MODELING THE CLOUD ENVIRONMENT IN DISTRIBUTED INTERACTIVE SIMULATIONS

Maureen E. Cianciolo
TASC
Reading, Massachusetts

Brian Soderberg
LORAL Advanced Distributed Simulation
Bellevue, Washington

ABSTRACT

This paper describes an ongoing effort to develop and integrate an empirical cloud model within a Distributed Interactive Simulation (DIS) environment in support of high-fidelity training and simulation applications. TASC is developing a cloud model (known as the Cloud Scene Simulation Model) to simulate realistic high-resolution cloud features within domains defined by larger-scale weather conditions. The cloud model generates four-dimensional (three spatial and time), multi-layer cloud fields using a combination of stochastic field generation techniques and convection physics, where internal model parameters are tuned to fit observed cloud data. One data set is generated for each specified output time and contains cloud water density values arranged on a regular volumetric grid. A typical output field contains tens of thousands of data points covering simulation domains of 50 km or more.

Because these data sets are too dense to be transferred across the DIS network or rendered in real-time, we have developed an approach that approximates the complex cloud formations generated by the model as a series of geometric primitives. The cloud data sets are filtered to the level-of-detail appropriate for a particular simulator. The approach uses a planar-wise approximation of a volumetric phenomenon that takes advantage of today's state-of-the-art image generator hardware. The cloud model runs in real-time, allowing for smooth transitions as the weather conditions evolve over the simulation domain.

In this paper, we present an overview of the Cloud Scene Simulation Model (CSSM); its inputs, outputs, and overall methodology. We describe a DIS architecture which enables distributed real-time calculation of large cloud fields, and address usage of and extensions to the standard DIS network protocol. We follow with a description of the volumetric rendering techniques employed in this effort. Finally, we summarize and briefly discuss the application of our methodology to other atmospheric phenomena in future implementations. We conclude our oral presentation with a video tape showing real-time cloud field generation and visualization within a DIS training environment.

ABOUT THE AUTHORS

Maureen Cianciolo is a member of TASC's System Sciences Division. She has over six years of experience in atmospheric physics, software development, and data visualization. As principal investigator of the atmospheric modeling and visualization efforts at TASC, she is currently using her background in physics and computer modeling while leading development of high-resolution models of atmospheric moisture.

Brian Soderberg is a LORAL Division Scientist and the manager of the Computer Image Generator (CIG) development and algorithm research group. Mr. Soderberg has ten years experience in computer graphics and image generator research and development. His research and development focuses on development of new algorithms for image generation to support LORAL's ongoing CIG development and DIS-related programs.

MODELING THE CLOUD ENVIRONMENT IN DISTRIBUTED INTERACTIVE SIMULATIONS

Maureen E. Cianciolo
TASC
Reading, Massachusetts

Brian Soderberg
LORAL Advanced Distributed Simulation
Bellevue, Washington

MOTIVATION

Atmospheric moisture in all of its forms (humidity, precipitation, clouds, etc.) can have significant effects on many aspects of military and commercial systems. However, current simulators are generally limited in their ability to provide a high-fidelity weather environment in order to fully simulate these effects. In this section we briefly outline some of the effects of weather on the operation of sensor systems and local terrain conditions. We discuss the inability of raw weather data alone to satisfy the fidelity requirements of some sensor and terrain models, and point out the challenges of real-time weather data collection.

All of these discussions serve as the motivations behind our ongoing effort to develop a computationally-efficient atmospheric moisture model (consisting of cloud, rain, and humidity models, although we discuss only the cloud model in this paper) that can function with a wide range of input data, generate scenes consistent with these inputs, and provide the necessary level of fidelity for typical sensor and dynamic terrain models.

Weather Effects on Sensors

Atmospheric moisture can impact the operation of military and commercial electro-optical sensors through absorption, scattering, and emission of radiation, and modification of the local atmospheric microphysics (such as aerosol particle size distributions). Examples of these impacts include: variations within moisture fields (e.g., near cloud edges) introducing background clutter that can severely degrade target detection and acquisition; the presence of water vapor attenuating electro-optical signals, thus reducing visibility and transmission; the presence of rain regions which are completely opaque to most common sensor wavelengths.

These atmospheric effects are ubiquitous and can represent a tactically significant tool to a well trained staff on the battlefield. Including these highly variable and highly local effects within the Distributed Interactive Simulation (DIS) training environments provides the opportunity to understand the effects of weather on specific systems and potentially take advantage of them.

Weather Effects on Terrain

The weather affects terrain and terrain-based models in two primary ways: 1) by modifying the radiation received at the ground, thereby introducing temperature and illumination differences across the ground domain, and 2) by modifying the ground characteristics (e.g., trafficability) directly in the presence of rain, snow, ice, or fog. The highly local nature of these effects can be tactically significant and therefore should be accounted for in realistic training environments. Along with the spatial distribution of the weather parameters, their temporal distribution is important. Ground force and ground vehicle simulators need to know the time-history of any precipitation to update soil characteristics and respond accordingly.

Weather Data Availability

Much of the raw weather data that are readily available (including temperature, moisture, pressure, wind, cloud layer information, rain rates, etc.) come from national or global networks of numerical weather prediction models, surface observations, and upper air balloon measurements. The resolution of these data sources is very coarse (on the order of tens of kilometers or more), which, although useful to characterize the general state of the atmosphere on a regional scale, does not capture the higher-resolution features within local regions and therefore does not satisfy the fidelity requirements of many simulators. (A limited number of higher-

resolution meteorological data sets exist in the vicinity of some airports or in support of special military training exercises, however, they are not generally available.)

In addition, we must consider the operational availability of these data to support war-time mission planning and rehearsal. During periods of conflict only limited weather data may be available due to data black-outs, limited communications, etc. Training/planning systems being used and developed for these conditions cannot depend on using special databases or full complements of meteorological data.

CLOUD MODEL OVERVIEW

Methodology

The Cloud Scene Simulation Model (Ref. 1) is an empirical model that generates high-resolution, four-dimensional (three spatial and time), multi-layer (low, middle, high, and convective layers) cloud fields consistent with larger-scale input weather conditions. That is, it simulates realistic structure (typical resolutions of 1-100 meters) within a domain defined by general meteorological characteristics. One output field is generated for each specified output time and contains cloud water density values arranged on a regular volumetric grid. The model can simulate a variety of cloud types including: cirriform (high, thin, cloud streaks), stratiform (low, homogeneous cloud layers), and cumuliform (puffy, vertically-developed convective clouds). Examples of three different cloud types are shown in Figure 1. Terrain-induced cloud types, including fog and roll clouds, will be added in the near future.

The model uses a fractal algorithm (the Rescale and Add algorithm, Ref. 2) to specify the horizontal distribution of cloud elements across the user-specified model domain, where parameters within the fractal algorithm are tuned to fit observed cloud data (e.g. the variability of liquid water density within cloud elements of differing types is controlled by a "length" parameter within the algorithm which is determined by an analysis of aircraft-based cloud measurements). The vertical growth of the clouds is modeled using convection physics and knowledge of atmospheric structure. Comparisons with real data have shown that the model captures the characteristically complex internal and external structure of cloud fields observed in nature.

The fractal algorithm within the model controls the stochastic distribution of the clouds. It is initialized with a number seed. By using the same seed and initialization data, one can reproduce identical scenes and ensure that individual simulators use identical cloud fields. By varying the number seed, one can simulate large numbers of unique fields, all consistent with the input weather state, which could then be used for Monte Carlo studies.

The model simulates three-dimensional cloud fields evolving in time. For simulations evolving in time, the model uses the bounding weather conditions at the beginning of the time period and the end when determining intermediate cloud fields. For example, a high-resolution simulation may require cloud fields every 1 minute although weather data (from a forecast model or archived data base) may only be available every one hour. The model correctly "interpolates" for all time periods in between taking into account temporal evolution (growth, dissipation, and diffusion are simulated using a fractal algorithm) as well as advection (cloud motion due to local winds).

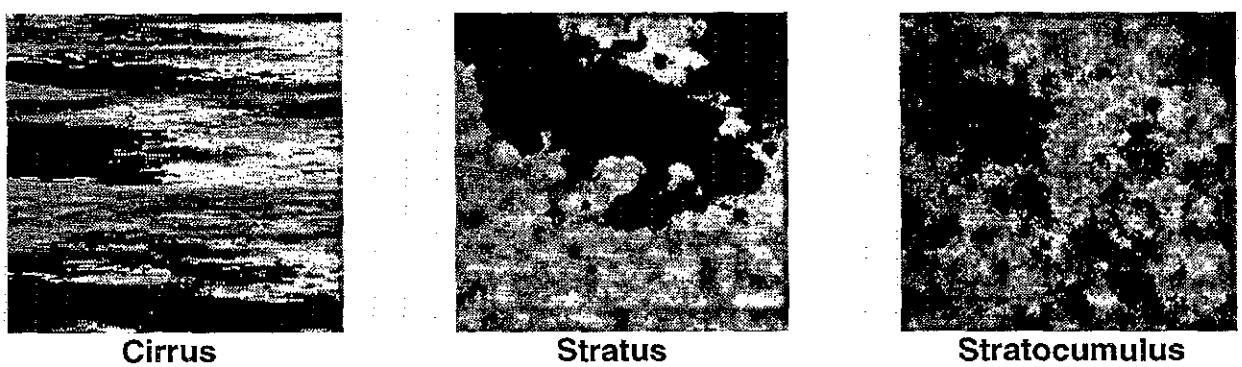

Figure 1 Two-dimensional slices (nadir view) of sample cloud types generated with the CSSM

The model is written in ANSI C. It currently runs in two modes: 1) stand-alone mode in which it generates sequences of data files that are then rendered independently, and 2) DIS-integrated mode in which it is initialized with an environmental PDU (Protocol Data Unit), and then simulates fields that are rendered within a CIG. The integration of the cloud model within a DIS environment is the primary focus of the next section. But first we provide more detail concerning the input and output of the cloud model.

Model Input

The model simulates cloud fields at very high resolutions (on the order of 1-100 meters) consistent with coarser-resolution user-supplied meteorological input data. Along with providing the meteorological inputs, the user also defines the simulation domain by specifying an (x,y,z,t) origin, spatial extent and resolution, and a temporal period and resolution. (Within the DIS-integrated mode, these parameters can vary during the simulation, e.g., the origin can vary as the region of interest changes.)

The model can use single-valued inputs over the region of interest. For example, a user simply specifies overall cloud layer information (e.g., 50% stratus layer at an altitude of 1000 meters and 80% cirrus layer at 3500 meters) and a single wind, temperature, and relative humidity vertical profile for the simulation domain. Or the model can be initialized with gridded inputs where cloud layers and meteorological variables can vary across the simulation domain. In either case, the model generates a realistic higher-resolution scene that satisfies these input criteria.

In our DIS-compatible implementation, we provide all model inputs through an environmental PDU. The structure of that PDU is based on the environmental state PDU proposed at the most recent DIS Workshop Simulated Environment Working Group (Ref. 3). We added to it the parameters required to initiate the CSSM software. (The PDU extent field designates the number of extra parameters required for running the cloud generator. See Table 1.) A list of the parameters required specifically by the CSSM follows:

Meteorological data (gridded or single-valued)
- wind as a function of altitude
- temperature as a function of altitude
- dewpoint temperature as a function of altitude
- pressure as a function of altitude

Cloud layer data (gridded or single-valued)
- cloud amount (0-100%) by layer
- cloud type by layer
- mean cloud base height by layer
- maximum cloud top height by layer

Domain parameters
- domain origin (x, y, z, and time)
- domain extent (e.g., 100 x 100 kilometers)
- domain resolution (e.g., 10 meters)
- temporal extent (e.g., 60 minutes)
- temporal resolution (e.g., 3 minutes)

Miscellaneous parameters
- root filename for output files
- number seed (to initialize the fractal algorithm).

The input parameters concerning the location of the domain origin (in space and time) have been included to satisfy simulations that move over time or cover a large spatial extent (beyond any single simulator's field of view). In these cases it is not necessary to model the atmosphere over the entire simulation domain, but instead just over the region of interest to each simulator. The domain origin parameters ensure that all the simulation participants see a consistent and continuous cloud environment which is paramount in DIS applications.

Model Output

The Cloud Scene Simulation Model generates a single output file for each user-specified output period. The files contain liquid/ice water content (LWC) values (grams/m^3) at every point in the user-defined model domain. That is, each output file is a three-dimensional snapshot of the scene at a given time. Visualization of these fields within a DIS environment is the subject of a later section. How these synthetic output fields compare with actual cloud fields is the subject of the next section.

Comparison to Real Cloud Data

A number of studies have shown that clouds show characteristic fractal behavior over a broad range of spatial scales (Refs. 4, 5). We use the Rescale and Add (RSA) fractal algorithm (Ref. 2) within the model to simulate the spatial distribution of cloud cover and liquid water content. A number of parameters within the RSA

algorithm control the statistical characteristics of the resulting fields. In our previous modeling effort, we used a limited set of cloud observations to tune these fractal parameters by comparing LWC time series sampled using conventional hot-wire probes mounted on aircraft to series sampled from the model output fields. Figure 2 shows two sample cases. We are currently collecting a large number of additional observations spanning a wide range of climatological conditions for further analysis. We will use a portion of these observations for parameter estimation and set aside the remainder for model validation.

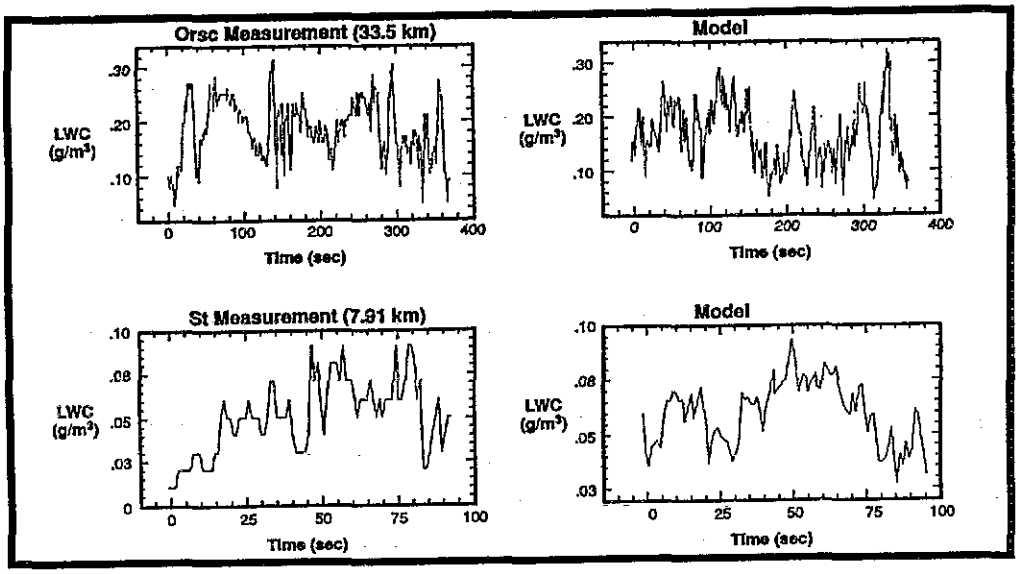

Figure 2 Comparison of measured and model-produced LWC spatial series for two different cloud types: OrSc (orographic stratocumulus) and St (stratus)

CLOUD SIMULATION WITHIN A DIS ENVIRONMENT

This section briefly presents the traditional DIS simulation node architecture, followed by a detailed description of the working architecture developed under this effort which allows integration of the cloud scene simulation and visualization models.

Standard DIS Architecture

Figure 3 presents a logical view of processing at a simulator node on the DIS network. Information on the DIS network is exchanged by way of Protocol Data Units (PDUs). PDUs describe the position and state of vehicles and other dynamic entities on the network. The *simulation host* is responsible for dynamics simulation and processing messages sent over the network. The host sends the computer image generator ownship vehicle data such as position, rounds fired, etc. as well as positional updates of other dynamic entities on the network to be displayed. The front end of the CIG is the *graphics processor* which is responsible for processing messages from the host, managing the viewport configuration, paging data from disk, and sending commands and data to the graphics hardware pipeline. State-of-the-art graphics workstations, such as the SGI Onyx, allow the simulation host and graphics processor to perform on common processors, blurring the distinction between these two logical functions.

We use this same basic architecture in our implementation with a number of extensions to support real-time cloud visualization. We describe those extensions in the next section.

DIS Architecture with Clouds

The CSSM requires substantial time (up to 90 seconds for a typical scenario) to calculate a single cloud data field. Typical DIS simulations run at 15 to 30 frames per second. To accommodate the CSSM processing time, the system architecture was designed to allow for concurrent synchronous (with frame rate) and asynchronous dynamic environmental model processes to run within the DIS simulation. The CSSM runs in the background storing its output

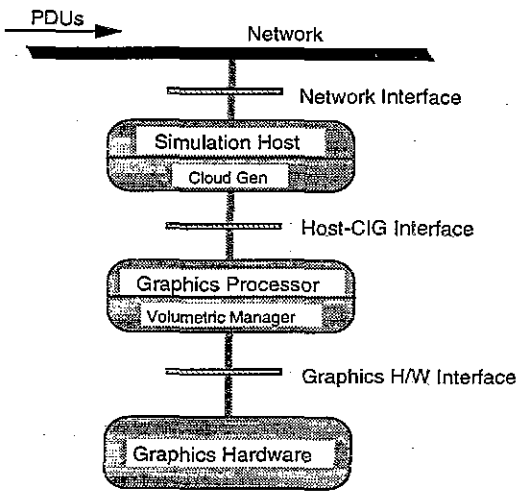

Figure 3 Logical view of DIS node processing

data in shared memory for subsequent processing. With each frame, the Volumetric Primitive Manager performs conversion, integration, and sorting of the volumetric cloud data with other synchronous model primitives prior to developing graphic element calls for visualization. This must be done to accommodate the graphics requirement to sort all transparent objects and render them back to front. (This is specific to the SGI Onyx Graphics Workstation.) This basic architecture, with the functionality to allow concurrent synchronous and asynchronous processes, should allow future incorporation of new models requiring volumetric simulation and visualization.

Figure 4 shows the entire data flow through our DIS-compatible cloud visualization system (the system also simulates smoke plumes, but we focus on cloud simulation in this paper). Data flow begins with PDUs, continues onto cloud modeling and the volumetric primitive handling which generates the hardware specific graphics calls. Each of the major components of the system is described in the following subsections.

Cloud Generator (CG). The CG executes under the control of the simulation host and is built around the Cloud Scene Simulation Model. The CG uses input cloud parameters specified in an environmental *PDU*, sounding data, and other parameters which depend on the type of simulation host (e.g., ground or air vehicle). It generates a temporal sequence of three-dimensional cloud fields as output. Interpolation between any two sequential cloud fields in the CIG results in smooth translation from one time to the next, and therefore results in smooth cloud motion and evolution.

Note: An alternate approach (not implemented for this effort) is for the CG to execute on a server and pass the cloud model data to individual simulators over the network. This approach reduces the local computation at nodes at the expense of increased network traffic.

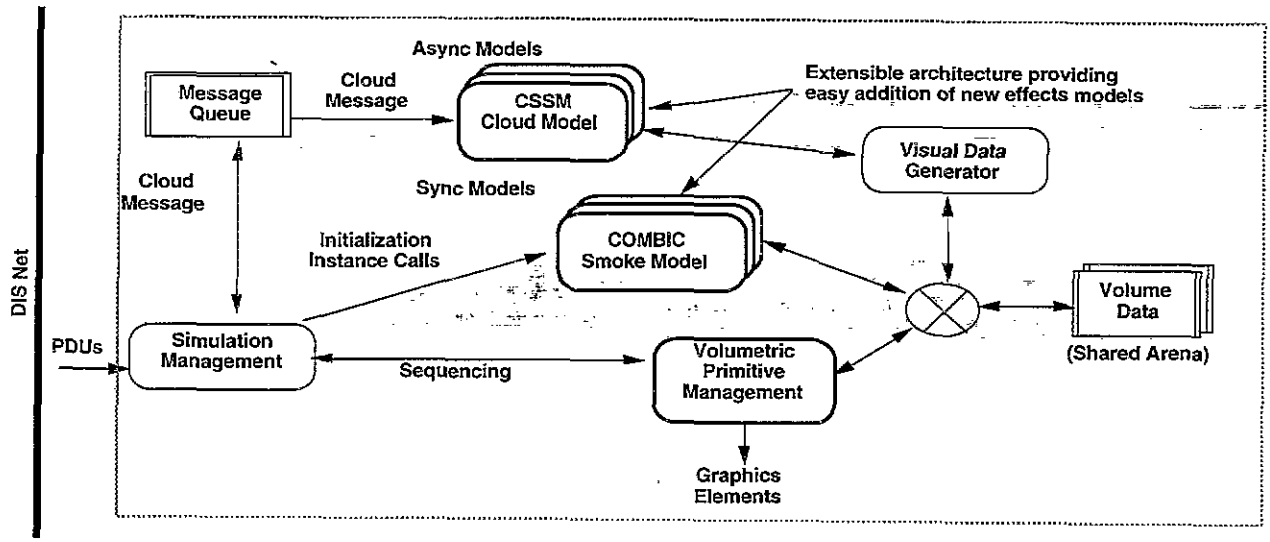

Figure 4 System data flow with integrated cloud simulation and visualization

Visual Data Generator (VDG). This module uses two sequential cloud fields to create the required data structures for the visualization of the clouds. It performs the following functions:

1) Calculate interpolation data for each primitive position, opacity, and dimension to provide the volumetric manager the capability to smoothly interpolate at up to 30 frames per second for smooth visualization.
2) Add random perturbations in position, dimension, and density of the graphics primitives used to represent the gridded cloud field. This produces a more realistic rendering of an otherwise regular Cartesian volume grid. (The number seed used to initiate the random perturbations is stored in the environmental PDU to ensure that all DIS nodes calculate the perturbations in an identical fashion so primitives are fully correlated across DIS applications.)
3) Set the primitive ambient and diffuse lighting components based on the relationship of each primitive with other primitives and the global illumination vector. This allows for self shadowing of the cloud primitives.

Volumetric Manager (VM). The VM executes on the CIG under the control of the graphics processor. The VM manages the loading of volume data and interpolation of the current field between two time periods. The VM then sorts the data primitives back to front to allow for graphics processing.

Network Data Interface. An entity on the network, known as the environmental manager, is responsible for controlling the overall environment during a DIS exercise. Periodically, the environmental manager issues an environmental PDU specifying cloud model parameters. These parameters include cloud type, fractional coverage, base and top layer heights, along with the simulation domain extent and resolution, etc. (See Model Input section presented previously.)

Graphics Hardware Data Interface. The cloud data primitives are displayed as two-dimensional textured or vertex-colored polygons with variable size, opacity, and intensity. The CIG's graphics library calls define the interface between the graphics processor and the graphics hardware. These calls are platform dependent.

GRAPHICS RENDERING PROCESS

One of the critical components to this effort was to find an efficient approach to rendering volumetric phenomena on off-the-shelf graphics workstations or image generators. Current DIS software packages are good at representing surfaces, but not very good at representing fuzzy volumetric phenomena. Our challenge was to come up with a *polygonal method* to approximate volumetric elements required for the cloud simulation and visualization. That method is described in the following subsections, where we begin with a discussion of the method for approximating the overall cloud structure and follow with an outline of the lower level graphics primitive implementation.

Cloud Structure

The requirements for the high level cloud geometry structure are: 1) it must be derived from the CSSM software 2) it must allow interoperable visibility between any two vantage points to ensure a fair fight among DIS users, and 3) it must accommodate varying levels of data resolution to support varying levels of fidelity graphics hardware and database configurations.

There were two basic choices for the graphics geometry representation. The first involves use of a complex polygonal hull to describe the cloud shape or structure. This method was deemed unresponsive to the requirement for interoperability. Although providing realistic surface representation while outside the cloud structure, a polygonal hull exhibits improper intervisibility when two vehicles are viewing each other from within the cloud. The second method (selected for implementation) is described in the following section.

Geometric Primitive Aggregation. This approach groups many small graphics primitives together to provide the total cloud structure. For example, a large number of small spheres, each with their own individual attribute settings, could be used to form a complex cloud structure.

Advantages-- The primary advantage of such a method is that the overall look can be very accurate with many small primitives. Each primitive, with its own attribute settings, can accurately model shape, lighting, and density variations throughout the cloud domain. In addition, it can be accomplished on today's

graphics workstations such as the SGI. The resolution of the primitives can be adjusted for varying levels of fidelity and processing speeds. The viewing interoperability requirement is satisfied with this technique.

Disadvantages-- The primary disadvantage of this method is that at high primitive density, a large number of polygons is required. When looking through many layers of primitives, the pixel processing load can become excessive. Hence, some form of pixel processing control is required. We plan to implement a simple level-of-detail control to handle this problem in the future.

A high-level schematic of the rendering process is presented in Figure 5. That figure shows the structure of the cloud field throughout the visualization process. As shown in the upper left, the CSSM generates a three-dimensional regular grid of LWC data. These data are converted to spherical visual primitives for graphical representation with associated radius, location, opacity, and lighting characteristics. Random perturbations to the location, radius, and opacity of the primitives are then imposed for additional realism. Self shadowing is later added.

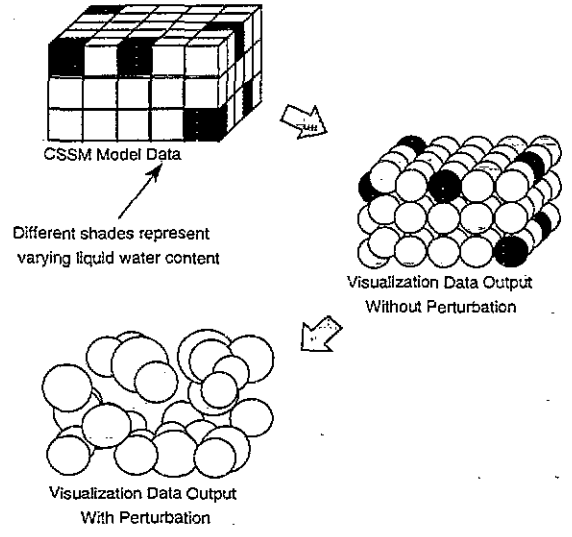

Figure 5 Cloud model data structures within the visualization process

Low-level Graphics Primitives

Under this effort, we developed a method of aggregating many simple low-level primitives to perform a planar-wise approximation of volumetric phenomena. The previous section discusses that approach. This section describes the attributes of the low-level primitives that we selected. The primary requirements that drove our selection were: 1) real-time processing speeds, 2) realistic visual representation, and 3) minimal intervisibility problems.

Volumetric Primitive Attributes. We considered several different primitive types in our implementation. Some of these are shown in Figure 6. Each of the base geometric primitives includes attributes to allow realistic visual processing. They include color or texture pattern, opacity, shading attributes (flat or smooth), and material type to support sensor visualization. We selected angle-oriented disks and polygons for our implementation as they both consist of the fewest number of polygons and produce a realistic visual effect.

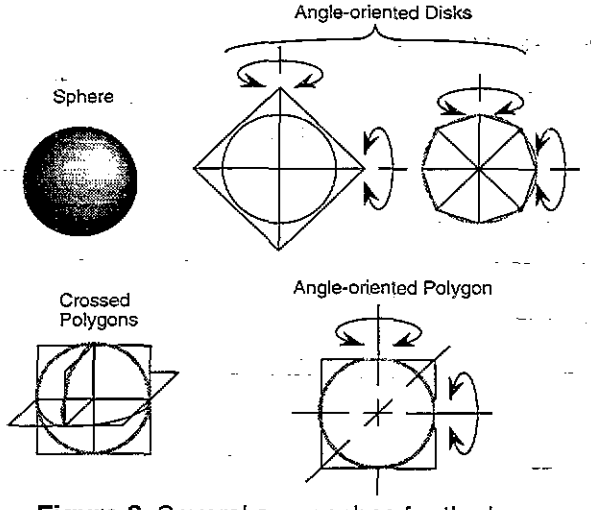

Figure 6 Several approaches for the base volumetric primitive geometry for cloud visualization

Self Shadowing. The shading parameters for each cloud primitive are calculated as a function of the number of cloud primitives between it and a light source and the density of the cloud field along that path. This results in variable shading across the cloud field which produces a more realistic visualization.

RESULTS

The two photo plates (Figs. 7 and 8) show output from the cloud visualization system. Both scenes show cumulus cloud layers from different viewpoints: an airplane view (Fig. 7) and a low-flying helicopter view (Fig. 8). Both scenes were generated in real time on a Silicon Graphics Onyx deskside workstation.

SUMMARY

In this paper we describe an ongoing effort to develop a high-fidelity cloud visualization system to provide physically-based dynamic environmental effects in DIS applications.

Under this effort with Phillips Laboratory and TEC, the CSSM software has been successfully integrated into a networked simulation software package. In addition, a new volumetric primitive management method has been developed to visualize dynamic cloud entities in real time. (The overall architecture has also been extended to simulate other physics-based models such as smoke plumes. See Ref. 6.)

The resulting DIS visualization architecture will be used in future dynamic environment modeling experiments to support a variety of simulation efforts such as upgrades to the Modular Semi-Automated Forces (ModSAF) simulation. These ModSAF upgrades will include simulating rain, dust clouds, smoke plumes, and other environmental weather conditions along with clouds to determine their effects on automated forces behavior. We believe that the basic modular architecture described in this paper will be extendible to those experiments.

ACKNOWLEDGMENTS

Special thanks to Jeff Turner, Chris Moscoso and George Lukes, of the US Army TEC, and Don Grantham of the US Air Force Phillips Laboratory who supported, advised, and sponsored this DIS integration and development effort.

REFERENCES

[1] Cianciolo, M.E., and R.G. Rasmussen, "Cloud Scene Simulation Modeling - The Enhanced Model," Technical Report, PL-TR-92-2106, April 1992.

[2] Saupe, D., "Point Evaluation of Multi-Variable Random Fractals," Visualisierung in Mathematik and Naturwissenschaft, H. Jurgens and D. Saupe, Eds., Springer-Verlag, Heidelberg, 1989.

[3] Beverina, A.F., and J.R. White, "An Alternative Environment PDU for Initiation, Control, and Hand-off of Embedded Processes," 10th DIS Workshop, March, 1994.

[4] Lovejoy, S., "Area-Perimeter Relation for Rain and Cloud Areas," Science, 216, 185-187, 1982.

[5] Cahalan, R.F., and J.H. Joseph, "Fractal Statistics of Cloud Fields," Monthly Weather Review, 117, 261-272, 1989.

[6] Soderberg, B.T., and J. Gardner, "Dynamic Environment Modeling in Distributed Interactive Simulation - Final Report," June, 1994.

Figure 7 Real-time cloud scene generated with the CSSM and volumetric processing software (view from above a cumulus cloud layer looking down toward an airport runway)

Figure 8 Cumulus cloud layer with cloud shadow processing in a dusk environment. In addition, physics-based smoke plume modeling is added using the same volumetric processing technique

SIZE	FIELD TYPE	DATA	EXPLANATION
96	PDU Header	Protocol Version - 8-bit enumeration *Exercise ID - 8-bit unsigned integer* PDU Type - 8-bit enumeration Padding - 8-bits unused Time Stamp - 32 bit unsigned integer Length - 16-bit unsigned integer Padding - 16 bits unused	PDU Header is same format as Version 2.0.3 Entity State PDU (ESPDU)
64	Environmental Entity ID	Site - 16-bit unsigned integer Application - 16-bit unsigned integer entity - 16-bit unsigned integer Padding - 16 bits unused	ID is same format as ESPDU
64	Environmental Entity Type	Domain - 8-bit enumeration Kind - 8-bit enumeration Country - 8-bit enumeration Subcategory - 8-bit enumeration Specific - 8-bit enumeration Extra - 8-bit enumeration Padding - 16 bits unused	This type record is unique to the Environment PDU. Specific enumeration for the record is being worked in the Atmospheric Subgroup of the Synthetic Environmental Working Group
192	Location in World	X-Component - 64-bit floating point Y-Component - 64-bit floating point Z-Component - 64-bit floating point	Same as ESPDU
96	Linear Velocity	X-Component - 32-bit floating point Y-Component - 32-bit floating point Z-Component - 32-bit floating point	Same as ESPDU
96	Entity Orientation	Psi - 32-bit floating point Theta - 32-bit floating point Phi - 32-bit floating point	Same as ESPDU
32	Appearance	32-bit record of enumerations	Same as ESPDU
320	Dead Reckoning Parameters	Dead Reckoning Algorithm - 8-bit enumeration Other Parameters - 120 bits unused Entity Linear Accel - 3*32-bit floating point Entity Angular Velocity - 3*32-bit integers	Use Standard dead reckoning algorithms plus new TBD algorithms for embedded process
32	PDU extent	*Number of Parameters - 16-bit unsigned integer* Size of each parameter - 16-bit unsigned integer	The following allows the user to define parameters for specific entities
96	Weather Information	Number of Sounding Levels - 16-bit unsigned int ID of Sounding Data - 16-bit unsigned integer Random Seed - 32-bit unsigned integer	Sounding data specifies pressure, temperature, dewpoint temperature, and wind as a function of altitude across domain
128	Cloud Temporal Information	Number of Output Periods - 32-big floating point Temporal Resolution - 32-bit floating point Starting Time - 64-bit unsigned integer	Starting time gives the seconds since midnight of Jan. 1, 1970
32	Cloud Layer Information	Number of Layers - 16-bit unsigned integer Padding - 16 bits unused	This number includes cumulus and non-cumulus cloud layers
nx128	Cloud Layer Parameters	Fractional Coverage - 32-bit floating point Base Height - 32-bit floating point Top Height - 32-bit floating point Cloud Type - 8-bit enumeration Padding - 24 bits unused	Cloud layer parameters describe the thickness, cloud type and fractional amount of each layer. At most 4 layers are allowed in the current cloud model

Table 1 Cloud Entity State PDU Definition