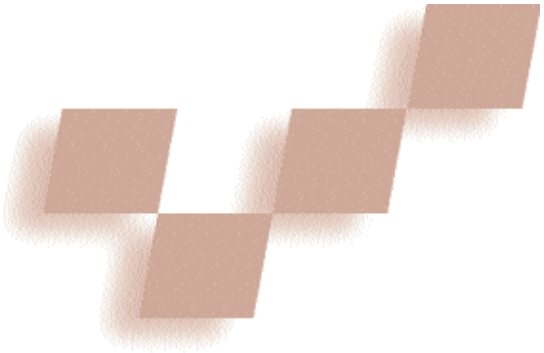


Visualization for Situational Awareness

Eliot Feibush and Nikhil Gagvani
Sarnoff Corporation

Daniel Williams
Systems & Scientific Software

Visualization techniques in a situational awareness system aid rapid comprehension of a complex battlespace. Hardware scalability lets remote users share situational awareness with the command center.

Situational awareness visualization of joint military forces requires the representation of large geographic areas and thousands of military units. High-level commanders are interested in the overall tactical deployment of their forces, enemy forces, and noncombatants. The commanders are not looking for highly detailed views of engagements between units. A Joint Forces operation simultaneously involves ground, sea, and air forces, and can include forces from a coalition of different nationalities. The visualization shows if the different forces are operating in support of each other.

Military units must be displayed in the context of terrain, on the sea, under the sea, and in the air. The extent of the area of interest (playbox) is typically one million square miles. Observers want to see the highest resolution data available and to navigate through the model at interactive rates. The geospatial information displayed consists of elevation data, a variety of maps, and high-resolution imagery. The size of the playbox and the amount of data place a considerable demand on both the terrain model and the visualization software. Collateral maps and images aid in understanding the movement and placement of units.

The battlefield is a large, complex arena. Researchers have developed computing environments to host situational awareness applications. Durbin et al. incorporated virtual reality to enhance human understanding and interaction.[1] Hix et al. reported on a virtual environment for battlefield visualization and its evaluation.[2] Rosenblum et al. described a situational awareness environment for directing the US Marines.[3] The term situational awareness has been applied to other aspects of the military, such as the situational awareness of an individual pilot or gathering information about specific assets. We focus on theater-wide situational awareness covering a large geographic area containing many military units.

The Joint Operations Visualization Environment (JOVE), developed at Sarnoff to assist military decision makers, optimizes the display of tactical data for information presentation. JOVE uses three rear-projected displays driven from an SGI Onyx2 Infinite Reality2 computer. Stereoscopic display is available. A joystick, a graphical user interface, and speech commands provide interaction. JOVE delivers accurate navigation through the model over a range of viewer positions. A typical scene from the situational awareness application appears in Figure 1.

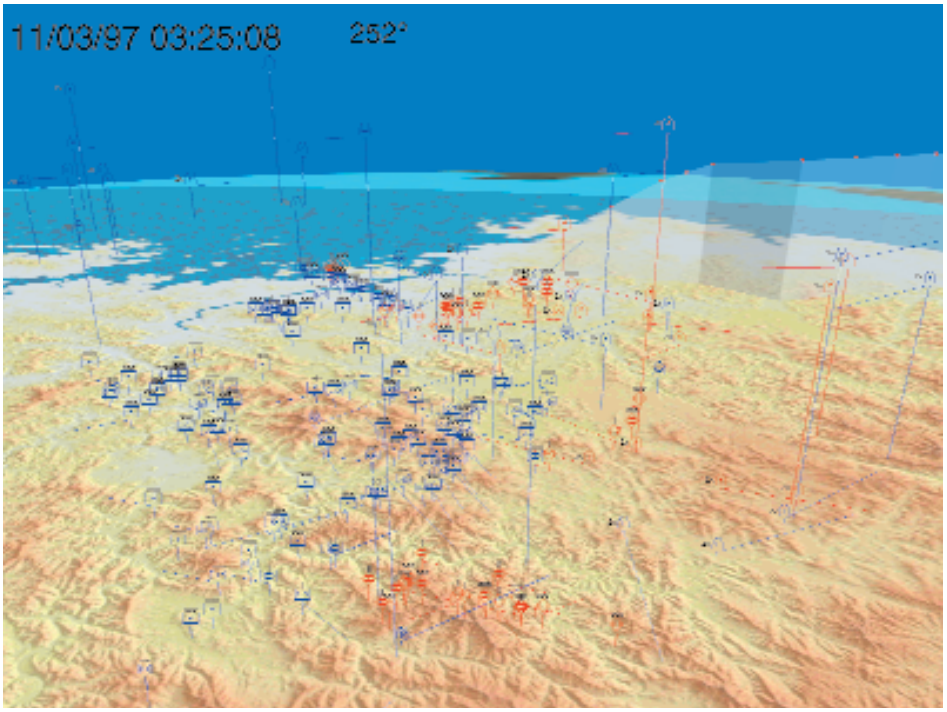
This case study describes techniques used for effectively modeling and navigating geospatial and tactical data for situational awareness. The visualization technology explained here could be applied to commercial air traffic control, emergency management response, and geographic information systems as well.

Software architecture

Our visualization software is based on the client-server paradigm. Several independent software modules communicate through CORBA (Common Object Request Broker Architecture).[4] This approach facilitates well-defined interfaces between modules, independent development of modules, and communication between modules running on different computers. The central module, the Display Server (shown in Figure 2), is written for Iris Performer,[5] a software toolkit containing rendering techniques for visual simulation. It maintains and traverses the scene graph for rendering by the graphics system. The graphical user interface for the application runs as a client and submits requests to the Display Server. Typical requests include setting the view and loading geospatial data.

The military data feed sends information about the units. The data, consisting of time-stamped messages describing the position and attributes of a unit, is stored in the tactical database. The system stores every data point so that the events can be replayed over time, and

0272-1716/00/$10.00 © 2000 IEEE



1 Military units are represented symbolically for situational awareness. A stalk pinpoints the location of each unit and the altitude of each aircraft. The red dots on top of the gray curtain show the location of an airplane during a time period. The terrain is drawn as a texture, shaded and color-coded by elevation. The time and date of the situation appear at the top along with the compass heading of the view direction.

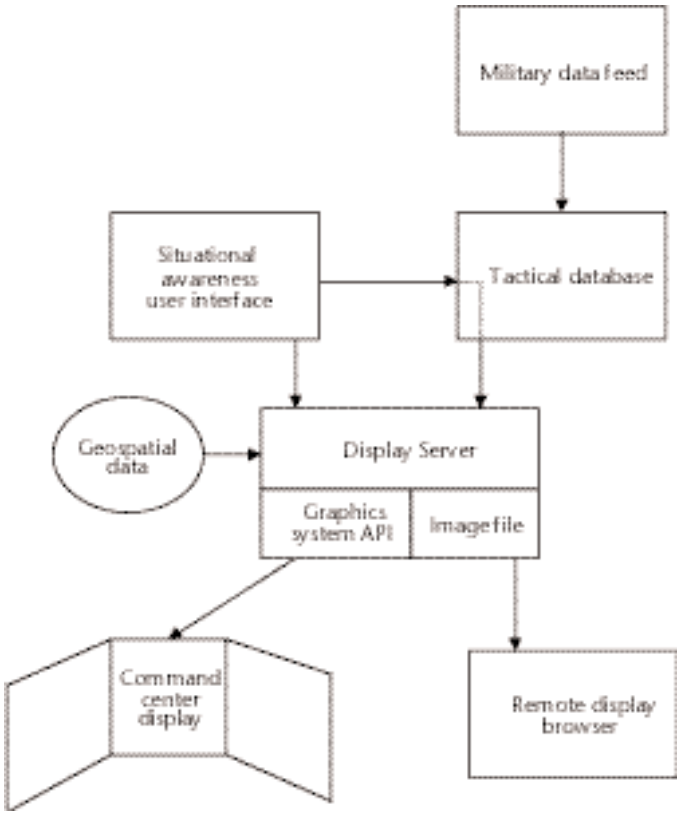
historical tracks can be generated. The tactical database sends requests to the Display Server to draw the units. The user interface lets the user specify exactly which units to display and at what point in recorded time.

JOVE has been deployed at training exercises where computer simulation provides the unit data. Each branch of the military has its own simulator that only displays its own units. The simulators broadcast their events through a standard protocol.[6] JOVE receives the messages and is the first system to display the units of all forces in a common picture.

The Earth model

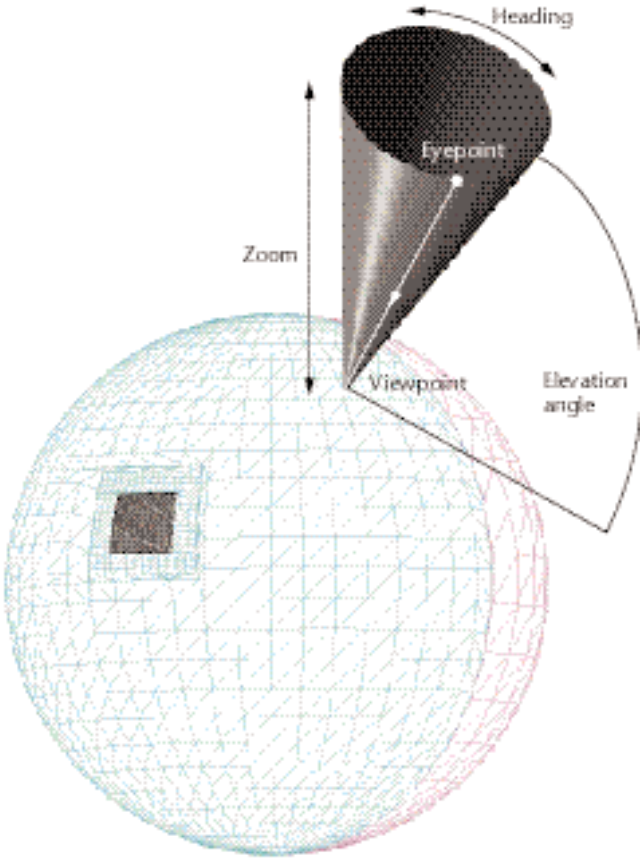
Our model starts with a sphere representing the Earth. We portray the entire earth because

- Units can move out of the playbox.
- Far-ranging units, such as aircraft, may be stationed anywhere around the world.
- Additional situations could develop outside of the detailed playbox.



2 Visualization software architecture.

Satellite imagery of the Earth is texture-mapped onto the geometric model. The cloudless geosphere imagery (http://www.geosphere.com) portrays the earth at 4 kilometer resolution. Lines of latitude and longitude overlay the imagery at 10 degree intervals. We modeled the sphere geometry to use polygons efficiently and also to avoid thin triangles near the poles. The highest polygon density occurs in the playbox, where most of the action takes place. Similarly, we modeled the near hemisphere containing the playbox with more polygons than the distant hemisphere. The entire Earth and playbox can be



3 The Earth model has the highest polygon density in the playbox (black), lower in the surrounding hemisphere (cyan), and lowest in the far hemisphere (magenta). The Cone Viewer navigation model positions a cone perpendicular to the Earth. The eye looks from the rim of the cone to the tip of the cone on the Earth's surface.

modeled with about 4,000 triangles in this way, as shown in Figure 3. We generated the polygon geometry using the OpenGL Optimizer toolkit, which stitches together the two hemispheres and the playbox seamlessly. By specifying the topology as three regions defined by shared edges, Optimizer created a crack-free triangle mesh.

The sphere geometry is initially calculated in spherical coordinates (latitude, longitude, and elevation), then converted to Cartesian coordinates. All graphics calculations use a Cartesian coordinate system with its origin at a corner of the playbox. We chose Cartesian coordinates instead of spherical coordinates for fast processing by the hardware. The SGI graphics hardware implements a nonlinear *Z*-buffer with maximum precision at low *Z* values. Therefore, having the origin at a corner of the playbox allows the highest numerical precision in the most critical area and avoids artifacts during the hidden surface removal process.

The geometry for the terrain elevation model, maps, imagery, and military units are geographically registered to and integrated with the Earth model. The geometry is initially calculated in spherical coordinates and then converted to Cartesian coordinates to optimize graphical rendering. The point of view moves seamlessly across the different features, so the observer maintains context even with large changes in viewpoint.

Navigation model

We designed our model for controlling the view of the scene so that users could not crash into the Earth's surface or get disoriented, flying off into space. The point of view is restricted to locations on the Earth, and the eye point lies on a virtual cone perpendicular to the earth's surface. We have observed inexperienced computer users navigate comfortably with the Cone Viewer on their first try. This model works well for navigating environments positioned on a surface when the viewpoint is primarily from above. The Cone Viewer could benefit geographic information systems, weather displays, or looking at parametric values projecting from a surface.

The viewing model consists of positioning the tip of the cone at a point on the Earth's surface, as shown in Figure 3. Moving the tip of the cone to a new point on the Earth changes the point of view displayed at the center of the screen. The eye is restricted to locations on the rim of the cone and always looks toward the tip of the cone. This type of view fosters situational awareness of large areas on the Earth. For our application it is not necessary to look up at the sky or fly through a valley.

The cone's height and diameter are adjustable. Scaling the size of the cone changes the zoom factor by moving the eye either farther from or closer to the tip. Changing the apex angle of the cone, while keeping the distance between the eye and the tip constant, changes the elevation of the view direction. A top-down view is like looking at a flat map. A low-angle view is useful for looking at 3D terrain features. Moving the location of the eye around the circular base of the cone changes the compass heading of the view. We have found it useful to display this heading value (0 to 359 degrees) in the visualization.

The initial user interface to the Cone Viewer used a mouse to manipulate on-screen thumbwheels corresponding to the degrees of freedom. We subsequently added a joystick interface to make navigation more user friendly. The four geometric features of the cone control the view parameters:

1. Location of the tip controls point of view.
2. Location of the eye on the rim controls heading.
3. Scale controls zoom.
4. Apex angle controls elevation angle.

These features map to joystick functions. Pushing the joystick moves the tip in the compass direction and moves the viewpoint. Twisting the joystick moves the eye around the rim and rotates the view. Twisting while squeezing the trigger button changes scale (zoom). Pushing while squeezing the trigger changes the height and diameter of the cone proportional to the elevation angle. To maintain interactive rates when navigating, some of the symbols are not drawn while the joystick is in use. When the navigation stops and the view remains constant, then the system redraws all the symbols. Combining interactive navigation with a single, integrated geospatial model helps users build a coherent mental model of the terrain and its features.

The Cone Viewer with the joystick interface has performed successfully in many military exercises that we have observed. Its simplicity and reliability attract the interest of first-time users, encouraging them to try other aspects of the system.

Elevation visualization

Elevation postings are typically measured at 100-meter intervals and stored in data files. A square degree of land at 100-meter resolution contains about 1,440,000 sample points. Since the playbox can cover more than 200 square degrees, representing the terrain at full resolution with a uniformly triangulated polygonal model greatly exceeds the capabilities of current graphics computing. More efficient triangulation methods will yield better performance.[7,8] However, current high-end graphics hardware can display very large textures at high performance. Our first elevation visualization technique takes advantage of high-resolution texture rendering. Our second technique uses a more display-efficient polygonal model.

The first technique represents the entire elevation data as color-coded texture. The value of each elevation point is color-coded by height and entered into a large texture map. This texture is mapped onto a spherical patch of geometry corresponding to the playbox. This approach enables displaying the terrain at its full resolution and makes for well-defined coastlines and other detailed features as shown in Figure 1. The color-coding enables the user to develop a perception of the terrain, such as areas having tall mountains and areas of flat plains.

Initially we used just the color-coded value for the texels (pixels in the texture map). Subsequently we shaded the color-coded texels according to the 3D shape (relief) of the terrain. Observers preferred combining color-coding with the visual cue of relief shading because it reinforced their perception of the terrain. We describe the implementation of the relief shading technique in the next section.

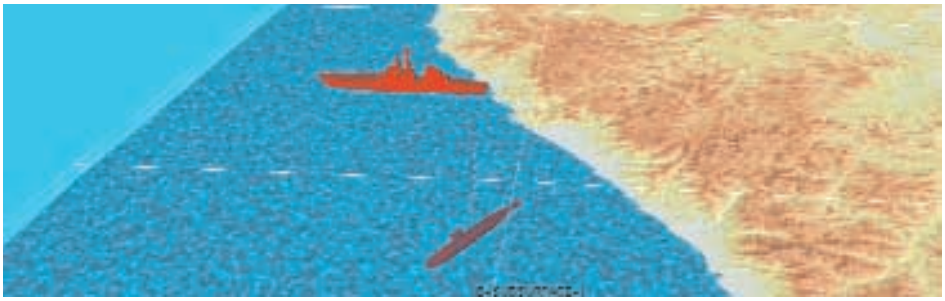
The second technique uses a 3D model of the terrain—a triangulated irregular network model—to optimize rendering performance. Close to the surface, a 3D picture of the terrain is preferable. This enables users to not only perceive hills and plains, but also gain line-of-sight information. For example, if two opposing tanks are near each other but not firing, a hill might block their view. We describe the implementation of relief-modeled terrain in a following section.

4 (a) Elevation postings are color-coded by height. (b) Reflectance map of elevation values with illumination from the northwest. Gradient information from the elevation data is used to produce relief shading. (c) Color-coded, relief-shaded elevation. The color-coded values (a) are modulated by the reflectance map (b) to produce this shaded elevation texture.

We have used the two techniques to create terrain models deployed at military exercises. Users involved with ground-based units preferred the 3D relief model. Users of aircraft were satisfied with the shaded, color-coded texture because the relief shading resembles the hill shading printed on maps that pilots typically use for mission planning.

Color-coded, relief-shaded, textured terrain

We combined two methods to better convey the topography of the digital elevation model. First, we developed a color-coding scheme to reinforce the perception of elevation. Blue and cyan cannot be used because they are reserved for water and sky. Green is perceived as vegetation, and white looks like snow. Eliminating blue, cyan, green, and white leaves gray, yellow, and red and their intermediate blends. Observers preferred gray for low land and red for high elevations. We used this color scheme to translate each posting of the elevation model into an appropriately colored texel in a texture (Figure 4a). Then we computed a reflectance map using Method



5 The submarine and its label are drawn below the translucent, textured water.

P, A Simple Alternative,[9] which roughly simulates how the terrain will look if viewed from above with the sun's illumination coming from the northwest (Figure 4b). The method uses gradient information from the elevation data to produce a relief-shaded texture. We used this reflectance map to modulate the colored texels to achieve the final texture map (Figure 4c).

Relief-modeled terrain

Visualization of the 3D terrain promotes understanding the speed and movement of ground forces and planning tactical maneuvers. Elevation data points are organized into a triangular, irregular network model using the method proposed by Heckbert and Garland.[10] A monolithic model for the entire playbox cannot be rendered interactively, so we split the model into multiple tiles. Tiles outside the view volume are culled prior to rendering. Each tile also has multiple levels of detail, which switch to lower detail as the eyepoint moves further away.

Contextual cues accurately model coastlines, especially for low-resolution tiles. Threshold-based segmentation followed by edge detection creates a coastline mask from the elevation data. This mask serves as an input to the triangulation algorithm to generate a larger number of triangles near the coastline.

If each tile is individually modeled, the triangles do not match up at the edges between tiles. To alleviate this, we create a huge model for the entire playbox and then split it into tiles. Such an approach provides seamless terrain at the cost of more triangles. However, when adjacent tiles have different levels of detail, cracks can appear in the terrain. One solution eliminates the cracks with additional geometric modeling.[11] Our approach displays texture-rendered, flat terrain below the relief model to achieve a visually continuous appearance.

The relief model typically occupies several hundred megabytes of disk space and takes considerable time to generate. To facilitate rapid color changes without regenerating the relief model, the system handles color as a one-dimensional texture. It assigns texture coordinates to the relief polygons according to elevation and creates a very small texture map using the elevation colors. The 1D texture rendered on the relief model reloads instantly. We implemented rapid traversals for coordinate and normal computations in Iris Performer to achieve scaling. Two-dimensional textures from maps and satellite imagery can be draped on the terrain. Again, the JOVE software includes texture coordinate computation to enable switching between 1D and 2D textures for runtime switching of maps.

The actual vertical displacement of terrain relative to its area is small. The vertical dimension only covers a few pixels when a large geographic area is rendered on the screen. To enhance the observer's perception, the vertical displacement is scaled up to exaggerate elevation. In the JOVE user interface, a slider interactively controls the vertical scale. Users typically prefer a scale factor of 5 for the relief model when viewed from high in the air. For line-of-sight evaluation along the ground, the user can set the vertical scale factor to 1, the original elevation data. This helps in determining if one ground unit can see another. The terrain is modeled on a spherical Earth, so it is not possible to apply a simple scale transformation to the original 3D model. Therefore the 3D coordinates of each point must be recalculated based on the scaled elevation.

Sea and under sea visualization

Joint Forces operations include surface craft and submarines. Our technique for displaying sea units is to model a translucent, textured surface for water areas within the playbox. Initially we created a noise texture tile that covered a 1×1-degree square area (about 60 miles on a side). Although adequate for close-up views of the coastline, it became visually objectionable for playboxes with large water areas. In a distant view, showing a number of noise texture tiles, the macro pattern of identical, adjacent, water texture tiles created a visual artifact. To eliminate the repetitive patterning, we generate a fractal sum noise texture[12] with enough texels to cover the entire playbox. By merging this water texture with the elevation postings, the land area texels replace the fractal pattern in areas not covered by water. Each texel in the merged texture has a transparency value. The land texels are opaque, and the water pixels have varying degrees of transparency according to the noise function.

Opaque blue sea floor and walls modeled below the water surface visually enclose the sea area. The texture on the water enhances the perception of the surface and partially obscures the submarines. This clarifies the position of the submarine as below the water. The lines of latitude and longitude are drawn over the submarine symbol and label, as in Figure 5.

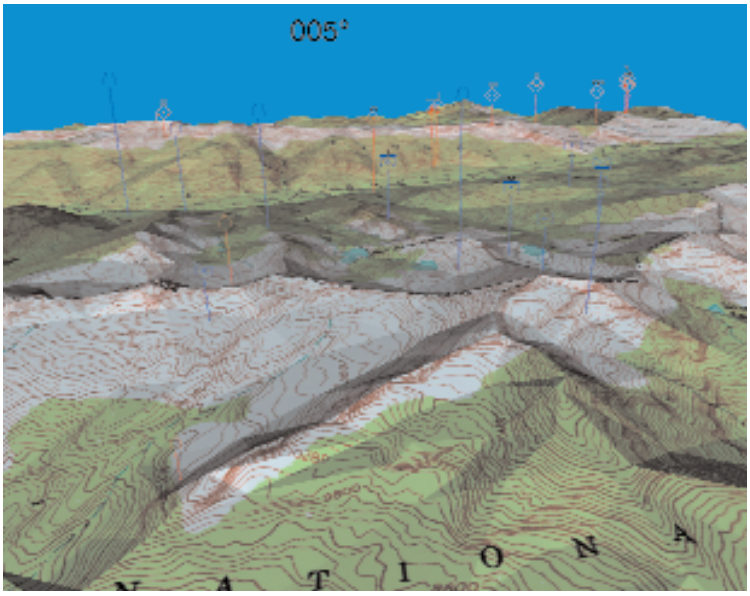
Maps and imagery

Digital maps and imagery, available for display in the terrain model, provide contextual geographic detail to enhance the observer's perception of the features pertaining to the tactical situation. We have used maps that vary in scale from 1:2,000,000 to 1:24,000. The map's content varies with the scale. The scale of the content

of 1:2,000,000 matches the scale of a far-ranging view of the entire playbox. The content of more detailed maps cannot be resolved from a distant view. As the view moves in, the content of the more detailed maps becomes visible, so it is effective to switch the display to a different map.

Maps and images are stored as high-resolution textures. The origin of each texture is registered to a point of latitude and longitude, so it can be positioned within the Earth model.

Figure 6 shows a 1:24,000 contour map texture rendered onto the 3D relief model—useful for determining where ground forces can move and how quickly. The land areas in other maps we have used are shaded corresponding to the terrain. Texture rendering a shaded map onto the 3D model visually reinforces the map shading with the geometrical shape. Pilots and aircraft commanders felt this was a very effective presentation. If the map is higher resolution than the terrain, the map will appear distorted and visually conflict with the relief model. If the map or imagery is significantly higher in resolution than the elevation postings, then it is rendered onto a small geometric surface positioned above the terrain as a spherical patch.



6 A 1:24,000-scale map draped onto the 3D relief model. Military units are displayed as symbols.

Displaying large maps

Combining adjacent digital maps (pixel arrays) into a large texture facilitates their display on certain graphics hardware. We combined sixteen 8K × 8K arrays into one 32K × 32K texture map. This texture of more than 1 billion texels is stored on disk as a precomputed mipmap of 16 levels, with each level organized in 512 × 512 texel tiles. Large textures are displayed using clipmaps[13] when running on an SGI Onyx2 computer with Infinite Reality2 graphics hardware. Clipmaps permit displaying very large textures by efficiently loading texture tiles from disk to the texture memory of the graphics hardware as needed. An 8K × 8K clipmap for one digital map occupies 8.66 megabytes of hardware texture memory. The 32K × 32K clipmap uses 12.66 Mbytes of texture memory. It is much more efficient for display to composite geographically adjacent digital maps into one large clipmap than to create a clipmap for each individual pixel array.

The *clipcenter* is the point on the model that is given the highest resolution texture, with the resolution decreasing at points away from it. The default algorithm in Iris Performer for setting the clipcenter places it at the point on the model closest to the eyepoint. For our viewing model, this holds true only when the eye is overhead and looking straight downward. We implemented our own clipcentering algorithm using a ray from the eyepoint outward into the view frustum. We use the viewing model elevation as the input to a function that bends the ray from the center of the frustum (when looking straight down) to the bottom of the frustum (when looking straight ahead.) The intersection of this ray with the model (or the closest point) serves as the clipcenter.

Hardware support for clipmaps is not available on lower end machines. On such machines hardware texture memory (typically 4 Mbytes) can only accommodate a 1K × 1K texture map. Hüttner[14] proposed a mipmap pyramid to overcome this limitation. The method requires potentially time-consuming runtime modification of the geometry.

We use a simple paging scheme to browse large maps on an SGI Octane workstation with 4 Mbytes of texture memory. The map is divided into 512 × 512 texel tiles. We create flat, geo-positioned geometry corresponding to the latitude and longitude extents of each tile and preload it into the Performer scene graph. Initially, traversal for all such geometry is turned off, incurring no rendering burden. The four texture tiles closest to the current viewpoint load in response to user demand. The system maps these textures onto their corresponding geometries, which it makes visible in the scene graph. When the view position changes, new tiles can load automatically or when the user requests. With automatic loading we achieved update rates of 3 frames per second on an Octane.

Tactical visualization

Theater-scale situational awareness focuses on the military units' tactical information. The input data feeds provide information about each unit. The information typically consists of loyalty (friendly, neutral, or hostile), location, role, size, heading, and speed. The size of ground units ranges from squad to brigade, but only a single location value is specified. Role refers to function, such as infantry or motorized armor for ground units and fighter or tanker for aircraft. Aircraft have altitude and submarines have depth. Realistic rendering of fine detail is not required. Efficient presentation of the information in the data feed is critical.

We implemented two techniques for visualizing tactical units—symbols and 3D models. Although the 3D

models have an intuitive appeal, we have observed that experienced users derive more information from the symbols. For aircraft, in particular, the same airframe can be equipped for a variety of roles, such as transport or surveillance. The functionality—indistinguishable in the 3D models—is very obvious in the symbolic representation. We based the symbols on Military Standard 2525A, developed by the US Department of Defense. The color and shape of the symbol indicates friendly, neutral, or hostile. The shape indicates air, sea, or ground. The icon in the center of the symbol represents role. The size of a ground unit appears symbolically above its rectangular frame. For aircraft, we display a number and an X, such as 4X, to indicate a flight of 4 aircraft reported in the data feed.

The system renders the symbols as billboards with depth, ensuring that the front of the symbol always faces the observer while drawing the symbol at its true location in the 3D world. If the symbol had a fixed orientation, then rotating the view could position the symbol at an angle to the observer and make it impossible to recognize.

We scale the billboarded symbols and the models so that they always cover a constant amount of area on the screen. Otherwise, the units would get very large as the eyepoint approaches or become very small and possibly disappear when the eyepoint moves away.

A stalk line drawn from each unit pinpoints its location on the Earth's surface. This is especially important for clarifying the location of aircraft. Portraying the location and history of air units in 3D has had compelling results compared to systems that only draw the path in 2D on a flat map. We observed a simulated theater ballistic missile launch at an exercise. From the 3D visualization of the trajectory and rapid rise in altitude, the commander could immediately identify the air unit as a missile. The 2D display only showed an air unit, and several minutes elapsed before the operators at the 2D display could identify it as a missile and not an airplane.

A heading vector drawn for each unit shows direction of travel. The length of the vector is proportional to the unit's speed. We have observed situations where diversely oriented ships simultaneously turned toward the coast to begin an amphibious landing. Seeing the alignment of the heading vectors provides an early indication of coordinated movement. Large ground units, such as a brigade, occupy an area in reality, but only a single point of latitude and longitude is reported as their location. Therefore we display the stalk instead of outlining an area.

We have participated in joint force exercises where the data feeds reported up to 15,000 military units. The screen resolution is much too low to display all the units simultaneously. A display filter controls which units are shown. The user sets the filter by specifying a combination of attributes. This is a very flexible and general approach. For example, the user can request the system to display all hostile ground units whose role is "special operations" and all hostile helicopters. Limiting the display to these units makes it possible to locate the air support for an advanced ground assault. This helps the commander understand the situation and formulate a response.

Movement over time proves useful for understanding strategy. The location of each unit is stored in the tactical database for every update. For ships and ground units, the system draws a single line to show each history path. For an airplane it draws a translucent, gray history curtain stretching down to the ground from the plane's flight path. Showing the location and altitude of a plane during a time period helps to determine its mission. Figure 1 shows the history for an airplane diving down for an attack. The history for a submarine also uses a gray curtain, but drawn below the translucent water surface.

We used history tracks to resolve questions of territorial incursion. Without visualization it is a time consuming task or sometimes impossible to manually reconstruct the path of a unit. In one military training exercise where JOVE was deployed, there was a report that a red (hostile) aircraft flew over blue (friendly) territory. This affects the blue commander's response provided it can be authenticated. By recalling the red plane's history curtain, it was immediately clear that the incursion occurred. None of the other computer systems at the exercise could answer this question.

The movement of units can be replayed faster than they actually occurred. Users are comfortable with a replay rate 30 times faster than real time. Fast replay offers an excellent tool for showing prior events to the next shift of personnel. An entire 8-hour shift can replay in 16 minutes, enabling the new shift to see all the events of the previous shift. This is a very efficient method for conveying information.

The tactical database for a one-week exercise typically contains 100,000 events describing the creation, update, or deletion of units. Replay used at training exercises provides after-action review, enabling users to search through time and space to verify the movements and engagements of military units. Visually replaying these events gives participants a coherent picture of the tactical situations that occurred during the exercise.

Collaboration

The military has a hierarchy of command locations ranging from the command center to remote outposts. Thus, the ability to share the visualization throughout the chain of command is desirable. A large JOVE system in the command center can have a 12-foot-wide display, so a group of people can readily look at the display and discuss the situation.

Commanders have reported that one of their biggest challenges is mutual understanding of the tactical picture between the command center and the units in the field. The Display Server runs on a computer in the command center. We developed a technique for remote users with minimal computing resources to collaborate with the command center. If the remote user can run a browser and connect to the server in the command center, then collaborative visualization becomes possible.

We developed software for both the client and the server. The client software runs in the browser and can request the current picture in the Display Server. The remote user can change the view, set the time, and specify which units to display. The request is sent to the server, which generates a new picture that is stored in an image file and transmitted to the client.

Sharing the command center's visualization enables the unit commanders to verify their positions and status. This can be key to preventing friendly fire.

Conclusion

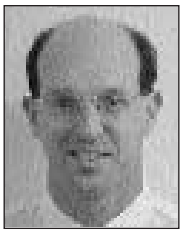
Visualization of tactical and terrain data increased situational awareness. When deployed at exercises, JOVE rapidly clarified complex environments containing large amounts of data. Symbolic representation of military units conveyed more information than geometric models. Polygon-based terrain modeling would have exceeded the capacity of current display systems. However, current graphics hardware supports very large texture maps. Therefore we applied texture-based terrain modeling techniques to situational awareness. Organizing the software as independent modules connected by CORBA facilitated extending the system for collaborative visualization between distant users. For the future we are developing additional user interfaces, reading new data feeds, and developing metrics for evaluating battlefield visualization systems. ■

Acknowledgments

This work was sponsored by the National Information Display Laboratory (NIDL), Princeton, N.J. The Cone Navigation model was developed by Michael Amabile and Charles Asmuth of Sarnoff Corporation.

References

1. J. Durbin et al., "Battlefield Visualization on the Responsive Workbench," *Proc. IEEE Visualization 98*, ACM Press, New York, Oct. 1998, pp. 463-466.
2. D. Hix et al., "User-Centered Design and Evaluation of a Real-Time Battlefield Visualization Virtual Environment," *Proc. IEEE Virtual Reality 99*, IEEE Computer Society Press, Los Alamitos, Calif., 1999, pp. 96-103.
3. L. Rosenblum et al., "Situational Awareness Using the Responsive Workbench," *IEEE Computer Graphics and Applications*, Vol. 17, No. 4, July/August 1997, pp. 12-13.
4. Object Management Group, *The Common Object Request Broker: Architecture and Specification*, Tech. Report 1.2, Needham, Mass., 1993.
5. J. Rohlf and J. Helman, "Iris Performer: A High-Performance Multiprocessing Toolkit for Real-Time 3D Graphics," *Proc. Siggraph 94*, Computer Graphics Proc., Ann. Conf. Series, ACM Press, New York, July 1994, pp. 381-394.
6. R. Weatherly, D. Seidel, and J. Weissman, "Aggregate Level Simulation Protocol," *1991 Summer Computer Simulation Conf.*, Mitre Corp., McLean, Va., July 1991.
7. P. Lindstrom et al., "Real-Time, Continuous Level of Detail Rendering of Height Fields," *Siggraph 96 Conf. Proceedings,* Holly Rushmeier, ed., Ann. Conf. Series,. ACM Press, New York, Aug. 1996, pp. 109-118.
8. R. Pajarola, "Large-Scale Terrain Visualization Using the Restricted Quadtree Triangulation," *Proc. IEEE Visualization 98*, Oct. 1998, ACM Press, New York, pp. 19-26.
9. B. Horn, "Hill Shading and the Reflectance Map," *Proc. IEEE*, Vol. 69, No. 1, Jan. 1981, pp. 14-47.
10. P. Heckbert and M. Garland, *Fast Polygonal Approximation of Terrains and Height Fields*, Tech. Report CMU-CS-95-181, School of Computer Science, Carnegie Mellon University, Pittsburgh, Penn., 1995.
11. L. Willis, "Who Says You Can't Teach an Old LOD New Tricks," *Proc. Image 98*, The Image Society, Chandler, Ariz., 1998.
12. K. Perlin, "An Image Synthesizer," *Proc. Siggraph 85*, Computer Graphics Proc., Ann. Conf. Series, ACM Press, New York, July 1985, pp. 287-296.
13. C. Tanner, C. Migdal, and M. Jones, "The Clipmap: A Virtual Mipmap," *Proc. Siggraph 98*, Computer Graphics Proc., Ann. Conf. Series, ACM Press, New York, July 1998, pp. 151-158.
14. T. Hüttner, "High Resolution Textures," *Late Breaking Hot Topics, IEEE Visualization 98 CD-ROM Proc.*, IEEE Computer Society, Los Alamitos, Calif., Oct. 1998.



Eliot Feibush *is a member of the technical staff at Sarnoff Corporation in Princeton, N.J.. His research interests include modeling, rendering, human-computer interfaces, and display systems. He received both his B. Architecture (1979) and his MS in computer graphics (1981) from Cornell University.*

Nikhil Gagvani *is a member of the technical staff at Sarnoff Corporation in Princeton, where he is part of the Microsystems Imaging Technologies Group. His research interests include computer graphics, visualization, networking, and most recently, volume graphics and animation. He holds a B.Tech degree from the Indian Institute of Technology, Kharagpur, and an MS degree from Rutgers, the State University of New Jersey. He is also a PhD candidate in computer engineering at Rutgers.*

Daniel Williams *has been an independent consultant specializing in computer graphics software development for 14 years. He holds a BS in mechanical engineering and an MS in computer science from the University of Pennsylvania. His interests are visual simulation, scientific visualization, and distributed systems. He is a member of the ACM, Siggraph, and IEEE Computer Society.*

Readers may contact Feibush and Gagvani at Sarnoff Corp., 201 Washington Road, Princeton, NJ 08540-6449, e-mail {efeibush, ngagvani}@sarnoff.com. Contact Williams at Systems & Scientific Software, 263 Forrest Ave., Elkins Park, PA 19027, e-mail sass@acm.org.