3D Visualization for Point of Need and Cloud Based Training

Greg Dukstein
Dignitas Technologies, LLC
Orlando, Florida
gdukstein@dignitastech.com

Keith Nielsen
PEO STRI, PM ITE
Orlando, Florida
Keith.b.nielsen.civ@mail.mil

Paul Dumanior
PEO STRI, PM ITE
Orlando, FL
paul.h.dumanoir.civ@mail.mil

ABSTRACT

US Army Modeling, Simulation, and Training (MS&T) is moving toward cloud based solutions to meet the future training needs of the Warfighter. Cloud based solutions will need to provide easily accessible synthetic training tools that will enable soldiers all over the world to train together. The Army is investing in cloud technologies and leveraging research to provide Point of Need (PoN) training services and reduce the cost of deployment and sustainment. This paper captures the lessons learned from a government funded effort to develop an open source light weight cloud based 3D visualization tool for MS&T applications. In addition to looking at broader applications, our effort is specifically targeting needs of the Army's Live Virtual Constructive – Integration and Architecture (LVC-IA) program as a transition target. Our work has resulted in a thin client 3D viewer that runs in a browser and leverages common US Army standards and components for exercise data, terrain, and model data. We will discuss how open solutions and leveraging new government developed technologies can provide a cost-effective solution while still achieving commonality and interoperability. We will discuss the approaches used, such as streaming terrain elevation data, ground surface imagery, and 3D models from cloud based servers to thin client 3D viewers and examine different technologies for rendering a 3D scene in a web browser. We discuss the practical challenges of transitioning this technology to meet cloud requirements of LVC-IA and other Programs of Record. Lessons learned will be presented regarding implementation and fielding of a web based solution into a large, complex training environment.

ABOUT THE AUTHORS

Mr. Gregory Dukstein has over 20 years of experience in modeling and simulation applications, with a focus on terrain services, terrain databases, and modeling behaviors. Mr. Dukstein has worked CGF applications, SAF database formats, and terrain services for CCTT, UKCATT, Warfighters Simulation (WARSIM), and Synthetic Environment Core (SE Core). Mr. Dukstein has been involved in large programs and smaller efforts filling roles such as software developer, systems engineer, team lead, and Chief Engineer. Mr. Dukstein is currently the Director of Engineering at Dignitas Technologies where he manages research and development projects for PEO STRI and RDECOM.

Mr. Keith Nielsen is the Lead Development Engineer for the Synthetic Environment Core (SE Core) program at the U.S. Army Program Executive Office for Simulation, Training, and Instrumentation (PEO STRI). Mr. Nielsen received his undergraduate degree in computer engineering from the University of Central Florida. Mr. Nielsen has more than 15 years of system engineering experience in modeling and simulation applications, with a specific focus on simulation protocols, synthetic natural environment, and semi-automated forces.

Mr. Paul Dumanoir is the Chief Engineer for the Product Manager for Warrior Training Integration (PdM WTI) under the Project Manager for Integrated Training Environment (PM ITE) at PEO STRI. He leads several ITE modernization risk reduction activities and has over 30 years of experience in simulation and training programs as Product Manager, Project Director, and Systems/Software Engineer. His interests include enterprise architectures, model-based and product-line engineering, and system of systems integration and interoperability. He holds a B.S. in Electrical Engineering from the University of South Alabama and M.S. in Computer Systems from the University of Central Florida.

3D Visualization for Point of Need and Cloud Based Training

Greg Dukstein
Dignitas Technologies, LLC
Orlando, Florida
gdukstein@dignitastech.com

Keith Nielsen
PEO STRI, PM ITE
Orlando, Florida
Keith.b.nielsen.civ@mail.mil

Paul Dumanior
PEO STRI, PM ITE
Orlando, FL
paul.h.dumanoir.civ@mail.mil

INTRODUCTION/BACKGROUND

US Army Modeling, Simulation, and Training (MS&T) is moving toward cloud-based solutions to meet future training needs of the Warfighter. Cloud-based solutions need to provide easily accessible synthetic training tools to enable soldiers all over the world to train together [1][2]. PM Integrated Training Environment (ITE) is investing in cloud technologies and leveraging research to provide Point-of-Need (PoN) training services and reduce the cost of deployment and sustainment of their training systems and services.

The objective of this Program Executive Office for Simulation, Training & Instrumentation (PEO STRI) sponsored project was to research and develop a lightweight 3-Dimensional (3D) visualization tool that runs as a thin client in a browser to support PoN and Cloud-Based training. The intent was to leverage an existing software application which currently provides 3D visualization capabilities to Synthetic Environment Core (SE Core) and the Live, Virtual, Construction Integrating Architecture (LVC-IA) Program of Records (PORs) in a desktop based configuration. The desktop application is called Veritas, and is currently fielded and widely accepted by the LVC-IA community, however it does not have cloud-based support. The focus of the effort was to develop the thin client 3D viewer to support current LVC-IA V3 requirements for exercise monitor and After Action Review (AAR) activities, yet be based on a flexible architecture which enabled accommodating additional ITE 3D visualization requirements as they become better defined. The web-based 3D viewer architecture includes: a web server to receive and process exercise events and stream these events to the thin client; a web server to provide terrain and 3D model data to the thin client; and the thin client 3D viewer that runs in a browser.

In this paper, we discuss our research, development, experimentation, challenges, and lessons learned to prototype an open source, lightweight, and web-based 3D visualization tool for MS&T applications, which we named webVeritas. We discuss our analysis of current 3D viewer requirements for ITE simulation systems to identify the common set of 3D viewer requirements to define the requirements for web-based 3D viewer prototype. We discuss how open solutions and leveraging government-developed technologies provide a cost-effective solution while still achieving commonality and interoperability. Finally, we discuss the approaches used, such as streaming terrain elevation data, ground surface imagery, and 3D models from cloud-based servers to thin client 3D viewers, and examine different technologies for rendering a 3D scene in a web browser.

This paper also captures transition targets for webVeritas. We discuss our collaboration with ITE programs to leverage technologies and ensure our approach supports their current and future cloud requirements. In addition, we document remaining tasks and recommendations for future extensions to move the web-based viewer from a Technology Readiness Level (TRL) 6 prototype to a production ready TRL 9 product.

RESEARCH AND ANALYSIS

Requirements and Use Cases

Our research and development began by identifying PoN use cases and current PM ITE 3D viewer requirements. Point of Need use cases include: 3D viewer is accessible from a browser; web applications are lightweight and easily deployable; web services support multiple thin clients; exercise and AAR events are processed by web service; terrain and 3D models are stored and streamed from a web server; and the thin client renders geospecific terrain and 3D models. We analyzed 3D viewer requirements from four PM ITE PORs, One Semi-Automated Forces (OneSAF), Joint Land Component Constructive Training Capability (JLCCTC), LVC-IA, and Synthetic Environment Core (SE Core) to identify common viewer requirements across the ITE portfolio. Our requirements effort leveraged previous

work on a desktop viewer developed for research use cases [3] which is now the LVC-IA 3D Viewer. The resulting set of requirements and use cases drove our research and development of webVeritas.

Thin Client Framework Analysis

In the initial phase of our research we investigated existing technologies that could provide a thin client framework to render 3D terrains, 3D models, and exercise/simulation events (e.g. entities, fire, detonations, battlefield effects, etc.). The thin client framework provides the foundation of webVeritas. Key requirements for the framework include: utilize modern web technologies such as HTML5; run in a browser without the need for a plugin; and display correlated terrain data and 3D models. Other requirements include low cost, high performance, networking support, ease of development, and community support. We looked at several technologies including Unity, Unreal 4, and Cesium as possible frameworks to leverage for the thin client 3D viewer. We developed prototypes using all three frameworks and ran a series of experiments to determine which one best supported our thin client framework requirements. Based on the results of our analysis and experimentation, we selected Cesium [5] to provide the foundation for webVeritas. It is open source with an active developer community, performs well, uses a straight forward approach for rendering terrain and 3D models, and it utilizes HTML5. Below we discuss each of the thin client frameworks we investigated and the results of our experiments.

Cesium

Cesium is a JavaScript Application Programming Interface (API) for geospatial visualization in a WebGL enabled web browser. It streams global terrain height data and imagery, and renders on a virtual globe. The API allows connections with different terrain and imagery providers, including locally hosted servers. 3D models of buildings and vehicles can be streamed, given that the models are available.

Cesium is a client framework that runs natively in a browser and therefore supports Representational State Transfer (REST) style networking as well as WebSocket connections. Since there is no bundled server infrastructure, use of this framework requires some initial setup of open source web servers to provide terrain data, as well as development of a custom server to feed simulation data. Terrain heights and imagery are imported from DEM terrain and GeoTIFF imagery. While adding new terrain and imagery is relatively simple, the main challenge is generating the static models. The Cesium 3D model format is glTF.

Since Cesium appeared to be the most viable amongst the framework candidates, a prototype python based server was created to facilitate further testing. The server translates Distributed Interactive Simulation (DIS) data to a format that is sent to the 3D Viewer Application. Specifically, the server converts DIS messages to Google Protocol Buffer format that are sent to 3D Viewer Applications via a WebSocket established between the client and server. The messages received by the 3D Viewer Application are used to display entity representations and events on the virtual globe. The performance, ease of development directly in JavaScript, and no cost make Cesium a viable option for the framework of the 3D viewer. The table below lists the Cesium pros and cons identified during our analysis.

Table 1 Cesium Research Results

Pros	Cons
✓ Open source with active developer community	✓ Graphics are not as impressive as with UE4 or Unity
✓ Minimal coordinate conversions needed for global entity positions. Cesium uses Geocentric Coordinate System.	✓ Programming is not in a type safe language (easier to introduce errors that are discovered at runtime rather than build time)
✓ Straight forward automation to add terrain	
✓ Instantaneous application startup	
✓ Free to use	
✓ Programming in JavaScript, access to Web Workers API for multithreading available	

Unreal 4

Unreal Engine 4 (UE4) is a popular game development framework able to support multiple platforms including internet browsers with HTML5. UE4 application development is done in C++, but the code can be exported to HTML5 to run in a web browser. UE4 HTML5 export support "uses the emscripten tool chain from Mozilla to cross-compile

C++ into JavaScript"[1] so the game engine code can run in a browser. The HTML5 export feature is considered experimental. Terrain can be imported through the UE tools as "Landscape Actors" using heightmaps and textures. It can also be imported on the fly by code using a procedural mesh. Both have their challenges; the first approach requires a lot of manual labor; the second approach using the procedural mesh requires code to read a data source during runtime and renders the 3D terrain along with textures. 3D Models must be imported into the UE4 editor for use during runtime. FBX is the preferred import format. Despite the lure of using the advanced game engine that UE4 provides, the lack of HTML5 maturity and multithreading support, long load times, and complex royalty costs make UE4 framework inadequate for the web based 3D Viewer. The table below lists the Unreal 4 pros and cons identified during our analysis.

Table 2 Unreal 4 Research Results

Pros	Cons
✓ Sophisticated game engine with advanced graphics	✓ Royalties (5% of gross revenue after the first $3,000 per product per calendar quarter)
✓ Source code available	✓ Built-in multiplayer framework not compatible with "world origin rebasing"
✓ Built in multiplayer framework	✓ HTML5 export is experimental
✓ World origin rebasing for unlimited size terrains without the problems of floating point precision (Not compatible with multiplayer)	✓ No documentation on how to send messages to and from UE4 HTML5 code and "native" JavaScript in the webpage
	✓ Multi-threading not supported in UE4 for HTML5 export (but planned for future)
	✓ Long load times

Unity

Unity is a game development engine supporting deployment to a wide variety of platforms, including the web. Unity 5 introduced support for WebGL, allowing content to be run in compatible browsers without the use of a plugin. Unity converts code written in C# to Instruction Line (IL) to C++ to JavaScript (IL2CPP) to convert game engine code to HTML5 runtime. The Unity web export functionality is more mature than UE4's. We successfully tested the WebSocket connection between HTML5 export and web server. Several model formats can be imported into the Unity project assets directly without any prior conversion. These formats are FBX, dae (Collada), .3DS, .dxf, and .obj files as well as SE Core's Gen1 and Gen2 common moving models (CM2). Despite the positives of the Unity game framework the lack of multithreading support, long load times, debugging difficulty, and development costs make this framework inadequate for the web based 3D Viewer. The table below lists the Unity pros and cons identified during our analysis.

Table 3 Unity Research Results

Pros	Cons
✓ Sophisticated game engine with advanced graphics	✓ Multi-threading not supported for HTML5 export
✓ Built in multiplayer framework	✓ No built-in mechanism for loading extremely large terrains
✓ No royalties	✓ Long load times
✓ Ability to send and receive messages to and from Unity export code and "native" JavaScript code in web page	✓ Hard to debug WebGL runtime issues due to language conversion to JavaScript
✓ Doesn't require a web browser plugin	✓ Build sizes are very large
	✓ $125 per seat/month to develop with framework

[1] https://docs.unrealengine.com/latest/INT/Platforms/HTML5/GettingStarted/

DEVELOPMENT OF WEB-BASED 3D PRODUCTS AND TOOLS

Three main design goals guided the development of webVeritas. First, webVeritas should be developed with minimal licensing costs using open source technologies, and be provided with government purpose rights. Second, the user interface (UI) should be simple, intuitive, and similar to the current desktop version of Veritas. And third, the web-based 3D viewer prototype should be delivered at TRL 6 to support exercise monitor and AAR operations.

See Figure 1 for a high-level system architecture description of the web-based 3D viewer which consists of three main components: Terrain/Model Server, Exercise Service, and Thin Client 3D Viewer.

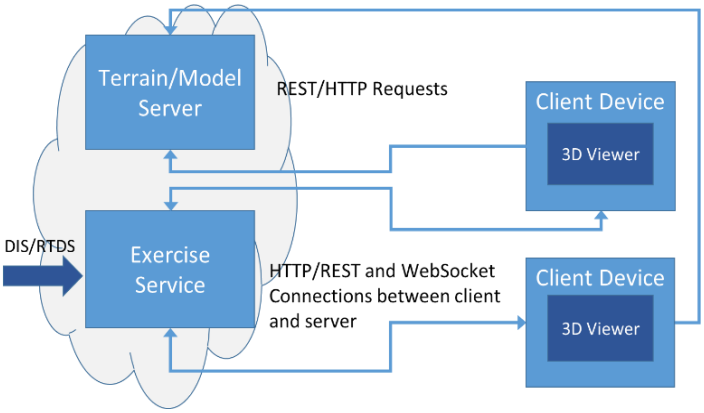

Figure 1 web-based 3D Viewer Architecture

Thin Client 3D Viewer Component

The thin client 3D viewer component provides an interface for the end user. The 3D Viewer runs in any HTML5/WebGL enabled browser. It is simulation protocol agnostic, using a shared communication data model between the client and server applications. It runs in an internet browser on one or multiple workstations. The thin client retrieves terrain, elevation, and 3D model data from the Terrain/Model Server component, and exercise data, such as entities, objects, and battlefield events are received from the Exercise Service component. Rendering of terrain data and exercise events is handled in the thin client by the Cesium framework. Elevation data is rendered by Cesium runtime as a triangle mesh. Cesium requests appropriate zoom level and upsamples, if necessary, to smaller area tiles by splitting an elevation mesh into 4 smaller area elevation meshes. The imagery is texture mapped on the elevation mesh by Cesium. Cesium requests appropriate zoom levels and uses appropriate imagery providers. Multiple imagery providers can be registered so different Tile Map Service (TMS) imagery can be used simultaneously. When entities or objects are received from the Exercise Service the thin client requests their corresponding 3D model from the Terrain/Model Server to then render in the viewer. The Cesium framework supports 3D models, including key-frame animation, skinning, and individual node picking, using glTF. gITF is an emerging industry-standard format for 3D models on the web by the Khronos Group, the consortium behind WebGL and COLLADA[2]. Simulation effects such as fire, detonations, dust clouds, smoke, and muzzle flash are rendered in the thin client viewer. However, Cesium does not have any direct support for particle systems to render simulation effects such as smoke or detonations. We developed a custom implementation on top of Cesium's BillboardCollection class. The developed particle system can be composed with multiple parameters such as particle texture, particle lifetime, emitter lifetime, emitter rate, etc.

Figure 2 webVeritas Displaying Virtual Globe

[2] https://cesiumjs.org/

We identified a number of use cases for the thin client. These include: view terrain locations; connect to a running exercise of an AAR playback session; observe a real-time exercise or AAR playback data such as entity events, weapon events, object events, and tether to entity; and disconnect from an exercise or AAR playback session. Figure 2 shows how the thin client first looks when brought up in a browser, displaying Blue Marble imagery.

Thin Client User Interface

The 3D viewer UI must be simple to use, intuitive, and easy to navigate, like the desktop version of Veritas. A user-friendly interface is one of the main reasons the desktop Veritas application is chosen as an exercise monitor and AAR 3D stealth tool. In addition, a similar UI will feel familiar to current Veritas users, and will ease the transition from a desktop tool to web-based one. Therefore, we modeled the web-based 3D viewer UI after the desktop UI. One webVeritas UI requirement was for an architecture that enables easy extension of tools and plugins that interface with the 3D view. Another requirement is a single view user experience, which is similar to desktop Veritas, with all functions available on one screen. We evaluated two popular UI frameworks: React and Angular2. React[3] is a JavaScript open source library maintained by Facebook, primarily used to generate user interface views. One advantage of React is it utilizes JSX that allows mixing XML and JavaScript in one file. Angular2[4] is a JavaScript component based framework maintained by Google, with the advantage of being a framework containing everything required to develop a client side website.

We selected Angular2 for the thin client UI framework because it provides a complete framework with preferred libraries and functionality. It uses the TypeScript language, which is a superset of JavaScript. Because it provides strong typing and classes, and can be compiled back into JavaScript, it has improved compatibility in all browsers. Additionally, Angular2 provides an easy to use framework for extending UI components. For databinding and communication, we selected RxJS (Reactive Extensions for JavaScript). Angular2 uses Observables as part of its asynchronous interfaces, such as making HTTP requests and communication between components. The RxJS library comes with a WebSocket Observable. Combining this WebSocket Observable with other RxJS operator extensions allowed easy filtering of messages and handling reconnections when the websocket was disconnected due to an error.

We designed and implemented the thin client UI to have a single view providing a simple menu to access all 3D viewer user functions. Pictured below are a few examples of the UI. The Entity List function accesses a list of entities in the training exercise with the capability to sort, filter, and tether to entities. The Fire Lines function provides the capability to enable/disable fire lines. The Goto Location function moves the eyepoint to a user provided coordinate. The finally the Saved Locations function allows the user to save important or interesting locations. The images in *Figure 3* provide two examples of the thin client UI.

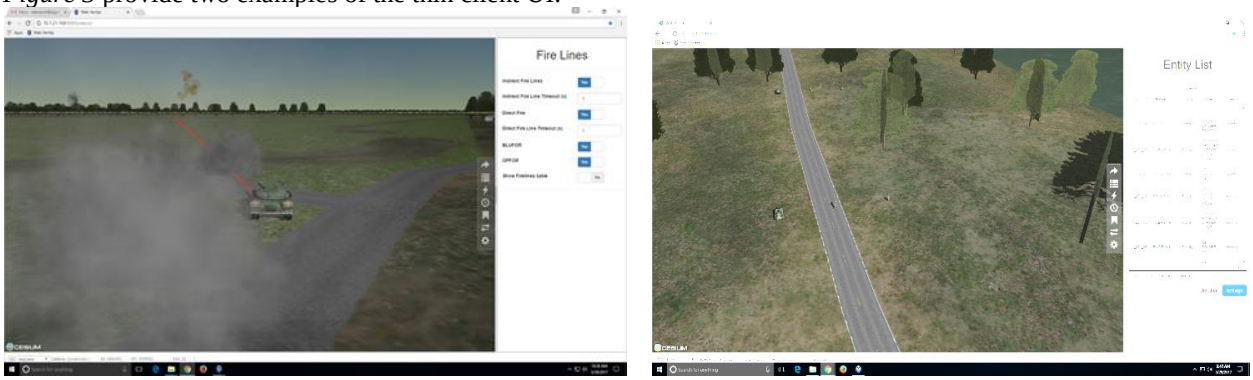

Figure 3 Example Thin Client UI

[3] https://facebook.github.io/react/
[4] https://angular.io/

Terrain / Model Server Component

The Terrain / Model Server component is an HTTP based server that supplies terrain information consisting of elevation data, terrain skin textures, and 3D models to the thin client 3D viewer component. We reused a Go based open source Cesium Terrain Server compiled to run as a native executable on either Windows or Linux. The terrain data includes 2D imagery tiles and terrain elevation data. The model data provided by the Terrain/Model Server consists of 3D models of features (e.g., trees, buildings, etc.), entities (e.g., vehicles, lifeforms, munitions, etc.), and objects (e.g., log cribs, fences, barriers, etc.). It should be noted that the Terrain and Model servers can be run together as a single server (depicted in Figure 4) or run as separate processes on different machines with minimal configuration changes on the Exercise Service. This allows flexibility in environments with limited or abundant hardware, and supports future scalability.

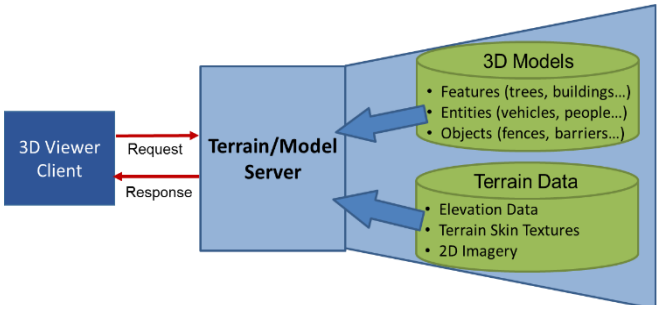

Figure 4 Terrain and Model Server

Exercise Service Component

The Exercise Service component hosts the 3D viewer HTML5 client code. The component handles simulation exercise specific operations such as decoding simulation protocol data and forwarding the events and data to the thin client 3D viewer components to be rendered. The primary role of the Exercise Server is to convert simulation specific protocol information (e.g. DIS) into a more generic viewer data format that is forwarded to and then displayed by the thin client. The server is written in Python and consists of both third-party software (i.e. Bottle and CherryPy frameworks to handle web requests) and custom written modules to do the translation. The server defines a REST API to interact with the Exercise Service.

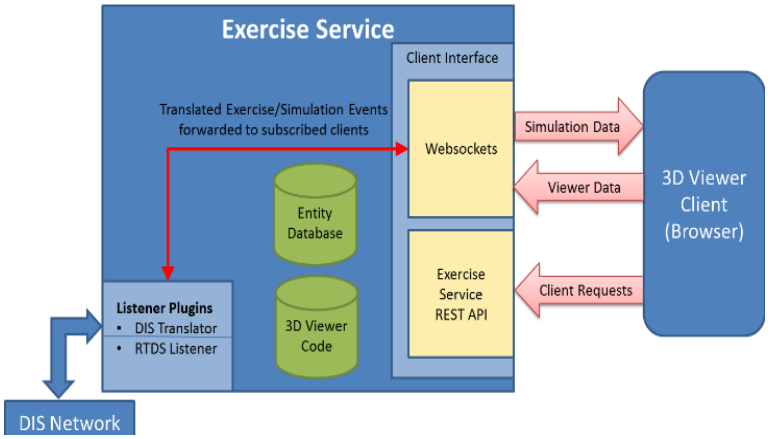

Figure 5 Exercise Service

Through Listener Plugins the Exercise Service can receive and process simulation data via any number of simulation protocols such as DIS, HLA, and Tena. The prototype version of webVeritas integrates a DIS Listener plugin, enabling interoperation with systems such as OneSAF for experimentation, test, and demonstrations.

Web-Based 3D Viewer Data Model

We analyzed DIS as a potential data model between the Exercise Service and 3D Viewer browser client since it is used widely in the MS&T community and supports the necessary data for a 3D viewer. However, DIS PDUs contain more data than necessary to render entities and battlefield effects in a 3D viewer. Instead of using DIS as-is, we developed a custom data model using Virtual DIS (VDIS) as a starting point. This custom data model enabled optimizations in data size, and allowed resource intensive calculations to be performed on the server, instead of in the light weight browser where resources are constrained. In addition, using a custom data model definition maintains independence from existing simulation protocols.

We selected the Google protocol buffers (protobuf) software library as the mechanism to represent the custom web-based 3D viewer data model. Google protocol buffers allow definition of a data model independent of programming

language, providing language and platform flexibility. For webVeritas, it outputs data model objects to JSON, our chosen data model format.

Communication Protocols

Communication between the client and server is done via REST (data upon request only) and WebSockets (2 way communications). The Exercise Service uses the DIS Listener module to capture DIS traffic from the network, then the DIS Translator module converts the DIS information into viewer specific data, which is then streamed to the browser 3D viewer client via websockets. The Exercise Service provides a REST API to allow the browser 3D viewer client to request information such as simulation time, list of terrains, and saved locations as needed.

The 3D browser client retrieves information from the Terrain and Model servers through standard HTTP requests. The Terrain Server implements the Tile Map Service (TMS) specification, allowing the 3D client to retrieve both imagery tiles as well as terrain height maps via a REST API that takes a zoom level and an x/y grid specified in the request URL. For imagery, a jpeg or png file is returned for the given request, and for terrain heights a file containing height data in the "heightmap-1.0" format is returned for the queried area. 3D models files (terrain features, entities and animations) are retrieved by URL by the client software using a plain HTTP GET request. File return types depend on the type of resource, e.g. glTF for entity models and b3dm for terrain features.

TERRAIN ELEVATION, IMAGERY, AND 3D MODELS

Terrain Elevation

Elevation data is rendered by the 3D viewer as terrain height maps in the Cesium heightmap-1.0 terrain format. We used SE Core provided DTED level 2 elevation data with a post spacing of 30m in GeoTIFF format. The SE Core elevation data is converted by dividing it into smaller tiles that can be easily streamed from the terrain server to the thin client. The conversion is done off-line and the data is stored on a server to be accessed by the Terrain / Model Server and streamed to the thin client as the eyepoint is moved. Elevation data is cached by the thin client along with other data streamed by the Terrain / Model Server.

Imagery

A 3D viewer must display realistic ground surface textures to support exercise monitor and AAR activities. Cesium uses imagery to display ground surface textures. Depending upon the use case, different imagery resolution may be needed. For example, a fixed-wing flight simulator is typically operating at high altitudes, so low-resolution imagery may be sufficient. However, ground, or near ground simulations like the Close Combat Tactical Trainer (CCTT) and the Aviation Combined Arms Tactical Trainer (AVCATT) will require high-resolution imagery so ground textures look realistic. Terrain imagery is draped over terrain elevation height maps. When webVeritas is started, a virtual globe is visible in the thin client. We are using Blue Marble imagery at 500m resolution for the virtual globe. This enables the operator to go anywhere in the world and have terrain even if it's very low resolution when zoomed in. A user can provide high resolution imagery for locations on the globe where needed for training. For our research and experimentation, we leveraged SE Core correlated terrain products such as satellite and synthetic imagery, elevation data, OneSAF Terrain Format (OTF), and OpenFlight terrain data.

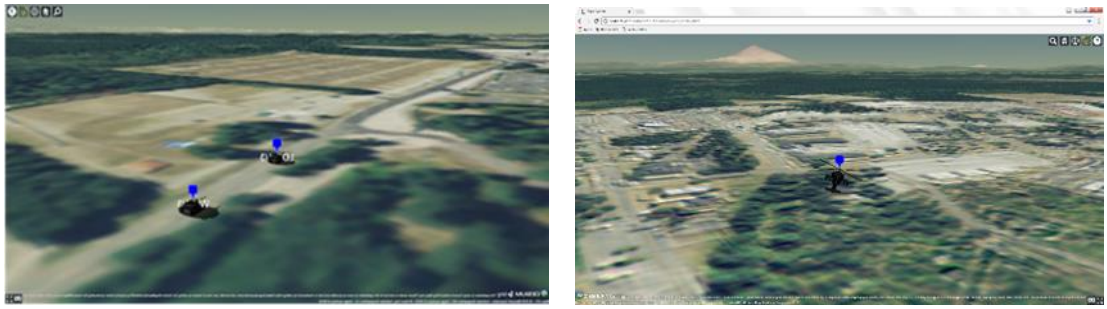

Figure 6 JBLM Satellite Imagery

In our initial experiments, we used the SE Core Joint Base Lewis-McChord (JBLM) data. The JBLM satellite imagery is provided at approximately 2m pixel resolution. Our evaluation of the satellite imagery found it to be unsuitable for ground- or air-based simulations because the imagery becomes stretched, blurry, and unrecognizable when viewed close to the terrain surface (see images in Figure 6).

SE Core produces synthetic imagery which can be produced at much higher resolutions than available satellite imagery. Synthetic imagery has no shadows, clouds, vehicles, and when zoomed in close, no tree or building images. Additionally, synthetic imagery correlates closely with other SE Core runtime formats because they are all produced from the same source data. We considered streaming performance and storage requirements for the synthetic imagery, and evaluated different resolutions to see which might be sufficient for viewing ground based simulations. We integrated 2m aerial imagery covering the entire database, 0.5m resolution ground surface imagery for an inset area, and 0.1m resolution ground surface imagery for a smaller insert area. The 2m aerial imagery included tree and building images which enables the viewer to render the scene faster because it is only rendering imagery and not 3D models of the tress and buildings.

Our evaluation of synthetic imagery found that the 2m resolution was of similar quality overall as the satellite imagery. However, it was better than satellite imagery because it correlates with other SE Core runtime formats and did not have clouds, shadows, and other artifacts often found in satellite imagery. The 0.5m ground surface imagery looked much better than the 2M aerial imagery and the 0.1M looked much better than the 0.5m ground surface imagery. The 0.1M imagery is of similar quality as terrain textures found in SE Core OpenFlight terrain databases. See the images in Figure 7 for comparison between the different synthetic imagery pixel resolutions.

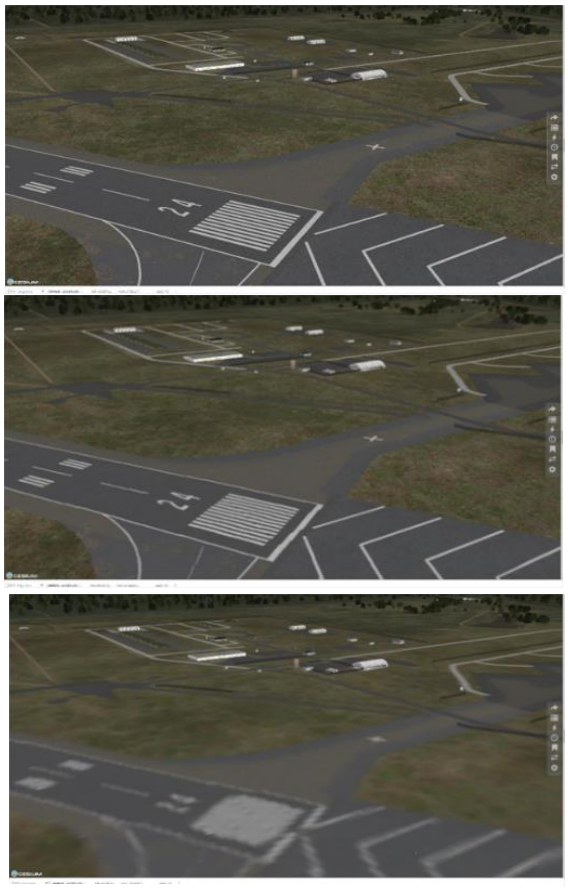

Figure 7 Synthetic Imagery (0.1m, 0.5m, 2m resolution)

3D Models

The thin client viewer renders 3D models of terrain features such as trees and buildings. As the operator moves the viewer eyepoint around the synthetic environment 3D models are streamed from the Terrain/Model Server to the thin client. For webVeritas we extracted tree and building 3D models from SE Core OpenFlight terrain and put them into the Cesium 3D tile format using an off-line conversion tool. The 3D thin client viewer renders the visual models for all entities and objects received from the Exercise Service. When the thin client receives an entity from the Exercise Service it retrieves the corresponding 3D visual model from storage in the Terrain / Model Server. The Terrain / Model Server stores all 3D visual models for vehicles, objects, munitions, etc. that are to be rendered in the simulation battlespace. The thin client is built on the Cesium framework and requires the 3D models to be in the format Cesium uses to render moving models which is glTF.

PERFORMANCE

We captured webVeritas performance metrics throughout development. Our objective was to meet performance numbers of the desktop Veritas application. We focused our performance metrics on thin client CPU usage and frames per second (FPS). We found that webVeritas performed as well as the desktop application when panning and zooming the terrain without entities. During an exercise, webVeritas performance varied depending on the number of entities and battlefield effects rendered by the thin client. See the table below for a comparison between desktop and web-based Veritas running the same scenario on the SE Core Camp Grayling terrain. Additional performance testing and optimization will be needed to fully support MS&T programs.

Performance Test	Desktop FPS/CPU	WebVeritas FPS/CPU
No entities panning/zooming	60/3%	50-60/3%
25 Moving ground vehicles with dense forest and buildings	60/5%	20/13%
Engagements fire and smoke effects in dense feature area	60/6%	10-15/13%
Ground vehicle with articulation in view w/other vehicles outside view	60/4%	30-40/10%
Single RWA in view of camera with other vehicles near	60/4%	30-45/9%
10 RWA with moderate building and tree feature density	60/4%	20/12%
Tethered to flight of 5 moving RWA. Dense trees & buildings	60/6%	40-50/9%

TECHNOLOGY TRANSITION

One of the objectives for this research project was the transition of developed technologies to US Army MS&T PORs with 3D viewer requirements. To this end, we developed the web-based 3D viewer with open source technologies, Army MS&T common components (e.g., SE Core terrains, 3D models, and V-DIS), and delivered it with Government Purpose Rights to reduce many of the traditional barriers to technology transition to MS&T programs. The maturity level of the software produced was at TRL 6 and included, design and test artifacts designed to enable easier transition.

PM ITE targeted LVC-IA POR as the first adopter of the webVeritas research product since many of the web-based 3D viewer research requirements were derived from LVC-IA 3D viewer requirements. LVC-IA v2.0 is currently deployed with the desktop version of Veritas.

Throughout development of webVeritas, our team met with the LVC-IA development team. We held periodic technical exchange meetings to describe our architecture, design, and implementation approach, and elicit feedback to ensure we kept implementation on track to be a candidate for transition to LVC-IA. We participated in LVC-IA design reviews and provided technical guidance on their approach for connecting the web-based AAR to webVeritas for AAR playback.

In addition to transitioning this research product to the LVC-IA POR, PM ITE also used the results of this research to feed the development of a 3D viewer sharable asset for the PM ITE enterprise. The 3D viewer sharable asset is one of several sharable assets being developed under a PM ITE product line engineering approach that provides common simulation & training capabilities across the synthetic (V, C, G) systems portfolio. The PM ITE product line approach is part of a larger enterprise modernization initiative known as the Synthetic Simulation Transformation (S2T) whose purpose is to facilitate communication with developers, users, and our industry partners to support standards and common solutions for synthetic simulation. S2T portal can be found at https://www.s2tportal.mil/

PROOF OF CONCEPT DEMONSTRATION

As a proof-of-concept we wanted to demonstrate the web-based 3D viewer prototype in a relevant or operational environment. The Army LVC-IA system provided an ideal operational environment to demonstrate the webVeritas capabilities. The demonstration showed how the webVeritas prototype capabilities met the requirement set and provided evidence that the webVeritas 3D Viewer is ready to be transitioned and productized by a Program of Record. We worked closely with the LVC-IA development and test teams to install and configure webVeritas, develop and test the demonstration scenario, and execute the demonstration. We installed the Exercise Service and Terrain/Model Server on two Linux Virtual Machines (VMs) on LVC-IA servers. The demonstration included three core systems—CCTT, AVCATT, and Games for Training (GFT) VBS3. We ran the webVeritas thin client 3D viewer in a FireFox browser on a Windows PC workstation. Finally, we also ran the desktop Veritas on a separate workstation to show a comparison of capabilities between the current 3D viewer and webVeritas.

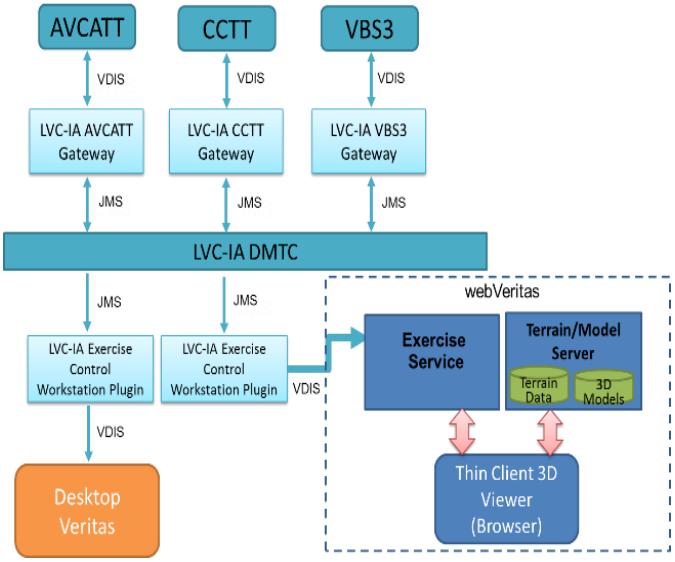

Figure 8 Proof-of-Concept Demonstration Setup

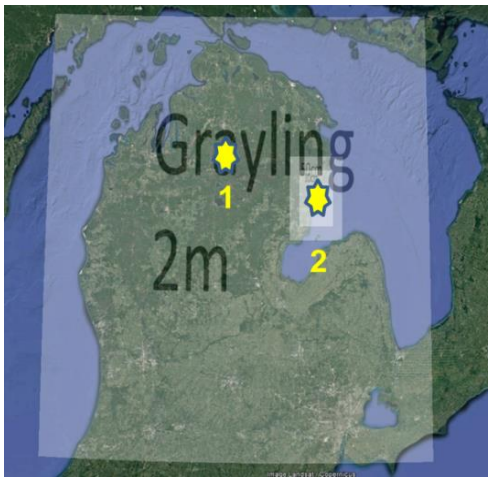

Figure 9: Demonstration Locations

The demonstration scenario ran on the SE Core Camp Grayling terrain. Figure 9 illustrates the demonstration locations on Camp Grayling. The demonstration scenario included approximately 100 entities from GFT, CCTT, and AVCATT executing at two separate areas on the Camp Grayling terrain. The image below shows the two areas used in the demonstration exercise. In the first scenario, AVCATT Blackhawks flew from the MOUT site to the location where two GFT DI were waiting. The GFT DI was mounted and was flown to the MOUT site to support CCTT Blufor infantry to engage with CCTT hostile forces. The second scenario took place on the east coast of Michigan in the 0.1M resolution inset. This showed the high resolution synthetic imagery, allowing comparison to the low resolution 2m synthetic imagery where the first scenario occurred. The scenario consisted of approximately 90 CCTT and AVCATT entities. First, we ordered AVCATT Apaches to fly to a location outside an airfield and engage with CCTT hostile ground forces occupying the airfield. Next, we showed a CCTT M1A2 platoon maneuvering through a built-up area. Lastly, we demonstrated a large AVCATT and CCTT Blufor and Opfor engagement in an open location north of the airfield. The demonstration successfully proved we met our research objectives and goals to develop a web-based 3D viewer that runs in an operational environment, uses common Army MS&T products and standards including SE Core terrain and 3D models, and exchanges simulation events via the V-DIS protocol.

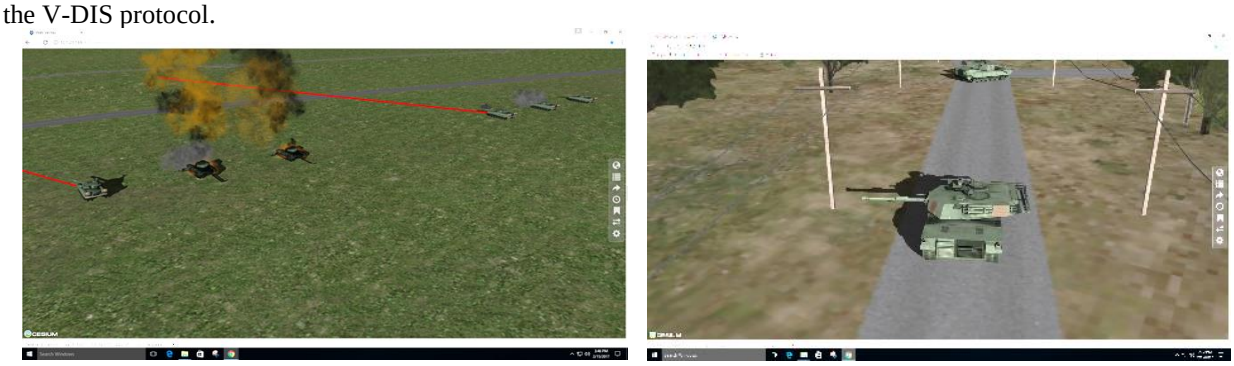

Figure 10: Proof-of-concept demonstration

LESSONS LEARNED

Throughout the research project we learned a number of lessons that will help us improve how we execute future research and development efforts, and help us improve the web-based 3D viewer prototype in follow-on efforts. In addition, these lessons learned can help others doing similar research. In this section, we share some of the lessons learned on this project.

Cesium Capabilities

Our selection of Cesium as the thin client 3D render framework provided time and cost savings because it already had many capabilities needed for our thin client viewer, and has been leveraged on related work [4]. Leveraging Cesium's existing functionality allowed us to focus on other capabilities, and mature the prototype further than planned. For example, Cesium provided a Geocentric coordinate space API for interacting with objects and entities making it unnecessary to convert to a local coordinate space. The Cesium engine supports downloading model files in the background and loading them when ready. Cesium also provides an optimized capability to load imagery and elevation tiles. In addition, a basic tethering capability is implemented in Cesium, which we extended to include tether from the UI entity list. We leveraged existing Cesium code to provide battlefield effects like smoke, fire, and dust. Cesium also provides ground clamping of entities and objects. We estimate that using Cesium saved us over two months of research and development effort.

Actively Support Open Source Project

Cesium is an actively supported open source project with a strong user community. This allowed us to leverage the work of other developers using Cesium and share ideas and solutions to issues through online forums. We cloned the Cesium open source third party APIs in our local git repositories, which allowed us to mirror Cesium, collada2gltf, and other tools locally. We created branches when updates to these baselines were committed and easily downloaded upstream updates and merged with our local branches.

Cesium Elevation Tile Issues

We represent terrain elevation in the 3D viewer using Cesium elevation tiles. After implementing and testing with Cesium elevation tiles we found that random breaks in the terrain surface can be seen between tiles during pan and zoom operations close to the terrain surface. Creating the elevation tiles is a simple process, but when Cesium renders an upscaled elevation tile at higher zoom levels, there is an error with precision in the resulting vertices. On something as simple as a sloped terrain, the result is blocky with a vertex count that is higher than necessary and also resulted in cracks between the upsampled tiles. We investigated alternative solutions to using elevation tiles, such as converting elevation GDAL to Cesium quantized mesh tiles, which will be considered as a future improvement to webVeritas.

CONCLUSION

This research proved the viability of a light-weight, web-based 3D viewer, using open source technologies and leveraging Army MS&T common components and standards. Cesium proved to be a successful solution for rendering 3D content in a thin client 3D viewer. Leveraging SE Core terrain and model data ensures a high level of correlation and supports interoperability with existing and future simulation systems using SE Core products. A web-based 3D viewer is easier to install, maintain, and configuration manage than a traditional desktop viewer application. The result of this research effort is webVeritas, a GPR, TRL-7 tool that runs as a thin client in an internet browser to provide 3D visualization for Point-of-Need and Cloud-based training.

REFERENCES

1. Powell Jr, D. A. (2013). "The Military Applications of Cloud Computing Technologies," Army Command and general Staff College Fort Leavenworth School of Advanced Military Studies.
2. Bair, L., Fairchild, J., Elkins, S. (2016) "High-Fidelity Training on Demand via the Cloud," Modsim World Proceedings 2016.
3. Dukstein, G., Watkins, J., Le, K., and Gonzalez, H., (2013) *Extending construction simulators through commonality and innovative research*, in Proceedings of the Interservice/Industry Training, Simulation, and Education Conference, pp. 2355–2365, Orlando, Fla, USA
4. Santiago, F., Verdesca, M., & Watkins, J. (2012) Geospatial Correlation Testing Framework and Toolset. *Proceedings of I/ITSEC 2012*
5. Cesium website: https://cesiumjs.org/