ARL-TR-8810 ● SEP 2019

DEVCOM
ARMY RESEARCH LABORATORY

Visualizing an Active Protection System Monte Carlo Simulation

by Vincent Perry, Michael Chen, Russell Binaco, Wendy Gao, and Simon Su

ARL-TR-8810 ● SEP 2019

Visualizing an Active Protection System Monte Carlo Simulation

Vincent Perry and Simon Su
Computational and Information Sciences Directorate, CCDC Army Research Laboratory

Michael Chen
Weapons and Materials Research Directorate, CCDC Army Research Laboratory

Russell Binaco
Rowan University

Wendy Gao
University of Maryland, Baltimore County

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**			
1. REPORT DATE *(DD-MM-YYYY)* September 2019	**2. REPORT TYPE** Technical Report		**3. DATES COVERED (From - To)** May 2018–July 2019
4. TITLE AND SUBTITLE Visualizing an Active Protection System Monte Carlo Simulation			**5a. CONTRACT NUMBER**
			5b. GRANT NUMBER
			5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Vincent Perry, Michael Chen, Russell Binaco, Wendy Gao, and Simon Su			**5d. PROJECT NUMBER**
			5e. TASK NUMBER
			5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) CCDC Army Research Laboratory ATTN: FCDD-RLC- S Aberdeen Proving Ground, MD 21005			**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-8810
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			**10. SPONSOR/MONITOR'S ACRONYM(S)**
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT			
13. SUPPLEMENTARY NOTES ORCID ID(s): Vincent Perry, 0000-0003-0429-1813; Simon Su, 0000-0002-2460-3899.			
14. ABSTRACT A recent undertaking by the US Combat Capabilities Development Command Army Research Laboratory to create a Vehicle Protection Software Suite for the modeling and simulation of active protection systems (APSs) has spawned many data-analysis efforts to better understand these systems. Furthermore, the software is being written to utilize the high-performance computing resources provided by the Department of Defense Supercomputing Resource Center. This report discusses a number of concurrent developmental efforts by the Data Analysis and Assessment Center to visualize an APS Monte Carlo simulation as well as improvements to current visualization applications to enable further analysis when the software suite matures. The visualizations include data-driven 3-D animations of the APS simulation as it plays out, 2-D charts and graphs for variable analysis of the simulation data, and a 2-D to 3-D hybrid visualization framework to coordinate different views of the entire data set for an all-encompassing analytical workflow. This report documents the visualization applications and some of the technical underpinnings to enable the data-analysis capabilities.			
15. SUBJECT TERMS visualization, simulation, high-performance computing, HPC, Monte Carlo, active protection system			
16. SECURITY CLASSIFICATION OF:	**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Vincent Perry
a. REPORT Unclassified / **b. ABSTRACT** Unclassified / **c. THIS PAGE** Unclassified	UU	31	**19b. TELEPHONE NUMBER (Include area code)** 410-278-4896

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

Acknowledgments

This work was supported in part by the Department of Defense (DOD) High-Performance Computing Modernization Program at the US Army Research Laboratory, DOD Supercomputing Resource Center.

1. Introduction

The US Army Research Laboratory* Department of Defense Supercomputing Resource Center (ARL DSRC) provides high-performance computing (HPC) resources and support to the Army's research, development, test, and evaluation communities.[1] Within these scientific communities, data analysis and assessment are vital components of the scientific process. Without understanding the data, there is little to conclude from these modeling and simulation efforts. Enabling scientists and researchers to gain insights from their data through visualization and interaction is an important component of the DSRC Data Analysis and Assessment Center (DAAC) operations. The DAAC team provides this analytical and assessment capability to users in many different forms, from traditional scientific visualization applications such as ParaView, Ensight, and Visit, to remote desktop environments for a familiar user interface (UI) to an HPC environment.[2] In the past couple of years, the DAAC team has been investigating moving beyond traditional scientific visualization capability to accommodate more unstructured data. This entails both 2-D visualizations that coordinate similar information within a data set, as well as 3-D visualizations that demonstrate how the data appear in a world-like environment. Furthermore, these capabilities can be coordinated across dimensions to provide a visual-analytics capability, where the user drives the data-analysis workflow by interacting with the visualization applications.[3]

One particular use case for this visualization capability is an active protection system (APS) Monte Carlo simulation. An APS is a defense system for Army tanks and other armored vehicles, and the Monte Carlo simulation enables scientists to better understand how the APS will perform. The ARL architecture for simulating and evaluating the performance of the APS is the Vehicle Protection Software Suite (VPSS), which has grown out of the legacy code called System Survivability Engineering Software.[4] The VPSS is currently under development to expand simulation features used for characterizing protection technology's performance against adversarial threats. Furthermore, the simulation software is being written to run on the HPC resources supported by the DSRC for both scalability and sustainability into the future. The APS refers to the response mechanisms equipped to a tank while VPSS refers to the simulation software encompassing all of the different aspects of an APS. The lifecycle of an APS is shown in Fig. 1, including the launch and detonation of the countermeasure (CM) weapon. Ultimately, the

* Some of the work outlined in this report was performed while the US Army Research Laboratory (ARL) was part of the US Army Research, Development, and Engineering Command (RDECOM). As of 31 January 2019, the organization is now part of the US Army Combat Capabilities Development Command (formerly RDECOM) and is called CCDC Army Research Laboratory.

goal is to visualize the entire life cycle including post-engagement residual-threat analysis using the data derived from the Monte Carlo simulation. More information on the APS project may be referenced in these reports.[4–7]

In this report, we present our data-analysis support for the APS project. This includes improvements to a 3-D playback animation of the simulation data and enhanced filtering capability to visualize relevant subsets of the data set on 2-D charts, as well as improvements to our hybrid visualization framework to enable coordinated data analysis across the 2-D–3-D threshold.

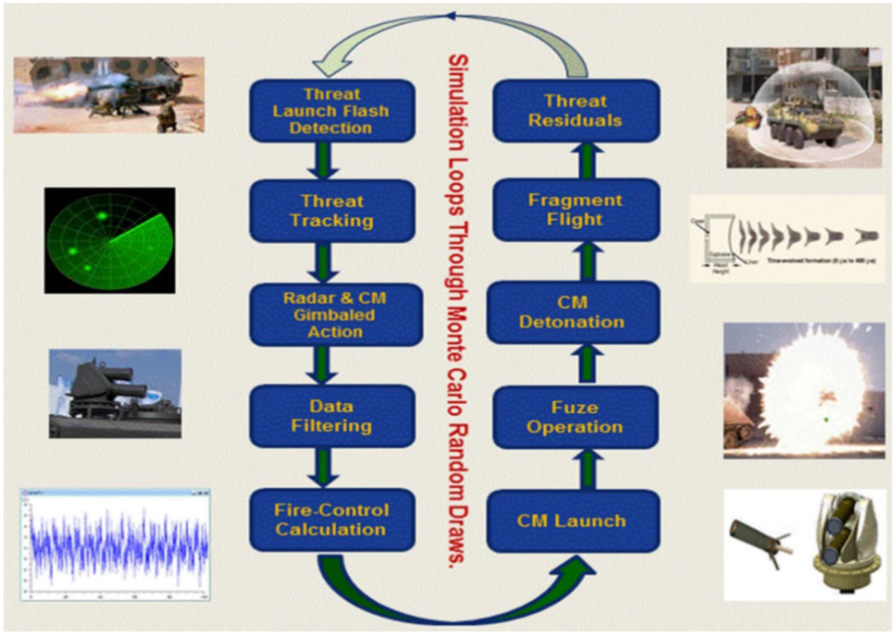


Fig. 1 Engagement life cycle of an APS

2. 3-D Visualization

At the inception of the project, the DAAC team inherited the previous data viewer, which was an OpenGL application over a decade old. The data viewer would read in the data file from the Monte Carlo simulation and animate the engagement life cycle of an APS. This involves an incoming threat trajectory toward a tank, a CM being fired from the tank once the threat is detected, and the CM engaging the incoming threat, fragmenting the threat by detonating within the vicinity. As an initial data-analysis capability, the DAAC team sought to update the current visualizer with an updated graphics engine. As mentioned in Graham's contractor report,[7] the DAAC team was given data and 3-D models of the simulation to create higher-fidelity 3-D visuals than the one previously used. The first game engine used was Unity, followed by Unreal Engine 4 (UE4) for more impressive graphics.

2.1 Unity Playback Application

Unity is an application development platform that allows the import of 3-D assets and models into a 3-D environment and manipulating those assets via scripting and interaction.[8] In the development of the Unity-based visualization application, we used the 3-D models provided by our collaborator and developed C#-based animation scripts to animate the 3-D models using the positional and orientation data from the simulation data. We started the development process by loading the 3-D models provided into Unity to ensure they were importable with the correct scale and dimensions. Figure 2 shows a scene from our Unity-based application illustrating one time step from the data file.

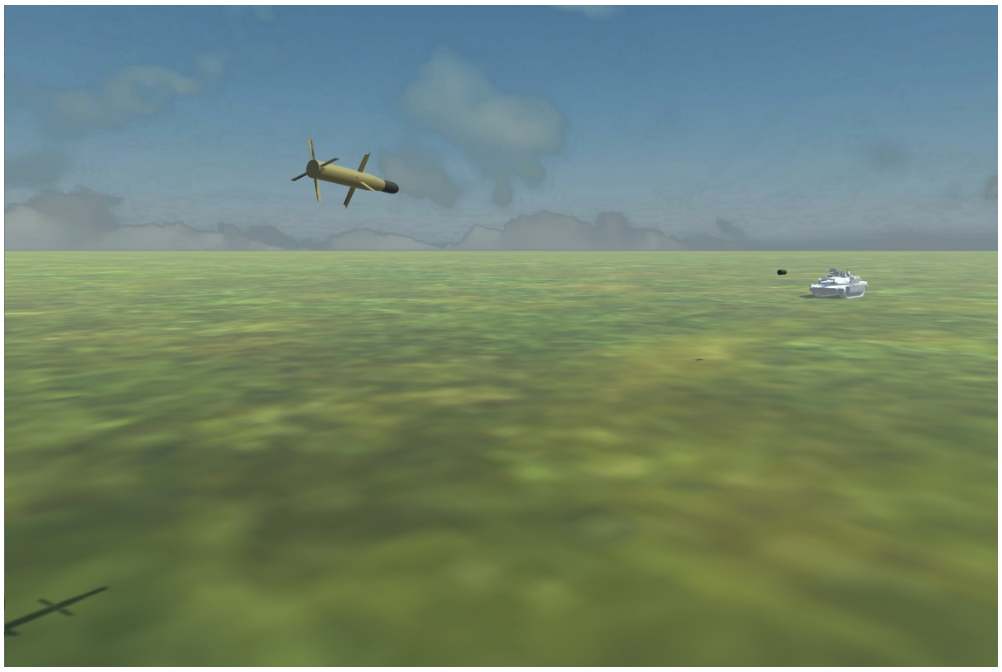


Fig. 2 Unity application showing a single animation time step based on the output of the Monte Carlo simulation run

The threat projectile and tank model had to be manipulated within the 3-D environment to correctly rotate and scale them to a world-like environment. Furthermore, as the tank model was made up of many pieces, we had to manipulate the model by using a script to iterate through the pieces separately. We had to recreate the CM model in Unity, as it was not provided by our collaborator. We also used a skybox model that was provided to add realism to the animation, and the terrain model was modeled and textured using one of the image assets provided.

The next step in the development was to correctly animate the scene using the simulation data provided. The data file provides the timing of the simulation together with positional and orientation data for all the 3-D objects in the simulation. The entire data file was processed as a comma-separated values file into arrays in Unity with all of the data from the same reference frame stored accordingly.

The main challenge is the synchronization of the time provided by the simulation into animation timing in Unity. Although the data contain the timestamps for each simulation step, Unity uses timing differently, using frames and pauses in loop iterations in the scripts. We had to track Unity time and compare it with the simulation time in order to synchronize the two times manually. Although we are not using the simulation time to drive the timing of the Unity animation at this time, each 3-D object moves according to its coordinates from the data set.

For each of the Monte Carlo runs the entire animation is for a few seconds' duration, so we gave the user the ability to slow the animation in Unity to allow them time to analyze the simulation. The user can control the timing of the animation by pressing "s" for slower or "f" for faster animation speed. We also included the "r" or reverse command to go backward through the data set at the same rates of speed that was set. The "p" key will toggle the pause and play of the animation. We also included different observation points in the scene for the user and the "c" button changes the view of the user in the scene to follow the missile or the canister. Furthermore, we added the ability to view the animation within virtual reality (VR).

Everything the user sees in the animation up to this point is based on the data from a scientifically correct simulation output. However, we do not have data for the detonation of the CM and its fragmentation of the incoming threat. Instead, we implemented the CM's detonation using a particle system for aesthetics. Figure 3 shows use of the Unity particle system to show the CM's detonation. Instead of animating many small spheres and scripting them, it is quicker and less resource heavy to implement a particle effect. To maintain the scientific accuracy up to the point of the particle-system effect, we only trigger the particle system when the data from the simulation indicate the CM detonates.

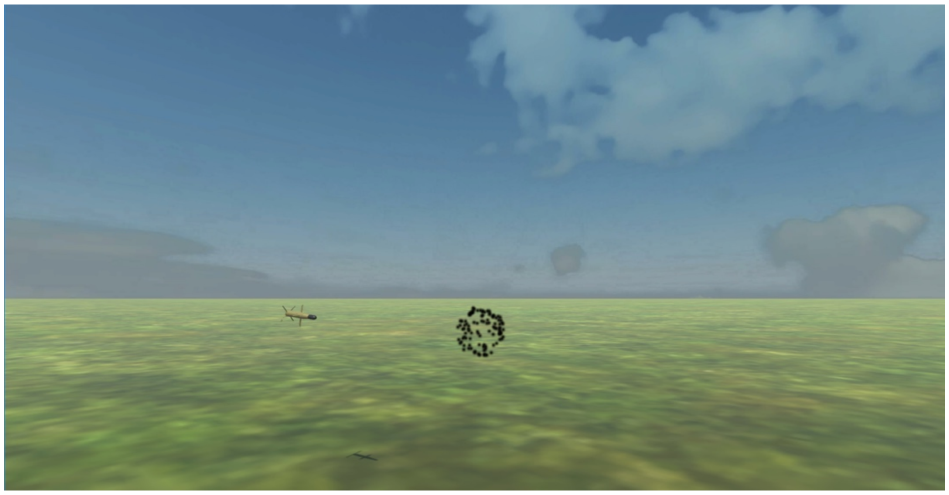


Fig. 3 Unity application showing the use of the particle system to illustrate the CM's explosion

2.2 UE4

As mentioned previously, the DSRC DAAC team historically has used the Unity game engine to visualize 3-D environments and simulations. However, within the past couple of years UE4 has become Unity's number one competitor. UE4 is an open-source game engine that has been gaining in popularity and user base, with a growing developer community. The transition to UE4 from Unity was due to its rapid development workflow, better quality, and higher resolution graphics. UE4 focuses on visual scripting through blueprints instead of traditional programming, which is a process described at great length within its documentation.[9] Features of UE4 include the integrated physics engine with various adjustable factors such as gravity, force, mass, and object collision, as well as the integrated Apex Destruction plugin that takes a static mesh and fractures it to form a destructible mesh. We utilize both the physics engine and Apex Destruction plugin during the formation of the engagement and post-engagement phases of the simulation.

When transitioning our visualization from Unity to UE4, we originally sought to recreate the Unity playback application described previously. This proved relatively straightforward, utilizing UE4's data-table structures to read and store the data set and animating the 3-D assets in the scene accordingly. For the CM's detonation, the physics engine and Apex Destruction plugin replicate the particle system used in the Unity application. We were then able to view the animation in VR using the OculusVR plugin for UE4 and use the hand controllers for movement and interaction. In total, recreating the Unity application in UE4 was a rapid development workflow that produced very promising results. Figure 4 shows the APS visualization from all three applications.

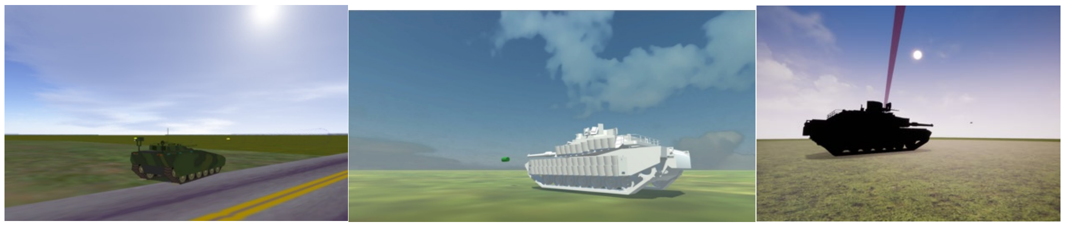


Fig. 4 OpenGL application (left), Unity application (middle), UE4 application (right)

Once we recreated the existing application, we chose to continue iterating upon our data-analysis support for the APS project.[10] Instead of just showing the APS animation playback of a single threat, we created a visualization to show the entire engagement life cycle of an APS including post-engagement fragmentation. This includes multiple land assets and incoming threats, multiple different kill measures, pre-engagement radar detection of threats, and post-engagement fragmentation containing residual-threat analysis. At this time, the simulations to produce the data for these components are under development. Thus, none of these components of the application were data-driven, but instead stubbed to be able to read data in from the VPSS simulations when the data become available.

2.2.1 Full Engagement Life Cycle

In a typical engagement scenario, friendly tank(s) and other units are in formation and encounter one or more threats launched by hostile forces. The two common threats an APS is designed to counter are rocket-propelled grenades (RPGs) and antitank guided missiles (ATGMs). The engagement life cycle begins in pre-engagement, which is composed of threat detection and tracking as well as targeting systems. When an appropriate response has been derived in the pre-engagement phase, a CM is launched that can be either hard kill or soft kill. The hard-kill mechanisms use an explosive canister that will physically intercept the threat via detonation, whereas the soft-kill mechanisms use jamming or other nonlethal technologies to neutralize the threat. After this engagement has occurred, there is a post-engagement phase where residual fragments from the neutralized threat can still potentially cause damage to friendly units.[11]

The engagement scenario for the application is an array of tanks and accompanying forces moving forward in battle formation that encounter enemy threats. Figure 5 shows the friendly units in this formation, as well as the text interface that displays how many times the vehicles have been hit by missile fragments in the post-engagement phase.

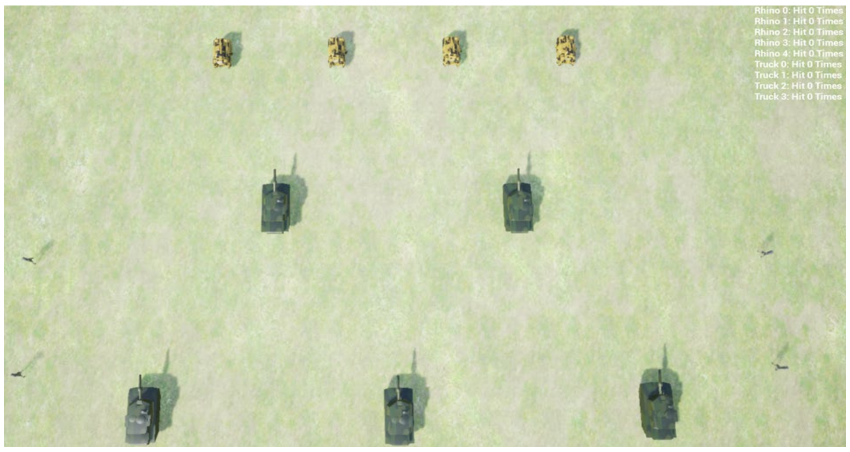


Fig. 5 Pre-engagement unit formation

In this simulation, a missile appears every 5 s in the field in front of the units and randomly targets a tank. When the missile spawns, the radar system first detects the threat. Currently, the missiles spawn in random locations, providing variation as to when the radar system detects them, but this can be modified to more realistically match the behavior of any existing radar technology should that information be provided. Once the targeted tank's radar detects the incoming threat, the tank responds with one of three CMs. Figure 6 presents these responses.

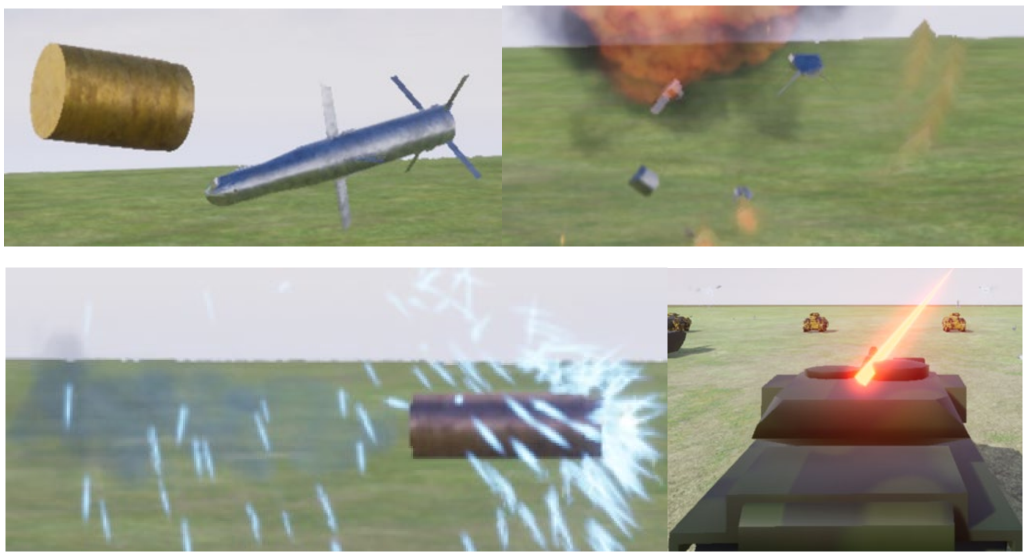


Fig. 6 Three CMs in the engagement phase: hard-kill canister (top left) and canister detonation (top right), flare (bottom left), and soft-kill electromagnetic pulse (bottom right)

The top of Fig. 6 shows a hard-kill CM before and after detonation. The hard-kill CM is a canister that targets the missile and explodes in close vicinity, causing the missile to fragment. The bottom left of Fig. 6 shows a flare CM, which is meant to circumvent the targeting system of the guided missile and causes the missile to reroute and target the flare instead of the tank. Upon colliding with the flare, the missile fragments. Lastly, the bottom right of Fig. 6 shows a soft-kill mechanism, demonstrated by a laser representing an electromagnetic pulse that disables the missile. The missile's targeting system becomes disabled, causing it to fall directly to the ground and fragment upon impact. After these three cases, the post-engagement phase consists of the fragments' trajectory. Figure 7 shows fragments lying on the ground as a result of the engagement.

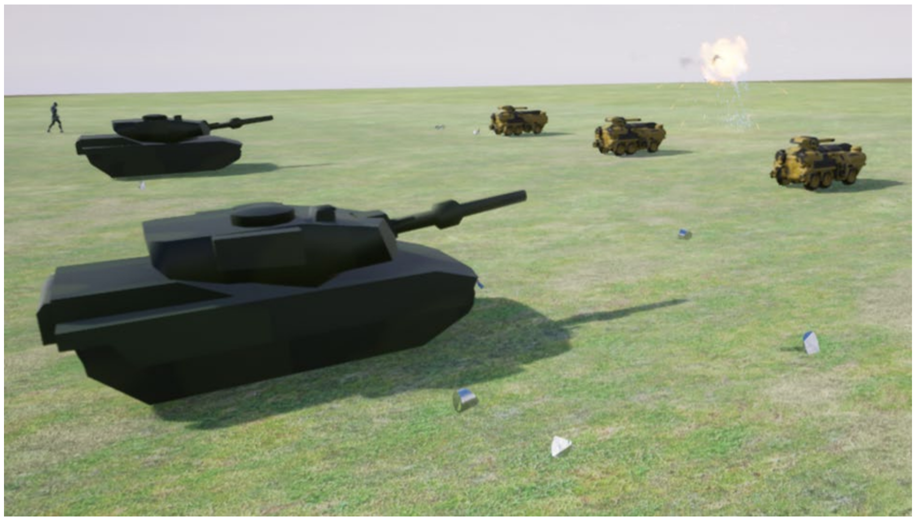
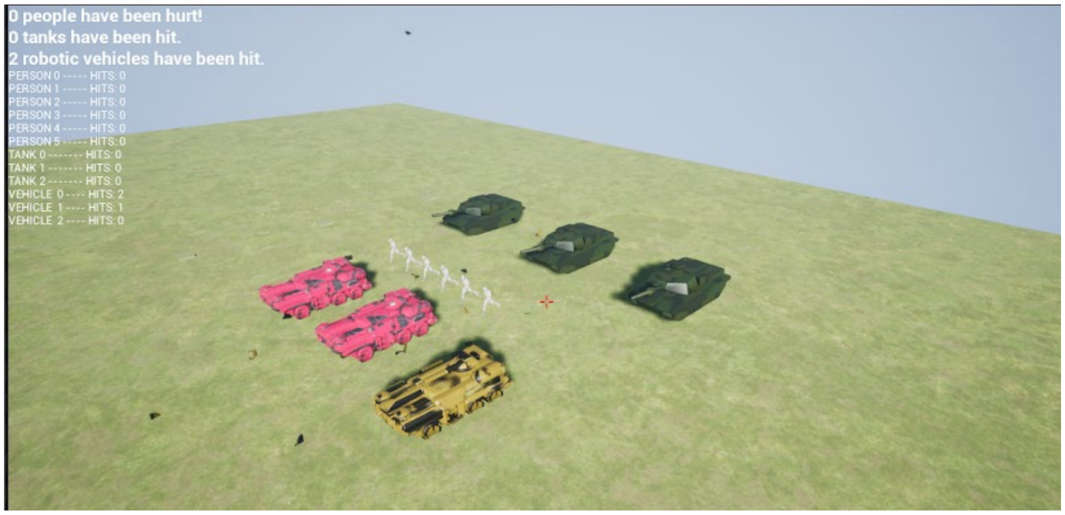


Fig. 7 Post-engagement phase of engagement life-cycle simulation

2.2.2 Post-engagement Threat Analysis

A visual casualty-analysis system was developed to analyze the collision of the hostile projectile's debris and is shown in Fig. 8. As the friendly forces are struck by the threat residuals, the material of the vehicle or personnel changes to become red temporarily, clearly displaying where fragments have collided with the units. A quantitative visual in the top-left corner displays the total number of each type of friendly unit that is hit, along with the number of times each individual unit is hit. Other features of the application include a simulation pause, user movement with keyboard and Oculus Rift controls, and partial time compression to allow the user to observe the environment in slow motion while moving at a normal speed.



```
0 people have been hurt!
0 tanks have been hit.
2 robotic vehicles have been hit.
PERSON 0 ------ HITS: 0
PERSON 1 ------ HITS: 0
PERSON 2 ------ HITS: 0
PERSON 3 ------ HITS: 0
PERSON 4 ------ HITS: 0
PERSON 5 ------ HITS: 0
TANK 0 -------- HITS: 0
TANK 1 -------- HITS: 0
TANK 2 -------- HITS: 0
VEHICLE  0 ---- HITS: 2
VEHICLE  1 ---- HITS: 1
VEHICLE  2 ---- HITS: 0
```

Fig. 8 Visual analysis of threat residuals colliding with friendly units

This application focuses on the optimization of the CM's interception for the least-incurred damage to the friendly forces. Because the simulation data are not yet available, we were able to manipulate variables such as the force and height of the threat explosion and the angle of approach of the threat to gain variability in potential approaches of threat residuals. Fifteen and 25 m from the formation were used to display the range for an RPG, while 50 and 75 m demonstrated the range for an ATGM. Currently, the threat–CM engagement is emulated with UE4's physics engine, but it has demonstrated the ability to manipulate objects based on given data, as well.

Currently, the post-engagement threat analysis consists of logging the number of collisions between threat fragments and friendly units. Given additional information about the durability of the vehicles and impact damage of missiles, analysis can be integrated into the simulations to provide a more detailed threat assessment. An animation pause, partial time compression for slow-motion, and user movement are included features of the application. The user is also able to view and move around the 3-D environment within VR, allowing the analyst to view the animation and engagements from different angles and distances. In addition, the keypress "o" produces a JavaScript Object Notation (JSON)-structured log file, for further analysis, that reports when each threat was engaged.

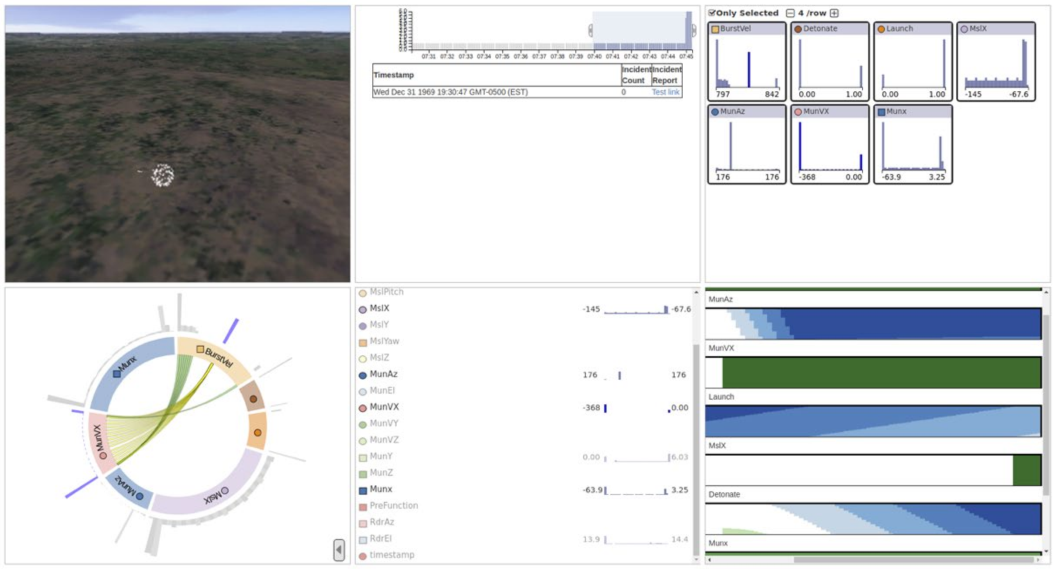
3. 2-D Visualization

Although the original data viewer provided for the APS simulation was a 3-D playback animation driven by the data file, there are other ways to analyze data beyond just viewing a simulation realization. With exploratory data-analysis problems involving many variables of heterogeneous types, it is useful to have several charts showing different angles of the overall data. Further, creating a full visual analytic solution involves the interoperation of a collection of charts and visualizations that represent the same slice, combination, or configuration of data that the analyst wishes to see. This enables the synchronization of each view such that any user action in one view changes all others, guaranteeing a unified representation. This is referred to as brushing and linking.[11] Charts are linked because they are a connected to the same data set, and when that data set changes or is filtered, those changes are then brushed across all of the charts and graphs. Thus, to enable a more complete data-analysis capability for the APS simulation, we created a 2-D visualization application containing multiple coordinated views for the analyst to further inspect the simulation data.

3.1 SyncVis Application

In earlier work, we created a front-end application layer on top of the ParaViewWeb[12] framework, which we named SyncVis.[13] We used ParaViewWeb because its InfoViz application programming interface (API) supports interactive coordination across multiple views, and also supports linking of common data across multiple charts. SyncVis is a web-based application that contains a composite of different visualization tools and components meant for rendering simulation data. It is a prototype data-visualization tool that integrates dc.js,[14] Unity, and ParaViewWeb to visualize data through multiple 2-D visualizations. All of these visualizations are linked to the same data server such that whenever a data filter is applied to the main data server, the brushing of those data is coordinated server-side and applied to all of the synced visualization tools. The web page itself is organized through React and has six different application boxes. The boxes are spread evenly across the webpage in a grid (Fig. 9). Each box and its respective charts are set up in a different way, but they are all united because they all receive their data from the same underlying ParaViewWeb server. In order to communicate with the server, SyncVis uses a one-way Remote Procedure Call (RPC) to send and request data. These are all done through a separate communication file that allows SyncVis to coordinate with other client applications.

ParaViewWeb contains provider modules that facilitate data flow from data source to visualizations. That is, each provider module is responsible for managing a particular piece of data and/or part of the application state, potentially coming in remotely, and interested visualization components are notified when there is a change in state through a publish/subscribe approach. In addition to the provider design, the InfoViz API also includes a collection of useful charts for visualizing more complex correlations between variables.



Timestamp	Incident Count	Incident Report
Wed Dec 31 1969 19:30:47 GMT-0500 (EST)	0	Test link

Fig. 9 SyncVis showing multiple coordinated views of 2-D visualization components: Unity playback animation (top left), a UI for time-range selection (top center), histogram of selected variables (top right), chord diagram of selected variables (bottom left), variable selector (bottom center), and cubism charts of selected variables (bottom right)

In addition to data management, SyncVis manages visualization-component rendering by using the React framework.[15] React makes it simpler to create composable UIs in a declarative fashion and also allows us to use state-of-the-art, open-source React-based visualization components more easily. To create new web-based visualization front ends, we would simply have to create a new provider module to deliver new types of data to the visualizations. Existing React-based visualization components can directly be used in SyncVis. ParaViewWeb composite providers are available throughout the React component hierarchy; hence, individual visualization components can subscribe to provider events to retrieve new data and update the state of the visualization component, which triggers React to re-render the visualization automatically.

3.2 Filtering Improvements

For the APS data it is helpful to understand both individual and pairwise relationships of variables from the simulation. For time-based simulations, a natural filtering method is based on time, where an analyst may choose to only look at a specific period of time within the simulation. Previously, the only filter the SyncVis application supported to filter the main data server was a time filter. However, the APS data set brought to light new variables that an analyst may wish to filter by, in particular binary variables. This section describes adding a new filtering method for binary variables in the data set instead of just the time stamp, as well as a way to filter by multiple variables at once.

For visualizing the binary filters, dc.js "donut charts" were used because they are useful for filtering data and the concept of "on" and "off" is very intuitive for analysts. On initialization, the server automatically calculates and records the binary variables of the data set. As described previously, the donut charts must communicate with the ParaViewWeb server through asynchronous RPCs, where a communications JavaScript file coordinates all of the RPCs to the server.[16] To accommodate this mode of communication, two new RPCs were added to the communications file for getting and setting the binary variables upon user interaction.

To interact with SyncVis, the variables may be selected by the user on the variable selector shown in the bottom center of Fig. 9. Whenever a variable is selected, the subsequent filter is applied if applicable. If the variable is binary, this filter simply combs through that variable's values and calculates the ratios for its Boolean values. Once the ratios are calculated, the ensuing donut chart will appear underneath the time filter as seen in Fig. 10. From there, the user can interact with SyncVis by selecting the donut chart. On selection, the newly set RPC sends the name of the selected binary variable and the section of the donut chart that was selected (i.e., "true" or "false") to the ParaViewWeb server for further filtering of the data set.

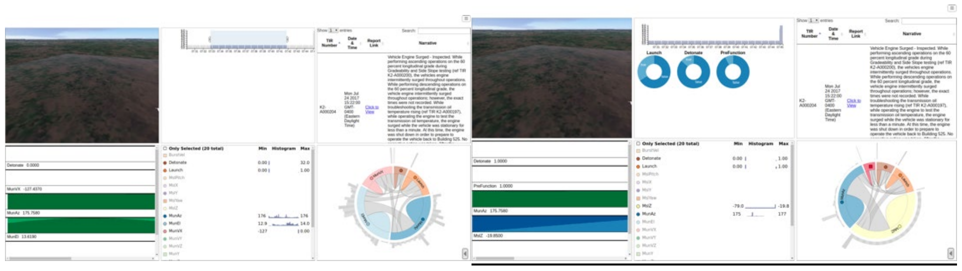


Fig. 10 SyncVis before donut charts added (left) and SyncVis with the addition of donut charts (right)

To include Boolean variable filtering in addition to the time range, filtering in SyncVis needed to be redesigned to accommodate the application of multiple filters to the data set. Instead of just the time filter, SyncVis needs the ability to cross-filter over multiple different data points. Previously, the time-stamp filter just removed the indices that did not correspond to the selected range of data in the data set. At first, the solution was to just manipulate this object by removing more data points from it, but it conflicted with other modules within the code base. Instead of this system, the time filter and donut charts were each given their own lists to filter the data into one master list of data points. These filters were originally intended to be composed of "true" and "false" values that determine whether the data point is included or excluded by the filter. However, when applying multiple filters, there was no way to efficiently reverse one filter while maintaining the structure of the other filters.

To allow for binary variable filters to stack on top of each other within one filter module, the filters were redesigned to be arrays of numbers in which 0 represents "true" and any number higher than 0 represents "false". The system is designed so that any new additional filters should act similarly. After each individual filter has been calculated, they are then compiled into a single abstract filter array. After all data points have been added to the array, the data are served to the client with the relevant filtering.

4. Hybrid Visualization

In a paper for the Software Engineering and Architectures for Realtime Interactive Systems workshop,[17] we described the creation of a hybrid visualization framework to visualize the APS simulation by coordinating both 2-D and 3-D visualizations. The APS use case requires that data be represented by both 2-D charts and graphs and 3-D representations that visualize the data's spatiotemporal dynamics. Further, there are benefits to viewing these data in an immersive 3-D VR environment, for it can allow for novel, more cognitively intuitive ways of interacting with the data. Previous work has used Unity to visualize the dynamics of an APS simulation and have this playback coordinated to a 2-D collection of data visualizations through a simple web interface for state communication. This section briefly discusses the previous implementation, followed by improvements made to the visualization framework to enable further data-analysis capability.

4.1 Previous Implementation

In terms of the 3-D visualization, the initial approach was simply to view the dynamics by playing back an entire simulation realization. The work developed a way for the simulation playback to coordinate with the SyncVis web-app dashboard and respond to a time-selection filter applied in SyncVis. When a time selection is applied, the Unity application would restrict the simulation data for that particular selected time range, allowing the user to view the corresponding clip of the animation on loop. For example, Fig. 2 in Section 2.1 shows a snapshot of a scene. While 3-D playback animation provides the user with a visual of the test scenario as it would play out in real time, it does not facilitate any exploration of the data other than watching the scenario play out. Unfortunately, there is no way for a user to interact with this animation other than starting or stopping the playback. In addition, there is not very much information to be gained from stopping the animation playback at one single time step. Thus, to support a more in-depth and useful data-analytics experience, we decided to represent the 3-D data in a different way.

In terms of communication of input and filter information between Unity and web application servers, this was previously done by exposing filtering information through a separate microservice as a Representational State Transfer (REST) API. In this setup, the data server, upon changes in the time filter range, would send a POST request to deliver this update to the microservice. The Unity application would constantly poll the API and determine when the time filter has changed. This design was not an ideal choice but was chosen due to lack of better options on the Unity side of implementation. In our reconfigurable visualization ecosystem,[18] state and data changes are normally communicated through the Web Application Messaging Protocol (WAMP), in contrast to REST. WAMP allows two types of high-level messaging patterns to be communicated across distributed, connected devices: RPC and publish/subscribe type messaging. Unfortunately, the current stable .NET version available for Unity does not support the desired WAMP libraries we would have liked to use to establish a full, high-level bidirectional communication capability between the data server and the Unity app. This setup would have been ideal because filter changes would be able to be pushed directly to the Unity app, rather than the app having to poll the server.

4.2 Improvements to the 2-D–3-D Hybrid Framework

This work aimed to improve upon our initial implementations, discussed previously, with the following:

- Create a more robust pipeline between Unity applications and our server side, such that the Unity application can interplay with other data visualizations connected to it and have it work in a coordinated fashion.
- Create a more interactive 3-D environment that better facilitates a user to peer into their data instead of just a playback over time.

4.2.1 Bidirectional Communication Pipeline between Unity and Data Server

As previously mentioned, to establish bidirectional communication between Unity applications and the WAMP data server, we designed a middleware that serves as an event translation service. Figure 11 shows a schematic of the pipeline.

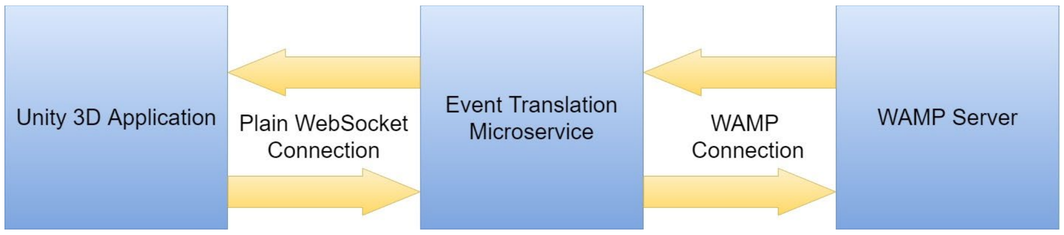


Fig. 11 Message-passing pipeline between WAMP server and unity

The event-translation service is designed to map WAMP functionality over to the Unity app. The event-translation microservice is designed to forward specific cases of these messages to and from the Unity app, as needed for the APS 2-D–3-D visualization. Specifically, the microservice will receive a payload from Unity specifying the RPC to call, as well as the argument inputs, and will forward this to the WAMP server by calling the corresponding RPC on it with the given inputs. In addition, the event-translation microservice can subscribe to a topic published by the WAMP server with a corresponding event handler that pushes the published topic and payload to the Unity app as a serialized JSON message. At that point, the Unity app can respond to the message accordingly.

In this particular application, the Unity app can be used to select a time range of the simulation to drill into, whose filter would apply to other visualization apps connected to the WAMP server. Further, when the time range is changed in another app, the Unity app can (and should) respond to the change accordingly. For the former, upon selecting a desired range in Unity a JSON payload will be sent to the microservice consisting of the RPC name to call on the server, called "timerange.set", and the actual time range as a two-element array containing the start and end time. The microservice will take the RPC name and run it through a switch statement containing the corresponding RPC calls mapped by RPC name, then call "timerange.set" on the WAMP server. In terms of the latter, the microservice subscribes to the "timerange.updated" topic. When the WAMP server publishes this topic coming from another application, the microservice receives it along with the new time range and sends both pieces of information to the Unity app. Figure 12 illustrates this pipeline, which will be illustrated further in the next section.

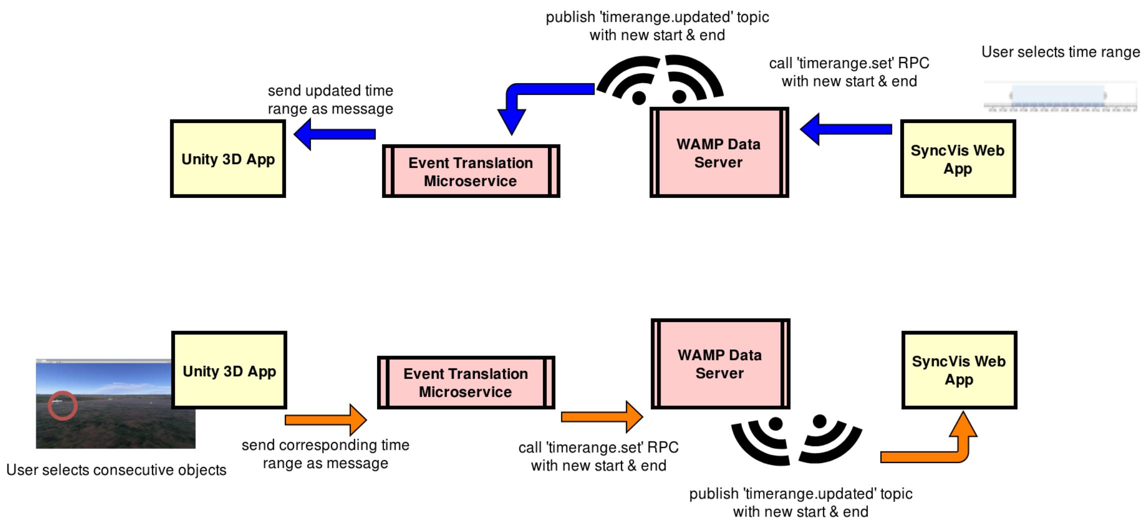

publish 'timerange.updated' topic with new start & end
call 'timerange.set' RPC with new start & end
User selects time range
send updated time range as message
Unity 3D App
Event Translation Microservice
WAMP Data Server
SyncVis Web App
Unity 3D App
Event Translation Microservice
WAMP Data Server
SyncVis Web App
User selects consecutive objects
send corresponding time range as message
call 'timerange.set' RPC with new start & end
publish 'timerange.updated' topic with new start & end

Fig. 12 Communication pipeline from SyncVis to Unity (top) and Unity to SyncVis (bottom)

4.2.2 Interactive 3-D Immersive Environment

In our updated visualization, we show the incoming threat and CM data at increments within the 3-D space instead of playing the animation on a loop for the given time range. When the entire time range of values is selected in SyncVis, the 3-D visualization shows a snapshot of both the threat trajectory and CM trajectory for incremental time steps within the entire time range.

There are 57 static threats present in the scene, aligned along the entire threat's trajectory path at increments of about every 5 time steps for the entire simulation. There are 66 static CMs present in the scene, one at each time step during the CM's trajectory. The 3-D visualization does not provide any insight into how the scenario plays out in real time, no longer playing the actual animation over time but, rather, contains a number of static 3-D assets showing the trajectories within the scenario. The 3-D assets still contain references to their time-step data from the simulation data, but the scene allows the user to inspect the trajectory paths as a whole, not just as they play out over time. Figure 13 shows the visualized scene.

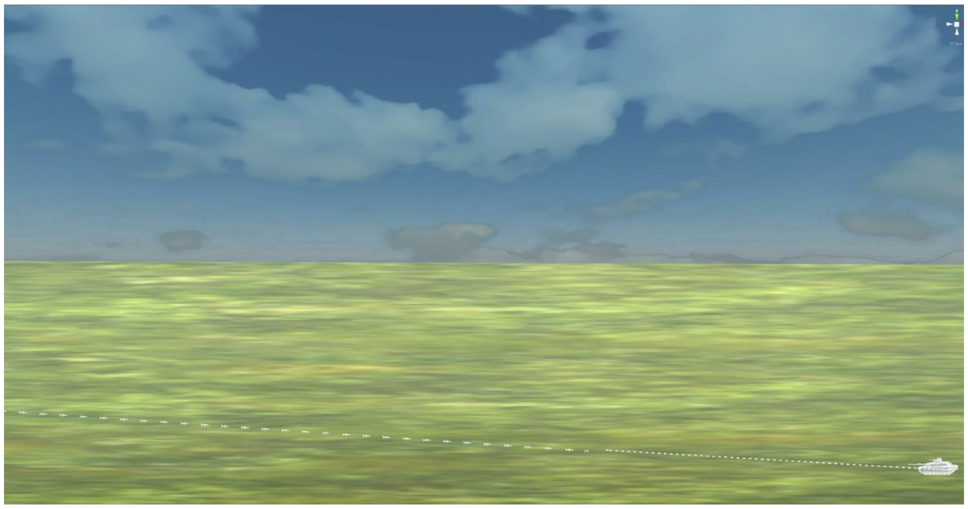


Fig. 13 Unity application with static asset trajectories

With the integration of the WAMP data server and event-translation microservice we have a more robust pipeline in place to push state changes within either application. At the front-end application view, nothing appears different to the user. As the user selects a smaller time range to visualize in SyncVis, the 3-D visualization reflects the change by showing only the threat and CM data that are present within that time range. When a time-range selection occurs in SyncVis, SyncVis will call the "timerange.set" RPC on the WAMP data server, the data server will publish a "timerange.updated" notification, and the microservice will respond by forwarding the updated time range to the Unity application. The Unity application will then turn the renderers off on the assets in the scene that do not fall within this time range. Thus, the selection of the time bar in SyncVis updates the scene in Unity to only show the corresponding threat and CM positions within that time range. (Figure 12's top half illustrates this particular data flow through the pipeline.) This allows the user to further inspect the trajectories of the data and to visualize when the CM gets fired in response to the incoming missile threat.

To create a more interactive 3-D environment for data exploration and analysis, the 3-D assets in the scene are selectable. This allows the user to spatially select the assets of interest in the 3-D environment to initiate a visual filter on the data from 3-D back to 2-D. In VR, the user is able to use a laser pointer from the hand controller to select a range of 3-D assets worth analyzing. This occurs by pointing the controller at an asset and pressing the trigger to select the asset. When an asset is selected, it will highlight in the 3-D scene. While still holding the trigger, the user may then drag the pointer across multiple assets, releasing the trigger when finished. This will highlight all of the assets in the scene that were toggled over while the trigger was pressed. The starting time point and ending time point of the range of assets selected are calculated, a request for time range update is sent as a message to the microservice over the web-socket connection, and in turn the corresponding “timerange.set” RPC is called on the WAMP data server. (Refer to the bottom half of Fig. 12 for an illustration of this data-flow scenario.) Further, the microservice, being subscribed to “timerange.updated” topic occurring on the WAMP data server, will relay these published updates over web socket back to Unity. When the Unity application receives an updated time range, it disables the renderers of the assets in the scene containing a time step not within the updated range. When the selection comes from Unity, all assets that are not highlighted in Unity will disappear after the selection occurs. To reset the time range in Unity, the user may select anywhere in the scene that is not a threat or CM asset. When this click registers, a message is sent to the microservice over the web socket to reset the minimum and maximum values for the time range to the minimum and maximum of the entire dataset. Then, all assets in the Unity scene will reappear. Figure 14 depicts users interacting with the 2-D–3-D hybrid visualization. One user interacts with SyncVis on the display wall while the other user interacts with the Unity application in VR. This allows for collaborative data analysis where any update in either application is reflected in the other.

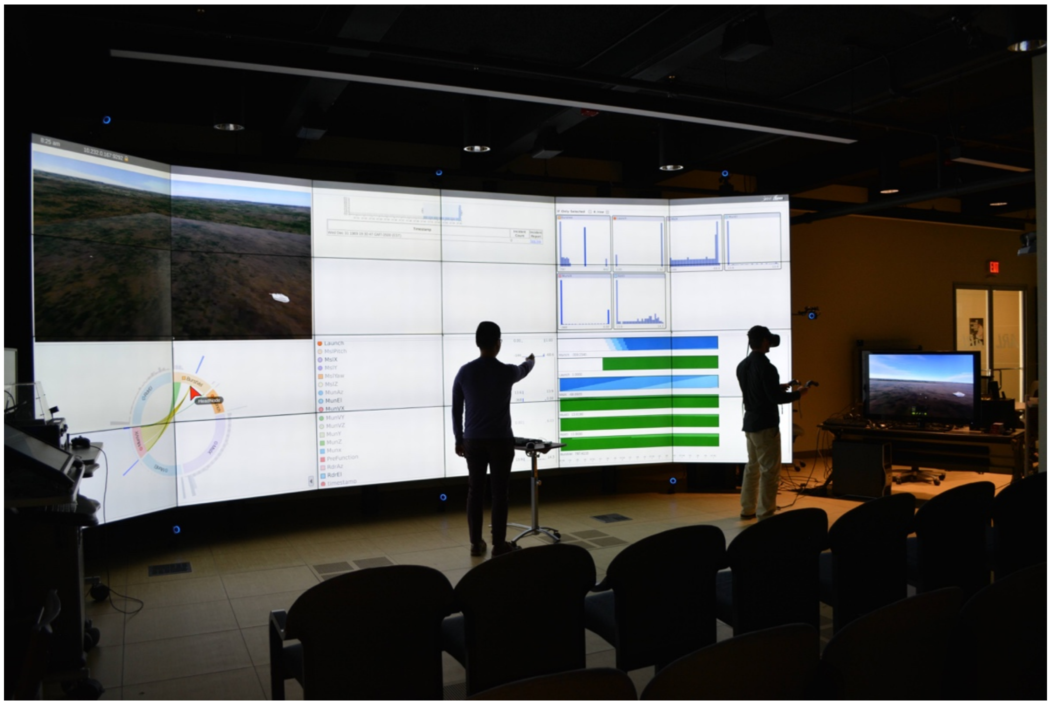


Fig. 14 2-D–3-D hybrid visualization framework for APS

5. Conclusion and Future Work

The DSRC DAAC team has provided ample data-analysis support for the APS project. The data-driven 3-D animations created using both Unity and UE4 provide high-fidelity graphics for analysts to visualize the Monte Carlo simulations. From a technical standpoint, UE4 has proven itself capable and flexible to rapidly develop visualizations of these types of simulations. It supports data-driven visualization in both static data-table and live data-streaming formats and in situ communication with external processes or simulations. UE4 can render high-resolution visuals without experiencing performance issues, even with multiple units and simultaneous events. Furthermore, development for VR technology is seamlessly integrated with UE4.

Because APS is just a subset of the VPSS project as a whole, the DAAC team has worked with the project collaborators to anticipate future data-analysis capability. The two new UE4 applications have been created to visualize various components of an APS life cycle once the simulation data become available. These applications provide a structure for optimization-based immersive analysis, as well as an architecture in which visualization can be coupled with other processes that define the behavior of military systems being simulated. This includes radar, targeting, and weapons systems. The rest of the VPSS simulation code is currently being developed, optimized, and/or parallelized to run on the HPC resources. Once these components are functioning and can produce data, those data can be used to drive the UE4 applications.

Once these 3-D applications are data-driven, the next step will be to hook the applications directly to the code running on the HPC resources for in situ analysis. That is, as the simulations run as a job on the HPC, the simulation output is directly injected into the UE4 application such that the analyst may view their simulation running in real time. UE4 has shown to be capable of communicating with external processes, but these complex systems still need to be integrated into the project. Once VPSS contains the software to accurately simulate the behavior of military systems, these UE4 applications can be used to visualize those simulations.

For the 2-D visualization, we were able to accommodate a new type of filtering with which the analysts can drill down into their simulation data, as well as demonstrate the ability to cross-filter the data using multiple filters at once. The multiple coordinated charts and graphs provide a means for in-depth analysis beyond just viewing the 3-D simulation realization. It enables both single-variable analysis and pairwise analysis between variables. With multiple coordinated views demonstrating the same data with the ability to visually select the data to analyze, the analyst can explore and discover anomalies within the data set that may not be apparent otherwise. The next steps for the 2-D visualization are to enable more filtering capability based off the variables in the data sets and to optimize the cross-filtering used for even quicker drill-downs as the size of the data sets continues to grow.

The enabling of the 2-D–3-D hybrid visualization framework allows analysts to coordinate their data-analysis efforts across the 2-D–3-D threshold. With both visualization applications displaying the same set of data, we have provided the analysts with multiple modes to suit their analytical needs. Currently, the hybrid framework only supports the analysis of one simulation run of the APS. Ideally, the analyst would be able to compare the data across multiple simulation runs; however, this may require some novel, new visualization capability to coordinate multiple variables over time over multiple simulation runs. We are currently researching visualization strategies to depict multivariate spatio-temporal correlation over multiple simulation runs. One way to do this is to move our current 2-D visualizations to a 3-D environment where the third dimension can be used for simulation runs, effectively "stacking" the 2-D visualization for each simulation run to compare. We would also like to hook the new UE4 applications we have developed to the hybrid visualization framework similarly to how we hooked in the Unity application.

6. References

1. US Army Research Laboratory DoD Supercomputing Resource Center [accessed 2019 Aug]. https://www.arl.hpc.mil/about/index.html.

2. Data Analysis and Assessment Center [accessed 2019 Aug]. https://daac.hpc.mil/index.html.

3. Su S, Barton JM, An M, Perry V, Panneton B, Bravo L, Kannan R, Dasari V. Reconfigurable visual computing architecture for extreme-scale visual analytics. In: Blowers M, Hall RD, Dasari VR, editors. Proceedings of SPIE 10652, Disruptive Technologies in Information Sciences, SPIE Defense and Security; 2018 Apr 15–19; Orlando, FL.

4. Chen M. Vision, strategy, and tactics of system survivability engineering software (SSES) for adaptive and cooperative protection simulations–an enabling technology for next-generation combat vehicles. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2018 Feb. Report No.: ARL-TR-8290.

5. Chen M, Graham MJ. Transition and implementation of survivability suite engineering simulation (SSES) for active protection systems. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2017 Jul. Report No.: ARL-TR-8048.

6. Chen M, Graham MJ, McNeir M. Preliminary development of survivability suite engineering simulation (SSES): software architecture and design. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2016 Oct. Report No.: ARL-TR-7855.

7. Graham MJ. Active protection system (APS) end-to-end modeling (EEM) animation and analysis tools. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2018 Jan. Report No: ARL-CR-0823.

8. Unity User Manual. San Francisco (CA): Unity Technologies; 2019 [accessed 2019 Aug]. https://docs.unity3d.com/Manual/index.html.

9. Unreal Engine 4 Documentation. Cary (NC): Epic Games; 2019 [accessed 2019 Aug]. https://docs.unrealengine.com.

10. Binaco R, Gao W, Su S. Assessment of collateral damage in post-engagement of an active protection system: visualization and simulation of an active protection system engagement lifecycle. High Performance Computing Modernization Program (US); 2019 Aug. Report No.: HIP-19-005.

11. Gough J, Chen M. Visual vulnerability assessment of nonstationary vehicles subject to threat residuals in active protection systems. High Performance Computing Modernization Program (US); 2018 Aug. Report No.: HIP-18-041.

12. North C, Shneiderman B. Snap-together visualization: a user interface for coordinating visualizations via relational schemata. Proceedings of the Working Conference on Advanced Visual Interfaces, AVI 2000. New York (NY): ACM; 2000. p. 128–135.

13. ParaViewWeb: a JavaScript library for building web-based applications with scientific visualization. Kitware; 2019 [accessed 2019 Aug]. https://kitware.github.io/paraviewweb/docs/.

14. dc.js – Dimensional charting Javascript library [accessed 2019 Aug]. https://dc-js.github.io/dc.js/.

15. React: a JavaScript library for building user interfaces. Facebook Inc; 2019 [accessed 2019 Aug]. https://reactjs.org/.

16. Richard J, An M, Su S. HPC enabled visual analytics for heterogeneous big data: Cross-filtering weapon defense data on binary variables by integrating dc.js and ParaViewWeb on the Syncvis Application. High Performance Computing Modernization Program (US); 2018 Aug. Report No.: HIP-18-030.

17. Su S, Perry V, An M, Chen M. Real-time interactive hybrid 2D and 3D visual analytics on large high resolution display and immersive virtual environment. SEARIS 2018 11th Workshop on Software Engineering and Architectures for Realtime Interactive Systems; IEEE Virtual Reality; 2018 Mar. Accepted for publication.

18. Su S, Bravo L, Dasari V. Reconfigurable visual computing architecture for extreme-scale visual analytics. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2018 Sep. Report No.: ARL-TR-8504.

List of Symbols, Abbreviations, and Acronyms

2-D	2-dimensional
3-D	3-dimensional
API	application programming interface
app	application
APS	active protection system
ARL	US Army Research Laboratory
ATGM	antitank guided missile
CCDC	US Army Combat Capabilities Development Command
CM	countermeasure
DAAC	Data Analysis and Assessment Center
DSRC	Department of Defense Supercomputing Resource Center
HPC	high-performance computing
JSON	JavaScript Object Notation
REST	Representational State Transfer
RPC	Remote Procedure Call
RPG	rocket-propelled grenade
SSES	System Survivability Engineering Software
UE4	Unreal Engine 4
UI	user interface
VPSS	Vehicle Protection Software Suite
VR	virtual reality
WAMP	Web Application Messaging Protocol

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	CCDC ARL FCDD RLD CL TECH LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
3 (PDF)	CCDC ARL FCDD RLC S V PERRY S SU FCDD RLW PA M CHEN