Engineering Visualization

David E-P. Colbert and R. Edward Ralston

Physics-based, high-fidelity modeling and simulation (M&S) tools that the engineering community develops and employs to analyze the performance of modern weapon systems are necessarily becoming more complex. As the fidelity of these tools increases, so does the volume of pertinent information that they generate. To interpret the wealth of resulting engineering information, APL developed "engineering visualization" as a tool to facilitate a better understanding of weapon systems and their respective models and simulations. Engineering visualization allows the engineer to present the results of M&S-based analysis to the sponsor with unrivaled clarity and efficacy. By providing the highest level of insight possible to both engineer and sponsor, engineering visualization has established itself as an essential systems engineering tool. This article describes the history, development, implementation, and future of engineering visualization in the Laboratory's Air Defense Systems Department.

INTRODUCTION

An important role of a weapon system engineer has been to interpret analytical results and present those results to the sponsor. Many early engineering efforts at APL produced tabular engineering information printouts, which an engineer pored over, looking for patterns and trends in the array of numbers. Later, two-dimensional (2D) plotting tools became readily available. The engineer now had the ability to study a single metric versus another metric in a graphical representation, looking for trends or anomalies in the graph. As time progressed, three-dimensional (3D) plotting applications became available to the weapon systems engineer that facilitated the ability to study three-parameter modeling outputs, either as a line or a surface in 3D space. Adding a fourth dimension, usually time, to these plots enabled a unique quality of animation that allowed visualization of four-parameter engineering information.

The volume and complexity of the engineering information resulting from these weapon systems models grow directly as the complexity of the threat increases. Also, with recent trends in DoD research focusing more on using statistical computer models of weapon systems, the weapon systems engineer has enormous amounts of information to analyze and interpret. In addition, these analytical results must be communicated to the sponsor in a clear and concise manner.

Commercially available engineering analysis tools are not capable of studying all aspects of a weapon system or presenting comprehensive analysis results succinctly. Often, complex analysis can result in a presentation with hundreds of slides and can require several hours to

present to the sponsor. To make this problem manageable, APL has developed "engineering visualization" as a tool to study such voluminous amounts of complex engineering information and to present findings to the sponsor efficiently and effectively.

EVOLUTION

Single-Screen Nondistributed Visualization

The first class of engineering visualization, single-screen nondistributed visualization, has been used to analyze several weapon systems. This section describes the genesis of this fundamental visualization tool.

Visualization Entities

The visualization software's basic purpose is to take weapon system simulation data and efficiently build a 3D representation of those data within a visualization scene (Fig. 1). Thus mapping the data of an object (e.g., a missile) to a meaningful visual representation of the object is the fundamental objective. The core C++ object within the visualization software is, therefore, the entity object, which contains a reference to the graphical representation of the object within the visualization scene, a reference to the data source, and the mapping of the data to the graphical object's various controls (such as position, orientation, and articulations). In the visualization software, anything that has a visual representation and is driven by data is represented as an entity object. The visualization software consists of about 250 C++ classes and contains a visualization scene; a motif interface; a math library; networking code, camera, and event scripts; a 3D graphical user interface; and 2D overlays.

Defending Missile Entity

Air defense systems that use a defensive missile have detailed physics-based models of that missile, and these models produce information that must be correctly depicted. Visualizing the important aspects of the defending missile begins by creating a 3D representation. To create such a representation, the engineer must obtain engineering diagrams, blueprints, photographs, computer-aided design models, and detailed descriptions of the missile to the highest possible fidelity. Using all of these resources, he creates via software a polygonal 3D representation (or framework) of the missile to scale. Next, photographs are used to create computer images, known as textures, that mimic the surface appearance of the missile body. With the textures mapped onto the polygonal 3D representation or "wireframe," the engineer can effectively reproduce the size, shape, and appearance of the defending missile body. Then additional texture-wrapped polygonal representations of each stage's rocket motor flames may be attached to the tail of each respective missile stage. Finally, each part of the missile that is to be driven by modeling data (such as aero-control surfaces) must have an "articulation," which describes its location with respect to the center of gravity of the missile and its range of motion relative to the missile's axes (Fig. 2).

The resultant texture-wrapped polygonal representation can, however, be so highly detailed that computer graphics hardware becomes saturated, especially if numerous such representations are simultaneously displayed in a visualization scene. Although the performance of the visualization software is greatly enhanced by object culling (removing "hidden" objects), most of the graphics optimization comes from polygon reduction. APL achieved polygon reduction by creating several versions of each graphic object, each with a different level of detail. When rendering a scene, the appropriate version is chosen based on the distance of the object from the viewer: the farther an object is from the viewer, the simpler the graphic object can be.

The engineer must next focus on the need to visualize from the perspective of the six-degree-of-freedom (6-DOF) model of the defending missile. The primary parameters that must be visualized are the data

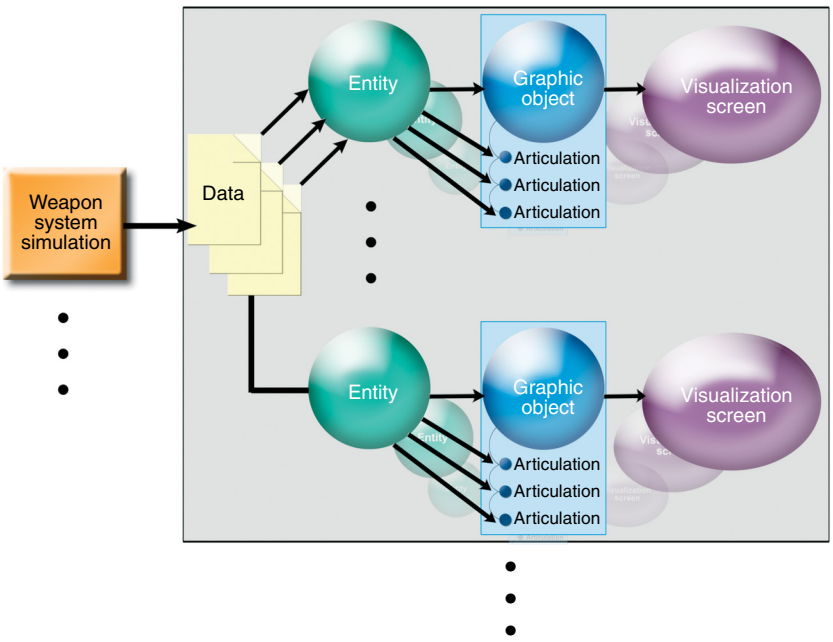

Figure 1. The weapon system simulation provides data to the entity, which attaches the data to the graphics object and its articulation in the visualization screen.

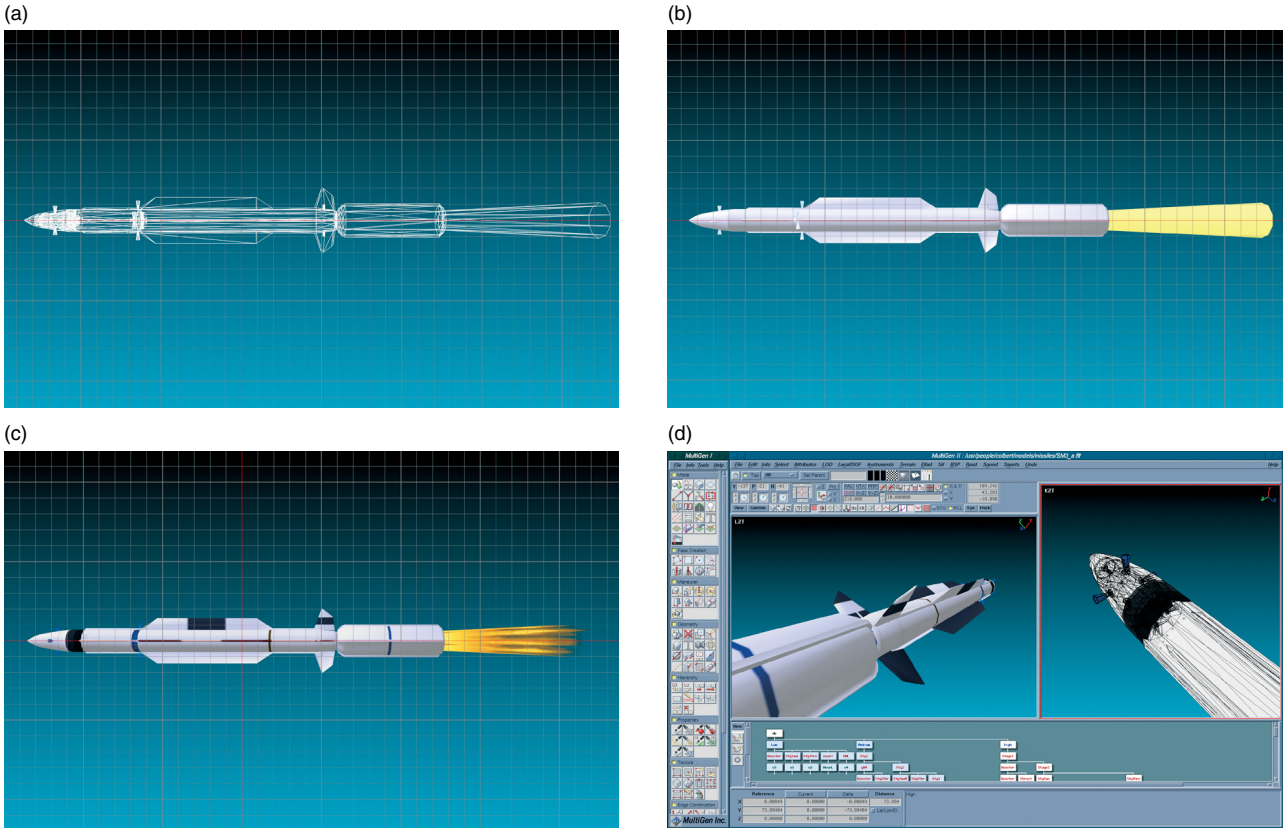

Figure 2. Polygonal (a), surface (b), and textured (c) representations of the missile are shown, as well as a missile graphic object in the process of being constructed (d).

that describe the actual translation and rotation of the missile at a fine time-step granularity (i.e., time, position, velocity, attitude, and acceleration). By visualizing these parameters, the engineer can clearly see for each time step where the missile is, where it is heading, and where it is pointing (Fig. 3).

Event information also facilitates the visualization of how the body of the missile changes size and shape as the missile continues along its flight path. Event information includes the missile launch time, each missile stage's rocket motor burnout and separation times, the ejection time of minor elements of the missile body (e.g., nosecone, dome cover, clasps or nuts, bolts), impact events (e.g., debris clouds, explosions), and translucent spheres around the missile that represent its lethal kill radius.

Additional parameters that the engineer must visualize are the thrust and aero-control surface data. These include the main thrust of the missile, the deflections of the missile's fins, and any attitude control thrusters on the missile. With these

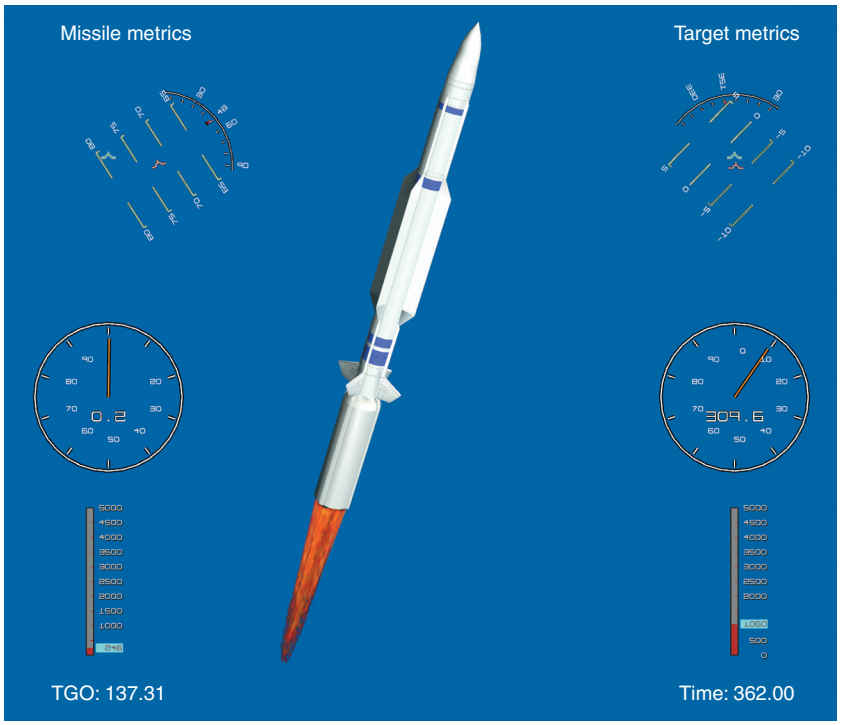

Figure 3. The missile entity, in this case a Standard Missile-3.

parameters added to the missile visualization, the engineer can scale the rocket motor flames in proportion to the main thrust data; rotate each fin according to its fin deflection data; and turn on, turn off, and scale the attitude control thruster flames according to the attitude control thruster data. Having incorporated the thrust parameters and aero-control surface into the visualization, the engineer can visualize the effects of the forces that affect the flight of the defending missile. To add realism to the visualization, the engineer may even add scaled smoke trails to the rocket motor flames to simulate plume from the rocket motors.

After obtaining the missile body representation and corresponding data driving each element of the body, the engineering information that describes the missile's onboard sensor must be added. The onboard sensor usually comprises a protective cover and the sensor itself. A translucent cone or beam emanating from the sensor face is used to represent its field of view. Sensor information includes the half-power beamwidth of the sensor, the range (if appropriate) of the sensor, the azimuth and elevation of the beam at all times, and the sensor activation time. Combining the protective cover ejection time from the stage data with the parameters for the sensor beam, the engineer can visualize the precise location, size, shape, and pointing angle of the sensor beam at all times, as well as any entities which happen to be contained within its field of view.

Threat Entity

The threat visualization process closely resembles the process for a defending missile except that intelligence information is now a critical input. The 3D representation of the threat is again created from any available engineering diagrams, blueprints, etc., pertaining to the threat, either from freely available or, more often, intelligence sources. Using these resources, the engineer generates via software a technically accurate wireframe of the threat and maps textures onto the polygons that mimic the threat's appearance. Finally, the engineer adds flames to the rocket motors and articulations to any elements of the threat body, both of which are dynamically driven by data. The threat's launch vehicle can also be visualized using the same process.

The flight of the threat for each time step can be visualized from data on its position, velocity, attitude, acceleration, main thrust, attitude control thrusters, and stage events. The engineer follows a visualization process similar to the one described for the defending missile to produce the trajectory, forces, and stage events for the threat. The engineer may now see where the threat is, where it is going, how it is oriented along its flight path, whether it is speeding up or slowing down, what forces are acting upon it, and how its size and shape change as its flight progresses (Fig. 4).

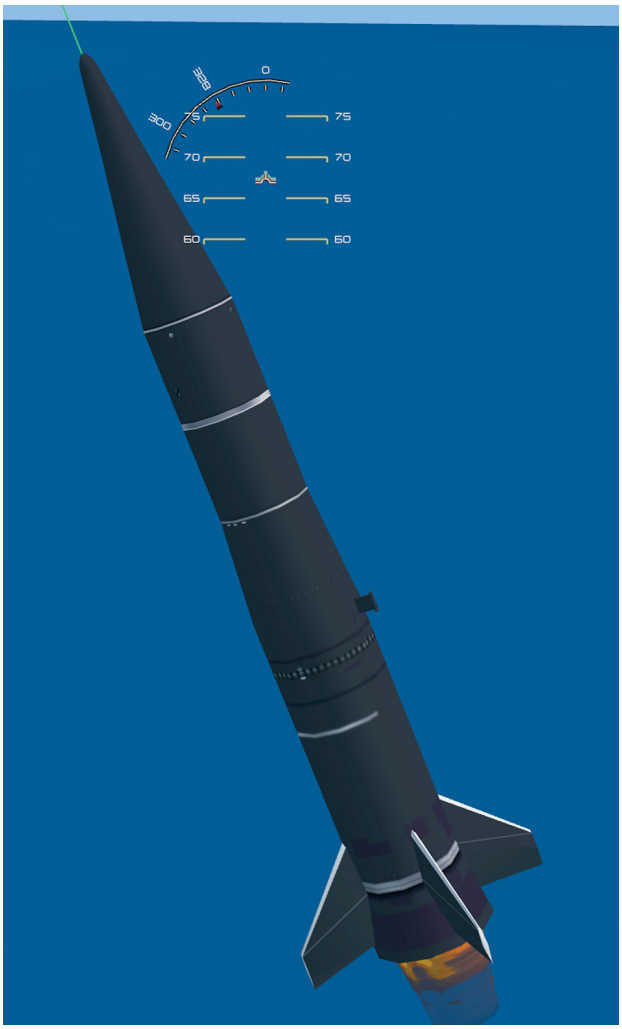

Figure 4. The threat entity, shown here as a target test vehicle.

Ship Entity

The engineer must next add the launch platform for the defending missile, typically a ship. Using available Navy ship engineering diagrams, blueprints, etc., a 3D representation of the launching ship is produced in a similar manner as the defending missile and the threat. The trajectory for the ship has been less complicated to date than for the missile and threat because it can be described by a single latitude, longitude, and heading (Fig. 5). Adding roll, pitch, and yaw as well as surge, sway, and heave is simply a matter of attaching data from a ship motion model to the ship entity.

Radar Entity

Housed on the ship entity, the radar entity (typically a phased array SPY face) must now be added to the visualization to show search and track parameters. The metrics for the radar include the azimuth, elevation, and instrumented range extent of the volume in which the radar searches for the threat, as well as the

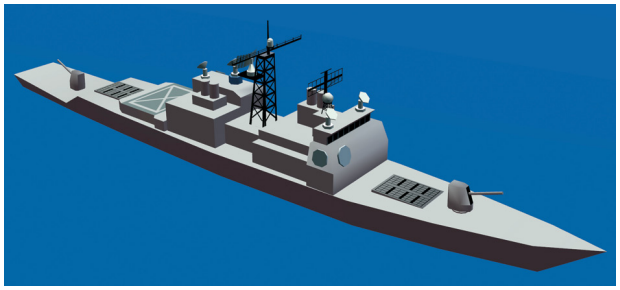

Figure 5. The ship entity, an Aegis cruiser in this visualization.

half-power beamwidth, instrumented range extent, and dynamic pointing angles of the beam(s) with which the radar searches the volume. With this information, the engineer may construct 3D representations of the radar beams and search volume. These 3D representations are created by constructing polygonal representations of the beams and volume, adding color and translucency so that the visualization simulates peering into the beams and volume to determine what entities (such as threats) are present.

The engineer must attach the beams and volume to the appropriate SPY radar face on the representation of the ship using information from the detached radar model (typically the SPY FirmTrack model) to describe their motion. The search volume frame time determines how long it takes the radar beam to scan the entire search volume. Radar beam azimuth and elevation describe exactly where each beam is pointed at each moment in the visualization. The radar model (Fig. 6) also collects the state, type, group, cluster, object, and correlation information pertaining to the tracks for all threats that the radar is tracking.

Fire Control Entity

Also attached to the ship entity is the fire control entity, or in some cases the illuminator(s). This entity is represented as a set of beams and volumes similar to the radar beams and volumes. The illuminator beams typically have different dimensions than those for the radar entity. Illuminator beams are attached to the illuminator dishes on the ship, and the beams are pointed according to the desired pointing angle of the dish. The illuminator dishes on the ship must also be rotated toward the threat because they are mechanically aimed (Fig. 7).

Aircraft Entity

All 3D representations of both friendly and threat aircraft that the engineer visualizes are created in the same way as a defending missile or threat is created. The position and attitude of the aircraft at a fine time-step are needed to visualize its flight. The visualization now shows where the aircraft is at each point in time, as well as how it is oriented at that time. For aircraft with an onboard sensor, the 3D representations for the sensor typically include the sensor itself, the sensor's search volumes, and the sensor's search beams. The beams and volumes for the sensor are similar to those of the radar and illuminator, except that they have different sizes and shapes (Fig. 8).

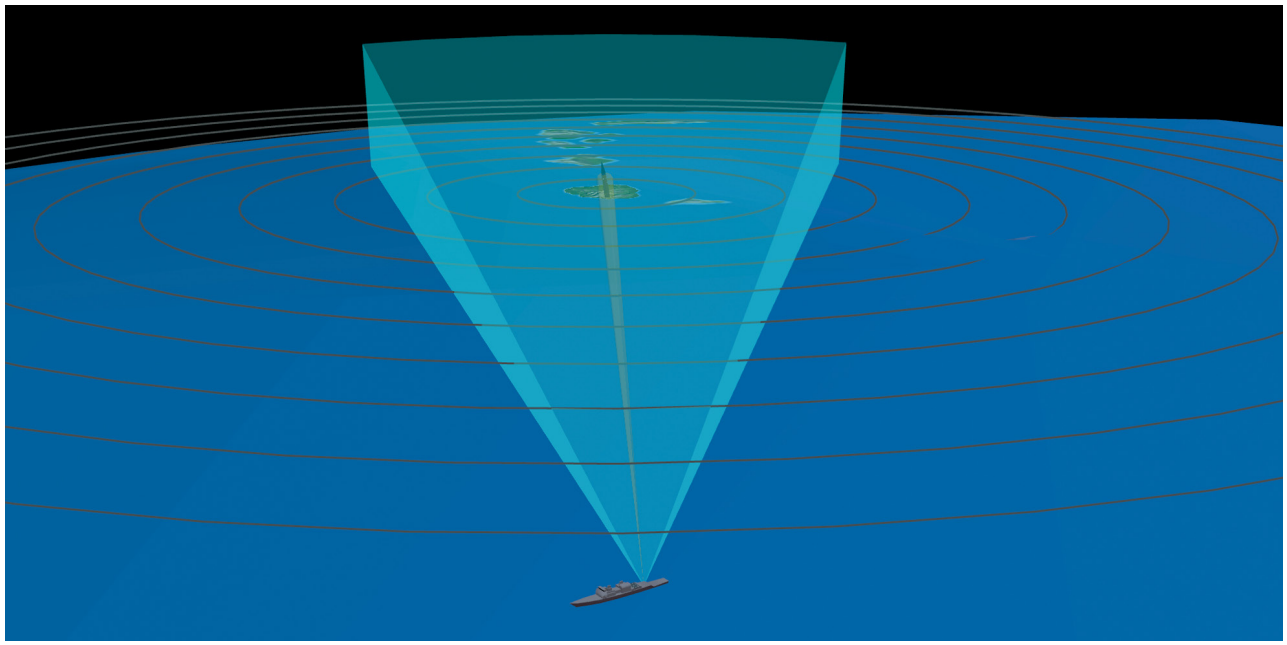

Figure 6. The outer search volume, the inner search beam, and the threat can be seen in this image of a radar entity.

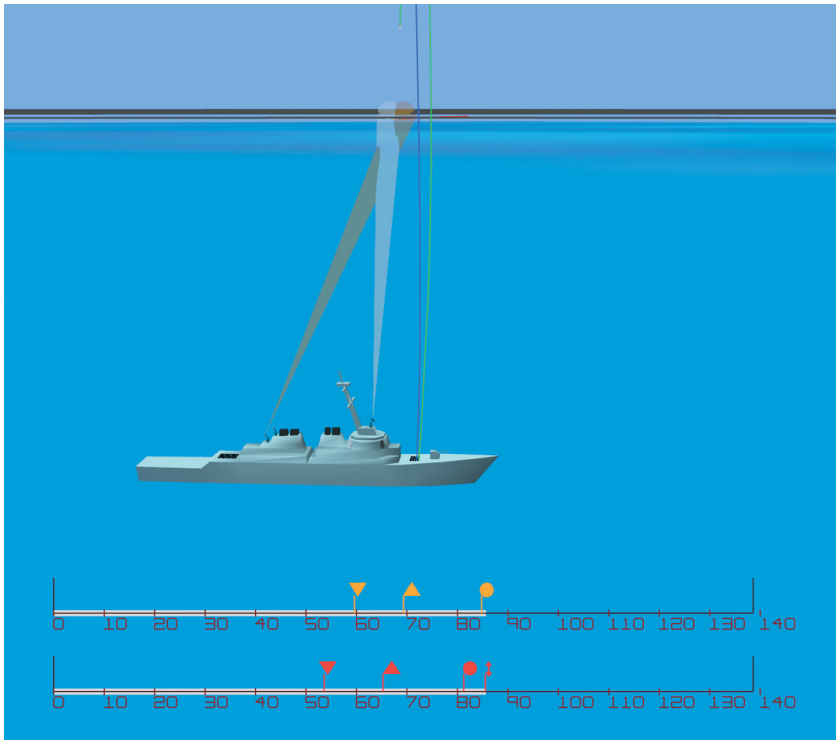

Figure 7. The two beams emanating from the ship toward the threats on the horizon are the illuminators of the fire control entity.

Celestial Entity

If the defending missile guides to the threat with a seeker that may be affected by stars entering its field of view, then celestial entities are important. Celestial entities are represented as emissive points on a celestial sphere that is at a great distance from the Earth. The instantaneous star and planet locations on the celestial sphere are calculated from their right ascension and declination according to the U.S. Naval Observatory's *Multiyear Interactive Computer Almanac* for the specific launch date and time. Using this information, the engineer constructs a geocentric celestial sphere at a distance of 10 R_E from the Earth.

Figure 8. The aircraft entity, in this case an E-2C.

Terrain Entity

The engineer must also create the terrain over which all other entities are located. The terrain entity is either a 3D representation of the entire Earth or a small patch of it. Terrain entities may be generated in several different ways and to several different fidelity levels, depending on the specific requirements of the visualization.

Several geographic datasets are used to create a terrain visualization. One source is digital terrain elevation data (DTED) from the National Imagery and Mapping Agency (NIMA). DTED datasets contain information for most of the Earth on ground-level elevation above mean sea level. This information is obtained via satellite. Another source is digital elevation map data from NIMA, which includes similar elevation information in a different file format and for different resolutions than DTED information. In addition, digital feature analysis datasets contain cultural features for most of the Earth, e.g., buildings, roads, canals, railroads, airports, etc. The engineer uses several software packages to generate terrains and integrates the data from several such geographical datasets to create any of three terrain classes.

The first class is low-fidelity terrain, which is typically used for visualization of Anti-Air Warfare analysis where scenarios usually occur over a small area and are generally not near major land masses. Several steps are necessary to generate this type of terrain. First, the appropriate DTED data for the area of the world to be visualized are loaded into polygon reduction software to construct the initial flat-Earth 3D polygonal representation of the location. Next, the engineer creates a low-fidelity image texture from the DTED or digital elevation map dataset using any of several software packages and then maps the texture onto the polygons. The engineer must now convert the terrain from its flat-Earth representation into a non–flat-Earth model which is usually the 1984 World Geodetic System's ellipsoid Earth. In this manner, the engineer creates a terrain that contains elevation information, not only in the polygons that determine the shape of the terrain but also in the texture that is mapped onto the polygons (Fig. 9).

The second class, medium-fidelity terrain, has a relatively low-fidelity polygonal representation but a relatively high-fidelity texture mapped onto these polygons. This terrain class has been used mainly for Theater

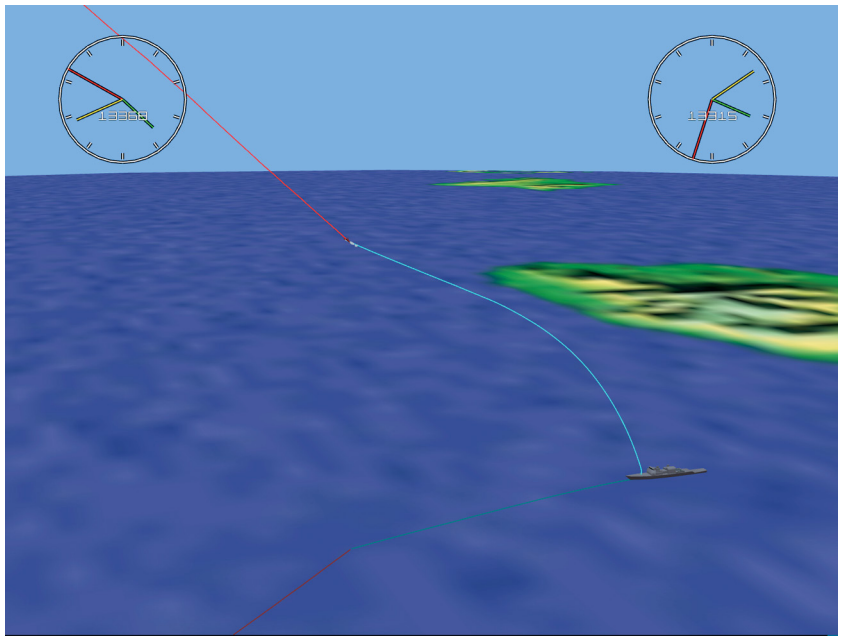

Figure 9. The terrain in this image is a low-fidelity graphic representation of Hawaii, which was created for an Anti-Air Warfare reconstruction visualization. Kauai may be seen just behind the ship.

Ballistic Missile Defense (TBMD) analysis, which takes place far above the Earth and thus does not require a high-fidelity polygonal representation of the Earth's surface. However, visualization cameras are often in the exo-atmosphere looking down on the Earth, which requires a large and relatively high-fidelity texture to cover the camera's field of view. To generate a terrain of medium fidelity, the engineer has to follow steps that are slightly modified from those for the low-fidelity terrain. First he loads the appropriate DTED data for the pertinent area of the world into polygon reduction software. He may need to load several datasets into the software and concatenate them into one large flat-Earth terrain to cover the entire area. Following this, either DTED or digital elevation map datasets are loaded into the software, which reads the datasets and converts them into images. In this way, the engineer creates a relatively high-resolution texture wrap for the terrain. Finally, the terrain is converted from its flat-Earth representation into the particular coordinate system and Earth model. The result is a terrain similar to the low-fidelity terrain, except that the medium-fidelity terrain covers a much larger area of the Earth (Fig. 10).

The third class of terrain, and by far the most complicated to generate, is the high-fidelity terrain. This terrain has a high-fidelity polygonal representation and a high-fidelity texture. It is used only when it is necessary to have all of the characteristics of the medium-fidelity terrain plus the highest geometrical accuracy and precision possible at the surface of the Earth. These terrains have been generated for joint mission analysis of Overland Cruise Missile Defense (OCMD) and TBMD, which involve defending against exo-atmospheric theater ballistic missiles and terrain-hugging overland cruise missiles, respectively.

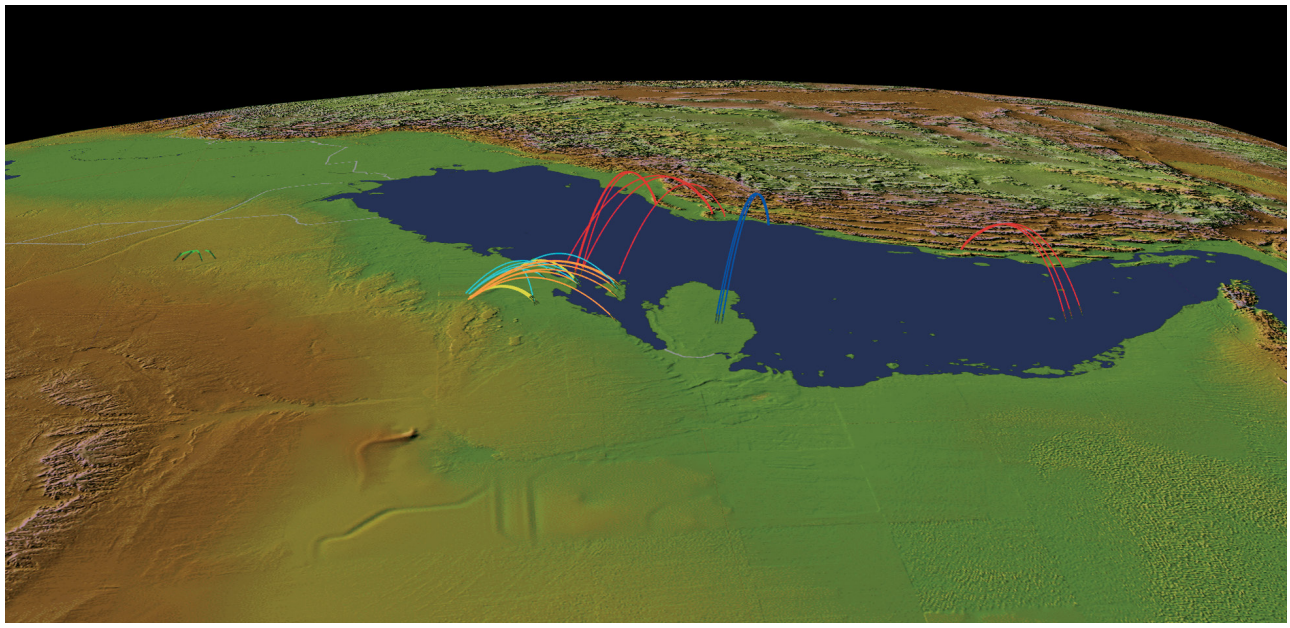

Figure 10. This middle-fidelity terrain of the Persian Gulf was created for a single-screen nondistributed TBMD visualization.

To build a terrain of this class, the engineer must first determine what computer resources are available on the visualization platform, including main memory and geometry as well as the texture limits of the graphics card. The terrain is designed based on these hardware constraints. To generate a high-fidelity terrain the engineer must first follow the steps for building a medium-fidelity terrain, then begin again with the same DTED datasets and batch-process the terrain as a set of rectangular tiles with far more polygonal detail. The terrain is divided into this set of rectangular tiles so that hardware resources are conserved by only having the hardware render high-fidelity tiles which are closest to the viewpoint. Similar to the level of detail optimization used for entities such as the missile and target, simpler versions of the terrain tiles are used as the distance from the viewer increases.

Even with this terrain optimization, the engineer still must perform more optimizations to ensure that the capabilities of the target platform's graphics hardware are not exceeded. Therefore, a critical step in generating a high-fidelity terrain is selecting an appropriate polygon reduction method and reduction parameters for the chosen method, customized to the specific capabilities of the computer platform. The engineer then maps the highest-fidelity texture onto the tiles. If cultural features for the terrain are required, then the digital feature analysis data dataset overlays cultural features onto the terrain as a polygonal representation of the selected features (Fig. 11).

2D Overlays

Since not all data of interest can be meaningfully shown as a 3D object, the engineer needs to also support other forms of data display. This is accomplished by overlaying 2D graphic representations of the data such as 2D plots, rotary dials, scales, compasses, and attitude indicators onto the 3D scene, thereby allowing visual correlation. In addition to displaying the 2D data as graphical overlays, the data can also be displayed as text. Although this is not the optimal display choice for most data items, it is essential to be able to display text for such information as titles, time of flight, miss distance, and missile stage. It is also useful to have text displays in the 3D visualization environment such as annotations for track numbers attached to threats and labels attached to stars in the visualization. The text is rendered using a texture-mapped font for efficiency and to give the engineer the flexibility of editing the pixmap-based fonts.

Configuration Files

As the visualization software matured and APL amassed a large library of terrains and graphic models, users began to realize that most of the subsequent work in visualizing new input data sets would entail manipulating the input data rather than developing new features or graphic models. The input data come from a multitude of sources and are usually given to APL in a format that needs to be manipulated, resampled, or combined before they can be used by the visualization software. APL improved the process of visualizing new datasets somewhat by integrating a library of classes that provided useful mathematical operations (e.g., unit conversions, coordinate transformations, and quaternion-based spherical interpolations). However, the availability of certain data would sometimes necessitate the modification of the C++ visualization code. For example, if APL received data that included an articulation that had not been previously visualized, then the software would have to modified so that these data could drive the articulation control points of the graphic representation.

The solution was to develop a configuration file that would fully describe the data and the control points of the graphic representation. This configuration file would also specify how the data were to be mapped to the corresponding graphic representation. Developing a visualization was then a matter of modifying the configuration file instead of modifying the visualization code.

APL also wanted the configuration file to be flexible enough to allow it to read each data file in its native format. Rather than writing a static configuration file format, the Laboratory decided to embed a scripting language into the visualization software. This would allow the flexibility to perform arbitrary data manipulations, define the control points, and specify the data-to-graphic mapping within a single configuration file. Perl was chosen as the scripting language because it is optimized for scanning arbitrary text files such as APL's visualization data files, has object-oriented support, can be embedded into existing applications with relative ease, and has an established community that has developed a voluminous software code base from which APL draws.

Visualization Products

Because visualizations allow for user interaction, they are most useful when viewed from a display while it is being rendered directly from the computer. Being able to interact with the visualization (e.g., viewing a scene from varying angles, changing the playback speed, enlarging screens on a subdivided display) enables the engineer or sponsor to focus on any aspect of the visualization desired.

Since the host platform for the visualization software is too large to permit ease of physical portability, APL has used several methods of capturing the visualization onto analog or digital media to create portable versions for distribution. The simplest but least preferred method is an analog VHS video capture of the visualization. To

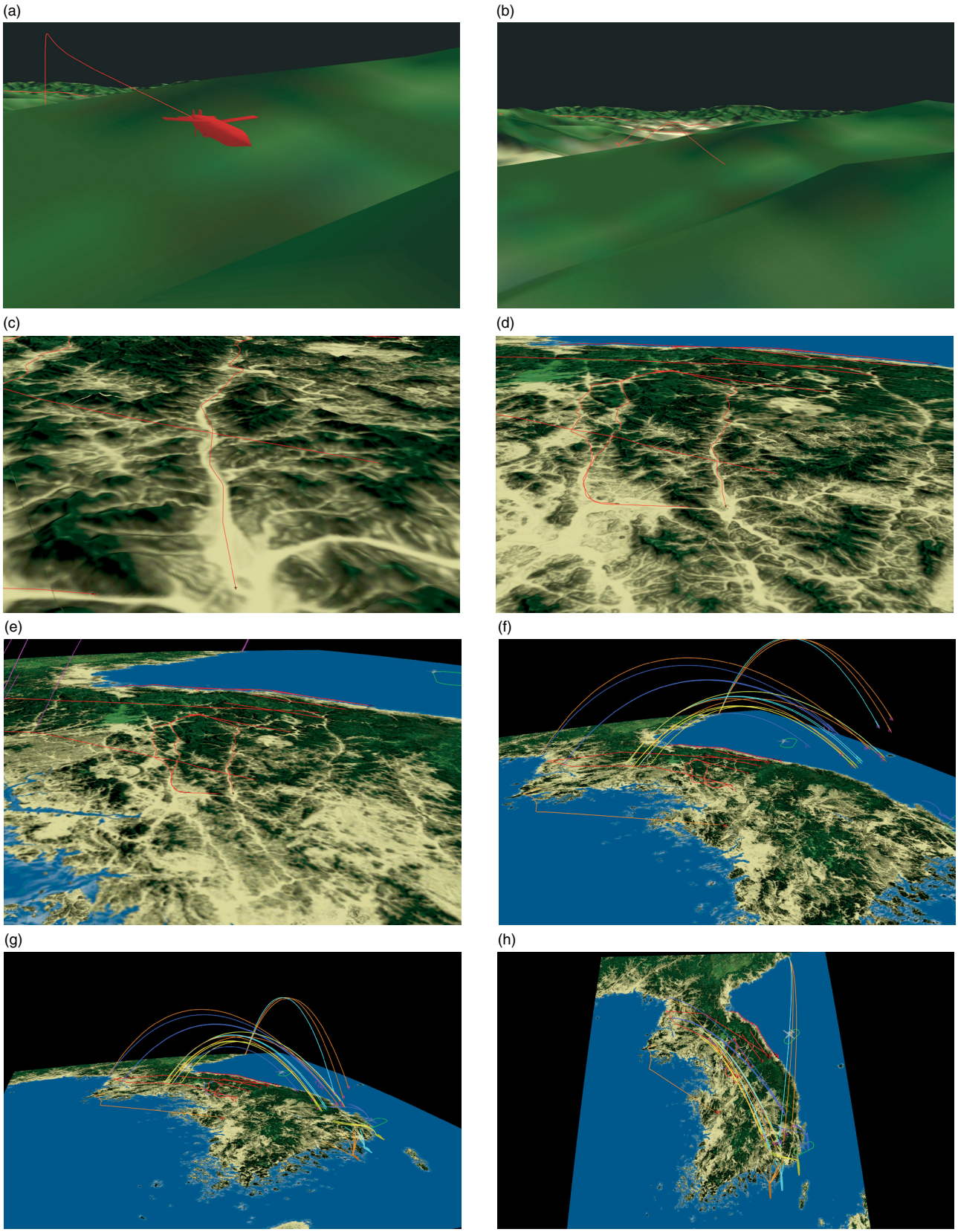

Figure 11. These eight images display the level of precision to which a high-fidelity terrain must be generated while still covering a large area. In (a) and (b), the relatively small distance between the missile and the mountain ridge over which it has just flown may be seen. In (c)–(e), the red profile of the cruise missile is seen weaving through valleys (light green) in the terrain. The scenario as seen from space is depicted in (f)–(h). This high-fidelity terrain was generated for a mixed-mission TBMD and OCMD multiscreen distributed visualization, another class of visualization which is discussed later in this article.

create the VHS videotape, a video camera is mounted directly facing the computer monitor of a running visualization.

An alternative and preferred method of capturing to VHS videotape is to first create a digital video capture and then create the videotape from the digital version. To make a digital capture of the visualization, APL added code to step through the visualization (usually at 30 frames per second), capturing each individual frame to disk. After the visualization has saved all of the frames to disk, movie-editing software is used to assemble the frames and compress them into a single quicktime movie. Once the quicktime movie has been generated, it may be postprocessed to include titles, sound, labels, transitions, and other effects which improve the quality of the movie. The final movie is then written to videotape.

To produce an even higher-quality visualization product, the quicktime movie may be written to a DVD-R disk for playback on a television with a DVD player. For the highest-quality portable visualization product, the engineer writes the quicktime movie to a compact disk for playback on a desktop or laptop computer.

The digital capture process can be lengthy because, although the visualization software can display the rendered images at high frame rates, the capture software is limited by the speed at which it can write the images to disk. APL accelerated this process by purchasing specialized computer hardware that could save the images to disk as fast as they could be rendered. The addition of this hardware allows a visualization that had previously required days to be captured to now be captured in a matter of minutes.

To allow digital capture of visualizations, it was also necessary to have scripted interactions (e.g., camera angle changes, screen zooms, pauses, playback speed changes) so that the digital capture could be performed without user interaction. To facilitate this, the visualization software was written with a generic event interface that allowed all actions to be driven either by a user interactively invoking events through the keyboard and mouse or by inserting fixed events into the configuration script.

Reconstruction Visualization

The second class of engineering visualization—reconstruction visualization—is built on the same visualization framework as single-screen nondistributed visualization. Reconstruction visualization allows the weapon system engineer to simultaneously compare the actual performance of real-world weapon systems to the performance predictions of M&S tools. The sources of the engineering information for this visualization class include data from actual weapon systems and telemeters from weapon systems participating in an at-sea or land-based test firing. Reconstruction visualization allows the engineer to validate the M&S tools and present the findings to the sponsor.

The visualization entities in a reconstruction visualization are the same as those for the single-screen nondistributed class of visualization. The data, however, are collected from several of the participating weapon systems in the test. The data germane to the ship and shipboard weapon systems—ship location, ship heading, raw radar track from the SPY radar, filtered radar track from the weapon control system (WCS), and illuminator information from the fire control system—are all collected by the ship systems and sent to APL via a secure link to the Aegis performance assessment network. Engineering information germane to the defending missile system—the missile state bilevels, position, attitude, velocity, acceleration, and gimbal angles from the seeker—are collected from telemeters onboard the defending missile and transferred to APL via a secure network link with the Naval Warfare Assessment Station. The engineer must use several data fusion techniques to overcome the disparities in the data and integrate all of the engineering information into a single visualization.

Telemetry data for the reconstruction is collected from separate physical telemeters, all of which have unique clock and coordinate systems. To integrate these data sources, the engineer must align all telemeter clocks to a standard clock (usually Greenwich Mean Time) and resample all of the variable frequency data to a single, fixed sampling rate. Then he must align the coordinate systems. The coordinate system of the data collected from the missile telemeter is usually missile body frame, whereas the coordinate system of the data collected from the ship is usually either downrange-crossrange-up or east-north-up. To import these data into the visualization, the engineer must transform the data from their native coordinate systems into the coordinate system of the visualization and then merge the processed data from the various telemeter files into separate data files for each visualization entity.

The next step is for the engineer to collect the data from actual sensors in real-world weapon systems. These data will inherently contain noise and singularities due to the physical characteristics of the sensor, so the engineer must remove the singularities from the data or replace them with an average of surrounding points. Noise from the data must be removed with filtering techniques appropriate to the type of trajectory that is being reconstructed. Low-resolution, high-altitude, radially inbound threat tracks require different noise filters than high-resolution, low-altitude, weaving threats. The engineer determines the proper filter based on the quality of the track, the dynamics of the track, and any known peculiarities specific to the particular tracking sensors and telemeters recording the data.

To produce a best estimate of the actual filtered trajectories, the engineer works with several quantities and weighs them based on confidence in the data. For example, if there is high confidence in the actual intercept point of the threat and missile, the engineer may choose to fix the final point in both trajectories to the actual intercept point and filter the position data from either the missile telemeter or the SPY radar track data back to the trajectories' respective origins. However, if there is no confidence in the actual intercept point but instead confidence in the launch position and the velocities of the two trajectories, the engineer may fix the launch locations and determine the position of the two entities using the velocity from either the SPY radar track data or the missile telemeter data.

It is obvious from these two examples alone that the process of converting raw real-world data collected from sensors and telemeters into a filtered best estimate of the actual trajectory is a delicate and sensitive one in which data integrity must be maintained yet known anomalies filtered out. Through optimization and automation techniques, the time required to process these data from the time that they arrive at APL has been shortened from several weeks to about a day.

The visualizations of this real-world engineering information are used to analyze the actual performance of the SPY radar, illuminators, defending missile, and threat weapon systems participating in an actual at-sea test. Reconstruction visualization may be compared to M&S predictions for the at-sea test performed before and after the actual test. By overlaying all of this engineering information into a single visualization, the engineer can simultaneously observe APL's pretest modeled performance prediction, the actual performance of the actual weapon system from the at-sea test, and the posttest modeled performance prediction (Figs. 12 and 13).

Quicktime Visualization

The third class of visualization, quicktime visualization, is inherently distinct from the other classes. It is by nature a 2D visualization and may be created offline by virtually any third-party software package that is used to analyze engineering information. APL developed quicktime visualizations to meet a requirement for integrating 2D engineering information into 3D visualizations and synchronizing the playback of the information to the executing quicktime. Examples of such 2D image sequences are returns from onboard sensors, results of finite element analysis of intercepts, results of discrimination and handover algorithm analysis, quad charts displaying engineering information exported from third-party applications, and slide show presentations (Fig. 14).

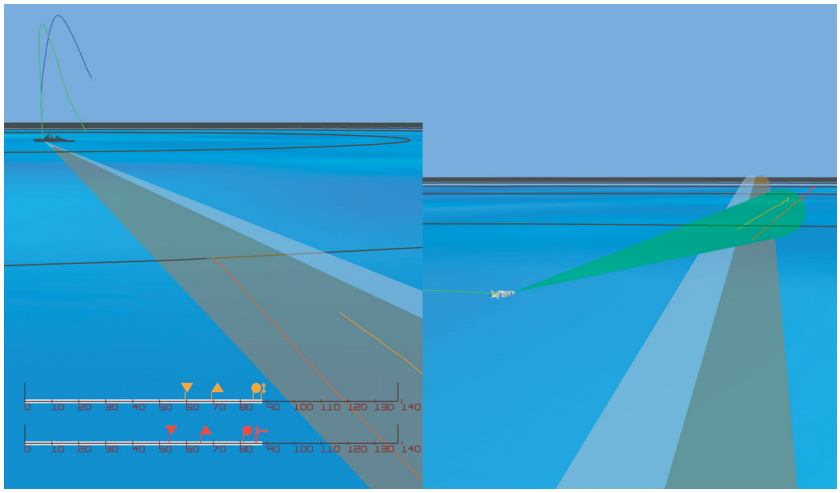

Figure 12. To the left is a reconstruction from the threat perspective. The beams are from the ship's illuminators. To the right is the same reconstruction from the missile's perspective. The green cone emanating from the missile is its seeker.

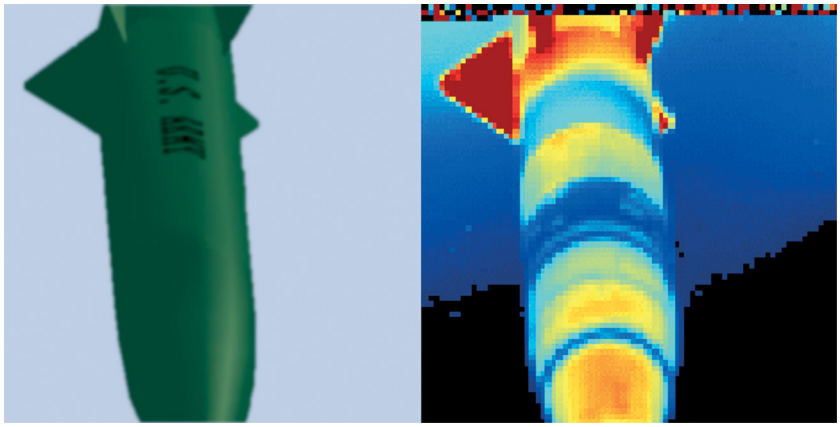

Figure 13. A reconstruction of the successful Standard Missile-2 engagement of a Lance target over the White Sands Missile Range in 1997. The SM-2 telemetry data were analyzed to determine the relative range, velocity, and attitude of the missile and target at endgame and were then used as input to the visualization. The visualization screen is divided into two parts: the first screen shows a scene of the endgame from a virtual camera oriented according to the SM-2's infrared (IR) camera gimbal angles, and the second screen shows the seeker IR images captured from the actual flight test. By setting the field of view of the virtual camera and orienting the camera according to the actual IR seeker's orientation, as reported in the telemetered data, APL was able to visually confirm that the visualized geometry was correct. The visualization was then used to extrapolate past the last available telemetry data point to not only conclude that the engagement was a "direct hit" but also pinpoint where on the target the missile collided.

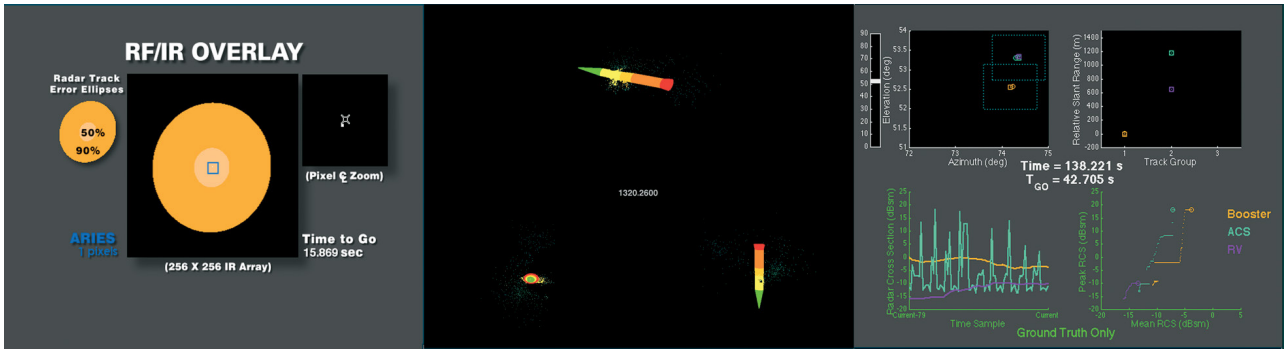

Figure 14. Three examples of quicktime visualizations: the left image is a visualization of the handover and discrimination process, the center image is a visualization of the finite element analysis of intercept, and the right image is a quad chart of some metrics relevant to the radar entity.

To generate the quicktime visualization, the frames of the quicktime movie are produced using third-party software applications. Next the images and times for each respective frame of the visualization are collected and the images are compressed into a quicktime movie. The engineer must associate a time to each of the frames in the quicktime movie. Then he begins playback of the quicktime visualization and synchronizes the current frame time of the running quicktime movie to the current time in the running master visualization. Quicktime visualizations may either be viewed as a stand-alone visualization or as a node in a multiscreen distributed visualization, which is described in the following section.

Multiscreen Distributed Visualization

The fourth class of visualization that APL has developed is a multiscreen visualization that is distributed across several computers. The Laboratory had a requirement to simultaneously visualize several weapon systems in great detail. This could not be accomplished with a single screen, so the visualization was designed to span multiple screens. Instead of attempting to render the visualization onto multiple screens using a single computer, multiple computers were used, with each computer rendered to an individual display. This allowed APL to take advantage of the resources of several midrange visualization computers for the multiscreen visualization that would otherwise require a single high-end visualization computer.

Multiscreen distributed visualization is accomplished by building a visualization for each midrange computer that displays a subset of the visualization windows and then running the visualizations simultaneously on each computer. The visualization software contains network software that enables APL to keep the visualizations synchronized, creating the appearance of a single, coherent visualization (Fig. 15).

The synchronization software used for the multiscreen visualizations also enables synchronization of other software with the visualization. Most notably, it

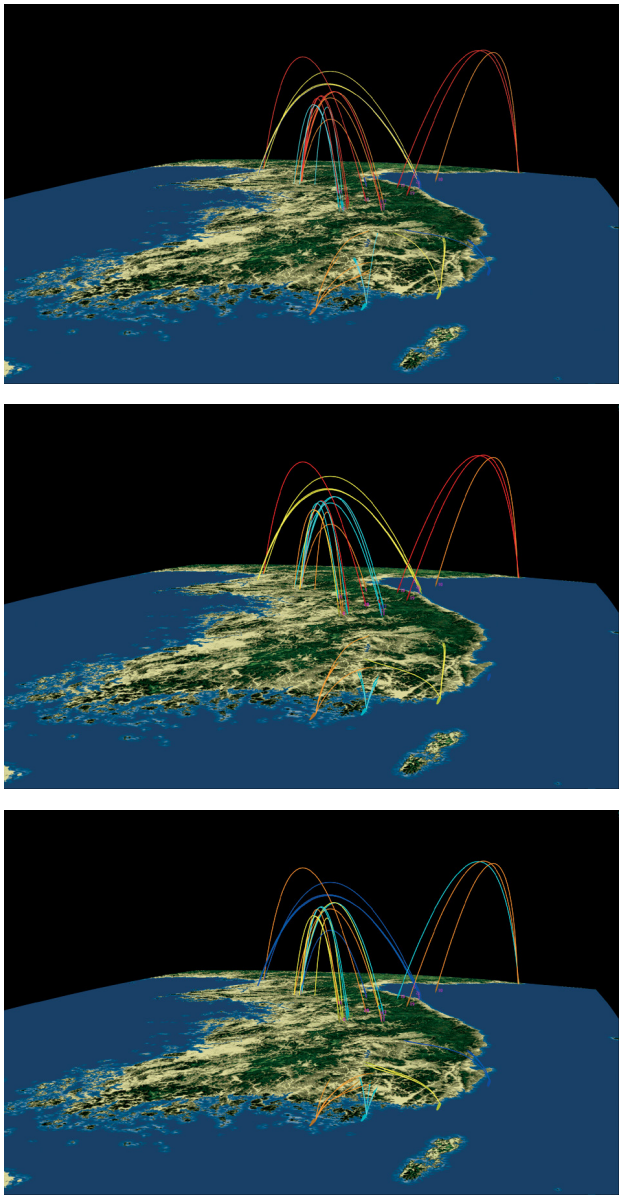

Figure 15. A multiscreen distributed visualization of the performance evaluation of the same scenario with three separate coordination algorithms. By visualizing all three algorithms simultaneously, APL was able to clearly see how each algorithm affected the overall performance.

allows for the integration of the quicktime visualization captured from other sources.

ARTEMIS Three-Screen, Nine-Window Visualization

APL developed the most recent visualization class—the APL Area/Theater Engagement Missile/Ship Simulation (ARTEMIS)[1] three-screen, nine-window visualization class—to realize specific visualization goals for a new type of M&S effort in the Navy Theater Wide TBMD program. ARTEMIS is a high-level-architecture federation of engineering models that integrates existing high-fidelity weapon system models into a distributed architecture. These distributed models, or federates, exchange engineering information among themselves as they execute, thus creating a closed-loop, end-to-end simulation.

The ARTEMIS simulation consists of several separate federates, each requiring a unique window in the visualization. To simultaneously visualize all of these federates, APL designed the ARTEMIS visualization for viewing in the System Concept Development Laboratory as a three-screen display in which the left and right screens are subdivided into quadrants. The center screen is a 3D window that shows a wide-area view of the entire scenario; each of the quarter-screen windows on the left and right shows information pertaining to a single federate. Taking advantage of new high-end computers with advanced graphics capabilities, APL has developed a single visualization for ARTEMIS that simultaneously displays each federate, allowing engineers and sponsors to view the enormous amount of engineering information from an ARTEMIS simulation run (Fig. 16).

To best represent each individual federate, APL decided that several federates would be shown as 2D plots or as text readouts instead of the 3D-rendered images for which the visualization software was geared. The solution was to write these non-3D federate windows using the Motif widget set and combine the widgets with the 3D software. The Motif-based federate windows were written into a library separate from the core 3D visualization software so that these federates could run in a separate process, taking advantage of APL's multiprocessor system and minimizing the impact on the 3D rendering software. Shared memory was used for communication between the Motif and the 3D processes and for integrating the two processes onto a single display by reparenting the 3D windows into the Motif screen.

Even with three 1280×1024 pixel screens, the display real estate is at a premium when trying to simultaneously visualize all of the federates; therefore, any of the quarter screens can be interactively selected and enlarged to take up one, two, or all three screens. When a federate window is enlarged, the engineer populates the remaining window area with additional metrics and other engineering information pertinent to the window's federate. To conserve screen space, displayed data are dynamically changed through the use of Motif selection lists, dynamic and logarithmic scales, scrollable text lists, and context-sensitive popups that can display extended data.

The global window is the center visualization window for ARTEMIS. It is the only window that occupies one entire screen and is similar to a single-screen nondistributed visualization. The global window visualizes the ARTEMIS run from a fixed view far above the Earth. It contains the terrain visualization, the ship from which the defending missile is launched, the threat and its launcher, and any relevant beams or volumes from the SPY radar or infrared (IR) seeker onboard the fourth stage of Standard Missile-3 (SM-3). It also contains readouts for metrics such as time and launch position for the missile and threat.

In the upper left-hand quadrant of the left screen, the engineer visualizes engineering information from the scenario manager federate. This window also shows the message traffic exchanged among all of the

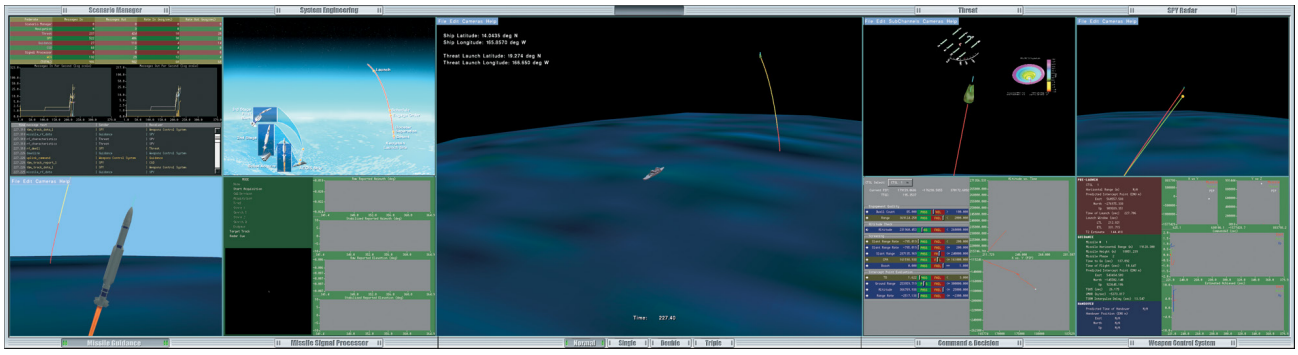

Figure 16. The ARTEMIS three-screen, nine-window visualization. From upper left to bottom right: the scenario manager federate window, systems engineering window, missile guidance federate window, missile signal processor federate window, global window, threat federate window, SPY radar federate window, command and decision federate window, and WCS federate window.

ARTEMIS federates. As the visualization progresses, federate-to-federate message metrics scroll by in the scenario manager window. The window contains the sender of the message, the receiver of the message, the message type, the time at which the message was sent, and plots of the message traffic rates. Also, if the message contents are available, the user can select messages and display their contents in a pop-up window.

To the right of the scenario manager federate window is the systems engineering window, which displays a slide show that is synchronized with the visualization. As the visualization progresses, the slide show presents critical events, such as launching, staging, and intercepting, as they occur.

In the lower left quadrant of the left screen is the missile guidance federate window which displays a 3D visualization of the missile guidance federate and is similar to the missile entity single-screen visualizations for which the software was originally written. Entities displayed in this window are the terrain, the ship from which the missile is launched, the defending missile itself, the beam representing the seeker onboard the fourth stage of SM-3, and the threat as it enters into view during endgame.

To the right of the missile guidance federate window is the missile signal processor federate window which contains engineering information that is germane to the missile signal processor, including the azimuth and elevation of the IR returns from the SM-3's fourth stage IR seeker, the current mode of the seeker, and raw and stabilized animated plots of the reported elevation and azimuth.

In the upper left quadrant of the right screen is the threat federate window. Like the missile guidance federate window, the threat federate window displays a 3D visualization similar to the original single-screen visualizations. It contains the terrain entity, the threat and launcher entity, and the missile entity as it enters into view during its fourth stage. It also provides menu-selectable options to enable subwindows, which display the current IR signature and radar cross-section signature that the threat is projecting to the defending weapon systems.

To the right of the threat window is the SPY radar federate window which contains search and track information from the SPY radar federate. Here, a 3D visualization of the SPY-filtered threat track and the ground-truth threat track provides a visual representation of the SPY track errors. These errors are also displayed in three overlay strip charts to better show their scale and direction.

In the lower left quadrant of the right screen is the window for the command and decision federate. This window visualizes the engageability tests performed by the command and decision federate preceding the missile engage order. The instantaneous results of engageability tests for engagement quality, altitude check, screens, and intercept point evaluation are presented as a table of pass/fail bars. To the immediate right of the table, any of the individual engageability parameters may be plotted.

To the right of the command and decision federate window is the WCS federate window. This window shows such metrics as prelaunch calculations performed by the WCS federate; metrics from the midcourse guidance of the missile, which is handled by the WCS; and metrics from the handover event in which the WCS "hands over" information to the missile signal processor. To the immediate right of the tabular data, the user may view plots for any of the federate's metrics.

THE FUTURE

APL intends to develop many additional visualization capabilities and incorporate them into the current engineering visualization:

- Reconstructions: Include video footage from launch site as well as footage from airborne and onboard missile/threat video cameras and engineering information from the ground station and satellites in the visualization.
- Communications: When relevant to the visualization, add a display of communications as they pass back and forth among the visualization entities.
- IR signatures: Develop a method by which instantaneous IR signatures of the threat can be dynamically mapped onto its skin as it flies.
- Finite element: Improve upon the Sphinx hydrocode tool's 2D visualizer used at APL to evaluate postintercept lethality of threats through finite element analysis.
- ARTEMIS: Make APL a visualization federate in the ARTEMIS federation of models, enabling users to visualize ARTEMIS concurrent to the simulation's execution.
- Undersea: Develop an undersea visualization capability that will allow APL to study undersea-launched weapon systems.
- Multimission: Develop an engineering visualization for nodal analysis tools, such as the APL Coordinated Engagement Simulation or the Extended Air Defense Simulation.
- Radar: Add fuller functionality to the SPY radar visualization in a scenario, including error ellipsoids, launch event correlations, and increased precision for search beam locations, clusters, groups, and any other metrics which may be pertinent.
- Particle system: Develop a multiprocess particle system for efficiently rendering great numbers of particles for improved smoke and debris representation in the visualization.

SUMMARY

High-fidelity engineering information is critical to facilitate the design, analysis, and M&S of complex weapon systems to defend against threats to our nation. The technical community and the associated DoD sponsors must have confidence in that engineering information and must understand it at various levels, depending on the functionality of the technical community and sponsors in the overall acquisition process. Contemporary engineering information is becoming even more complicated because of advances in threat technology, which drive the need for more complex weapon systems to defeat them. Also, the more recent DoD-wide focus on a distributed simulation capability, which integrates physically separate high-fidelity physics-based simulations, is also contributing to the increasing complexity of the engineering information. APL has developed a state-of-the-art engineering visualization capability which has become an essential systems engineering tool by providing confidence in and comprehension of complex engineering information.

REFERENCE

[1]Pollack, A. F., and Chrysostomou, A. K., "ARTEMIS: A High-Fidelity End-to-End TBMD Federation," *Johns Hopkins APL Tech. Dig.* **22**(4), 508–515 (2001).

ACKNOWLEDGMENTS: The authors wish to acknowledge Bernie Kraus, R. Kent Koehler, Kevin Wilmore, Doug Ousborne, Dave Wu, Phil Miller, Richard Freas, Bill Critchfield, Nancy Crowley, Chad Bates, and Simon Moskowitz, as well as the myriad APL modelers who provided their expertise and data to us.

THE AUTHORS

DAVID E-P. COLBERT received a B.S. in engineering physics from the University of Illinois at Urbana-Champaign in 1996. He joined the Air Defense Systems Engineering Group of ADSD at APL in 1996. Since then, Mr. Colbert has been developing visualizations of naval weapon systems for Area and Self-Defense Anti-Air Warfare, Area TBMD, Navy Theater Wide TBMD, Overland Cruise Missile Defense, Cooperative Engagement Capability, and Joint Mission Defense. He is currently working on oceanographic surface wave analysis, oceanographic environmental analysis, and engineering visualization as a member of the APL Associate Professional Staff. His e-mail address is david.colbert@jhuapl.edu.

R. EDWARD RALSTON is a member of APL's Senior Professional Staff in the Air Defense Systems Engineering Group of ADSD. He received a B.A. in mathematics from St. Mary's College of Maryland in 1992 and an M.S. in mathematics from Texas Tech University in 1995. Previously he worked for Veridian and Electronic Arts as a graphics software engineer. He first came to APL as a subcontractor in 1996 and was tasked with redesigning existing visualization software into a reusable object-oriented framework. Mr. Ralston became an APL staff member in 2001 and is currently developing testbed software for the MESSENGER team. His e-mail address is ed.ralston@jhuapl.edu.