Tactical Application of Gaming Technologies for Improved Battlespace Management

David Silvia
Naval Undersea Warfare Center DivNpt
Newport, RI 02841
401-832-2869
silviada@npt.nuwc.navy.mil

Ken Doris
Applied Visions, Inc.
Northport, NY 11768
631-754-4920
kend@avi.com

Keywords: Tomahawk, mission planning, time sensitive targets, game engine, visualization

Abstract

This paper discusses a research project that employs Gaming Technologies to improve the ability of military planners to more effectively locate and engage Time-Sensitive Targets (TST's). The battlespace is modeled and simulated through the use of Artificial Intelligence, Physics Modeling and Visualization capabilities employed in modern commercial computer games. This not only supports the ability to understand the spatial relationships of weapons, sensors, targets, and threats within the context of a mission but, also provides the ability to predict changes in these relationships through the mission's timeline. The rationale behind the selection of the specific technologies, as well as progress being made to develop a prototype workstation for future incorporation into the Tactical Tomahawk, is also detailed.

1. INTRODUCTION

The efforts described within this paper are the results of an on-going Small Business Innovation Research (SBIR) grant between the Navy and Applied Visions, Inc. The SBIR Topic, now entering its 2nd year of Phase II funding, is called Display and Visualization of Movement Predictions for Ground Vehicles and is managed by the Naval Undersea Warfare Center, Newport, RI. Its primary focus is the Tactical Tomahawk missile and the associated control system, though the research and its application are extensible to other Command and Control (C2) systems. The research and development will be embodied in the delivered application called Tactical Target Analysis and Prediction System (TTAPS).

2. PROBLEM

The Tomahawk Land Attack Missile (TLAM) is a long range, subsonic cruise missile, launched from U. S. Navy surface ships and U.S. Navy and Royal Navy submarines. Tomahawk missiles, used for land attack warfare, are designed to fly at extremely low altitudes at high subsonic speeds and are piloted over an evasive route by several mission-tailored guidance systems. Their first successful operational use was in Operation Desert Storm.

The missiles currently in deployment are the Block III Tomahawk missiles. They feature an Inertial Navigation System (INS) aided by Terrain Contour Matching (TERCOM) for missile navigation. In addition, the Digital Scene Matching Area Correlation (DSMAC) and the Global Positioning Satellite (GPS) System are coupled to the guidance systems to provide precision navigation. The Tomahawk missile has become the weapon of choice for the U.S. Department of Defense because of its long range, lethality, and extreme accuracy.

The Block III Tomahawk missiles are used against high-priority, long-dwell targets whose priority does not change during the missile's transit time[1].

The latest generation Tomahawk missile is the Block IV, or Tactical Tomahawk. It features the ability to reprogram the missile in-flight and strike alternate targets at any Global

[1] United States Navy Fact File, "Tomahawk Cruise Missile", 11 Aug. 2003. Office of U.S. Navy Information. 4 Feb. 2004

Positioning System (GPS) coordinates. It also has the ability to loiter over a target area and provide target battle damage assessment using its on-board camera.

Nevertheless, despite these new capabilities, the Tactical Tomahawk still has limitations. If opposing forces observe Tomahawk strikes in their operating area, they need only move `to evade incoming missiles. This type of target is referred as a short-dwell or Time-Sensitive Target (TST). TSTs present a problem for the Tomahawk missile because it cannot be retargeted quickly. Furthermore, the Tomahawk missile has limited endurance, increasing the likelihood that it will run out of fuel before relocating the target. In order to increase the effectiveness of the Tactical Tomahawk, the relocating and retargeting times must be reduced[2].

TSTs are defined as "those targets requiring immediate response because they pose, or will soon pose, a clear and present danger to friendly forces, or are highly lucrative, fleeing targets of opportunity"[3]. TSTs may include both stationary and mobile objects. Examples of TSTs stationary objects might be a bridge that an adversary's forces might be moving towards in order to gain a positional advantage, while mobile objects would include Transporter Erector Launchers (TEL) or Surface-to-Air Missile (SAM) systems. It is the latter mobile objects that are the focus of our work.

Mobile targets are attacked through Precision Engagement. Precision Engagement is the ability of to locate survey, discern, and track targets; generate desired effects; assess results; and reengage with decisive speed. The pivotal characteristic of Precision Engagement is the linking of sensors, delivery systems, and effects.

Our approach is to provide information superiority to decision makers; enabling them to understand the situation and select a course of action[4]; in essence to provide the operator with Situational Awareness. Simply put, Situational Awareness is knowing what is going on around you. Inherent in this definition is a notion of what is important and it is most frequently defined in operational terms. Therefore, Situational Awareness is defined in terms of the goals and decision tasks for a specific job as "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future"[5]. From an operational perspective, this definition can be recast, known as the 3-Questions model[6]:

Who is where?

What are they doing?

What will they do?

These questions map to Spatial, State, and Temporal factors.

Specifically, the destruction of TSTs is a problem of time, space, and force that can be addressed through improved Situational Awareness. The goal is to engage these targets as quickly as possible by conflating sensor, weapon, target, and operating environment information in order to provide a more complete understanding of the battlespace and the relationships within it. That is, the operator must be able to understand the attributes and spatial relationships of the weapon, sensor, and target, as well as the environment in which they are contained, in order to effectively engage a target.

The Tactical Target Analysis and Prediction System (TTAPS), now under development, uses gaming technologies to provide the ability to support strikes on mobile targets by Tomahawk Missiles through improved mission planning, optimized sensor (e.g., UAV) search routes, target movement prediction, and tracking of time-sensitive/mobile targets in a rapidly changing tactical environment. TTAPS will provide the ability to visualize the spatial relationships between the sensors used to detect the target, the weapon employed against the target, threats to the weapon, the movement of the target, and the battlespace itself, which includes the terrain and weather conditions.

3. APPROACH

The goal of this project is to develop a cost-effective, yet high-performance capability that provides the warfighter the correct level of SA necessary to engage mobile and TSTs. While developing an entirely new system for this purpose is certainly possible, it would also represent a high-risk, high-cost solution. A more efficient means to develop the required capability is to leverage mature technologies from other disciplines. Adapting an existing, mature technology can reduce development times while increasing the overall reliability if the end system. Approaches were evaluated and selected based upon the following criteria[7]:

Cost – Including licensing fees and development costs.

---

[2] Morrow, Capt. Steve. "What Comes After Tomahawk?" *Naval Institute Proceedings,* July, 2003.

[3] U.S. Joint Chiefs of Staff, DoD Dictionary of Military and Associated Terms, Joint Pub 1-02 (Washington, DC; 12 April 2001).

[4] U.S. Joint Chiefs of Staff, Joint Vision 2020, (Washington, DC: 2000)

[5] Endsley, M.R., (1988) Design and evaluation for situational awareness enhancement. *Proceedings of the 32nd Annual Meeting of the Human Factors Society*, 97-101 Santa Monica, CA: HFES

[6] Hone, G., Whitworth, W., Martin, L., Awareness is not a Stand-Alone Concept. *Proceedings of the Army Science Conference*, Orlando, FL, 2006

[7] Chrissis, M., Konrad, M., and Shrum, S., *CMMI: Guidelines for Process Integration and Product Improvement*, Pgs. 533-541, Addison-Wesley, New York, 2003.

Performance – Including speed and resource requirements.

Complexity – Including its affects on design, development, and maintainability.

Features – Determining how the technology meets the requirements of the capability.

Robustness – Determining if the approach can be extended beyond our current requirements set

Limitations – Including performance and lifecycle support, as well as future evolution

Availability – Vendors and/or sources for both the technology and documentation.

The *computer game* industry provided the candidate solution we were looking for in the form of a "Game Engine". A Game Engine is the core software component of a computer game that uses real-time graphics. The Game Engine itself is a middleware that provides a level of abstraction between the hardware and the application. It provides the underlying technologies for commercially produced computer games, simplifies development, and includes a rendering engine for graphics, a physics engine for vehicle dynamics and collision detection, an artificial intelligence subsystem to control the non-player characters, a sound engine for aural effects, and a networking subsystem. Game Engine technology is driven by a huge market of consumers and the technology continues to improve each year. Commercially available Game Engines are well-documented, open, modular products that combine high-performance visual rendering, sophisticated real-world physics and vehicle dynamics, as well as the reliability that our product requires. In addtion, due to its modular design, it is possible to easily upgrade the Game Engine to take advantage of new features as they are introduced. Although, Game Engines are not typically used in tactical applications, their capabilities and maturity are a natural fit for the requirements of this problem domain.

From the aggregation of the SBIR requirements and the common features of game engines it was determined that the problem of predicting vehicle movement can be broken down logically into three essential parts: *prior history, current state,* and *future goals.* These correspond to the following technology areas:

Artificial Intelligence – used to simulate the logic and doctrine used by the vehicle crew in deciding when and where to move. This simulation includes "memory" of prior events, "sensing" current status and events, and "decision making", all resulting in evaluation of holding position or moving to a new destination, including selection of the destination itself.

Pathfinding – given a decision to move to a new destination, determines what the routes the ground vehicle would take and what are the relative benefits and risks of each path.

Vehicle Physics – models the dynamic elements within the battlespace (i.e. targets, sensors, and weapons, both friendly and hostile) for purposes of movement prediction.

Visualization – although not needed to simulate the movement of the ground vehicle. This would provide the operator with a user interface (UI) that allows intuitive interaction and rapidly increases situational awareness.

The following sections describe our work in each of these areas.

4. ARTIFICIAL INTELLIGENCE (AI)

Our research looks into adapting the AI used by computer-controlled forces (often referred to as Non-Player Characters, or NPC's) that play a large part in most modern games. The technology has rapidly evolved to the point that, in some of the latest games, the NPCs exhibit behaviors that in many ways appear to be sentient. We are seeking to adapt this cutting-edge technology to predict the future actions of hostile ground vehicles.

After reviewing many potential methodologies, we determined that Goal Oriented Action Planning (GOAP)[8] would be the most suitable approach to extending the basic game engine artificial intelligence. GOAP was developed to handle real-time game action and is ideally suited to dynamic environments such as military operations. It evolved from earlier cognitive systems, such as GOMS (Goals, Operators, Methods and Selections)[9], with major influence from work done at Stanford University on the Stanford Research Institute Problem Solver (STRIPS)[10]. STRIPS consists of goals and actions, where goals describe some desired state of the world, and actions are defined in terms of preconditions and effects. An action may only execute if all of its preconditions are met, and each action changes the state of the world in some way.

The overall logical execution flow of a GOAP system is relatively simple. At a given point in time, each agent has a set of goals that need to be achieved, and tries to satisfy the goal or goals that are most relevant for the current situation. For a given goal, the logic regressively searches for actions that have an effect that matches the goal. For each matching

[8] Orkin, J. (2004), "Applying Goal-Oriented Action Planning to Games", *AI Game Programming Wisdom*, 2nd Edition, Hingham, MA, Charles River Media, Inc.

[9] Card, S., Moran, T., and Newell, A. (1983). *The psychology of human-computer interaction*, Hillsdale, NJ: L. Erlbaum.

[10] Nilsson, J. (1998), "STRIPS Planning Systems", *Artificial Intelligence: A New Synthesis*, Pg 373-400, Morgan Kaufmann Publishers, Inc.

action, the logic looks at its preconditions, determines if they are already satisfied, and if not, performs another regressive search to find actions that have effects that that match the precondition of the previously selected action. This type of regressive searching continues until it finds a path from the end goal all the way back to the current world state. The agent then begins executing the actions, rechecking the validity of the path and end goal at each step to accommodate for dynamic changes in the environment. Several interesting and useful concepts are embodied in this architecture, which makes it a prime candidate for our research:

- For a given goal, there may be multiple paths. The system can be designed to either simply pick the first path that works, or a weighting system can be applied to individual actions, thus providing the mechanism for searching for the lowest cost path. This can be employed to find the lowest risk plans for achieving military goals.
- There is no explicit mapping between goals and actions, thus allowing for dynamic resolution of unexpected conditions such as weather changes.
- The GOAP architecture lends itself to a separation of implementation and data. This separation of the coding from the data allows non-programmers the ability to create or modify behaviors, an extremely important attribute for the future deployment of this application.
- Regressively searching for plans in real-time affords opportunities to *learn* and find multiple solutions to problems
- Atomic goals and actions of a GOAP system are easy to read and maintain, and can be sequenced and layered to create complex behaviors.
- A GOAP system imposes a modular architecture that facilitates sharing behaviors among agents and even across software projects.
- The GOAP architecture can be defined using the Planning Domain Definition Language (PDDL), a machine-parsable standardized syntax used widely throughout the AI planning community.
- Translators exist between PDDL and DARPA Agent Markup Language (DAML)[11].

These characteristics of GOAP make it an excellent candidate for our application. We are now in the process of incorporating it into the Delta3D game engine, working with the MOVES Institute of the Naval Postgraduate School (NPS) under a Cooperative Research and Development Agreement (CRADA).

[11] See www.daml.org

5. PATHFINDING

As discussed above, we use AI to predict the next most likely actions of the hostile ground vehicles. In many cases, this will result in the decision to move to a new location. How the vehicle might traverse the distance from its current position to that location is the next problem to be solved.

A hostile ground vehicle at a given location will traverse to a new location in accordance with a number of dynamic factors. It will avoid exposure to threats; follow the most efficient path, etc. In our Phase I effort of the SBIR, we experimented with how the pathfinding logic in game engines could be adapted to the TTWCS application. Starting with our initial list of approximately a dozen candidate game engines, we narrowed the choice to UnReal and Torque since we had access to source code for both and were reasonably familiar with their code base. The versions of each engine we had in our possession employ relatively rudimentary AI pathfinding based on pre-scripted waypoints. We believe that entering waypoints into each terrain map is both error-prone and overly time consuming, so we decided to modify the Torque engine to include a dynamic path-finding capability based on the "A*" (pronounced "A-star") algorithm. We chose A* for its efficiency: it is widely used in modern games, and is the subject of continual analysis and improvement by the gaming community[12].

Given a starting point and a destination, A* dynamically builds paths by evaluating the "cost" of each possible route section, with an overall goal of generating the "least cost" path. For Phase I our only cost function was terrain slope. Using the slope of each terrain tile, A* will build a path that follows the flattest (easiest) path between the two points.

One of the strengths of this approach is that the cost function can be comprised of a number of factors. For example, adding a cost factor for exposure to danger would be useful in the TTWCS application. We expect, for example, that the hostile ground vehicles will often choose their routes based upon well they can hide from satellite and airborne sensors. We are currently incorporating these factors in our pathfinding improvements.

Figure 1 is a screenshot taken during this activity. In this example, the path illustrated is that from a TEL, located on a grassy area off the road, to a predicted destination which is also located some distance off-road. The white highlighted route indicates the most likely path, with the vehicle first traveling to the nearest road and then following the road network to a point close to the destination, with a final leg across open terrain.

[12] "Optimizing Pathfinding", Sean Barrett, *Game Developer Magazine*, a series of 5 articles running from Jan through May, 2005.

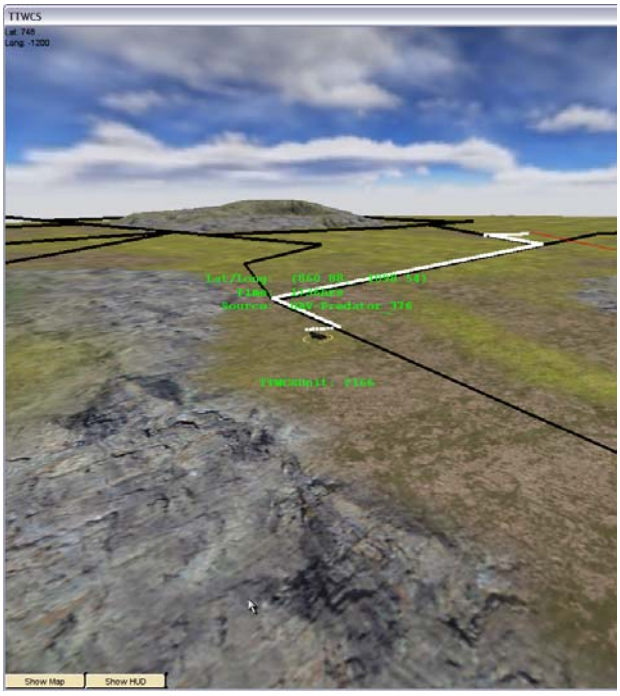

Figure 1 – Terrain & Road Network Path-Following in Torque Engine

6. VEHICLE PHYSICS

In recent years, one of the most successful genres of computer games has been the "driving" or "racing" games. Their success has largely been due to the emergence of a new class of middleware called "physics engines". Used in games such as Gran Turismo 4 (GT4), these products have reached the point where they can accurately simulate the real-world performance of vehicles down to minute detail, such as tire pressures vs. road surfaces, etc. This type of mobility simulation is a key feature in games that appeal to the young gaming audience, whether it be racing Porsches through city streets, driving SUV's off-road, or taking an M1A2 into the desert to hunt Iraqi T-72's. While entertainment value is what ultimately sells these games, the technical sophistication of the audience also requires an underlying feeling of realism that is based in real-world physics and lifelike action.

We started our evaluation with the Physics Engine that is currently incorporated in the Delta3D game engine – the Open Dynamics Engine (ODE), an open source project that has been used in several commercial driving games. We were able to quickly create a realistic vehicle driving simulation with an AI controller that would follow a path along a road network. While easy to work with, ODE's performance was disappointing, providing less than 2X real-time (e.g. a ten minute prediction into the future would take six minutes to calculate). We explored ways to increase ODE's execution speed, but soon realized that would become a full research project unto itself, and since the goal was to adapt and exploit existing game technology, it was eliminated from further consideration.

The next candidate was the Havok physics engine. Originally developed by a team of computer scientists at Trinity College in Dublin, Havok appeared to provide the most comprehensive vehicle modeling tools and included built-in object classes representing vehicle components. Working with an evaluation copy of Havok, we were able to fairly quickly determine it was capable of meeting our requirements. Unfortunately, our attempts to negotiate licensing fees for the full Havok development package to fit the budget of our project were unsuccessful. The result was that the Havok solution was discarded.

With Havok eliminated, our next candidate was the Novodex engine. This decision was partly influenced by the fact that Epic Games had recently chosen Novodex for its next version of the UnReal Engine, and bolstered by the fact that Novodex had chosen a different business model than Havok. Rather than license the software, Novodex, with their parent company, Ageia, provides the software for free and relies on sales of add-in cards containing their PhysX processor, which was developed to accelerate the highly specialized physics calculations. Although Novodex didn't provide a separate vehicle package, we found it relatively easy to assemble basic components into workable vehicle simulations; enough to show the feasibility of our initial concept. The results were much better than ODE, with Novodex able to run at approximately 10X real-time without hardware acceleration. Since a PhysX board sells for under $300 and installs in a common PCX motherboard slot, this is our current choice for our operational prototype. We are now working to integrate it into the Delta3D framework, replacing the current ODE subsystem.

7. VISUALIZATION/USER INTERFACE

The User Interfaces (UI) employed in the gaming industry incorporate many features that are applicable to tactical visualization application requirements. Modern computer games incorporate sound concepts of UI design, as they must operate in far more demanding user-interface situations than practically any other application developed today[13]. Additionally, today's computer users, and more specifically in our case shipboard operators, are comfortable and familiar with the UI employed by computer games. Today, the average age of operator on a U.S. Navy Submarine is 25 years old.[14] This age group constitutes 66% of the computer gamers today and they spend an

[13] Laurel, Brenda, T*he Art of Human-Computer Interface Design*, Addison-Wesley, New York, 1990.
[14] *Northwest Navigator*, USS Louisiana Blue, 17 Feb. 2006.

average of 7.6 hours per week playing computer games[15]. Our approach is to take advantage of the operator's familiarity with computer game interfaces in our design.

Our approach to this problem requires that the UI be simple, self-explanatory, adaptive, and supportive. The goal is for the user to interact with the task he needs to accomplish and less with the computer, making the technology subservient to the goals. In short, our goal is to immerse the operator in the TTAPS environment. Allowing the operator to visualize and explore the relationships between the targets that he desires to engage, the sensor he employs, the weapon, the threats to the weapon, and the environment itself. This understanding of the spatial relationships, both current and predicted, supports the operator's ability to effectively employ/route weapons, effectively employ sensors in the region, avoid threats to sensors/weapons, relocate fleeing targets, understand the goals of a target, and make strike decisions. The complex relationships between all the objects in the battlespace can be understood more readily through 3D visualization. In addition to the fact that studies have concluded that 3D visualization provides increased awareness over 2D presentations, the ability to represent the third dimension of height is particularly important to our approach[16]. For example, threats to weapons and sensors are limited by altitude and the threat area may be in-fact a hemisphere. In addition, the ability to change and manipulate the viewpoint allows the operator to fully examine the spatial relationships of all entities within the viewable battlespace.

The Delta3D open source game engine is well-suited to our visualization requirements as it incorporates advanced methods for rendering terrain. We chose it not only for its cost benefits and modularity, but also because it is one of the few engines that can support extremely large terrain databases, a necessary ingredient for viewing the potential target areas. Most game engines are limited to terrain expanses of only a few square miles, while Delta3D has no such limits. In addition, it can use mapping products from the National Geospatial Agency (NGA) with straightforward conversion. Another benefit of using Delta3D is that it is available for Linux, which may be a future requirement for TTAPS to enter the Fleet.

8. PROTOTYPE DEVELOPMENT

Now entering its second year, this Phase II SBIR has reached the stage where the individual elements are being integrated into a TTAPS prototype, to be hosted on a commercial laptop computer. As part of its development, we are leveraging an architecture called GameBridge, developed under a prior SBIR with the U.S. Army. Its purpose is to act as a translator between real-world data and the internal data structures of game engines. Built as a multi-layered framework, as shown in Figure 2, it contains a common core set of functionality tailored for a given application by customizing the C2 layer. The use of GameBridge will greatly simplify the integration of TTAPS into the overall TTWCS architecture.

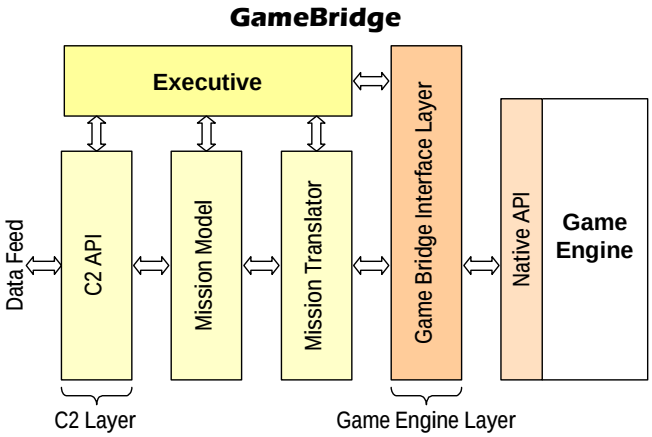

Figure 2 – GameBridge Architecture

9. SUMMARY

The SBIR project described in this paper is exploring the adaptation of gaming technology for predicting future movement of ground vehicles. Modern game engines incorporate sophisticated algorithms for artificial intelligence, path finding, physics and 3D rendering, all of which are directly applicable to this problem domain. While game engines are typically employed in training applications, tactical application is a natural extension of the technology. Currently in prototype development, the system is aimed at eventual incorporation into the Tomahawk weapon program and potentially other Time Sensitive Target applications. The application of Gaming Technology provides a cost-effective means to develop high-performance tactical applications that are easily deployed.

Biography

David Silvia is an Engineer with the Naval Undersea Warfare Center (NUWC) in Newport, Rhode Island. He currently serves as a liaison for the Tomahawk Weapon System (TWS) Advanced Concept Working Group (ACWG) participating in technical exchanges between Naval Surface Warfare Center, Johns Hopkins Applied Physics Laboratory, Lockheed Martin Corporation, and NUWC. In addition, Mr. Silvia conducts technical evaluations, surveys new candidate technologies, and

[15] "Entertainment Software Association, Facts and Figures", <http://www.theesa.com/facts/gamer_data.php >

[16] Carvajal, Alejandro, "Quantitative comparison between the use of 3D vs. 2D visualization tools", *Proceedings of the Ninth International Conference on Information Visualization*, 2005.

develops prototypes to address future requirements for the TWS. His past work for the Navy has included the development of applications in speech recognition, distributed database design, and peer-to-peer architectures. Mr. Silvia has over 20 years of experience in software development and engineering. He has a Bachelors Degree in Engineering from Roger Williams University in Bristol, Rhode Island. He has worked for the Navy for six years

Ken Doris is the Vice President of Engineering at Applied Visions, Inc. in Northport, NY. He serves as the Principal Investigator on the Navy SBIR project described in this paper as well an on-going Army SBIR that also uses gaming technology for battlefield analysis and visualization. One of the authors of IEEE 1278 (the original DIS specification), Ken has published numerous technical papers on subjects such as 3D visualization, real-time network traffic analysis and multicast addressing. He received his Bachelor of Electrical Engineering degree from Rensselaer Polytechnic Institute.