmessage",["to","bbn_fvwc","message","touched by"+llDetectedName(0)]);
  }
  http_response(key req, integer status, list metadata, string body) {
    llOwnerSay("Response from Twitter is "+body);
  }
}
```

Exhibit 2: Twitter Handler LSL code

to illustrate the succinctness of the language). Notice that outbound HTTP request done in the `tweet` function are part of the non event-handling auxiliary function, but `http_response` is a built-in asynchronous event handler. Figure 7 shows the result of virtual sensors "tweeting" to a game player's twitter account.



bbn_fvwc Sensor in alleyway near sign at <224.73030, 63.67002, 22.93421> Powering Down.@2009-11-16 14:28:24
2 minutes ago

bbn_fvwc Sensor Near Lon Won Restaurant at <223.48280, 39.73821, 22.91856> Detected : Hen272@2009-11-16 14:26:01
5 minutes ago

bbn_fvwc Sensor Near Lon Won Restaurant at <223.48280, 39.73821, 22.91856> Detected : Hen272@2009-11-16 14:25:51
5 minutes ago

Figure 7: "Tweets" from in-game Sensors

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the contributions to the design and implementation of the experiment of our colleagues Dr. Bruce Roberts and Kerry Moffit. We also owe a huge debt of gratitude to Tami Griffith (RDECOM/STTC) for her tireless support and advocacy of virtual worlds in the DoD and the Federal sector in general.

REFERENCES

Citysense09: CitySense - An Open, Urban-Scale Sensor Network Testbed: http://www.citysense.net/

COP06: http://proceedings.esri.com/library/userconf/proc06/papers/papers/pap_2220.pdf

ND10: National Defense, April, 2010. "Airmen to Live Out Their Careers in Cyberspace". http://www.nationaldefensemagazine.org/archive/2010/May/Pages/AirmentoLiveOutTheirCareersInCyberspace.aspx

FVWC09:http://fvwc.army.mil/FVWC-Main.html

LSLBook08: Scripting Your World: The Official Guide to Second Life Scripting D. Moore, M. Thome, and K. Z. Haigh Sybex Publishing, 2008

LSLBook09: http://syw.fabulo.us

Openmap08: http://openmap.bbn.com/ChickenChase/

SL: Second Life home URL: Secondlife.com

Soc09: http://twitter.com/peosoldier

Soc10: Marines Ban Twitter, MySpace, Facebook http://www.wired.com/dangerroom/2009/08/marines-ban-twitter-myspace-facebook/

Soc2-10: Soldiers in Afghanistan use iPhone app to battle Taliban: http://www.phonedog.com/2009/12/29/soldiers-in-afghanistan-use-iphone-app-to-battle-taliban/

Score09: http://openmap.bbn.com/ChickenChase/cgi/TeamPage.cgi?action=show

SL: secondlife.com

Process99: Processes, and Products. The US Army Training Support Center, (1999). Proceedings of the Second Training Effectiveness Symposium, Hampton, VA.