Leveraging HTML5 and WebGL to Address Information Assurance Barriers for Simulation Based Training in the U.S. Military

Douglas Maxwell
Advanced Training Systems Division
U.S. Army Research Laboratory
Orlando, FL
douglas.maxwell3.civ@mail.mil

Michael Heilmann
Institute for Simulation and Training
University of Central Florida
Orlando, FL
mheilman@ist.ucf.edu

ABSTRACT

A decade ago, it was assumed that simulation based training applications such as VBS3 (Virtual Battle Spaces 3) that use commercially available game engines would become a ubiquitous part of the infantry soldier's training curriculum. Unfortunately, the virtual environments for soldier training have encountered significant barriers to adoption as the underlying COTS technologies were not designed to conform to DoD information assurance policy. Most government employees and military service members are not privileged users with elevated security access to their computing equipment and cannot install the software. Further, depending on the network they are connected to, the software may not have the requisite Certificates of Net-worthiness or an Authority to Operate and cannot be installed. To use these technologies, classrooms at training sites with stand-alone networks must be specially prepared. This paper will discuss the common technical barriers preventing COTS game engines from gaining wide DoD installation approvals. This paper will also discuss a prototype virtual world simulation that has been developed at the U.S. Army Research Laboratory that leverages HTML5 and WebGL technology to create a simulation based virtual environment simulation that can be accessed via a common web browser. This prototype needs no third-party client software to be installed and should allow for military connected to the Internet to access training material. We will discuss the security implications for this approach, content protection, digital certificates for identity management, and the need to keep the simulator software and the training content separate.

ABOUT THE AUTHORS

Dr. Douglas B. Maxwell is a Science and Technology Manager at the U.S. Army Research Laboratory-Human Research and Engineering Directorate (ARL-HRED). He is the director of the Military Open Simulator Enterprise Strategy (MOSES) project. Dr. Maxwell earned his Ph.D. in Modeling & Simulation.

Mr. Michael Heilmann is a Software Engineer at the Institute for Simulation and Training, University of Central Florida. He is the lead programmer for the Military Open Simulator Enterprise Strategy (MOSES) project. He earned his M.S. degree in Computer Science.

Leveraging HTML5 and WebGL to Address Information Assurance Barriers for Simulation Based Training in the U.S. Military

Douglas Maxwell
Advanced Training Systems Division
U.S. Army Research Laboratory
Orlando, FL
douglas.maxwell3.civ@mail.mil

Michael Heilmann
Institute for Simulation and Training
University of Central Florida
Orlando, FL
mheilman@ist.ucf.edu

INTRODUCTION

Modern simulators for training have been around for more than 70 years with tremendous successes seen in aircraft pilot training (Blow, 2012), and game based virtual environments have been in mainstream use for almost 40 years in the entertainment industry. 20 years ago, proposals for linking simulation with game based environments for training gained traction and about 10 years ago the first simulators were introduced for infantry soldier training (Beal, 2009). In the early 2000's there was a wave of exuberance and optimism that virtual environments would be used for all manner of training in the military. The argument was cost savings, time savings, and safer training options making the technology adoption very attractive (Lackey, Salcedo, Matthews, & Maxwell, 2014; Rushmer, 2006; Stanney et al., 2013).

As of today, 2017, home gaming is a multibillion-dollar industry and vehicle operator simulators are growing in popularity and acceptance outside the military. However penetration of virtual environments for infantry soldier training is difficult to track and quantify (Pickup, 2013; Riecken, Powers, Janisz, & Kierzewski, 2013). There are barriers to entry that make widespread adoption of these technology by the military possible with possibly the most significant issue to deal with is information assurance and conformance to policy.

Currently, the program of record for infantry soldier virtual training is a program called Virtual Battle Spaces 3 ("Virtual Battle Spaces 3 (VBS3)," 2010; Wong & Nguyen, 2012). The VBS3 program is deployed to training sites all over the world and used for training activities such as the Basic Leadership Course, land navigation basics, convoy training and more. However, a gap analysis has revealed the program to be difficult to acquire, configure, maintain, and operate. Normal operation of the system is in a classroom setting with a local enclaved network configuration. Specially prepared workstations are provided to soldiers for training purposes. They generally do not have access to the VBS3 outside a training center. By placing these barriers in front of federal civilian and military users, it is not surprising to see usage of the program of record different than originally anticipated.

Rethinking how training and content is delivered while maintaining a cohesive 3D virtual environment is necessary. One idea is to deliver the content through existing software available to the unprivileged user. Recent advancements in networking, 3D graphics, and voice services make delivery of training content through the web browser feasible and attractive for investigation.

BACKGROUND

The U.S. Army has numerous virtual training systems based on commercial gaming engines. A common attribute that each of these software packages share is that if a user wants to use these applications on a military asset, they must obtain prior approval and have someone with administrative rights install the software. Further, if the software is to be used in a collaborative manner, the software must be cleared to operate on the network the military asset is connected to. Table 1 shows a matrix of a few selected available software titles and their intended use from the MilGaming web site.

Information assurance policy in the military is the reason for why it is so hard to obtain and deploy the software. There is no dispute that the IA policies are in place to protect the networks and users from penetration and data leakage, and are therefore a necessary part of the enterprise we work in. We must strike a balance of risk and functionality.

Since the computers provided to most military (civilian workforce and active duty) are managed by IT staff, the users have very little control over the computing asset. Software must be vetted and receive approval for installation, even if the software is developed for government use! There are instances where a software package can be approved for limited use on only certain networks. Usually the game based training systems are deployed to classrooms with stand-alone networks.

Table 1. Matrix of Selected Games for Training Available from MilGaming Website

Simulator	Intended Use
VBS3 (Virtual Battle Spaces 3)	3D first person game for training, mostly used for infantry skills training
BiLAT (Bilateral Negotiation Trainer)	Portable PC based training programed designed allow student to practice conducting negotiations.
Operational Language	Self paced foreign language training game to teach Iraqi, Dari, and more.
UrbanSim	PC-based virtual training game for practicing the art of battle command.
BioFor (Biometrics and Forensics)	Game based application designed to train how to successfully plan and apply biometric and forensic processes.
ELITE (Emergent Leader Immersive Training Environment)	Low-overhead interpersonal communication situation simulation used for counseling training and “soft leadership skills”.
DisasterSim	Low-overhead web-based (non-3D) training application used to train US military personnel on the unique nature of foreign disaster relief operations.

In addition to network restrictions, there is also the issue of software maintenance and deployment. Software must be maintained at a minimum for security patches and optimally provide functionality updates with fresh content. When the systems are deployed to disconnected/standalone networks, major updates become cumbersome and expensive to push out.

To address the restrictions imposed by an enterprise network policy, the web browser was examined for suitability to deploy virtual training applications. Attempts to do this in the past were met with mixed results. In the mid-90’s the VRML (Virtual Reality Markup Language) was created to display 3D information in the web browser (“The Virtual Reality Modeling Language,” 2002). The Navy created prototype applications with VRML and used it for online sharing of ship designs and the addition of intelligent features to machine parts as seen in Figure 1 (Summers, Maxwell, Camp, & Butler, 2001). A VRML plugin was required to view the content in a web browser.

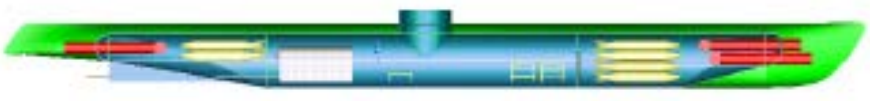

Figure 1. Submarine Hull Design Concept Viewed in VRML

In 1999, the Web3D consortium moved VRML to X3D with XML based encoding (Brutzman & Daly, 2007). The Naval Postgraduate School, working with the Office of Naval Research, was a major proponent of the development of these standards. X3D ("Extensible 3D (X3D) Encodings," n.d.) was an alternative to existing proprietary 3D web technologies of the day. It was explicitly designed to be royalty free, could be tuned for different classes of hardware, was capable of ingesting multiple data-encoding formats, and maintained backward compatibility to VRML.

X3D made it possible for developers to create more complex applications where 3D models and mathematical models could be combined to illustrate behaviors in the web browser. Building on VRML, even more functionality and interactive features could be added. The U.S. Navy was particularly interested since geoNodes were now available and undersea terrains could be represented as seen in Figure 2.

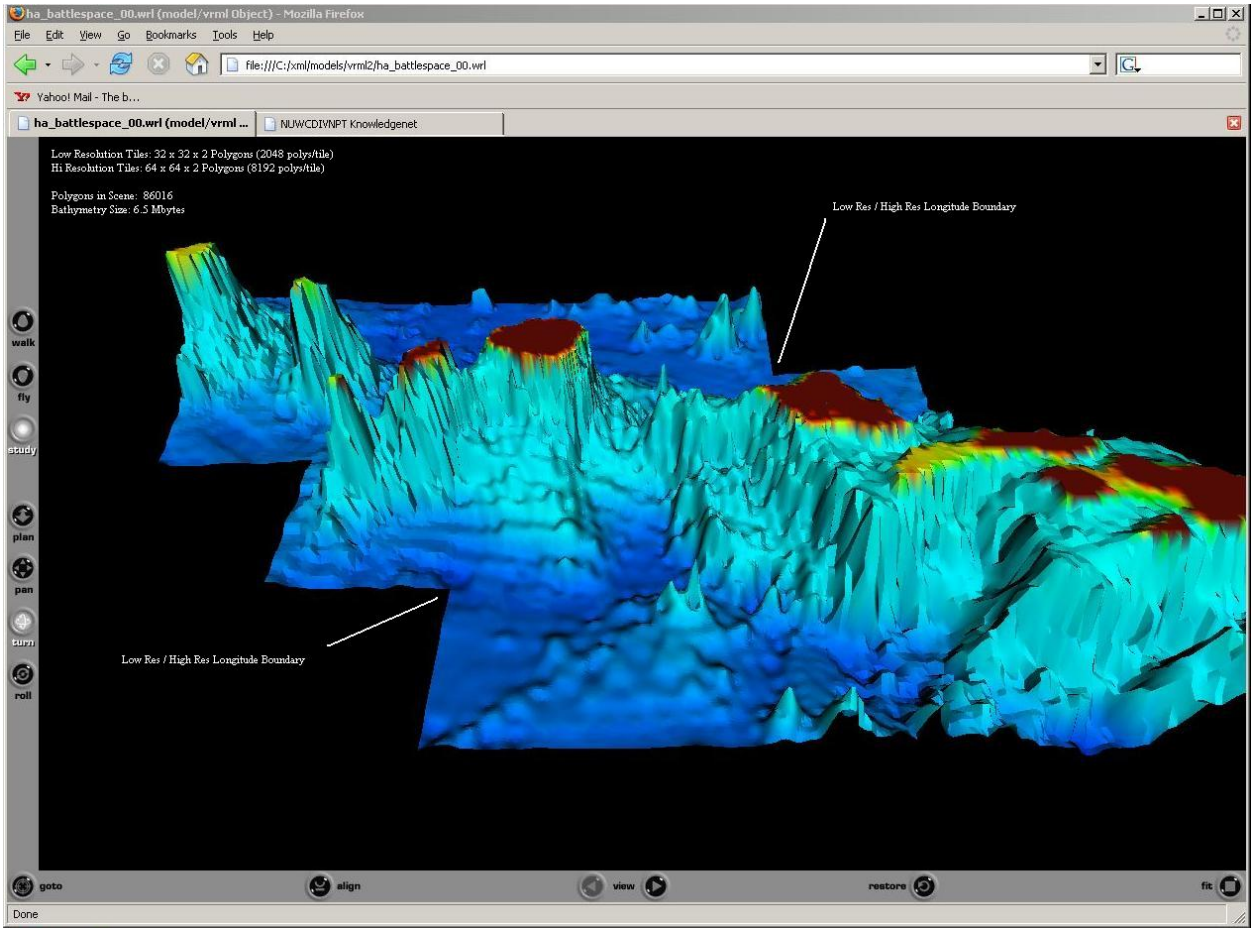

Figure 2. X3D Application Displaying Bathymetric Data

While X3D was an improvement over VRML, it still required a plugin to function in the web browser. HTML standards at the time did not natively interpret 3D data. The reason plugins were and are an issue is that they are treated just like stand-alone applications by enterprise IT management. The same information assurance policies that make it difficult to install regular applications to the military computing assets also make it difficult to install browser plugins. Although there exists an approved X3D plugin, it is unclear as to how well it has been maintained and it is the responsibility of the individual user to request that it be installed and cover the costs associated with the deployment. X3D was a good functional demonstration of how web browsers could be leveraged to view 3D content, it was unfortunately not adopted widely due to the barriers to entry discussed above.

To use the web browser to view 3D content, a more practical solution must be found. Fortunately, today the HTML standards have evolved to include 3D content. HTML5 and WebGL standards can interpret JavaScript

natively to view content. HTML5 could be used to provide simulation based training with 3D virtual environments to not only a desktop web browser, but also to portable systems such as tablets and phones. In the following sections, we will discuss current game based training simulator design and compare it to a proposed design that leverages HTML5/WebGL standards. The proposed design is currently in development and limited testing has been conducted to establish baseline performance benchmarks.

CURRENT DESIGN

Game based simulators typically follow two different paths for their virtual environment: sharding or replication. A sharding game server will subdivide the space into areas of interest, or shards, which together comprise the virtual area. This allows for a single virtual environment to be larger than any single process could support in a reasonable manner. A replicating game server loads multiple full copies of the virtual space. The virtual space is much smaller, but the server can distribute the player load across the processes, and support more players. A sharding server requires that there is not a high number of entities or users on a single shard for good performance. A replicating server requires that distributed players or entities have no need to interact with each other.

The simulation discussed here is a sharding simulator that has already been used as a tool in training based experiments. This game server subdivides its virtual space into 256 by 256 meter squares that are each a separate shard or server process. Multiple shards that are configured to run cooperatively may be aggregated into a single virtual space by co-locating their areas of interest in the virtual world's coordinates. The processes themselves may be widely distributed physically and still operate in this way. The aggregated virtual space is viewed by any connected client as a large continuous space. Figure 3 illustrates a 1.25 km by 0.75 km example of virtual space simulated with 12 simulator processes.

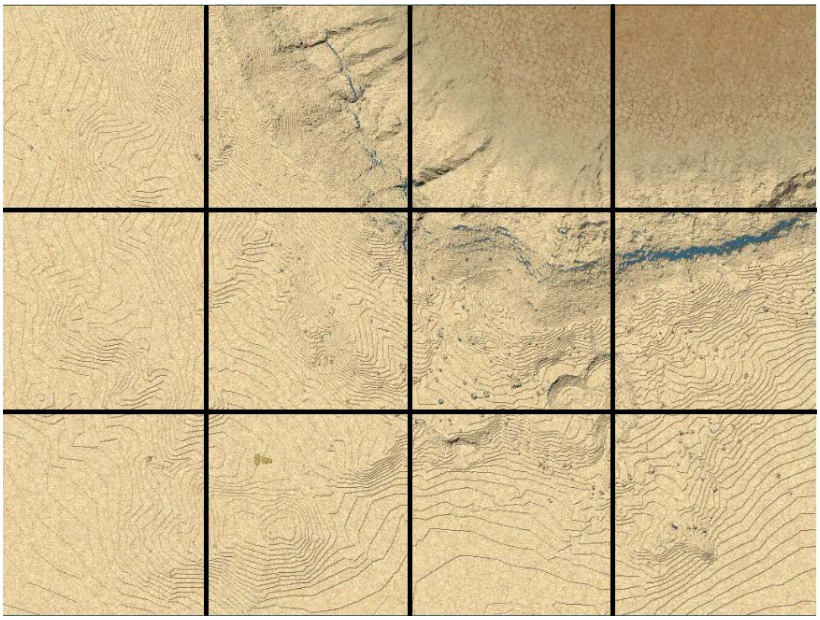

Figure 3. Illustration of Multiple Aggregated Simulator Shards in a 4x3 Configuration

Each server process is a self-contained simulator, with its own scripting, entity management, and physics responsibilities. The coordinating processes for this aggregation and cooperation are very small and lightweight, leaving the simulator shards to also perform tasks for user management, inventories, etc.

This game based simulator is written primarily in C#, using the .Net runtime. It uses MySQL as a data persistence layer. There is interaction with C++ components, such as NVidia's PhysX physics library for the physics components of the simulation.

The client for this simulation is an installed C++ and OpenGL application. Upon login, the client communicates with the coordinating cluster processes using HTTP protocols, and then connects to one or more region processes to view

and interact with the simulated scene. When the virtual space is aggregated, the client will connect to up to 25 simulator processes concurrently to communicate with each portion of the aggregated space. As the client moves through the space they may cross a simulator boundary, causing their primary point of contact with the space to switch to a new simulator process. The extra connections to surrounding simulator processes are added and dropped during this process. It does not matter how quickly or slowly a user navigates the virtual space, the process borders will remain at 256 meter intervals. During client-simulation interaction, almost all communication is handled by a single UDP socket between the client and each process. Some functions have been exposed through HTTP handlers, though it is not used for simulation interaction

This design is problematic for several reasons. An installed client is required for any interaction with the simulated environment. Users are free to traverse boundaries, but non player entities and scripted events may not cross between shards. The high number of dedicated network ports required to support a large virtual environment is problematic on restricted networks. Between the complex client and the multiple server processes, gaining accreditation on secure networks is difficult and costly.

An example network connection map that is simplified for readability is shown in Figure 4. One can see that each installed client will communicate with multiple simulator shards, and with the coordinator services. The simulators themselves communicate with the coordinator and the persistance layey. Omitted here is voice communications, inter-region communications, non simulation services such as asset caching, etc.

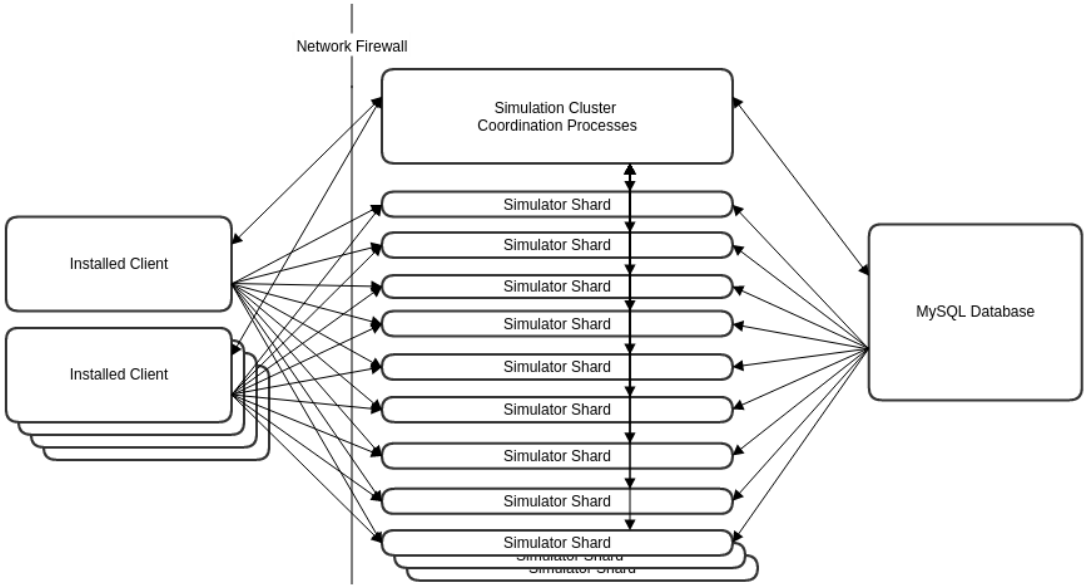

Figure 4. Traditional Game Based Simulator Server Layout

PROPOSED DESIGN

The proposed design stems from the desire to use web applications to interact with the simulation system. The system as it is currently implemented does not support the TCP based connections that web applications are currently restricted to. In analyzing the current design, two conclusions were reached: First, the simulator has too many concerns to be addressed in a scalable manner. The non-simulation services should be converted to web services on the coordinating processes. Second, connection management is a problem for connected clients. The web clients would benefit strongly from a unified interface, or connection gateway, with which to communicate.

These conclusions led to the desire for an interface layer, hereafter called the arbiter process, or layer. This arbiter would manage client network connections, and proxy simulator events between the client and the simulator. Non-simulation concerns can now be intercepted and serviced directly by the arbiter, freeing the simulator process from additional work.

The arbiter layer shown in Figure 5 is to be implemented as closely as possible to a scalable web application rather than a specialized component of the simulator. It is currently planned to be implemented using NodeJS to leverage community support and the broad availability of existing modules.

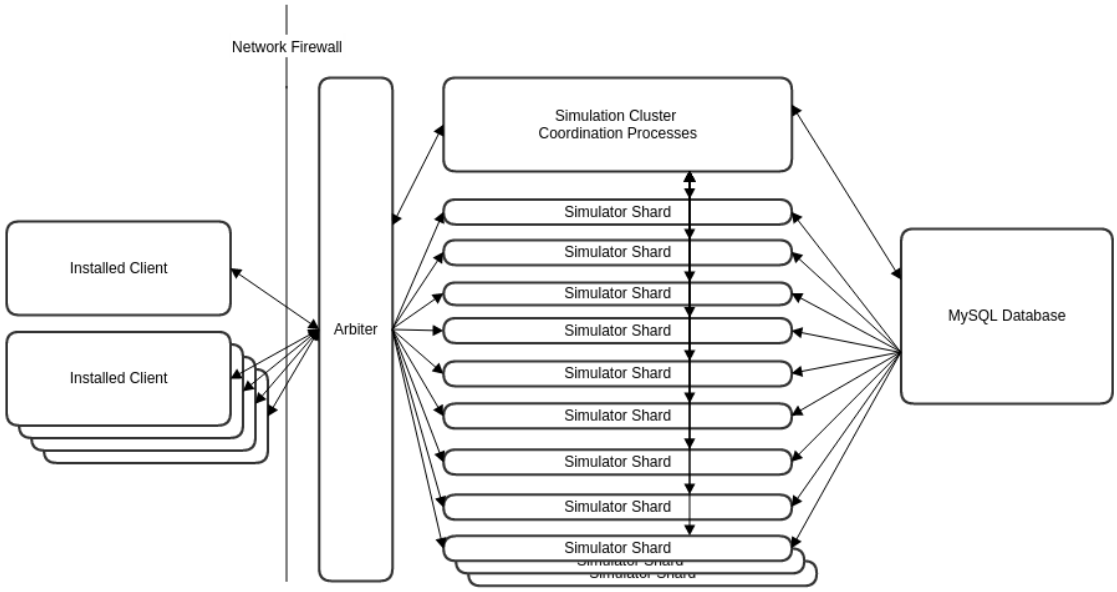

Figure 5. Proposed Design with Data Arbiter

This design allows for complete control over protocols, allowing for tcp and websocket clients to exist independently of the normal simulator communication. The separation of concerns for the simulator means that chat, user controls, inventories, etc… may be handled by the arbiter, allowing the simulator to focus on the actual simulation. Since there is reduced network complexity, the client connections are now made with a single interface, rather than connecting to multiple processes at multiple network endpoints. This greatly simplifies network security and configuration for all parties. It becomes feasible to operate over the commonly unrestricted ports 80 and 443.

TESTING

The implementation and testing of the complete arbiter layer is a complex and long-term goal. Instead of designing and implementing the entire application change, we have elected to implement in strategic stages. The incremental stages that we are implementing and testing are:

1. Simple HTML5/WebGL view of a static scene representative of the simulator.
2. Simple HTML5/WebGL view of an active simulation, fed with live data.
3. HTML5/WebGL application with basic interaction against a live simulator.
4. View-only Web application client that allows simulator participation.
5. Web application client that can replace the installed clients for all simulation interaction.

We have implemented increment 1 with a static scene viewable in html5 independently of any simulator or arbiter process as shown in Figure 6. Not all objects are placed correctly, but asset formats are transformed correctly, and display correctly in the HTML5/WebGL view.

The web application is currently implemented using the BabylonJS WebGL Framework. This allowed us to leverage an existing 3d visualization library for the client instead of having to implement our own. Included with this framework is a lighting model and mechanisms for dynamic shadows, entity controls, terrain management, and mechanisms for loading scenes from multiple reusable components.

The static display of a virtual scene is simple but very valuable. First, it allows us to emulate what the web application would need to do to display the contents of a live simulation, specifically the translation of asset formats and coordinate spaces for a WebGL based scene. It also does not require a running arbiter process, nor a functional data link to a running simulation. It will form the basis for the development and testing of the first stage of the arbiter process: a non-interactive live view of the simulation space. As the performance of the web application is suspected to be a problem area in this development, we wanted to expose any issues as early as possible.

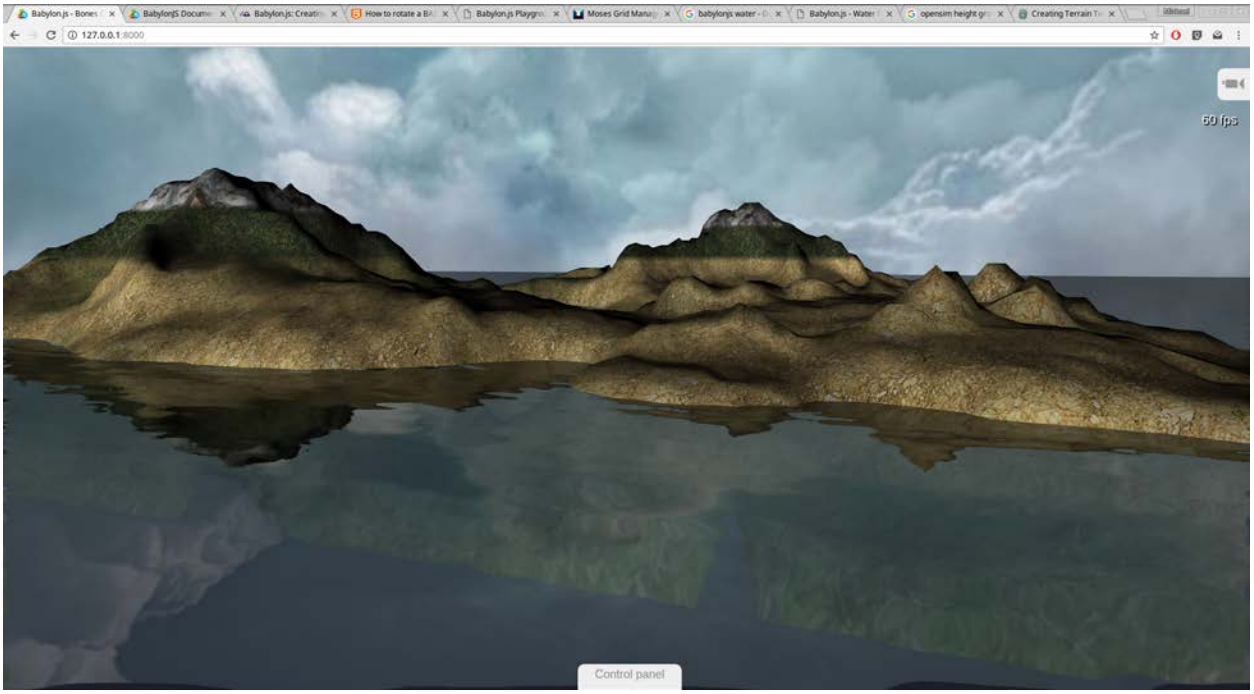

Figure 6. Initial Babylon.js Scene Derived from Terrain Database

The initial increment loads the simulator scene from a convenient archive file that the simulator can export. Scripting and dynamic information is ignored. The objects in the scene are converted from jpeg2000 images and proprietary geometry formats to png images and BabylonJS geometry. The resultant BabylonJS scene file is loaded into a BabylonJS web application. The objects in the scene are static, but the camera is interactive and free to navigate and explore the simulator.

It is easy to bundle a scene that is too large for a web browser to process. Research questions under current investigation include proper display of a scene with sufficient complexity to be comparable to the standalone clients the BabylonJS scene is intended to replace. Additionally, what is an acceptable time limit for asset conversion in real time to not impact user experience while providing the complexity needed to meet mission requirements? Last, what are the performance characteristics of the web application and how do they compare to standalone applications.

The next increment of development will begin the implementation of the arbiter layer. This will involve a new server-side application, and custom connectivity modules for the simulator. The increment will display a live view of the simulation in progress with dynamic updates of objects in the scene. Objects and interactions will be displayed in a WebGl scene but the web users will be unable to participate or affect the simulation.

FUTURE WORK

The first iteration of this work is was a web based interface to a 3D training environment that is static and non-interactive. As described above, the reason was to benchmark the amount of 3D content that can be delivered to various browsers, computers, and mobile platforms. Additionally, there is a need for a review of the current code to identify optimization opportunities for later implementation.

After performance metrics are drafted and web viewer performance is benchmarked additional functionality will be added to the scene. Capabilities such as user interaction with the environment will be added first. An example of this is the selection of an object by a user and allowing them to move it or selecting a vehicle to sit in and operate.

The overall objective is to replicate the utility of an installed 3D client application in a web browser without the need for third party applications or plugins. The current state of HTML5/WebGL technology shows promise in achieving this goal. Key issues such as identification of bottlenecks and working within the limitations of the web specifications make this a unique challenge. Figure 7 shows a comparison of the same content displayed in a traditional installed client and the web based prototype client. As can be seen in the picture, the results of initial testing and evaluation of the early work are encouraging.

Figure 7. Comparison of Scene from Installed Client (above) to Web Browser Client (below)

REFERENCES

Beal, S. (2009). EXPLORING THE USE OF A MULTIPLAYER GAME TO TRAIN INFANTRY COMPANY COMMANDERS EXPLORING THE USE OF A MULTIPLAYER GAME TO TRAIN INFANTRY COMPANY COMMANDERS. In *Interservice/Industry Training Simulation and Education Conference (I/ITSEC)* (pp. 1–12). Orlando.

Blow, M. A. J. C. A. (2012). Flight School in the Virtual Environment Capabilities and Risks of Executing a Simulations-Based Flight Training Program by School of Advanced Military Studies Fort Leavenworth , Kansas.

Brutzman, D., & Daly, L. (2007). X3D: Extensible 3D Graphics for Web Authors. *X3D*, (November), 130–135. http://doi.org/10.1109/MSP.2007.905889

Extensible 3D (X3D) Encodings. (n.d.). Retrieved January 25, 2017, from http://www.web3d.org/documents/specifications/19776-1/V3.3/index.html

Lackey, S. J., Salcedo, J. N., Matthews, G., & Maxwell, D. B. (2014). Virtual World Room Clearing : A Study in Training Effectiveness. In *Interservice/Industry Training, Simulation, and Education Conference* (pp. 1–11). Orlando. Retrieved from www.iitsec.org

Pickup, S. (2013). *ARMY AND MARINE Better Performance and Cost Data Needed to More Fully Assess Efforts*. Retrieved from http://www.gao.gov/assets/660/657115.pdf

Riecken, M., Powers, L. T. C. J., Janisz, C., & Kierzewski, M. (2013). The Value of Simulation in Army Training, (13073), 1–16.

Rushmer, R. (2006). Armchair warriors. *Professional Engineering*, *19*(20), 40.

Stanney, K. M., Cohn, J., Milham, L., Hale, K., Darken, R., & Sullivan, J. (2013). Deriving training strategies for spatial knowledge acquisition from behavioral, cognitive, and neural foundations. *Military Psychology*, *25*(3), 191–205. http://doi.org/10.1037/h0094962

Summers, J. D., Maxwell, D. B., Camp, C. M., & Butler, A. C. (2001). Features as an Abstraction for Designer Convenience in the Design of Ships. *Naval Engineers Journal*, *113*(4), 53–68. http://doi.org/10.1111/j.1559-3584.2001.tb00088.x

The Virtual Reality Modeling Language. (2002). Retrieved January 25, 2017, from http://www.web3d.org/documents/specifications/14772-1/V2.1/index.html

Virtual Battle Spaces 3 (VBS3). (2010). Retrieved from https://milgaming.army.mil/VBS3/

Wong, J. H., & Nguyen, A. B. (2012). Serious Game and Virtual World Training Instrumentation and Assessment, (December).