VIRTUAL WORLD ENVIRONMENT SIMULATION IN DISTRIBUTED SIMULATIONS

Xin Li, Mary Kruck, Henrik Lind and Dan Gilbert
Lockheed Martin Information Systems
Bellevue, WA

ABSTRACT

In early applications of distributed simulation, battles occurred during mid-summer at high noon on a clear day with just enough atmospheric attenuation to prevent the visual anomaly of seeing off the edge of the "world". The Synthetic Environments (SE) Program, part of the Synthetic Theater of War (STOW-97) Advanced Technology Concept Demonstration (ATCD), is developing an architecture and the processes to represent dynamic environments with increased fidelity. Part of the SE Program is the Dynamic Virtual Worlds (DVW) effort, tasked to enrich the virtual battlefield with a range of real-world environmental effects, such as clouds, battlefield smoke, vehicular dust, natural and artificial illumination, and atmospheric transmittance.

This paper describes the work performed under DVW to advance the current state of the art in the visual representation of environmental phenomena in virtual worlds. The implementation, known as the Virtual World Environment System (VWES), provides the management and computation of environmental changes in the virtual world. Physically-based environmental are incorporated into VWES, and provide a greater level of simulation fidelity. In a Distributed Interactive Simulation (DIS) environment, VWES receives Environmental State Messages, interprets the message data, executes the appropriate physical models, coordinates interactions among models and contributes to the visual image accordingly.

The VWES architecture is designed and implemented such that it can easily be integrated into visual simulation systems and can incorporate new physical models. The architecture includes two Application Program Interfaces (APIs), one which allows the visual simulation application to provide environmental data to VWES and control its operation, and one which allows VWES to communicate its results. A model registration mechanism provides a flexible means of registering one or more environmental models on a per exercise basis.

VWES has been integrated into two visual simulation systems: the NPSNETIV.6 Stealth and the VISTAWORKS ModStealth. Demonstration of the overall environmental simulation system is provided through interaction and correlation with its Computer Generated Forces (CGF) counterpart, ModSAF, and other environmental servers which issue environmental updates via the DIS network.

BIOGRAPHIES

Dr. Xin Li is a Real-time Software Engineer. He received his Ph.D. from the University of Central Florida and his M.S./B.S. in Computer Science from the Academic Sinica of China and the Northwest University of China. Dr. Li developed physically-based soil models while at the Institute for Simulation and Training. Since joining Lockheed Martin, Dr. Li led the effort to implement an interactive bulldozer demonstrating a real-time dynamic terrain capability on an image generator.

Mary Kruck is a Systems Engineer with a B.S. in Mathematics from California Polytechnic State University. She has participated in the development of air and ground-based simulation systems, and is currently leading the Dynamic Virtual World effort.

Henrik Lind is a Systems Engineer. He received his M.A. in Mathematics from the University of Washington, and an MSEE and BS in Physics from Carnegie-Mellon University. Since joining Lockheed Martin, Mr. Lind has participated in visual system software development and interactive game development.

Dan Gilbert is a Real-time Software Engineer with a B.S. in Electrical Engineering from the State University of New York at Buffalo. He has developed simulation software for various aircraft, and is currently the key developer of the Environmental Data Base employed by VWES.

The DVW Group can be reached at Lockheed Martin Information Systems, 13810 SE Eastgate Way, Suite 500, Bellevue, WA 98005, (206)957-3214 (Email: dvw@lads.is.lmco.com).

VIRTUAL WORLD ENVIRONMENT SIMULATION IN DISTRIBUTED SIMULATIONS

Xin Li, Mary Kruck, Henrik Lind and Dan Gilbert
Lockheed Martin Information Systems
Bellevue, WA

INTRODUCTION

The Synthetic Theater of War (STOW) is a major application of the Defense Advanced Research Project Agency's (DARPA) initiative in Advanced Distributed Simulations (ADS). One component of this application, Synthetic Environments (SE), seeks to advance the simulation of environmental effects in visual and computer generated forces (CGF) simulations. The goal of the SE Program is to extend simulation capabilities such that virtual battles may be fought under a wide range of environmental conditions including variable time of day, obscuration, local illumination and dynamically changing terrain. Further, these capabilities are to be employable within heterogeneous simulation platforms participating in distributed simulation exercises.

The Dynamic Virtual World (DVW) project is one element of the Synthetic Environments Program. It is responsible for the development of environmental effects, primarily atmospheric effects, for visual and CGF systems. This paper describes the visual implementation of DVW, the Virtual World Environment System (VWES).

VWES FEATURES

VWES provides an environmental modeling framework which can be tailored to individual visual simulation applications and individual simulation needs. It is designed to be easily integrated, providing an encapsulated modeling capability with simple, well-defined interfaces. A primary design goal has been an open architecture with no platform dependencies.

The VWES Framework

The VWES framework focuses on the visual simulation application interface and flexibility in the environmental modeling to be performed for a given simulation exercise. The interface allows the visual application to control environmental modeling capabilities, communicate required environmental data, and receive visual descriptions of environmental objects. Environmental modeling can be adapted to individual simulations through mechanisms for registering environmental models and controlling their processing. Figure 1 illustrates the VWES Framework.

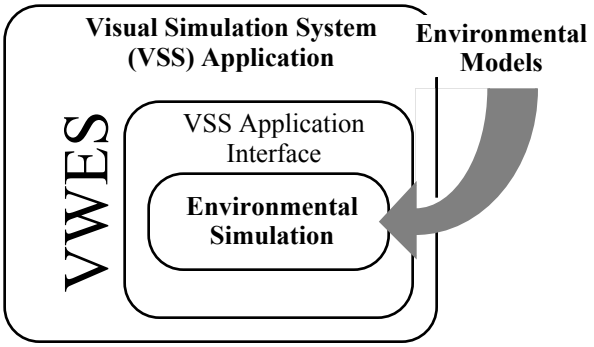

Figure 1 The VWES Framework

The visual application interface has two components: the "front-end" (VWES) Application Program Interface (API), and the "back-end" (VWES Agent) API. The VWES API allows the visual application to initialize VWES processing, update state information, control model execution, and terminate processing. The VWES Agent API allows VWES to communicate its results to the visual application and query the application for view and terrain surface information needed by modeling activities.

The integration of VWES into a visual application is reasonably straightforward. The developer incorporates VWES API function calls within the body of the visual application and develops a package (a VWES Agent) to translate the visual descriptions provided by VWES (via the VWES Agent API) into a form understood by the image rendering software.

In the simplest of integrations, the developer need only insert function calls to the basic VWES API initialization, message processing, update and termination functions. The incorporation of message processing function calls is conditioned upon the environmental models which may be registered, or rather the data upon which these models depend. More elaborate integrations may use environmental modeling control functions whose use is optional and left to the discretion of the integrator. The VWES model control functions allow individual models to be activated and deactivated during the simulation, and allow a model's processing characteristics (level of detail, update rate and fidelity) to be controlled. These functions are

intended to be used for scene control/load management and are effective only when a registered model supports such a capability.

The VWES concept of model registration allows any number of environmental models to be employed, and permits the visual application to perform their (the environmental models') initialization, update and termination through a small number of VWES API function calls. Model registration is performed on a per execution basis with models being identified by a text file. A user selects models for registration based upon simulation exercise requirements, and the capabilities of the host platform/Image Generator (IG).

The model registration process identifies to VWES, a set of software functions and dependencies for each model. The functions are associated with events which identify when each function is to be run e.g., at initialization, when environmental data is received, etc. The dependencies define the conditions under which the model is to be run e.g., when a wind direction change is detected, or when a battlefield obscurant is instanced.

The VWES Architecture

Five functional areas comprise VWES: the VWES API/Environmental Simulation Manager (ESM), the Modeling Simulation Manager (MSM), the Model Library, the Environmental Data Base Manager (EDBM), and the VWES Agent API (VWES Agent). Figure 2 illustrates the VWES architecture.

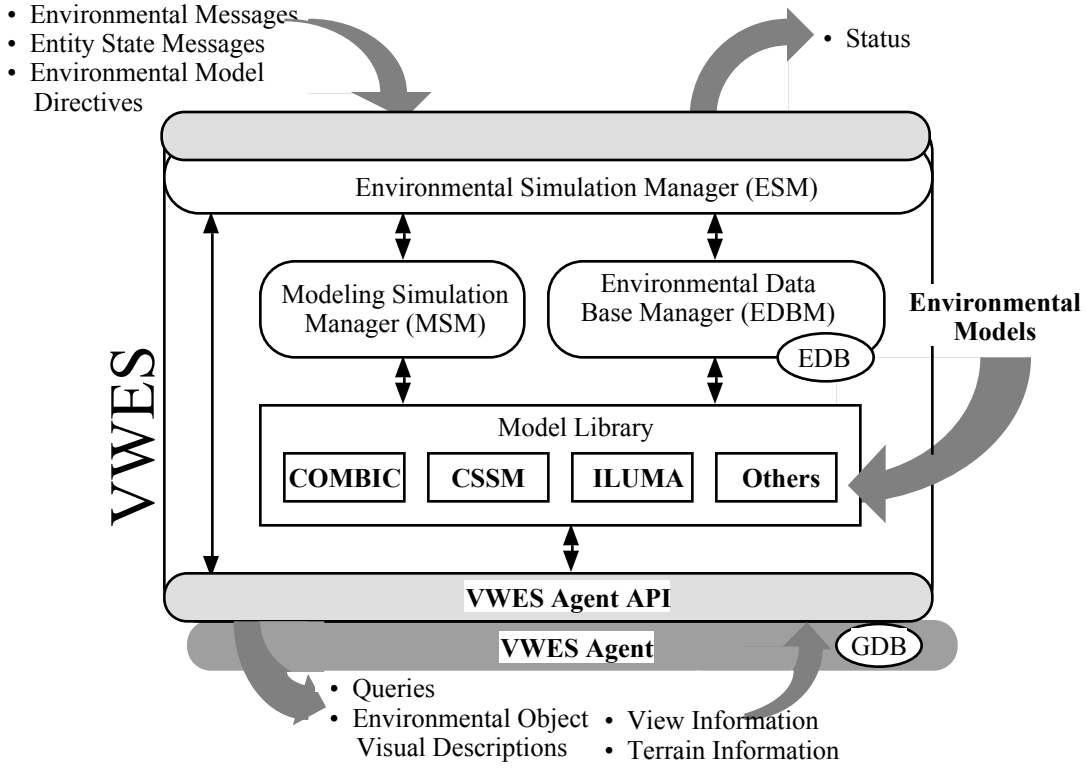

Figure 2 The VWES Architecture

Implemented through the Environmental Simulation Manager, the VWES API performs the overall management of environmental simulation activities and associated data structures. The ESM is supported by the Model Simulation Manager which manages modeling activities, including model registration. An internal data structure, the Environmental Data Base (EDB), contains pertinent environmental data. This data is managed by, and accessible through, the EDB Manager and its EDB API.

As previously mentioned, VWES itself does not provide all functionality needed to visualize the environmental objects/characteristics it models. A "VWES Agent" is needed to translate generic visualization data provided by VWES (via the VWES Agent API) into a form usable by the target IG.

Additionally, this VWES Agent is the means by which VWES accesses view and terrain information. Conceptually, the "VWES Agent" can be viewed as an implementation of the VWES Agent API.

Each of the VWES functional areas is discussed in further detail in the following sections. Interested readers are referred to [Lockheed Martin Information Systems (LMIS), 1996] for additional information.

ENVIRONMENTAL SIMULATION MANAGER

The Management of VWES

At initialization, ESM instantiates environmental models through a call to the MSM which registers models and establishes operational and data dependencies. Once models are registered, they are initialized along with the internal EDB, view information, communication with the VWES Agent, and utility services. All of these operations are achieved through a single call to the VWES API initialization function.

During the simulation, ESM's function is to keep the models updated with environmental changes to the virtual world, and maintain synchronization with the target IG. ESM monitors the current position of the eyepoint, maintaining the environmental state for the region of interest. Updates to the EDB, or movement of the eyepoint to a new geographic region, cause the state data, and geographically referenced update flags to be updated. These update flags indicate the environmental data type (e.g., temperature) which has changed. They are passed to the MSM, which triggers models dependent upon the changed data.

Additional operations performed by the ESM include assuring platform- dependent information in the Graphical Data Base (GDB) is current, synchronizing with the IG and resetting update flags once all environmental models are finished processing the new information.

At termination, ESM de-registers all environmental models and de-allocates all data structure memory associated with VWES software components.

Communication with the Visual Simulation Application

Two types of messages are recognized and processed by the ESM: environmental state messages (ENV_MSG) and entity state messages (ENT_MSG).

The ENV_MSG carries information describing environmental conditions: temperature, humidity, rain rate, wind velocity, cloud state, the current time, etc. Coordinates are associated with each data set to define the geographical location to which the data applies. In addition, the ENV_MSG may describe environmental events occurring in the virtual world. These include flare, smoke plume and vehicular dust instances. These differing types of information are grouped into records, known as environmental variants which form a part of the DIS environmental PDU [Neff, Haque and Shanks, 1995] from which the environmental state message is derived.

The ENT_MSG describes the state of a simulation entity, including entity id, type, location, velocity and current status. This data is used for creating, modifying and eliminating vehicle dust plumes.

When an environmental condition changes, or an object is instanced, the visual application passes information to the ESM through the VWES API. Models are invoked to process the information, update the EDB, and/or produce visual descriptions. Consider for example, the receipt of an Environmental PDU describing cloud state changes. The information is passed from the visual application and a registered weather model is triggered to process the data. ESM updates the EDB and through the MSM, triggers the Cloud Scene Simulation Model (CSSM). CSSM determines if the updates are significant, and, if so, executes, producing liquid water content distributions. This data is communicated to the VWES Agent, which translates it to semi-transparent polygons and submits the polygons to the IG for rendering.

Scene Content Control

ESM allows any number of environmental models to be employed and permits the visual application to control each registered model's operation. Through the VWES API, the application is able to access model names and ids, and activate and deactivate each model at any time during a simulation.

ESM also provides the application with a scene content control mechanism to manage computational resources and IG load according to variable simulation requirements. A set of VWES API functions allows the application to instruct environmental models to switch levels of detail, run at different update rates, or change modeling fidelity levels. If a model is not capable of reacting to these control adjustments, no changes in processing occur.

MODELING SIMULATION MANAGEMENT

Environmental models in VWES are self-contained and configurable according to simulation requirements. Models are bound by software wrappers added to support control and satisfy performance constraints. Models exchange information with the EDB and the VWES Agent. They identify environmental data dependencies and event handlers to process environmental information and produce visual representations.

The Modeling Simulation Manager utilizes a model registration mechanism that allows VWES to provide visual applications with an open architecture under which any model can easily be incorporated. This section describes how VWES manages environmental models.

Model Registration

A model registers itself to the MSM by providing information such as its name, trigger time, event handlers, and environmental data dependencies. A trigger time specifies how often the model wishes to be updated. Event handlers are software functions which a model wishes to be executed when specific events occur (e.g., when the system is initialized, uninitialized, updated each simulation loop, or when environmental conditions change).

There are nine events recognized by the MSM:

Table 1 Environmental Modeling Events

EVENT	DESCRIPTION
INITIALIZE	System Initialization
UPDATE	System Update*
PROC_ENV_MSG	System Receipt of an Environmental Message
PROC_ENT_MSG	System Receipt of an Entity Message
RERUN	System Detection of an Environmental Data Change
SET_LOD	System Adjustment of Model Level of Detail
SET_UPDATE_RATE	System Adjustment of Model Update Rate
SET_FIDELITY	System Adjustment of Model Fidelity
CLEAR	System Termination
* Generally each simulation frame.	

A model may register as many event handlers of each event type as it desires. When an event occurs, functions associated with the event type are executed in the order in which they were registered.

Environmental Data Dependency Management

When a model registers, it identifies a set of environmental sensitivities, i.e., environmental data dependencies. When any dependent data are updated in the EDB (within a region of interest specified by the ESM), event handler(s) registered with token RERUN are invoked to update visual description(s).

The MSM maintains variables used to manage data dependencies. A variable pair is defined for each environmental data type on which models depend. The first variable, TOTAL_DEPENDENTS, defines the total number of registered, active models which depend on the data type. This variable is used to establish an initial value for the second variable, CURRENT_DEPENDENTS. CURRENT_ DEPENDENTS is used to track the number of models which need to operate on the updated data. When environmental data is updated, update flags are set and CURRENT_ DEPENDENTS is set to the value of TOTAL_DEPENDENTS. Subsequently models are triggered, and upon a model's completion, CURRENT_DEPENDENTS is decremented. When all models are complete, CURRENT_DEPENDENTS is zero, indicating that the update flags may be reset.

This approach allows an order-independent execution of models processing environmentally sensitive data. It assumes however, that models have equal chance to access the data. In other words, models should be synchronized.

Scene Content Management

The MSM supports scene content management by providing functionality to instruct models to run at specified levels of detail (LODs), update rates, or fidelity levels. As previously noted, a model may be deactivated or activated through the MSM during run-time, allowing alternative methods to be employed for modeling environmental objects.

The Model Library

VWES currently supports 11 environmental models; Table 2 identifies each model's registration name and the environmental object/characteristic it models. The models themselves are too complex to describe completely. This section provides a brief overview of each model; references provide additional information.

Table 2 VWES Supported Environmental Models

Model Name	Environmental Object/Characteristic
CSSM	Clouds
COMBIC	Obscurants
ADVANCED_COMBIC	Advanced Obscurants
ILUMA	Global Illumination
LOWTRN	Haze
BEERS_LAW	Haze
FLARE	Flares
VDUST	Vehicular Dust
SIMPLE_WEATHER	Uniform Weather
OBSERVED_WEATHER	Observed Weather
GRIDDED_WEATHER	Gridded Weather

CSSM - The Cloud Scene Simulation Model is an empirical cloud simulation model developed by The Analytic Sciences Corporation (TASC). It simulates realistic, high-resolution clouds based on large-scale weather conditions [TASC, 1994]. Stochastic field generation techniques and convection physics are used to convert weather data into liquid water content.

COMBIC - The Combined Obscuration Model for Battlefield Contaminants was developed by the Army Research Laboratory (ARL) [Hoock, Sutherland and Clayton, 1987]. It is a comprehensive model for battlefield obscurants which produces time histories of an obscurant's envelope within which it assumes a Gaussian mass distribution. The model is capable of simulating 30 different obscurant source types, viewed by any of seven sensors. To allow real-time update of obscurants, efficient modeling and visualization algorithms have been developed by Northrop Grumman Data Systems [Gardner and Li, 1996].

ADVANCED_COMBIC - Advanced COMBIC is an extension of COMBIC which applies terrain-following and variable, local wind effects to obscurants.

ILUMA - The Natural Illumination Under Weather Conditions model was developed by the Army Research Laboratory (ARL) [Duncan and Sauter, 1993]. It is used to predict global illumination under a fairly wide range of sky and weather conditions. The model accepts meteorological data including: sun and moon position, moon phase angle, high-, middle-, and low-level cloud types and amounts, and the presence of fog or precipitation.

LOWTRN - The LOWTRAN atmospheric model, developed by the U.S. Air Force Geophysics Laboratory (AFGL), provides low spectral resolution (up to 20 cm^{-1}) transmission and/or background radiance of atmospheric paths [Pierluissi and Maragoudakis, 1987]. The transmittance along an atmospheric path is considered a function of the total amount of absorbing or scattering species along the path. Atmospheric conditions on which calculations are based include aerosol, rain, cloud extinction, pressure, temperature, and water vapor density.

BEERS_LAW - A simple closed form approximation for transmittance ($e^{-ext\_coeff*range}$).

FLARE - Unlike other models, the illumination flare model is not physically-based and does not react to environmental changes with the exception of time-of-day. The model controls flare motion, burn behavior and illumination characteristics. Provided with information such as initial position, velocity, color, illumination cone angle and light intensity, the model derives the characteristics of an animated sequence and local illumination effect.

VDUST - Supported by COMBIC, VDUST models dust plumes along a path behind a moving ground vehicle. The model inherits the physically-based features of COMBIC while taking into account vehicle attributes and soil characteristics.

VWES currently supports three weather models. The weather model's purpose is to respond to Environmental State messages, detecting changes to the environmental state and modifying the EDB according to geographic region and propagation mechanism.

SIMPLE_WEATHER - Applies weather uniformly across the virtual battlefield.

OBSERVED_WEATHER - Applies weather uniformly about a specified observation point. Observation points are assumed to have a circular region of influence of radius R.

GRIDDED_WEATHER - Applies weather locally, i.e., the weather is spatially variant.

ENVIRONMENTAL DATA BASE MANAGEMENT

To remain abreast of environmental changes occurring in the virtual world, VWES maintains environmental state information in a data structure known as the Environmental Data Base. The EDB is managed by and accessible through the EDBM which provides API functions to store, detect, and retrieve data.

The EDBM maintains spatially variant environmental data (e.g., temperature, precipitation, etc.), as well as data treated as uniform over the simulation region (e.g., sun and moon position, current time).

Spatially variant data, known as Atmospheric Data, are maintained in a grid structure. For each grid node, a data set is maintained; this data set describes the current environmental state for a corresponding geographical point. Uniform data, known as Global Data, is stored in a simple data structure.

Configuration of the gridded EDB is very flexible. Extents, resolutions, and initial data values are specified through an EDB API function call initiated by the ESM during initialization. The EDB may be configured as a 2-D or 3-D data base, with the extents in each dimension being independent of the others. Grid cell resolutions in each dimension are also independent of each other, allowing for non-square or non-cubic grid cells. The geometry of the EDB is selected prior to run-time, and can be configured in such a manner as to cover the entire virtual battlefield, while providing the best fidelity to performance tradeoff.

For most exercises, a single level of detail data base ensures adequate environmental fidelity. However, it may be desirable to represent some portions of the virtual battlefield at a higher resolution, for example, a tactically important area of an otherwise nondescript desert. The EDB satisfies this requirement by allowing specified locations of the data base to be represented at multiple levels of detail. LODs may be added or removed in real-time. Expanding a grid cell into multiple LODs allows VWES to access environmental data at varying resolutions for the corresponding region. Each finer LOD is derived by dividing the "parent" grid cell into eight "children". This scheme of multiple LODs for specific regions, versus the entire data base, is referred to as a hybrid oct-tree.

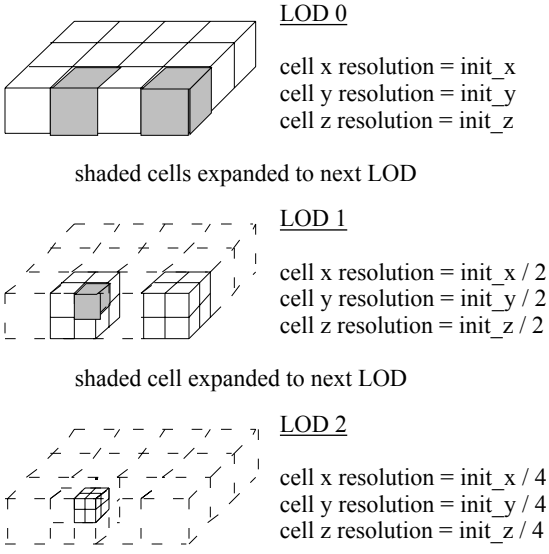

Figure 3 VWES Hybrid Oct-Tree

The EDB API allows the ESM or environmental models to update portions of the data base by specifying the affected location (origin, extents, LOD), a bit mask indicating the data type(s) to update, and the new data values. In the hybrid oct-tree structure, changes to the data are propagated from the point of update to other levels of detail. Modifications to the finest resolution are downsampled to update coarse resolutions; data input at a coarse resolution is duplicated to update fine resolutions.

Atmospheric Data updates are performed through the use of the EDB API "set data" functions. Updates can be distributed across the gridded EDB via two distinct mechanisms.

The first mechanism propagates uniform data values across a geographic region for a specified data type(s). VWES weather models SIMPLE_WEATHER and OBSERVED_WEATHER use this mechanism to update the EDB. The SIMPLE_WEATHER model, in which Atmospheric Data is uniform across the simulation region, specifies the entire data base as the affected geographic region. The OBSERVED_ WEATHER model, which affects a circular region of influence around an observation site, specifies the geographic point location of the observation site as the affected region and sets a special propagation flag. The propagation flag instructs the EDB to distribute the updates to all grid cells determined to be within the site's region of influence.

The second mechanism propagates varying data values for a single data type across a geographic region. The GRIDDED_WEATHER model uses this mechanism to update the EDB.

Global Data are updated through a separate EDB API function. This function requires only a bit mask indicating the data type(s) to update and the new data value(s).

The EDB provides a means by which ESM can detect environmental changes that affect registered models. In support of this capability, each grid cell of the EDB includes an "update flag" bit mask; each bit corresponds to an individual, unique data type. The appropriate update flag bit is set when a data type value for the grid cell is updated. The EDB API provides a function that retrieves and logically ORs the update flags of all grid cells for a specified region. A corresponding API function is provided to reset update flag bits. The Global Data structure maintains update flags which indicate changes to global data types. Similar API functions are provided to retrieve/reset global update flags.

Finally, the EDB allows ESM and the environmental models to retrieve environmental data associated with specific locations. Atmospheric Data is retrieved by specifying a desired geographic point, an LOD, and a bit mask indicating the data type(s) to retrieve. The EDB traverses the data base and retrieves the requested data from the appropriate grid cell. Data from the closest valid grid cell is retrieved for queries whose geographic point lies outside the defined dimensions of the EDB. A similar API function is provided for retrieving Global Data.

THE VWES AGENT AND ITS API

The VWES Agent is the platform-dependent component associated with VWES. It is used to (1) Generate visual representations in a format understandable by the target IG, (2) Interface to a Terrain Data Base (TDB), and (3) Obtain graphical view parameters. The VWES Agent is developed as a result of integrating VWES into a visual simulation application. It is generally perceived to consist of three sub-modules: VWES_AGENT_VIS, VWES_ AGENT_TDB, and VWES_AGENT_VIEW.

Generating the IG Representation (VWES_ AGENT_VIS)

VWES_AGENT_VIS accepts generalized graphic data, translating the data into calls to the IG graphics library. It provides the functionality to display each of the environmental objects/characteristics modeled within VWES. This currently includes: smoke, clouds, flares, and characteristics which are a function of time of day (e.g., sun angle, sun light intensity (diffuse [aka direct] and ambient [aka scattered] light), lunar illumination, sun and moon representations, sky color, directional horizon glow, and spline-based, colored haze, and local illumination (for flares)).

VWES_AGENT_VIS functionalities for processing environmental objects/characteristics are not necessarily independent; they may interact with one another. A prime example is the derivation of sky characteristics. Realistic sky representations are a function of time of day AND haze characteristics. VWES_AGENT_VIS, as we've chosen to implement it, re-generates sky characteristics if either the time of day or haze characteristics have changed. This implementation is supported by our separation of VWES_AGENT_VIS into (1) data gathering and (2) data rendering operations. During data gathering, VWES sends updated time of day and haze information to VWES_AGENT_VIS. This data is combined to create appropriate sky characteristics which is then rendered.

Since rendering time can be large compared to data gathering time (depending on scene complexity), we have chosen to employ jointly accessible data buffers. The graphics data buffers are filled during data gathering, and sent to the IG during data rendering. Thus, multiple VWES_AGENT_VIS calls can be made by VWES before the graphics process is able to render the data. Only the latest data appears in the image.

The VWES Agent API (VIS) enforces a rigid separation between environmental model data and the data used by the IG to render images. In support of the data gathering/rendering scheme described above, the API includes functions to "set" data as it is gathered, and to "update" data to be rendered. To render polygons representing smoke and clouds, for example, the following occurs.

Each time VWES generates new polygon data, a "set" function is called. This function copies polygon data to local (VWES_AGENT_VIS) buffers, and adds data needed for rendering, but does no rendering. Instead, a flag indicating the data has changed is set. The "update" function interprets the "changed data flag", sending the polygon data to the IG, clearing the flag, and resetting the data buffers.

Interfacing to a Terrain Data Base (VWES_ AGENT_TDB)

VWES accesses the visual simulation system's Terrain Data Base to obtain geographic information, including data base origin, Universal Transverse Mercator (UTM) designations, extents, and terrain elevations. The associated interface, the TDB API, consists of functions to retrieve general TDB information and terrain characteristics. Note that not all visual system terrain data bases provide the information identified above. In those instances, the developer may provide the data through a text file or some other means.

For optimization, our implementations return multiple elevation values for a single query (specifying multiple reference points). The method by which queries are handled is dependent on the data base used, and the underlying architecture of the data base and target IG.

Obtaining Graphical Parameters [VWES_ AGENT_VIEW]

The VWES Agent API (VIEW) provides ESM and the physical models with standardized access to graphical configuration information. Information available through this API includes clipping planes, eyepoint location and orientation, viewing vector, field-of-view angles, and frame update rate. VWES stores view information in the Graphics Data Base (GDB) which is

initialized by ESM and updated each simulation frame. Changes to the IG's software configuration do not incur interface modifications.

VWES maintains the GDB through queries to the visual application. View information is available to VWES components through the VWES Agent API (VIEW) and is used to bound visual representations. For example, smoke plume polygons are removed from processing if they fall outside the field of view of the eyepoint.

DESIGN ISSUES

Several issues have arisen during the VWES design process. Some are the result of attempting to define a generalized VWES Agent API usable by many heterogeneous visual systems. Others stem from the implementation of VWES Agents for multiple visual systems. The following sections address design issues raised during our development efforts.

Interface Content

What data should be included in the interface, and what form should it take? In answering this question, we analyzed the types of objects we wanted to represent and identified the basic image primitives most likely to be used. Further we identified the descriptive information needed to describe these primitives to various IG platforms and the most reasonable means of referencing environmental objects to be rendered. We noted that it is desirable to be able to reference an environmental object that is comprised of multiple primitives (such as a smoke plume) by a single reference identifier.

In practice, we found much commonality between the two systems into which we've integrated VWES, the VistaWorks ModStealth and the NPSNETIV.6 Stealth.

VWES Agent Functionality

In developing a VWES Agent, operations supporting efficient image rendering, and visual system limitations must be identified. Operations not performed by the system into which one is integrating should be performed by the VWES Agent. Visual system limitations must be "worked around", or VWES Agent functionality must be limited to only that which can be supported by the visual system.

Two examples which affect efficient image rendering are polygon sorting and object persistence. Both of the systems into which VWES has been integrated execute on a Silicon Graphics Inc. (SGI) platform. SGI platforms require semi-transparent polygons to be rendered back-to-front. In the VistaWorks ModStealth VWES Agent, no sorting is performed; all necessary sorting is performed by VistaWorks. In the NPSNETIV.6 Stealth VWES Agent, the Agent performs the sorting, as NPSNET provides none.

Does the IG retain a memory of each polygon from frame to frame, or does it begin each frame anew? The former holds for the VistaWorks ModStealth, the latter for the NPSNETIV.6 Stealth. In the VistaWorks VWES Agent implementation, a pool of graphical objects is maintained by the Agent. As new polygon data is received, it is bound to existing graphical objects, or new objects are allocated. The binding of VWES data to graphical objects changes from frame to frame, but objects are not de-allocated between frames which would be expensive and needless. Unused graphical objects in a given frame are deactivated. In NPSNETIV.6, graphical objects are not persistent, but are supplied anew each frame.

In some cases, visual system limitations may mean that VWES data may not be used at all or that the data may be used indirectly. Consider, for example, illumination flares. The VWES Agent API describes illumination flares as an animated sequence of textured polygons. Some visual applications, such as the NPSNETIV.6 Stealth, allow independent control of the animation; this control has been built into the NPSNETIV.6 VWES Agent. The VistaWorks ModStealth provides no such control. The VistaWorks Agent uses the initial flare data to trigger a VistaWorks-controlled animation sequence.

Limited IG Capabilities

Since not all IGs provide the same capabilities, the VWES Agent may need to provide functionality to compensate for these limitations. One example is the processing of local illuminations. Some IGs limit the number of local illuminations which may be active at any one time due to the computational expense of modeling them. The SGI hardware supports eight (8) local illumination sources; one is allocated for global illumination (the sun), leaving seven (7) for representing such things as illumination flares. Since distributed interactive simulations impose no limits on the number of active flares, the VWES Agent is responsible for managing the allocation of local illumination sources to active flares.

WHAT'S AHEAD

As this paper is written, VWES is in its fourth development phase. Enhancements expected to be implemented during this phase include sea state and wave modeling, additional explosions and weapons

effects, and IR Sensor modeling. Additional efforts focus on the coordinated (with other Synthetic Environment programs) development and integration of hydrology modeling, visual appearance, and weather modeling. Finally, overall VWES performance will be measured.

CONCLUSION

The need for enhanced simulation of environmental elements is clear. The uses and planned uses of simulation in the training of personnel, the development of tactics, and the evaluation of weapon systems is expanding. As such, these simulations must be comprehensive and well representative of today's battlefield. DVW and the other Synthetic Environment programs are furthering the realistic state of the battlefield and increasing the usefulness of simulation in today's defense environment.

ACKNOWLEDGMENTS

The technical development described herein was performed under the DVW contract, sponsored by the DARPA, the U.S. Army Topographic Engineering Center, and the Defense Modeling and Simulation Office.

REFERENCES

[Duncan & Sauter, 1993] Duncan, L.D. and Sauter, D.P., "Natural Illumination Under Realistic Weather Conditions", Technical Report, Army Research Laboratory, White Sands Missile Range, NM, 1993.

[Gardner & Li, 1996] Gardner, G. and Li, X., "Physically-based Battlefield Obscurants in Distributed Interactive Simulation", The proceedings of IMAGE Conference '96, Phoenix, NM, 1996.

[Hoock, Sutherland & Clayton, 1987] Hoock, D., Sutherland, R. and Clayton, D., "Combined Obscuration Model for Battlefield Contaminants (COMBIC)", ASL-TR-00221-11. U.S. Army Atmospheric Sciences Laboratory, 1987.

[Lockheed Martin Information Systems (LMIS), 1996] Lockheed Martin Information System, "System Description Document for the Virtual World Environment System (VWES)", Technical Document, 1996.

[Neff, Haque and Shanks, 1995] Neff K., Haque S. and Shanks G., "Revision of the Environmental PDU", Paper 95-13-054, Summary Report: The 13th Workshop on Standards for the Interoperability of Distributed Simulations, Vol. 1, 1995

[Pierluissi & Maragoudakis, 1987] Pierluissi, J.H. and Maragoudakis, C.E., "Atmospheric Transmittance/Radiance Module LOWTRN", Technical Report, TR-0221-4, Electrical Engineering Department, The University of Texas at El Paso, 1987.

[TASC, 1994] TASC, "Atmospheric Scene Simulation Modeling and Visualization", Technical Report, Contract No. F19628-93-C-0203, Electronic Systems Center Air Force Materiel Command, 1994.

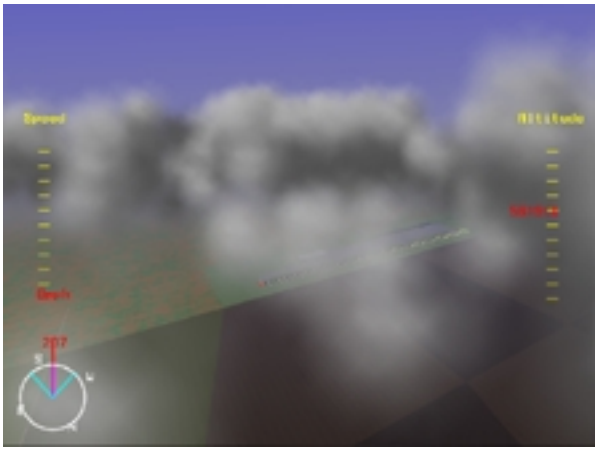

Figure 4 Cloud Simulation

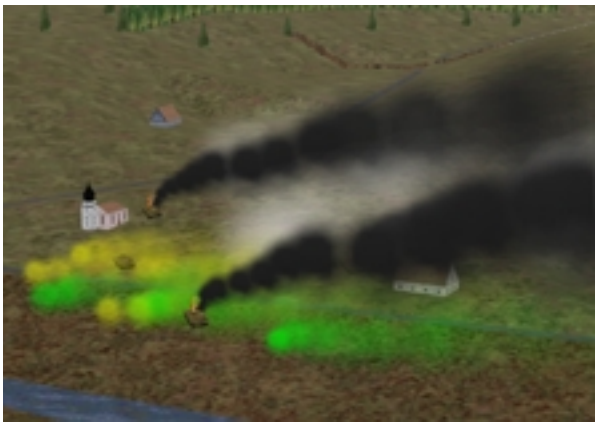

Figure 5 Obscurants Simulation