Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

CONSTRUCTIVE AND VIRTUAL MODEL LINKAGE

David R. Pratt
Matthew A. Johnson

Department of Computer Science
Naval Postgraduate School
Monterey, California 93943, U.S.A.

ABSTRACT

The U.S. Army has two disparate combat models, Janus and Distributed Interactive Simulation (DIS) based systems. Both facilitate training, tactical development and weapons analysis. However, a major problem is that entities existing in the Janus Combat Model cannot interact with DIS entities. This paper address how to make this interaction possible, producing a synergy between the combined models, each model benefitting the other.

The first step was to identify the differences between the Janus and DIS environments. Next, a software architecture was developed to store and manipulate data regarding both simulations. A communications architecture was created to allow data flow between the two environments. Finally, algorithms were developed to allow for interaction between Janus and DIS entities.

The resulting product, the World Modeler (WM), integrates Janus, a two dimensional, constructive combat model, into the three dimensional, entity-level virtual battlefield of DIS. Janus entities interact in real-time with other entities in the DIS virtual world. It is a software system operating on a low-end Silicon Graphics (SGI) workstation with TCP/IP and UDP/IP networking capabilities.

1 INTRODUCTION

Over the past several years there has been a rapid growth in the development and use of virtual models. Many of these models have been made possible by the tremendous increase in the computer power and the equally impressive decrease in machine cost. While few will argue the improvement and fidelity of the visual models, the same cannot be said for the physical models that drive the simulations. The use of "visual physics," the practice of making something look realistic, whether or not physically correct, is common. This, when combined with the real-time, asynchronous, and variable nature of these models, have made virtual models an analyst's nightmare. Accordingly, the analytic community has not been overly receptive to the use of such models. Distributed Interactive Simulation (DIS) is a typical paradigm used by these virtual models as described in (Institute for Simulation and Training 1993).

In contrast, the traditional models have relied heavily on deterministic, discrete, closed, event-driven models. The focus of these models has been on the calculations, reproducibility, and accuracy of the model. Many of the models have gone undergone a rigorous Validation and Verification (V&V) process to ensure the accuracy and truthfulness of their algorithms. Janus is a typical analytical constructive model. (Department of the Army 1993a and 1993b).

As a result of the different requirements of the models, the two types of models operate with radically different paradigms. The Janus LINKed to DIS (JLINK) project is an attempt to bridge these two distinct paradigms in order to develop a single cohesive view of the simulated environment (Marti 1994). In this paper we will cover the architecture and functionality of the WM, the interface between the two systems.

2 JANUS

The Janus Combat Model, as described in (Department of Army 1993b), is a constructive, monolithic combat model used primarily by the U.S. Army. Its main purpose is twofold. First, it is used as a combat development tool, analyzing both weapon system and tactics. Second, Janus is used as a training tool by leaders for the purpose of tactics training and staff planning. Janus' ability to train unit staffs in Command and Control (C2) makes it one of the most widely used of the Army's constructive models.

Janus portrays both individual entity level systems as well as aggregate level units such as platoons or batteries. Janus users develop combat scenarios consisting of force-on-force engagements between two opposing forces. Due to the stochastic modeling of entity engagements, execu-

tions (referred to as runs) of the same scenario will produce slightly different results. The study of several runs of the same scenario lend to the analysis of the tactics, force structure and weapons systems used. This analysis is used both for training and combat tactics development.

Janus employs high resolution algorithms which accurately model numerous weapons platforms; ranging from a dismounted riflemen to an MLRS battery. This high fidelity modeling allows Janus to be a valid environment for combat development. Military systems employed in ground combat are the primary modeling focus of Janus.

Janus is monolithic in nature, operating on a single computer system. Its numerous (up to sixteen) two-dimensional displays provide the user with a highly detailed, electronic map display of the battlefield. Traditionally, running a VAX/VMS and Textronix terminal system, the primary user interaction is through a four-button puck and tablet and the two-dimensional display. With the UNIX version of Janus, the workstation mouse serves as the primary input device.

This research is based on UNIX Janus Version 4.0 running on HP 715 and 735 workstations. This version utilizes polygonal terrain and individual building representations which differ significantly from previous versions of Janus.

3 BATTLEFIELD DISTRIBUTED SIMULATION - DEVELOPMENT

Battlefield Distributed Simulation - Development (BDS-D) is an Army program designed to equip the Louisiana Maneuver Battle Labs with the latest DIS systems. BDS-D will test future weapon systems and technologies, proposed force structure, and tactics. This concept is expected to save millions of dollars in development costs, and to reduce system development time by a factor of five. BDS-D provides a virtual battlefield where man-in-the-loop simulators and other intelligent platforms can interact and thoroughly test future technologies. If a particular technology application appears valid, continued investment occurs. If not, large amounts of money are not spent on continued development (McDonough 1993).

BDS-D represents the most state-of-the-art DIS synthetic environment. Through DIS, geographically-dispersed simulations are interconnected and interact in a distributed virtual battlefield. Advance Technology Demonstrations (ATDs) continually upgrade this environment. Linking Janus to DIS via the WM is one such ATD (U.S. Army 1994)

4 WORLD MODELER

Construction of the WM began in the Fall of 1993 in the Department of Computer Science, Naval Postgraduate School as part of the Army's Anti-Armor Advanced Technology Demonstration (A2ATD) research effort (U.S. Army 1994). As part of this project, the primary purpose of the WM is to allow interaction between Janus and DIS entities in support of A2ATD's Exercise Four.

The WM runs on its own SGI platform with a TCP/IP connection to Janus and a UDP/IP connection to DIS. Janus and the WM communicate with a local protocol. The WM communicates with the rest of the systems using DIS protocols. The logical architecture is shown in Figure 1. The actual system had the WM messages and the DIS Protocol Data Units (PDUs) going over the same physical wire as shown in Figure 2.

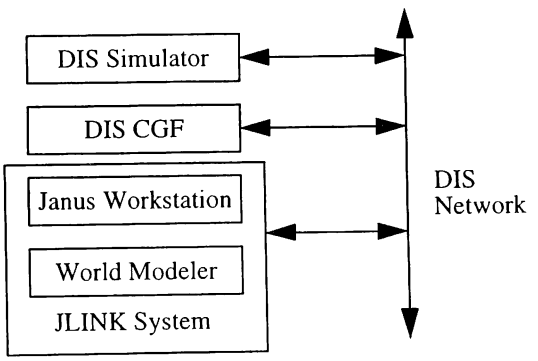

Figure 1: Conceptual System Structure

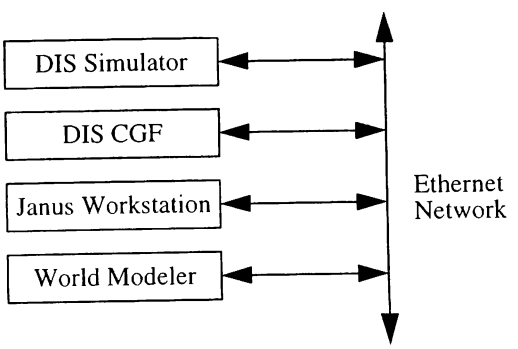

Figure 2: Physical System Structure

To reduce time, development of the WM took place on a low-end SGI workstation. This choice of hardware allows the use of existing NPS developed NPSNET software as necessary. Storage of information is patterned directly after NPSNET allowing significant reuse of NPSNET algorithms and code. Database formats are also consistent with NPSNET. (Pratt, 1993)

5 WORLD MODELER FUNCTIONALITY

To effectively act as a translator between the two paradigms, the WM must perform four major functions: manage the network, dead reckon Janus entities, reconcile Janus entity locations with the terrain, and arbitrate engagements between Janus and DIS entities. By performing these functions, the WM is the nexus by which Janus interacts in the DIS environment.

5.1 Network Management

Arguably, the most important function of the WM is network management. Due to the differences in the models, the type and frequency of required updates varies. Since Janus is an event driven model, a priority queue is maintained containing all scheduled events, typically, when an entity completes its assigned movement segment, when time delay for river fording is completed, and when artillery fire are scheduled. Entities are processed only when one of their events reach the head of the queue. Thus, a stationary entity will never be managed. This strongly contrasts with the DIS paradigm which is based upon the entity state vice discrete events. Whenever a DIS entity changes state (e.g. orientation, position error, or appearance) it sends out a PDU. Table 1 shows some of the differences in the entity terrain and movement state parameters. Furthermore, due to the unreliable delivery of UDP, a packet will also be sent out on a periodic basis, notionally every five seconds, to let the other systems on the network know it is still alive.

Table 1: Entity Movement and Terrain Parameters

Functionality	Janus	DIS
Movement	Speed	X, Y, Z Velocity X, Y, Z Acceleration
Position	X, Y	X, Y, Z
Orientation	Heading	Heading, Pitch, and Roll

To reduce network load, the WM only receives updates from Janus when an event for an entity has transpired. This mimics the event driven model of the system. To comply with DIS requirements, the WM dead reckons the Janus entities over the terrain and ensures that PDUs are sent out as required.

The reverse occurs when two update Janus. The WM receives all of the DIS PDUs. It then filters the packets to determine if events have occurred. If so, Janus is updated at the next heartbeat. This has the effect of discarding the DIS heartbeat messages and reducing the number of times Janus is updated.

5.2 Terrain Reconciliation

The two models also differ in their representation of the terrain. The Janus terrain is geared for efficiency of representation and computation and uses a two-dimensional regular grid of elevation posts and bit encoding. Each one of the grid elements are assumed to be homogenous over the entire grid. Since Janus only uses a two-dimensional iconic representation of the entity, it does not record entity orientation and elevation. Whereas, in the DIS environment the majority of the terrain consists of polygons which can be of arbitrary size and shape. This is directly related to the three-dimensional out-the-window view of most DIS systems. As a result, the WM must take the X, Y location of the entity and compute elevation based upon a polygonal terrain representation. To give the entity a realistic appearance, the pitch and roll of the entity must also be computed from the underlying polygon.

To further complicate matters, Janus uses a modified version of the UTM Coordinate scheme while DIS uses a geocentric Cartesian coordinate based upon WGS 84 datums. Hence, the WM must also perform proper coordinate conversion between the two models.

5.3 Dead Reckoning

Since Janus only processes the entities when an event occurs, the WM has to interpolate the location of the Janus entities between updates. In addition, DIS events which are not also Janus events must be handled. For example, when an entity crosses a polygon seam, it is not an event in Janus, but if the entity changes orientation, it is a DIS state change necessitating a PDU transmission. The WM is responsible for sending out all appropriate PDUs.

As stated above, the information required by the models varies, in some cases rather dramatically. Perhaps one of the most glaring examples is the way paths are maintained. In the DIS world, there is no concept of future path segments. Each entity is assumed to be moving according to some well-defined algorithm and state data. When the entity deviates from its projected location and orientation, a new PDU is sent out with the new state information. For most systems, this results in naturalistic motion. As an entity goes around a corner, a large number of PDUs will be sent out to represent the entity's curved path.

In Janus, the path that an entity will follow is known as the start of the exercise, it is part the scenario's set-up files. Currently the path is limited to fifty segments. When it gets to the end of a segment, the entity snaps to its new heading. This, however, results in a visual anomaly that

clearly delineates Janus entities. To seamlessly simulate the Janus entities into the DIS model, the WM performs basic turn smoothing. As a result, for a brief amount of time the WM controls the entities rather than Janus. By resolving the visual anomaly, Janus entities are virtually indistinguishable from other DIS entities.

5.4 Event Arbitration

The differences between Janus and DIS are also noticeable when resolving engagements. Since Janus is a monolithic model, it resolves all of the engagements internally. On the other hand, DIS handles engagements differently, as a notification and response process. This process is described briefly below.

5.4.1 Buffering of Events

At each synchronization point, Janus sends all fire and detonation events scheduled for the next time step to the WM. Since some of these events have not yet happened, the WM builds a time-based priority queue of the events. As part of the main simulation loop, the buffer is checked to see if any of the times of the events are less then the real-time clock. If so, the appropriate PDU is sent out to describe the event.

5.4.2 Impact Determination and Reporting

In Janus, all of the events are internal. This simplifies the firing and impact reporting process. In DIS, the firing entity sends a Fire PDU. The entity also determines the location and type of the impact and sends a Detonation PDU when the round impacts, as shown in Table 2. Thus, the munition's fly out is modeled by the firer. This works well in Janus since we are able to use the Probability of hit / Probability of kill (Ph/Pk) tables for direct fire to tell if we got a hit. For indirect fire, we just report the location of the hit.

Table 2: Which System does Impact Determination

Firer \ Target	Janus	DIS
Janus	Janus	Janus
DIS	DIS	DIS

5.4.3 Kill Arbitration

As in Table 3, the target is responsible for determining the effect of the detonation rather than the firer. When a Janus entity engaged another Janus entity or two DIS entities are involved, the problem is straightforward. Each respective system handles the arbitration and reports a change in status to Janus.

Table 3: Which system does Kill Arbitration

Firer \ Target	Janus	DIS
Janus	Janus	DIS
DIS	Janus	DIS

When a DIS entity engages a Janus entity, the models are forced to interact, imposing a significant arbitration challenge to Janus. The Detonation PDU contains the firing entity, the type of munition and the point of impact. From this information the WM reconstructs the engagement in order for Janus to reconstruct the event in a format that can used in Janus' Ph/Pk tables. The reverse is true when Janus engages a DIS entity. Rather than Janus doing the arbitration, the DIS entity remains alive until the entity determines its own state.

6 Structure of the World Modeler

In this section, we discuss the software structure used to track information about the virtual world, how we manage network traffic, and how we efficiently control the interaction between Janus and DIS entities. The architecture chosen for the WM allows for reading and writing to associated message buffers in parallel with the execution of the main application loop of the WM.

6.1 Initialization of the World Modeler

Before Janus can interact in the DIS virtual world, the two different environments of Janus and DIS must be reconciled within the WM. The initialization process of the WM performs this reconciliation.

The first step is to load the DIS-JANUS equivalence table into the WM. This data file contains information on the different entity types which Janus recognizes as part of its simulations. DIS entity type equivalences are matched with each Janus weapon system. The intent is to ensure that Janus entities are correctly portrayed in the DIS environment and vice versa. For example, we do not want Janus to mistake a DIS Soviet T-72 tank for a U.S. Bradley Fighting Vehicle.

Next the terrain data file is read. Currently, both Janus terrain and DIS terrain are derived from the same S1000 terrain database. Janus, however, does not explicitly track information on entity elevation values. The WM maintains a copy of the terrain to determine elevation and orientation values for Janus entity positions and munition effects.

Due to the time critical nature of the network connections, opening the network is the last step before execution starts. The WM establishes a TCP/IP connection to Janus and a UDP/IP connection to the DIS network. When these connections are established, the associated message buffers begin to fill with Janus and DIS entity information. The Janus message buffer is read and the Janus entities are loaded into the WM entity array which stores information on all Janus and DIS entities. Janus entities are flagged as such to allow for different processing in the main loop. These entities are the only Janus entities which will participate in the simulation, because Janus does not introduce new entities during the simulation.

After the Janus entities are loaded, the DIS message buffer is read. These entities are then loaded into the WM entity array. Due to the DIS paradigm, creation and deletion of entities in the DIS world is dynamic. The WM only tracks entities which are currently being updated via the DIS PDUs on the Ethernet. Once the DIS entities are loaded in the array, Janus is then updated with all the new entities. Since Janus does not allow the creation of new entity types during execution, all undefined types are mapped to default vehicles. Janus acknowledges the receipt of this data with a synchronization message. This message signifies the completion of the WM initialization. The WM clock starts and the main application loop begins.

6.2 World Modeler Main Application Loop

The main application loop continuously executes the primary functions of the WM. Figure 3 contains a graphical representation of the main software modules in the main loop. This loop continues until the simulation is terminated from either Janus or the WM. Another DIS application cannot terminate the Janus-DIS connection.

When the simulation begins, the Janus clock runs a heartbeat faster than the WM. The frequency of the heartbeat is based upon the Janus processing time. Janus takes time to do internal algorithms such as line of sight and Ph/Pk calculations. As a result, the Janus cycle time varies from one heartbeat to the next. The heartbeat can be set to different lengths prior to the simulation, but is fixed during a run. At the end of each heartbeat, Janus and the WM synchronize their world representations, as such they must have completed their respective processing cycles. The smaller the heartbeat, the less drift occurs between the two systems due do to the different algorithms, but the larger the chance of Janus falling behind. Four seconds appears to be a reasonable heartbeat for scenarios containing up to 200 Janus entities.

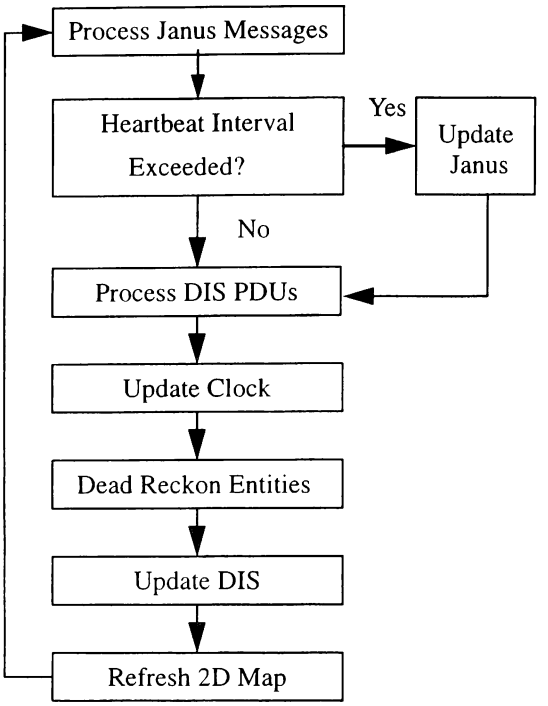

Figure 3: . Main Application Loop

6.2.1 Process Janus Messages

The WM reads Janus messages from the Janus message buffer. If the message deals with entity states, then the entity array is updated. If the messages deal with munition effects--fire and detonations--appropriate Fire and Detonation PDUs are created. Since Janus runs a heartbeat ahead of the WM, these events may not have occurred yet in WM time. Thus, time stamps on the message and the WM clock are continuously compared. If the event occurred, these PDUs are written directly to the Ethernet and broadcast to the DIS environment. If the events have not yet occurred, they are placed in an event queue which is read during each cycle of the application loop.

6.2.2 Update Janus

The WM keeps track of when it last updated Janus. If the last update occurred less than a heartbeat past, Janus is not updated. If greater than the heartbeat, the WM updates Janus with its current information on DIS entities. Also, any new DIS entities which may have entered into the simulation are also passed to Janus. Thus, the heartbeat is used to limit the number of updates Janus receives and ensure that Janus does not get too far ahead of real-time. Continual updating of Janus about DIS entities slows

Janus down. The heartbeat interval is designed to keep Janus informed but not overwhelmed with updating entity information.

6.2.3 Process DIS PDUs

The WM reads the DIS message buffer containing PDUs, parsing the PDUs by type. Entity State PDUs are used to update the entity array. If a new entity has entered the simulation, it is added to the array. Fire and Detonation PDUs are also filtered to see if they have any effect on Janus entities. If not, they are discarded. If they effect Janus entities, the information is processed and Janus is immediately updated.

6.2.4 Update Clock

During each pass through the application loop, the current time is assigned to the WM clock. This allows the WM to keep track of the time difference between the current time and the time that the entity array was last updated.

6.2.5 Dead Reckon Entities

The WM dead reckons entities based on their last known position, orientation, velocity, and elapsed time. As discussed above, entity and terrain reconciliation also occurs during this phase.

6.2.6 Update DIS

The WM is responsible for sending Entity State PDUs to DIS according to the DIS 2.0.3 standards (Institute for Simulation and Training 1993). To accomplish this task, the WM scans its entity array to determine whether any Janus entities meet the criteria for the generation of an Entity State PDU. If so, the WM creates and sends the PDU. The WM then reads the Janus event queue which contains Janus fire and detonations events in PDU format. If the current time is greater than the event time, the PDUs are broadcast onto the net. If not, they stay in the queue to be screened the next time through the main application loop.

6.2.7 Refresh 2D Map

The final portion of the loop redraws the WM two-dimensional display based on the most current information in the entity array. The cycle then begins again and continuously loops until Janus or the WM application has exited.

7 RESULTS AND CONCLUSIONS

A common problem in testing and evaluating systems like JLINK is the creation of an initial scenario which overloads the system to see if it works. This is commonly called the "Big Bang" approach. We decided early on to take a more rational approach. Testing of the JLINK system was conducted using a series of scenarios similar to those listed in Table 4. As shown in Table 5, each scenario was slightly more complex than the previous one to isolate the additional functionality. Consequently, we could debug errors before the scenarios became unmanageable. While other scenarios were used to test specific functions, the ones listed below were the "gold standards" for each phase.

Table 4: Scenarios Used to Test JLINK

Scenario	Red	Blue	Total	Interaction
A	1	1	2	None
B	1	14	15	None
C	19	17	36	Combat
D	123	75	198	Combat

Table 5: Entity State and Application Loop Time

Scenario	Activity	Ave. PDUs per Sec.	Ave. PDUs per Sec. per Entity	Application Loop Time in Sec.
A	Static	0.394	0.197	0.014
A	Static, Rotating Turrets	1.280	0.640	0.014
A	Moving, Static Turrets	6.066	3.033	0.014
A	Moving, Rotating Turrets	6.160	3.080	0.014
B	Moving, Rotating Turrets	88.800	5.920	0.020
C	Small Engagement	80.028	2.223	0.021
D	Large Engagement	161.568	0.816	0.200

Scenario A tested the basic functionality of the system. It ensured that the terrain and network portions of the code were working correctly, and also gave us benchmarks for the PDU traffic. It was found that the number of PDUs were well within acceptable limits for the various actions being performed by the entities.

Scenario B tested the creation of different types of entities, ensuring that the correct entities were created based upon the DIS PDU traffic and the types created in Janus. Aircraft, human, and ground vehicle movement routines were tested extensively in this scenario which accounts for it relatively high PDU per second per entity average.

Scenario C is of particular interest because the entities were actually engaged in a combat mission, allowing us to test the engagement arbitration code. As such, it was the first real test of the entire system

Stress testing was done via the final scenario, Scenario D. This represented almost a doubling of the number of entities expected in the A2ATD scenario. As shown in Table 5, the WM did not have trouble keeping up with the entity count. Of note, we ran a long-haul version of this test over the Internet from RAND in Santa Monica, CA to the Naval Postgraduate School (NPS) in Monterey, CA with no noticeable delays. The decrease in the average number of PDUs per entities is due the number of entities which remained stationary for prolonged periods of time.

The WM establishes a ground truth environment to process both Janus and DIS entity data and events. Through its process, the WM delivers the necessary functionality to successfully integrate Janus and DIS.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Jed Marti, Dr. Chris Burdorf, Keith Brendley, and Major Chris Pate for their help on this project. David Ward and Bill Caldwell have been extremely helpful in the continuation of the work, the Janus Fast Movers project, and have debugged some of the problems that were in the first version of the WM. It would not have been possible with out all of their contributions. Likewise, we owe a debt of gratitude to the U.S. Army Materiel Systems Analysis Activity (AMSAA) for funding this research as part of the AntiArmor Advanced Technology Demonstration (A2ATD).

REFERENCES

Department of Army. 1993. *The Janus 3.X/UNIX Model Software Design Manual*, Headquarters TRADOC Analysis Center, ATRC-ZD, Ft. Leavenworth, KS.

Department of Army. 1993, *The Janus 3.X/UNIX Model System Design Manual*, Headquarters TRADOC Analysis Center, ATRC-ZD, Ft. Leavenworth, KS

Institute for Simulation and Training. 1993. *Standard for Information Technology - Protocols for Distributive Interactive Simulation Applications*, Draft 2.0.3, Orlando, FL.

Marti, J. 1994. *JLINK Integrating JANUS and BDS-D Project Overview, Draft*, RAND Arroyo Center, Santa Monica, CA.

McDonough, J. G., 1993. *Doorways to the Virtual Battlefield*, Proceedings from The First West Coast EFDPMA Conference and Exhibition on Virtual Reality: The Commitment to Develop VR, Illusion Engineering, Inc., Westlake Village, CA.

Pratt, D. R. 1993. *A Software Architecture for the Construction and Management of Real-Time Virtual Worlds*, Ph. D. Dissertation, Naval Postgraduate School, Monterey, CA.

U.S. Army. 1994. *Anti-Armor Advanced Technology Demonstration (A2 ATD), Experiments 2, 3, 4 and 5, Independent Evaluation Plan and Test Design Plan, Draft*, Material Systems Analysis Activity, Aberdeen Proving Ground, MD.

AUTHORS' BIOGRAPHIES

DAVID R. PRATT is an Assistant Professor of Computer Science at the Naval Postgraduate School (NPS), Monterey, California. A former Marine Corps Captain, Dr. Pratt earned both his M. S. and Ph. D. from NPS. His research interests include the application of DIS technology to new and existing problem domains. He is currently a Principal Investigator on Synthetic Environments (SE) database, Constructive Model Integration, and SE modeling tasks.

MATTHEW A. JOHNSON is currently an instructor in the Computer Science Department of the U.S. Military Academy, West Point, New York. A Captain of Infantry in the U.S. Army, he earned his M. S. degree from NPS in 1994.