NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

NPSNET: ENVIRONMENTAL EFFECTS FOR A REAL-TIME VIRTUAL WORLD BATTLEFIELD SIMULATOR

by

Daniel P. Corbin

September 1993

Thesis Advisor: Michael J. Zyda
Co - Advisor: David R. Pratt

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	93 September 23	Master's Thesis From 6/91 to 09/93

4. TITLE AND SUBTITLE
NPSNET: Environmental Effects For A Real-Time Virtual World Battlefield Simulator

5. FUNDING NUMBERS

6. AUTHOR(S)
Corbin, Daniel Patrick

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Computer Science Dept.
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/ MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

10. SPONSORING/ MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release;
distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Computer Science Department at the Naval Postgraduate School in Monterey, California has developed a low-cost real-time interactive simulation system using the Distributed Interactive Simulation (DIS) Protocol, known as NPSNET, that works on commercially available Silicon Graphics IRIS workstations.

The DIS protocol has provisions for an environmental effects Protocol Data Unit (PDU), but effects of a changing environment have not been implemented to use it. Furthermore, this lack of environmental effects reduces the realism of the simulations, such as NPSNET, that use the DIS protocol. The challenge in implementing environmental effects such as smoke, dust and the passage of time is to develop a model that users perceive as realistic, but is computationally cheap enough to be used in real-time applications. It is the lack of environmental effects, usable in interactive simulations, that we attempt to solve.

This thesis focuses on creating a library of visually realistic environmental effects for NPSNET that includes smoke, flames, clouds, lightning, the passage of time and night observation devices. The algorithms were initially derived from physical models, but were found to be too computationally intensive to be used in a real-time application. Thus, it was necessary to simplify the model by depending mainly on visual realism over physical models in creating the effects presented here. The result is a library of environmental effects that are both "visually accurate" and usable in real-time applications.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Graphics, Smoke, Fire, Night Observation Device, Clouds, Environmental Effects	88
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Approved for public release; distribution is unlimited

NPSNET: ENVIRONMENTAL EFFECTS FOR A REAL-TIME VIRTUAL WORLD BATTLEFIELD SIMULATOR

by

Daniel P. Corbin
Lieutenant, United States Navy
B.S.Mathematics, University of Florida, 1985

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1993

Author: 
Daniel P. Corbin

Approved By: 
Dr. Michael J. Zyda, Thesis Advisor


Dr. David R. Pratt, Thesis Co-Advisor


Dr. Ted Lewis, Chairman,
Department of Computer Science

ABSTRACT

The Computer Science Department at the Naval Postgraduate School in Monterey, California has developed a low-cost real-time interactive simulation system using the Distributed Interactive Simulation (DIS) Protocol, known as NPSNET, that works on commercially available Silicon Graphics IRIS workstations.

The DIS protocol has provisions for an environmental effects Protocol Data Unit (PDU), but effects of a changing environment have not been implemented to use it. Furthermore, this lack of environmental effects reduces the realism of the simulations, such as NPSNET, that use the DIS protocol. The challenge in implementing environmental effects such as smoke, dust and the passage of time is to develop a model that users perceive as realistic, but is computationally cheap enough to be used in real-time applications. It is the lack of environmental effects, usable in interactive simulations, that we attempt to solve.

This thesis focuses on creating a library of visually realistic environmental effects for NPSNET that includes smoke, flames, clouds, lightning, the passage of time and night observation devices. The algorithms were initially derived from physical models, but were found to be too computationally intensive to be used in a real-time application. Thus, it was necessary to simplify the model by depending mainly on visual realism over physical models in creating the effects presented here. The result is a library of environmental effects that are both "visually accurate" and usable in real-time applications.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND

Current trends indicate decreases in funding to the Department of Defense (DoD) will continue and as the money available decreases, the need for alternative and less expensive forms of training and prototyping will be on the rise. As all available resources decrease, simulators can serve the purpose of testing tactics and doctrine prior to their actual use in the field. Once developed in the virtual world (VW) they can be practiced and refined in the field [PRAT93].

In order to properly test any doctrine for validity, attributes and characteristics of the real-world environment to which the doctrine apply must be addressed in the simulator. It is often desirable to use simple computationally inexpensive objects which give the viewer a representation of the item. This use of simple icons to represent more complex objects is referred to in [THOR87] as the "70% solution" and suggests that a participant given the most important cues of a scene can be expected to imagine the remaining details [PRAT93]. However, under certain situations, such as billowing smoke in the distance or dust seen on the horizon, the level of detail can determine the amount of information provided. In such a case, a simple icon representation is not sufficient. Here deviation from realism can cause improper actions on part of the player resulting in "negative training". A simulator which utilizes "realistic" obscurant will require the participant to expend resources determining what is burning or how many tanks are approaching a position.

Previous research at the Naval Postgraduate School has resulted in the development of a real-time VW combat simulator known as NPSNET [ZYDA93]. The main objective of the NPSNET is to provide a realistic real-time combat simulator that can be executed on "inexpensive" graphic workstations such as the Silicon Graphics IRIS 4D family of machines. It also communicates with numerous other stations using standard message formats such as the U.S. Army's Simulation Networking (SIMNET) [DARP89] system or the Distributed Interactive Simulation (DIS) [ISTB93] Protocol Data Unit (PDU) formats

[PRAT93][ZYDA93]. Research is ongoing to develop a fully interactive, believable environment while maintaining the ability to execute the simulator on the low cost workstations [ZYDA93].

B. OBJECTIVES

The objective of this research is to develop a library that provides smoke, fire, clouds, vehicle dust trails and the passage of time that can be implemented into a real-time distributed VW. This library of functions called the Environmental Effects Library (EEL) is implemented for use in NPSNET running on the Silicon Graphics IRIS 4D family of workstations using the DIS protocols.

To meet the objectives, the following list of assumptions were utilized (1) An algorithm is considered "real-time" if six frames per second is maintained. The increase in technology will allow speedup to an acceptable rate. (2) Realism of the effects is based on a simplified physical model. While this assumption provides sufficient effects for human training, it prevents its use for engineering purposes because the models have been simplified to meet the real-time objectives.

C. ORGANIZATION

The previous sections of this chapter have stated the objectives and reasoning behind providing environmental effects in the NPSNET. Chapter II reviews the previous work accomplished in this area of research while Chapter III discusses the algorithms used to generate the environmental effects including the smoke generator, cloud generator, dust generator and lightning models while Chapter IV discusses night observation devices (NOD) and passage of time models. The implementation details and performance characteristics of the algorithms are discussed in Chapter V. The networking method is discussed in Chapter VI. Chapter VII discusses limitations of the EEL. Chapter VIII describes future extensions, research topics and concludes with conclusions and a summary of the research conducted. The Appendix contains manual pages to explain the use of the Environmental Effects Library (EEL).

II. ENVIRONMENTAL RESEARCH

The environmental obscurant models presented here are based on the models and research described by Reeves in [REEV83] and Gardner in [GARD85][GARD92]. Those models are based on physically based equations that are so computationally complex that computation and rendering times are too long to be used in a real-time system. Many simplifications were used to reduce the computations of the models to allow their use in real-time applications.

A. PARTICLE SYSTEMS

Modeling objects such as clouds, smoke and water that do not have well defined shapes and surfaces using traditional graphic techniques is difficult since the most commonly used primitives consists of polygons, patches and surfaces. The surfaces of the fuzzy objects are irregular, ill-defined with dynamic surfaces, so traditional primitives are difficult to use. It is the dynamic and fluid property of the fuzzy systems that is desirable and should be preserved in the rendered object.

Reeves presents a method for modeling fuzzy objects using a technique called particle systems in [REEV83]. A particle system differs from an object represented using traditional image synthesis techniques. First, the object is represented as a cloud of primitive particles that define its volume rather than by a set of surface primitives. Second, the particles of the system are not static but change form, move, are born and die over time. Third, once the basic shape is determined using a procedural algorithm, the primitives are randomly placed in the volume defined by this shape resulting in an image with the desired dynamic and fluid properties.

The particle system has several advantages over surface-oriented techniques. The first being that a particle is much simpler to represent than a surface object because orientation is normally not a major concern. Hence the computation time for each primitive is reduced allowing objects made of more primitives to be rendered in the same time period. Second,

the model definition for translation of the particles is procedural, usually based on a physical law, controlled using random numbers, allowing the modeler to adjust the level of detail to fit the specific situation. Third, the primitives of the object are dynamic which allow then to change shape over time, an attribute that surface-based modeling has only limited capabilities to accomplish.

The basis of the particle system is that each primitive is given its own attributes. The attributes can control the age, color, size, lifetime and position. The individual characteristics allow the primitives to be controlled independent of the surrounding neighbors that make up the object. Over a period of time, particles are generated into the system, move, change attributes and then eventually die off. The steps are continued as long as the object exist.

B. OBSCURANTS

Geoffrey Gardner discusses a method to generate realistic smoke and clouds in [GARD92] and [GARD85]. The models he presents uses ellipsoids that are covered using a texture derived as a function of the transmittance of the obscurant.

A plume of obscurant is created by positioning five smoke columns using the footprint shown in Figure 1. Within the footprint, 13 individual subplumes generate puffs to define the shape of the smoke plume. The obscurant is produced by generating a series of textured ellipsoids that are translated as a function of the ambient wind conditions. The ellipsoids when generated, become part of a smoke column. The center column's puffs are translated along a path defined by the movement of the ambient air. The remaining columns are translated to fill the trajectory envelope defined using the Gaussian distribution of the puff mass.

The transmittance or transparency of each obscurant puff is calculated using (Eq 1). This is accomplished by first determining the transmittance through the center of the ellipsoid. A Gaussian variation is then applied to this value to increase the transmittance of the ellipsoid near the edges of the ellipsoid. Statistical variations in mass concentration are

simulated by applying a fractal function which modulates the transmittance over the ellipsoid puff. Lastly the total transmittance at each pixel is calculated by combining the transmittance through all ellipsoids covering the pixel.

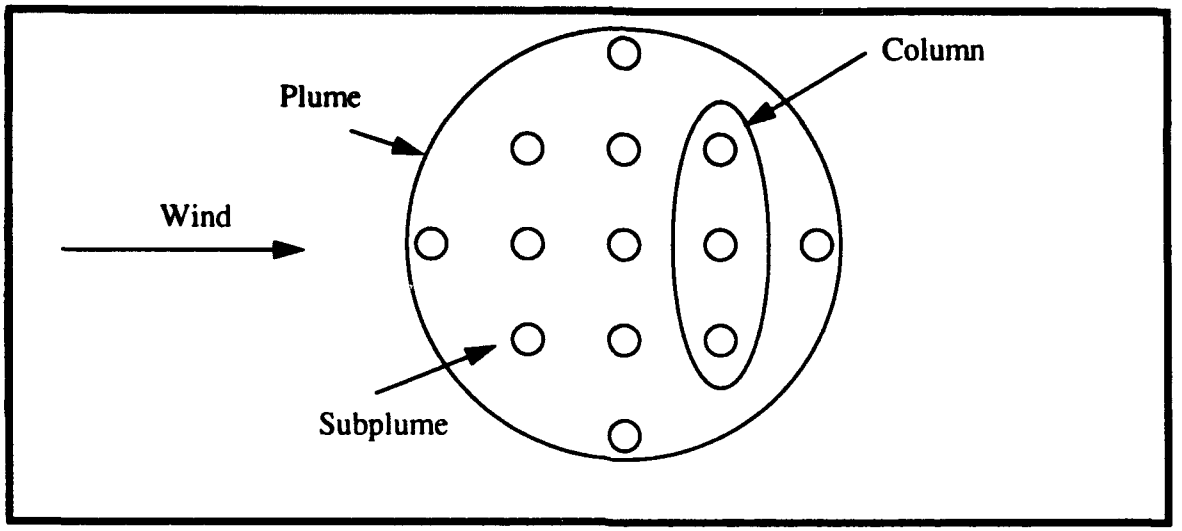

Figure 1. COMBIC Smoke Plume Footprint

$$T = e^{(-\alpha\lambda) \times \int C(X, Y, Z)\, dl} \qquad \text{(Eq 1)}$$

Where:

T: Transmittance of the obscurant.

α: The extinction coefficient of the obscurant.

λ: The wavelength of the sensor.

$\int C(X, Y, Z)\, dl$: The integral of the obscurant mass over the path length l.

X: The downwind direction of the cloud along the line of sight.

Y: The crosswind direction of the cloud along the line of sight.

Z: The vertical direction of the cloud along the line of sight.

The model represents the obscurant cloud using from one to five subclouds that are then positioned using a Gaussian distribution of the mass to form a Gaussian plume. The Gaussian ellipsoid shown in Figure 2 has a Gaussian distribution that can be used to represent the obscurant puff.

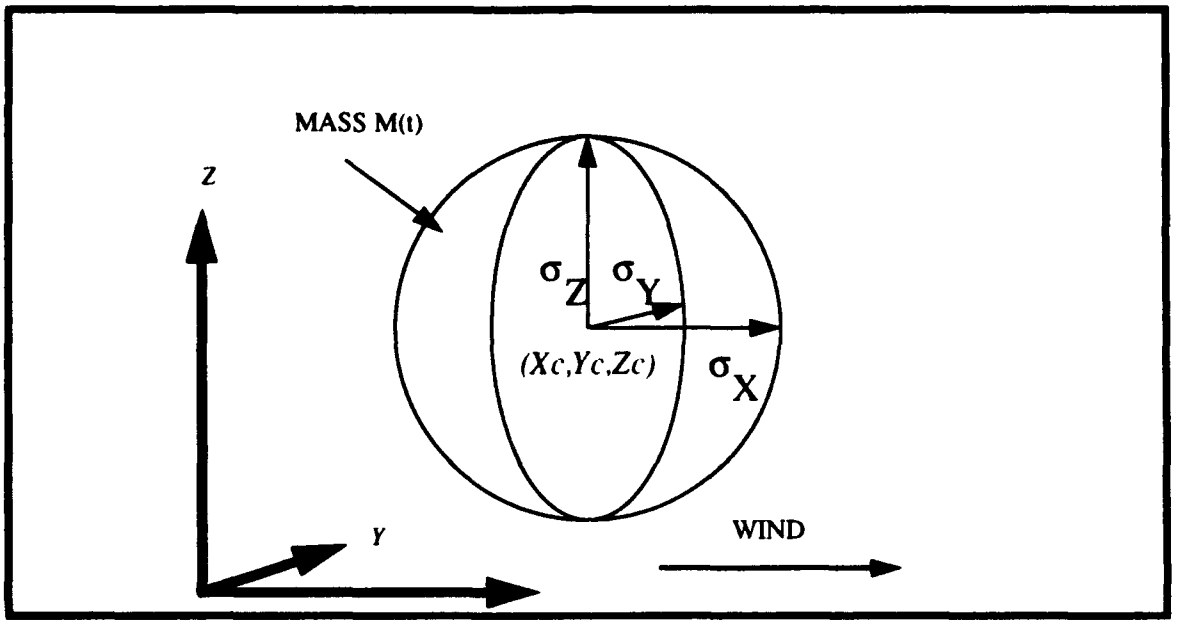

Figure 2. Gaussian Puff Geometry Used By Gardner In [GARD92]

The Gaussian variation of mass concentration is given by (Eq 2), (Eq 3) and (Eq 4). Where the variables are as shown in Figure 2. The generated Gaussian plume is an elongated envelope representing a continuous obscurant. A typical smoke plume is illustrated in Figure 3.

$$C = \left(\frac{(M(t))}{(2\pi)^{3/2} \times \sigma_X \times \sigma_Y \times \sigma_Z} \right) e^{\frac{-1}{2} \times D^2} \qquad \text{(Eq 2)}$$

$$D^2 = \left(\frac{X - X_c}{\sigma_X} \right)^2 + \left(\frac{Y - Y_c}{\sigma_Y} \right)^2 + \left(\frac{Z - Z_c}{\sigma_z} \right)^2 \qquad \text{(Eq 3)}$$

$$M(t) = M_o (f_d + (1 - f_d) \times e^{-\delta t}) \qquad \text{(Eq 4)}$$

Where:

M_o: The original mass.

f_d: The fraction of long term mass.

d: A scavenging coefficient defining the rate of mass decrease.

Gardner's model is conducted in two phases. The first phase computes a time history of the obscurant trajectory, mass concentration and mass production. The trajectory is used to represent the geometric envelope in terms of a downwind coordinate system. For any puff, the history contains the X and Z positions[1] and X, Y and Z dimensions for any specific time. The leading edge of the plume is defined by this time history.

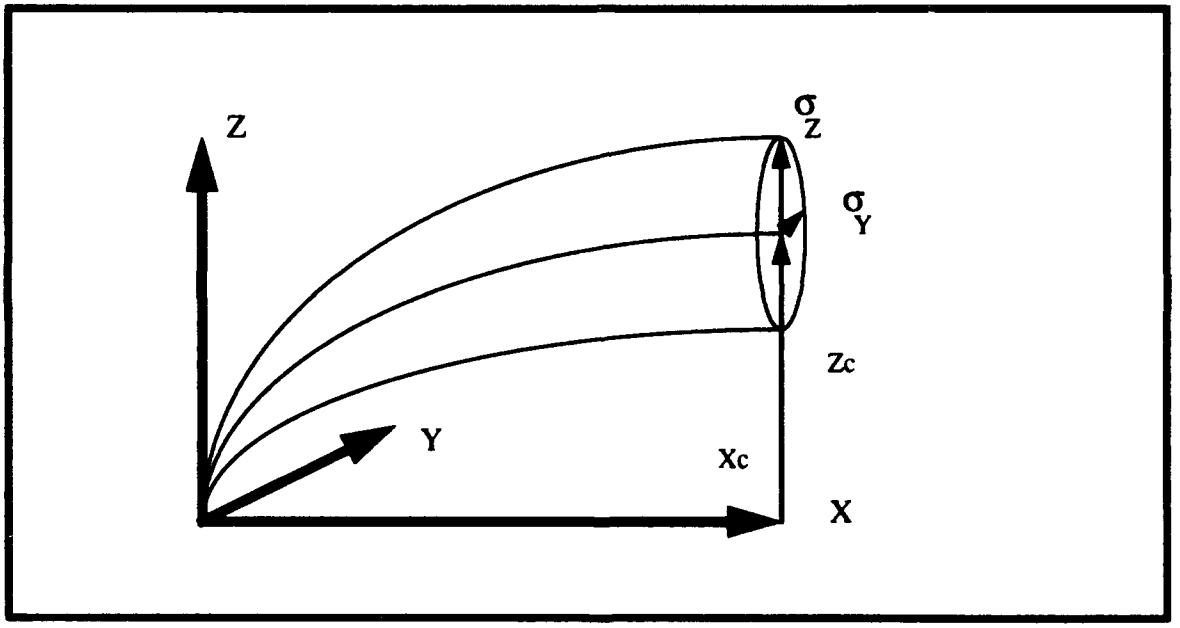

Figure 3. Gaussian Plume Geometry Used In [GARD92]

After the position and dimension of the leading edge have been determined the remainder of the plume is backfilled with scaled puffs to yield a continuous envelope of obscurant. The backfill is accomplished by spacing the filler puffs to minimize the amount of overlap between puffs.

1. Y is always zero because the plume coordinate system is setup so the X axis is oriented directly downwind.

The time history data used to determine the position of the puffs is computed in the first phase. In the second phase, after the position of each of the ellipsoids has been determined, the transmittance through a given line of sight is computed using (Eq 1).

Gardner's cloud model uses textured ellipsoids in the same fashion as the smoke model. The method of positioning the ellipsoids in the scene, however, is different. When generating a cloud formation, ellipsoids are grouped and positioned depending on the class of cloud being modeled.

The method presented by Gardner produces extremely realistic obscurant clouds however, the numerical computations are so extensive rendering frequency is measured in minutes per frame. We present a model that uses a plume made up of five columns of continuous puff primitives translated along a centerline. The columns are situated similarly to the foot print shown in Figure 1. but with five instead of 13 subplumes. Furthermore, the method we present is based on a particle system so each puff primitive is translated as a function of its own attributes. Using this method there is no need to backfill to maintain a continuous plume. Since our model uses an existing image file as a texture, there is no need to calculate the transmittance. The simplifications we made allows for an algorithm that can be used in real-time applications. The final obscurant cloud is not of as high quality as Gardner's model, but it provides for real-time obscurant usable on hardware currently available.

C. TIME OF DAY

Klassen [KLAS87] presents a method of determining sky color by approximating the atmosphere as multiple layers of parallel planes each containing a partial atmosphere of uniform density. His method calculates the reduction in light intensity caused by Rayleigh particle scattering. He then determines the color of each pixel in the scene by applying the change in light intensity to a precalculated sky color. The computation time using this method on a VAX 11/780 is approximately one tenth of a second per pixel for a 512 by 256 image.

Nishita [NISH93] describes a method to determine the color of the Earth's atmosphere as a function of position of the observer and elevation of the Sun. He discusses a spherical shell atmosphere that provides an accurate model of the sky's color. The model calculates the amount of color change caused by Rayleigh and Mia scattering of the incoming light. The time to calculate sky color for a 500 x 490 image ranged from just under four minutes to about 12 minutes per frame using an SGI IRIS Elan[NISH93].

Both models provide for accurate representation of the sky color. However because of the large number of calculations that must be performed to determine the color of the entire sky, the methods presented are not usable in real-time applications. As Klassen states in [KLAS87], there is no accurate method known to date that is able to produce a background sky quickly. Hence it is up to user to develop an algorithm best suited for the task at hand.

D. PREVIOUS RESEARCH SUMMARY

The methods presented here by Gardner and Nishita, although providing high quality images of high physical precision, cannot provide a model that is usable in real-time applications because of the number of calculations that must be performed during each cycle. We are willing to concede some of the realism their models provide if our simpler versions render images that are nearly what the participant of the simulation expects to see under the given situations.

III. ENVIRONMENTAL OBSCURANTS

The latest version of the NPSNET has a limited capability of providing obscurants such as smoke and flames. It does, however, provide fog and simple icons that can be used to give the participant a visual cue that there has been an event that causes smoke and flames to occur.

There are times when the use of a simple icon is adequate. The times include when computation time is so high that to compute the complex object properties prevents real-time display or when the object's significance is so small that rendering the object is not justified. However times do exist when expending resources to render a complex object is justified. This can occur when the presences of an object such as the smoke of a burning building in the distance or the generated dust cloud of a column of vehicles approaching from the horizon causes actions to be taken by the participant.

This chapter discusses briefly the underlying theory of the models used in the smoke, fire and vehicle dust trail algorithms. For a complete discussion on the topics of gas and smoke rise and dispersion see [BEYC79] and [BRIG69].

A. OBSCURANT PLUMES

1. Behavior Of Smoke Plumes Defined By Briggs' Equations.

Plume dispersion is most easily described by discussing separately the three aspects of plume behavior: (1) aerodynamic effects due to the presence of the source of the smoke; (2) rise relative to the mean motion of the air due to the buoyancy and initial vertical movement of the plume; and (3) diffusion due to turbulence in the air. In reality, all three effects occur simultaneously, but to simplify the model each is treated separately and are generally assumed not to interact. This simplification can be accomplished without losing the correctness of the model at the level of detail desired in our application.

Turbulence is created as gases and smoke are generated and as the shear wind force acts at the source of the plume. The turbulence causes a phenomena called

entrainment when the gases are mixed with the ambient air. The entrainment decreases the upward movement and buoyancy of the plume by changing the composition of the generated gases and as the entrainment continues, the plume increasingly takes on the characteristics of the surrounding air mass.

As the plume bends into the wind, it moves horizontally at nearly the wind speed of the entrained air, however it continues to rise relative to the position of ambient air. As the gases encounter ambient air above the plume, vigorous mixing occurs all across the top of the plume, causing the plume diameter to grow approximately linearly with the height of its rise [BRIG69].

If the plume gases have a temperature greater than that of the surrounding air or have a mean molecular weight less than ambient conditions then, because of their lesser density, they will be more buoyant. As the heat is lost, the total buoyant force in a given part of the plume will decrease causing the upward momentum of the segment to decrease until it eventually starts to lose height. It is important to note that as the puff moves downwind its relative velocity with respect to the wind will always approach zero because of the loss of momentum brought about through entrainment.

At some point downwind from the source of generation, the turbulence and vertical temperature gradient of the atmosphere begins to effect the rise of the plume significantly. If the atmosphere is homogeneously mixed, it is said to be neutral or adiabatic. In such situations, the temperature gradient is a constant at approximately 5.4 degrees Fahrenheit per 1000 feet. This is known as the adiabatic lapse rate. If the temperature lapse rate is less than the abiabatic lapse rate, the air is said to be stably stratified. Air lifted abiatically in a stably stratified environment tends to become cooler than the surrounding air and tends to sink after reaching some maximum height. If the temperature of the air actually increases with altitude, the air is quite stable with little air movement and is referred to as an inversion. This is the situation that exists in Los Angeles during the summer months when the smog layer becomes trapped in the valley.

If the temperature lapse rate of the atmosphere is greater than the adiabatic lapse rate, the air is said to be unstably stratified. Air lifted in such an environment becomes warmer than the surrounding air and all vertical motion tends to be amplified.

The potential temperature, t, is defined as the temperature that a sample of air would acquire if it were compressed adiabatically to standard pressure (1 atmosphere). The potential temperature can be used to measure the stability of the atmosphere as is shown in (Eq 5).

$$\frac{\delta t}{\delta z} = \frac{\delta T}{\delta z} + \Gamma \qquad \text{(Eq 5)}$$

Where:

T: Temperature of the surrounding air.

z: Altitude of the observer.

Γ: 9.8 degrees Celsius.

(Eq 5) shows the potential temperature gradient is positive for stable air, zero for neutral air and negative for unstable air. If the air is unstable, the buoyancy of the plume decays as it rises since the plume entrains air from below and carries it upward into regions of warmer ambient air. If the air is stable throughout the layer of rise, the plume eventually becomes negatively buoyant and settles back to the height where it has zero buoyance relative to the ambient air. If the atmosphere is neutral, the buoyancy of the plume remains constant in a given segment provided the plume does not experience loss of heavy particles or absorb significant amounts of heat. Since neutral stability is a sign of turbulence, there is increased activity in the plume resulting in increased air entrainment and hence there will be a rapid decrease in the upward movement of the plume.

The rise of a smoke plume is a function of the upward movement of the puff mass, while diffusion is an indication of the dispersion of the smoke particles in relation to the mean centerline of movement. The shape of the plume is a function of the traversal and dispersal as the plume ages. There have been numerous mathematical models proposed to

calculate the shape and resulting chemistry of a plume over time. We use the equations proposed by Briggs as our model, a thorough discussion of many of the other equations can be found in [BEYC79] and [BRIG69].

The Briggs' equations were derived by observing numerous smoke plumes under various atmospheric conditions and graphing the position of the plume over time and a method of best fit is used to develop an equation that defines the function of the smoke movement. All smoke plumes belong to one of several specific class of plumes depending on the conditions of the ambient environment under which they are generated. Within a class of plumes, the shape of a specific plume is found by using the dispersion coefficients as inputs to the appropriate plume equation. The dispersion coefficients are used in conjunction with the stability classes shown in Figure 4 to determine the movement of the plume over time.

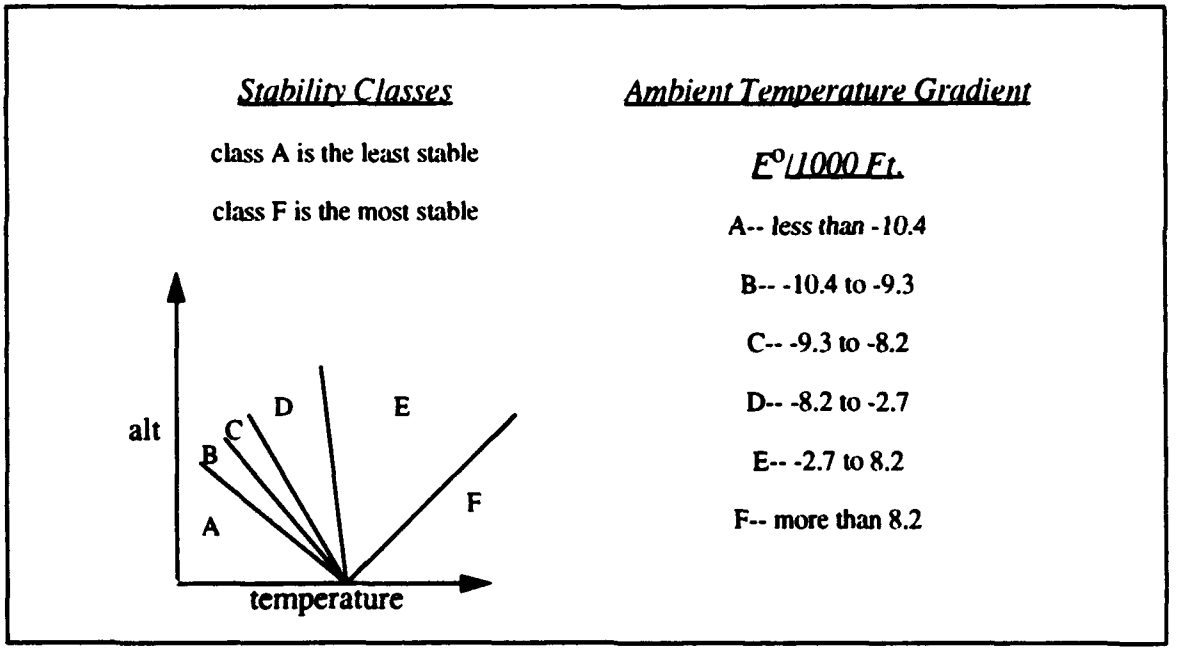

Figure 4. Description Of Stability Classes

Briggs divides plumes into four general categories based on the source of generation and the velocity of the prevailing winds. He considered the trajectory of cold

and hot plumes generated under calm and windy conditions, but we will consider only hot plumes generated under windy conditions here.

In general the Briggs equations for hot windy plumes are for bent-over, buoyant plumes and are based on observation data involving hot plumes from typical burning fossil fuels fires. Briggs equations assume the plumes are generated when draft velocities of the gases from the source are at velocities in the range of 6 - 30 meters per second and the exit temperatures are in the range 120 - 260 degrees Celsius.

The dispersion of a smoke plume is determined by statistical estimates derived using a Gaussian distribution [BEYC79]. The results of the estimates are used to determine dispersion coefficients that are in turn used in conjuction with the prevalent stability classes to determine the distance the plume translation differs from the plume centerline over time. One of the most accepted equation for determining the value of the dispersion coefficient σ was presented by McMullen in [BEYC79] and is presented in (Eq 6) where the values of I, J and K are taken from Table 1. The subscript y and z of Table 1 represent dispersion along the y and z axis in the coordinate system in which an increase in y indicates an increase in vertical movement and a change in z represents change in horizontal movement.

$$\sigma = exp(I + J \times \log(x) + K \times \log^2(x)) \qquad \text{(Eq 6)}$$

The values in Table 1 are based on data obtained in rural areas with open, level terrain. The effect of urban area or non flat terrain is an increased amount of turbulence on low altitude plumes. This has the same effect as if the plume is generated in an environment of less stable conditions. The dispersion values for urban environments can be found in [BEYC79].

The dispersion of the plume is directly related to the value the coefficients. Hence the resulting values of σ should increase as the stability of the surrounding atmosphere decreases.

On a clear night, the ground radiates heat which is lost to the atmosphere and in the process the air near the ground is cooled and an inversion is formed. A plume rising

through it quickly loses its buoyancy and levels off. The behavior of smoke generated under these conditions is called fanning and is illustrated in Figure 5a.

TABLE 1. RURAL DISPERSION CONSTANTS FOR USE WITH (Eq 6)

class	Iz	Jz	Kz	Iy	Jy	Ky
A	6.035	2.1097	0.2770	5.357	0.8828	-.0076
B	4.694	1.0629	0.0136	5.058	0.9024	-.0096
C	4.110	0.9201	-.0020	4.651	0.9181	-.0076
D	3.414	0.7371	-.0316	4.230	0.9222	-.0087
E	3.057	0.6794	-.0450	3.922	0.9222	-.0064
F	2.621	0.6564	-.0540	3.533	0.9181	-.0070

When the sun comes up, convection eddies are created and penetrate high into the atmosphere. When they reach the level at which the plume levels off, the eddies will cause the air to become mixed and fall toward the ground while the inversion aloft prevents upward diffusion. This phenomena is called fumigation and causes large amounts of the plume to be forced to the ground and is illustrated in Figure 5b. After an inversion is broken down by convection eddie currents, the atmosphere will be well mixed and neutrally stable and the plumes rise and diffuse in a smooth fashion known as coning which is shown in Figure 5c.

When diffusion is decreased and an inversion builds from the ground up, there is a ground inversion so weak that the plume can penetrate it and the plume diffuses upward but is prevented by the stability below from diffusing downward. This lofting causes the plume to remain high in the atmosphere for an extended period of time. Lofting is illustrated in Figure 5d.

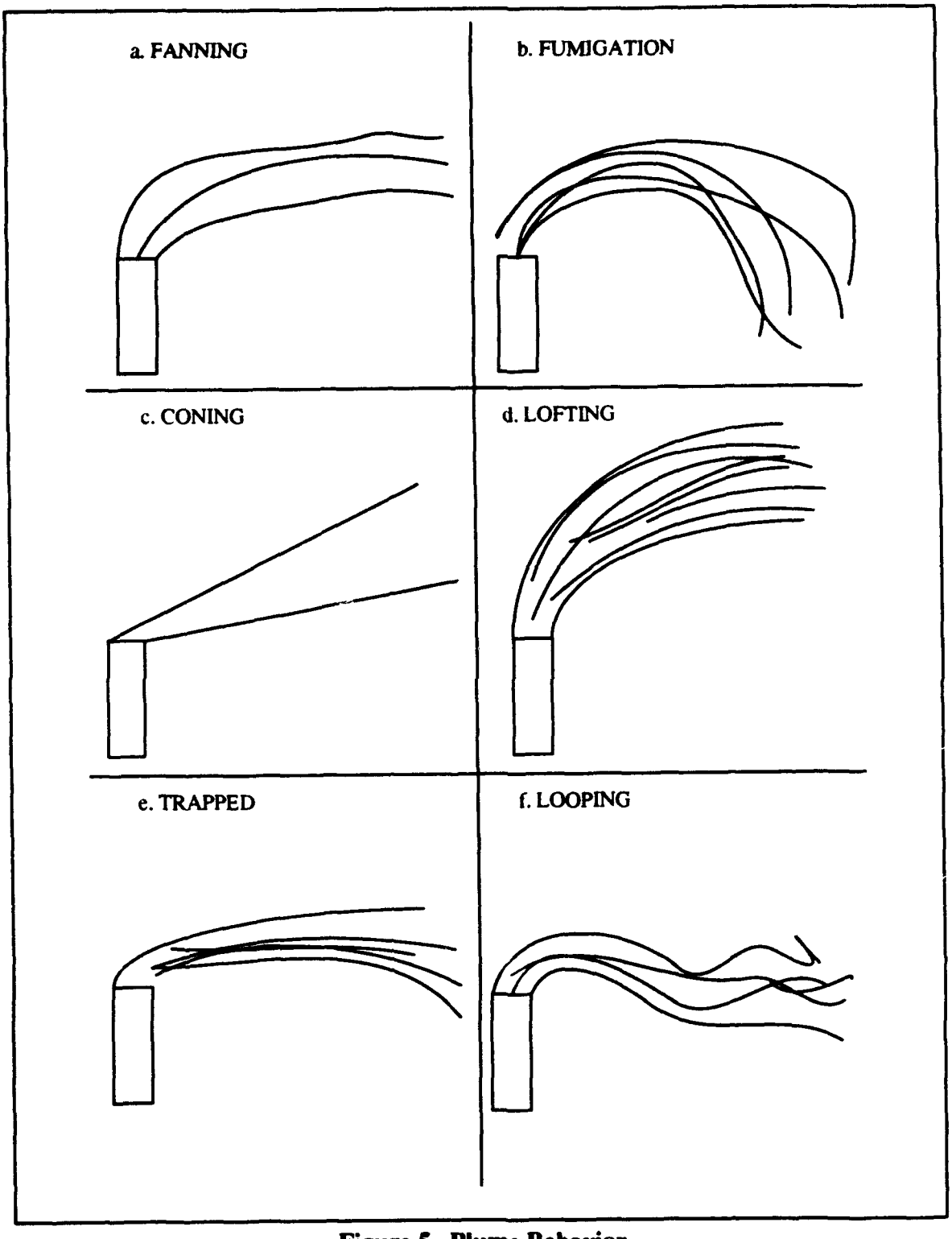

Figure 5. Plume Behavior

When an inversion layer exists above the plume, it is unable to ascend higher than the lower level of that air mass. Under this condition, a plume will be trapped as is illustrated in Figure 5e. As the heating of the ground intensifies, large convection eddies may develop and twist the plume in a looping manner which causes turbulence to the plume as shown in Figure 5f.

2. Control Of Particle Systems Using Newtonian Mechanics

According to Wejchert [WEJC91] models of natural phenomena are often too complex to be applied by an animator using traditional techniques because the numerical calculations are too intensive to be used for real-time systems. However simplifying the model by disregarding factors that are insignificant to the level of detail desired in the implementation can lead to applications that are usable in real-time graphic programs.

To determine the shape of the obscurant plume, we first determine the total force on the obscurant. We can integrate the acceleration to determine the velocity and likewise the velocity to calculate the position of each of the obscurant puffs.

(Eq 7) can be used to determine the force on an object when acted upon by a fluid and (Eq 8) is used to find the total force on an object by summing all the forces acting on that object.

$$F_{fluid} = \rho A \times v^2 \qquad \text{(Eq 7)}$$

$$F_{total} = \sum_i Fi \qquad \text{(Eq 8)}$$

$$a(time) = (F_{total})/m \qquad \text{(Eq 9)}$$

where:

F_{fluid}: The force of the fluid acting on the body.

F_{total}: The total force of the object.

ρ: The density of the fluid the object is immersed in.

A: The area of the body the fluid is acting upon.

v: The relative velocity between the object and the fluid.

m: The mass of the object.

a: The acceleration of the object.

Once the total force acting on the object is known, we can integrate (Eq 9) with respect to time using Euler's numerical method to calculate the velocity and position of the object over time.

Euler's method is a simplistic numerical method but does have the draw-backs that it is fairly inaccurate, can be unstable and is often inefficient because the time step must remain small to maintain its accuracy. These problems have little impact on our application since our time step is driven by the time through the graphics loop is relatively small compared to the speed of the objects. The inaccuracies do, however, provide the benefit of adding a sense of air turbulence by increasing the randomness between obscurant puffs without additional computational cost.

In using Euler's method, discrete time steps are used starting from the initial value when time equals zero. To take a step we use the derivative of the function to calculate the approximate change in the dependent variable with time. For a thorough review of numerical methods see [ANDR93]

The movement of the objects is determined by solving the force function for acceleration (Eq 10) and using Euler's numerical methods to integrate with respect to time to determine acceleration at each time step Δt for each object in the obscurant plume using. Likewise (Eq 11) and (Eq 12) are used to extrapolate the velocity and position for each cycle of the graphics loop.

$$a_i = (F_i)/m \qquad \text{(Eq 10)}$$

$$v_i = v_{(i-1)} + a_i \Delta t \qquad \text{(Eq 11)}$$

$$r_i = r_{(i-1)} + v_i \times \Delta t + \frac{1}{2} \times a_i \times \Delta t^2 \qquad \text{(Eq 12)}$$

This simplified model, as discussed previously, does not consider turbulence of the ambient air. However to give the appearance of turbulence random perturbations are

added to each of the objects to give a reasonable movement of the obscurant objects by making a call to a random generator and through errors inherent in the Euler method. The model presented is a balance between the physical model, speed of execution and control of the object immersed in the fluid.

B. FOG

A uniform density and linearly varying fog model is available on the IRIS series of machines using a call from GL so there existed no need to implement it into the EEL. A description of the GL fog is described below and a complete discussion of its use can be found in [SGIA92]. GL creates the uniform effect by blending the object and fog colors based on the distance to the object from the viewer. When objects are near, the objects appear as they would in the absence of fog, but as the distance increase the objects look washed out as the colors of the fog and objects are blended. At a certain distance from the object, the fog will completely obscure the object.

There are several different fog characteristics that can be set by the user such as color and density. The values of the density range from 0.0 where there is no apparent fog to 1.0 where fog totally obscures the color of the object at a distance of one eye unit in eye coordinates. The eye distance is the reference to which fog density values are normalized. The proportion of the objects true color that contributes to the apparent color is called the blend factor and is computed at the vertices of each of the graphics primitive when fog is turned on. The vertex blend factors are interpolated to determine the blend factor at the intermediate pixel values of the graphics primitive.

The blend factor is calculated using (Eq 13), (Eq 14) and (Eq 15). The first two equations are used to calculate fog in eye coordinates for uniformly distributed fog. The last equation calculates fog varying linearly with distance when given the distance the fog starts and stops from the viewer.

$$fog = (1 - e)^{(5.5 \times density \times Z_{eye})} \qquad \text{(Eq 13)}$$

$$fog = (1 - e)^{(-5.5 \times (density \times Z_{eye})^2)} \quad \text{(Eq 14)}$$

$$fog = 1 - \frac{(endfog + Z_{eye})}{(endfog - startfog)} \quad \text{(Eq 15)}$$

Where:

fog: The calculated blending factor in [0,1].

$density$:The fog density.

Z_{eye}: The eye space Z coordinate of the pixel or vertex being fogged.

$startfog$:The distance from the viewer where the fog begins.

$endfog$:The distance from the viewer where the fog becomes opaque.

The color of each of the pixels when the fog is active is determined using (Eq 16).

$$C = C_p \times (1 - fog) + C_f \times fog \quad \text{(Eq 16)}$$

Where:

C: The resultant color.

C_p: The color of the incoming pixel.

C_f: The color of the fog.

C. OBSCURANTS SUMMARY

There are numerous methods and equations available in the literature to model the movement of obscurants like smoke and fog. We have chosen to use Brigg's equations to model the translation and dispersion of our smoke plumes because of its simplicity and ease of implementation. The dust trail and flame generators determine the forces on each of the puffs and calculate the translation and dispersion based on the change of the force over time. The IRIS family of graphics workstation provides a uniform fog algorithm that calculates the color of each pixel based on the distance of the viewer to the object.

IV. NIGHT OBSERVATION DEVICES AND THE PASSAGE OF TIME

Those that use combat simulators need various battlefield characteristics modeled in the VW so all aspects of their warfighting capabilities can be tested. This should include every weapons system, environmental effect and possible advisory.

Characteristics associated with the passage of time, such as the amount and direction of available light, can play a role in determining if a certain action should be undertaken on the battlefield. Consequently, it should be modeled in the VW. If a simulator fails to address an environmental characteristic such as the approaching nightfall, the participant will have to contend with the changing environment for the first time in the field. As Zyda [ZYDA93] states

> Realism is an important factor in virtual world development, but truthfulness is as important to simulations. If we teach people incorrectly in a simulator, we put that individual in danger and lose the usefulness of the simulator.

The forces of the U.S. demonstrated in Panama and then again in the Persian Gulf Crisis of 1991, the advantage of using night vision devices to increase combat capabilities during night operations. The current advantage may decrease as opposing forces are outfitted with similar equipment. As this occurs, it will be the forces that have thoroughly trained and integrated the systems into their battle doctrines that will have the advantage.

The earliest versions of NPSNET did not model the passage of time but rather had one time setting for the entire time of play. The latest generation of NPSNET, NPSNET-IV, allows the participants to discreetly establish a time of day that sets the amount of light and color in the scene, but there is no change in environmental aspects such as light directionality. The players continue to play in the set time frame until actions are taken to modify the time setting. To increase the effectiveness of the simulator, the effects of the passage of time that occurs naturally without intervention should be integrated.

The following short discussion explains the basic concepts used in developing the model for the "Night Observation Devices" and "passage of time" algorithms.

A. NIGHT OBSERVATION DEVICES

Over the past forty years, night vision technology has steadily improved to its current state of relatively inexpensive, lightweight devices. The available devices are based on three varied systems each providing air and ground forces an extreme advantage over non-equipped advisories in all low light battlefield conditions. Each of the three technologies have advantages and drawbacks under varying conditions.

1. Image Intensification (I^2)

Image intensifying systems are passive instruments that intensify available light so the user can see surrounding objects and terrain in detail. Newer systems using this technology require a minimum of light and can be operated successfully in overcast or moonless nights but their use is limited in environments where there is virtually no light. Then image intensification systems are small enough to be used as a gunsight and are relatively inexpensive making outfitting large numbers of personnel possible. They do however have a relatively poor resolution. [LESS93]

2. Thermal Imaging

This type of system, unlike the I^2 systems, requires no existing light, but depends upon the temperature differential of objects in the viewed area. Because most objects in typical environments have unique temperatures, a viewer can easily distinguish between the terrain, vehicles and personnel in environments involving light foliage, smoke, dust and camouflage. However in situations involving rain showers or snowstorms, the effectiveness of the system decrease. Some models are small, weighing less than six pounds. [LESS93]

3. Laser Radar (LADAR)

LADAR technology operates much like radio-frequency radars, using the measured time-of-flight of its pulsed energy to produce an image. The components for scanning the energy beam and detecting the return are similar to those used in imaging

infrared sensors. This system is very effective in no-light situations and is effected less by temperature effects than the previous models but because of its non-passive method of imaging, is limited by Emission Control (EMCON) conditions. Current prototypes have dimensions as small as eight inches in diameter by eight inch length, weighing less than 12 pounds.[LESS93]

The advantage of having the ability to fight unhindered in the dark is well documented and the US armed forces are acquiring devices to outfit everyone from the aviators that fly close air support, to tank and truck drivers to the infantry soldier [LESS93]. The training of these individuals in operational exercises and in simulators should include NOD's if their use is to be as effective as possible.

B. PASSAGE OF TIME

1. Defining The Time Of An Event

The year, month, day, hour and location must be included whenever an event such as sunrise or moonset is defined for it to be meaningful to another person located at some other position on the earth. Specification of the date and time can be expressed as local mean time, relative to the meridian of the event, in Standard time, relative to a standard zone meridian, or to Universal Time which is relative to a world wide standard.

The local mean time standard is expressed in terms of the meridian of the location of the object and as such is an isolated time measure of no interest to any other meridian. The Standard system divides the Earth's into 24 time zones and all the clocks in any one zone are set to the same hour. The time zone designation of the event and the observer must be known for time correlation between two parties. The last method called Universal Time (UT) commonly called Greenwich Mean Time or Zulu military time uses a 24 hour clock as opposed to the standard 12 hour clock. When using this system when it is midnight at Greenwich, England, it is time 0000 UT. The advantage of using Universal Time is that individuals need only know their own zone designation to convert UT to their zone time [MILL78].

2. Specifying A Location Of A Body On Earth

In order to specify a location of an observer on the earth, two coordinates called the latitude and longitude are needed. Latitude, is the distance from the equator expressed as an angle measured northward or southward from zero degrees at the equator along the meridian of the observer to the observer. Longitude, expressed as an angle measured eastward from the prime meridian located in Greenwich, England to the local meridian of the observer. In this system of coordinates, the maximum possible latitudes and longitudes are 90 degrees North and 90 degrees South and 180 degrees West or East of the prime meridian [MILL78]

3. Specifying The Location Of A Celestial Body

As in the position on the Earth, two coordinates are required to specify the position of an object in the sky. One coordinate called the altitude is referenced skyward from the horizon and is the angle measured upward to the point occupied by the celestial body. The altitude in Figure 6 is represented by the angle DcP. The maximum value of the altitude measurement is 90 degrees

The second coordinate called the azimuth is the angle measured along the horizon eastward from true North to the observer. It is represented in Figure 6 as NcD. The location where the arc of the altitude intersects the altitude is the location of the body of interest [MILL78].

4. Defining Celestial Events Of Interest

When an individual is located on the surface of the earth, the area in the immediate vicinity appears as a flat plane,[1] while the sky appears as the inside surface of a sphere. The horizon is the intersection of the sky sphere with the plane (see Figure 6).

During the course of a day, the Earth rotates on its axis and the celestial bodies appear to rise in the east and set in the west. The most noticeable of these events is the rising

---

1. This disregards all geological structures such as hills, valleys or mountains.

and setting of the sun and the moon. The rise and set for the sun and moon is defined as the time when the upper semicircle of the body is even with the horizon of the observer [USNO87].

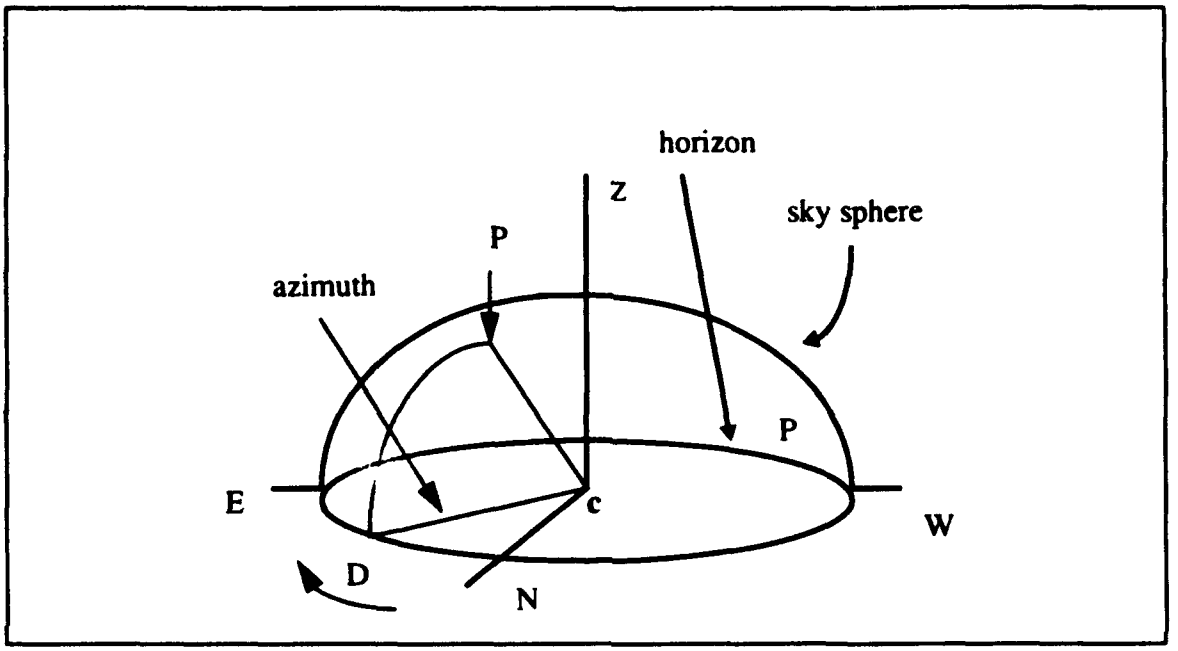

Figure 6. The Horizon-Sky Sphere Model.

In certain locations on the Earth, such as those areas north of the Arctic Circle and South of the Antarctic Circle, the phenomena of sun and moon rise and set do not occur as an unbroken interval. There are days when the sun and moon do not rise or set according to the model presented. This occurs because of the imprecision of the calculations and atmospheric conditions that may cause the light to be reflected so much that the phenomena does not appear to take place [MILL78].

There is a period of time prior to sunrise and after sunset during which there is natural light provided by the scattering of sunlight in the upper atmosphere. This period is subdivided into three periods depending upon the number of degrees the center of the sun is below the horizon. When the sun's center is between 0 and 6 degrees below the horizon, the period is known as civil twilight. The period when the center of the sun is between 6

and 12 degrees below the horizon is known as nautical twilight and when the sun's center is greater than 18 degrees it is said to astronomical twilight [USNO87].

5. Available Lighting

Illuminance measured in lux or lumens per square meter, is the flux or radiation per second received on a unit area of any surface. In ordinary terms, illuminace is the amount of natural light reaching the surface of the Earth. The ambient atmospheric conditions modifies the illuminance to a considerable degree. Hence, the amount calculated by the model presented here will most likely differ from the amount of light actually received on any given day. The recommended approach to interpreting the calculated illuminance is to consider the numbers as threshold values which, without additional information, determine only when certain activities can take place in natural light or when optical devices should be utilized [MILL78].

C. THE CELESTIAL MODEL

The EEL contains the function calls to model the movement of, and the light contributed by the sun and moon and the procedures to determine the times of nautical twilight. The equations used in developing the algorithms are briefly describe here. For a full description of celestial navigation see [MILL78]

The equations to model the movements and calculate the position of those bodies have been known since ancient times and in the past large almanacs with pre-calculated values were need to simplify the mathematics, but with the advent of computers the calculations are now straight forward.

$$\sin(a) = \sin(\varphi) \times \sin(\delta) + \cos(\varphi) \times \cos(\delta) \times \cos(LHA) \quad \text{(Eq 17)}$$

$$A = (\sin(LHA)) / (\cos(LHA) \times \sin(\varphi) - \tan(\delta) \times \cos(\varphi)) \quad \text{(Eq 18)}$$

$$LHA = GHA + \lambda \quad \text{(Eq 19)}$$

$$GHA = (GAST - RA) \times 15 \quad \text{(Eq 20)}$$

(Eq 17) through (Eq 20) are the used to determine the altitude a and the azimuth A of a celestial body where:

a: The altitude of the celestial body.

A: The Azimuth of the celestial body.

φ: The latitude of the observer.

δ: The declination of the body.

LHA: The local hour angle of the body.

GHA: The Greenwich hour angle.

λ: The local longitude of the observer.

$GAST$: The Greenwich apparent sidereal time that is a function of the Julian date and universal time of the observation.

RA: The apparent right ascension (referred to the true equator and the equinox of date) in hours.

For latitudes between 65 degrees North and 65 degrees South (Eq 21) through (Eq 26) can be used to determine the times of sunset, sunrise and twilight with an accuracy of about two minutes. Above latitudes of 65 degrees the results become less accurate so the results should not be relied on for purposes requiring high precision.

$$M = 0.9856 \times t - 3.289 \quad \text{(Eq 21)}$$

$$L = M + 1.916 \times \sin(M) + 0.02 \times \sin(2 \times M) + 282.634 \quad \text{(Eq 22)}$$

$$\tan(RA) = 0.91746 \times \tan(L) \quad \text{(Eq 23)}$$

$$\sin(\delta) = 0.39782 \times \sin(L) \quad \text{(Eq 24)}$$

$$\cos(H) = (\cos(z) - \sin(\delta) \times \sin(\varphi)) / (\cos(\delta) \times \cos(\varphi)) \quad \text{(Eq 25)}$$

$$T = H + RA - 0.06571 \times t - 6.622 \quad \text{(Eq 26)}$$

Where the variables are given the following meaning:

M: The sun's mean anomaly.

L: The sun's true longitude.

t: The approximate time of the local phenomenon in days since January 1.

z: The sun's zenith at rise, set or twilight.

H: The sun's local hour angle.

T: The local time of the phenomenon

In extreme northerly or southerly latitudes, the calculated events such as sunset or moonrise may not always occur because of round off error or other atmospheric conditions. The moon or sun may remain above or below the horizon for more than one day or the sun may remain above or below the horizon for months. Because of the intended use of the algorithm in the NPSNET simulator and the desire to reduce computational complexity, we determined that this slightly less accurate model provides results sufficient for our needs.

D. LIGHT MODEL

The Earth is continually bombarded by electromagnetic radiation. The waves in the electromagnetic spectrum seen by the human eye are said to be in the visible spectrum and have lengths ranging from 0.710 to 0.40 micrometers.

As the light from a celestial body enters the atmosphere it is refracted, scattered and reflected by clouds, fog, smoke haze and the molecules of the atmosphere itself before it reaches the Earth. The amount of available light and the extent to which it is scattered is the dominate factor in determining the color of the sky.

Rayleigh's Law indicates the light energy scattered per unit volume of air is inversely proportional to the fourth power of the wavelength of the illuminating radiation. This phenomena is called preferential light scattering. Rayleigh's Law predicts the probability that blue light will be scattered out of a volume of air is 16 times that of the red light. This is why the sky appears blue during the day. However, during twilights, the sun's altitude becomes small and the amount of air the light must traverse increases to the point where most of the blue light is reflected into space and prevented from reaching the observer at all so the majority of the light reaching the observer is scattered red light, so the observer will sense a red atmosphere. In contrast when the sun is directly above, the amount of

scattering of the red band is so small the sun appears to have a white appearance. A more thorough discussion on the optics of the atmosphere can be found in [CAMP77].

V. IMPLEMENTATION

A. IMPLEMENTATION OF THE ENVIRONMENTAL EFFECTS

The discussion of implementation includes an explanation of procedures, structures used in the algorithms and the methods used to achieve the desired affects such as the proper dispersion of a smoke plume and the effect of wearing night observation devices during daylight hours.

The methodologies used in the implementation of the smoke, flames, cloud and dust generators are similar, so they are discussed as a group. The "lightning", "passage of time" and "night observation device" algorithms are discussed separately, followed by a review of how the algorithms interact with one another.

A set of manual pages for the GL applications that describe the functions, the C specification and notes helpful to the user is provided in the APPENDIX.

1. Obscurant Generator

Reeves discusses the use of particle systems to render objects not readily modeled using simple polygons in [REEV83] and Gardner discusses a method to generate a smoke plume using a series of primitives in [GARD92]. The basis of our obscurant generator is derived using characteristics of both methods. We use a three-polygon primitive, considered as a point mass, that has its own individual attributes such as age, position and size. Unlike a true particle system in which every particle has its own attributes, the method used assumes particles near each other have similar characteristics and can be grouped and transformed as a group. This assumption can be used when the existing wind is uniform and obstructions to the fluid flow are considered insignificant.

The particle primitive of the obscurant cloud is called a "puff" and consists of three 12-sided textured[1] polygons each aligned perpendicular to the other as shown in Figure 7. The vertices and the texture mapping coordinates are generated using the

1. The vehicle dust trail generator uses a non-textured object. The non-textured object yielded better results than we could obtain using the method described for the smoke and flames.

equations $x = r \times \cos(\theta)$ $y = r \times \sin(\theta)$ and stored in a lookup table to reduce the number of sine and cosine calls

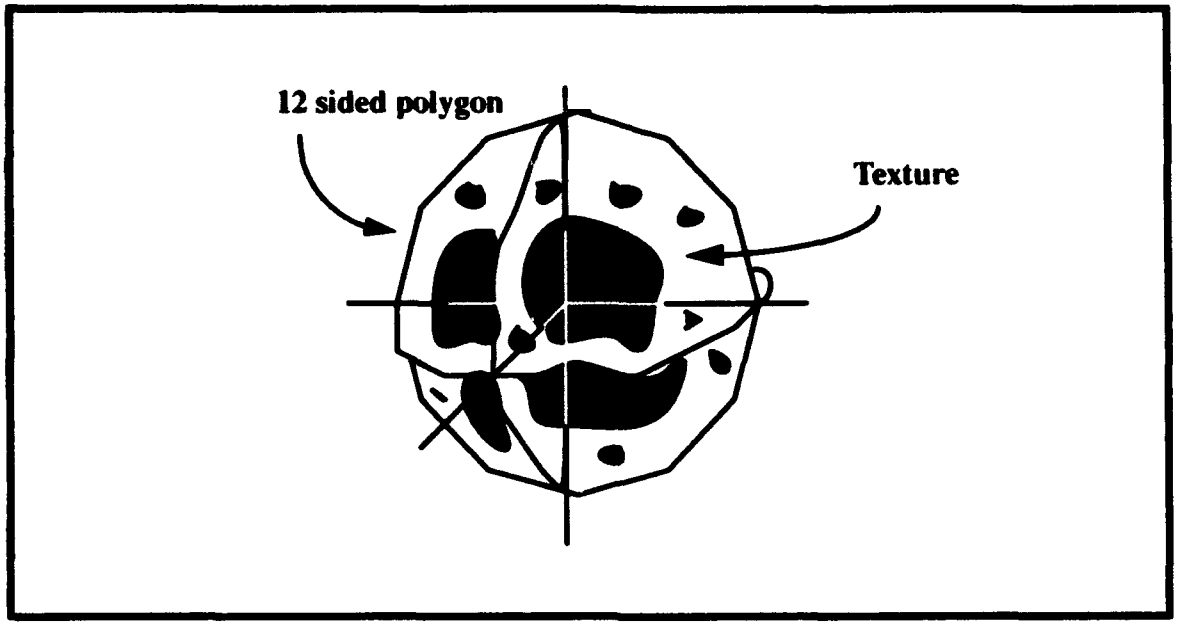

Figure 7. Puff With Texture Applied

The characteristics of a puff primitive are defined by the individual attributes maintained in the puff structure and are used to determine the size, shape, translucence, position and age of the puff. Since each primitive maintains its own characteristics and is considered a point mass, orientation in the world is insignificant and the resulting shape and appearance of the overall plume is determined not by one puff alone but by all puffs of the plume. Furthermore, Reeves states in [REEV83] the modeling of the transformation is essentially procedural with slight differences provided using random numbers. This allows a large number of primitives to be transformed in a short period of calculation time.

The age of the puff is calculated by determining the elapsed time since its creation and is used to calculate the position, size and translucence of each of the puffs. The maximum obtainable puff age is a function of the number of puffs available and the maximum downwind distance the puffs are expected to traverse. The limitation on the number of puffs is determined by the allowable amount of space allocated to puff storage and more importantly the acceptable increase in rendering time associated with the increase

in puffs. The current implementation of smoke uses 75 puffs per column, which is 1125 textured polygons for a plume that is in a full bellow. With three plumes of smoke and flames being put into the graphics pipeline, the simple driver program used in implementation and testing was slowed from 40 frames per second to just under 10 on a IRIS 4D/440 RE.

The polygons are textured using a four-component, image file of a smoke puff applied in an environment defined with the property TV_MODULATE which causes the RGBA characteristics of the resulting object to be derived from the RGBA values of the underlying polygon and the applied texture [SGIA92]. The alpha value of the underlying polygon is reduced with age so the resulting image appears to fade away as the puff moves far away from the source of generation.

A puff primitive is in one of the three states illustrated in Figure 8 during the life of a plume; the "unused queue", the "active queue" or in the "transformation and rendering phase". Puffs in the unused queue have initial attribute values and are ready to enter the smoke plume at the source as new virgin puffs. Those puffs in the active queue have been rendered at least once in the currently generating plume but have an age less than the maximum life of a puff.

Initially all puffs are in the unused queue but as the algorithm generates obscurant, they are popped off the queue, the attributes are updated, the puff becomes part of the obscurant column and are moved according to the appropriate translation model[2], scaled and placed in the queue of active puffs. Since each puff's size is known throughout its lifetime, we can reduce the number of puffs needed to generate a full plume by translating each puff a sufficient distance prior to generating the next one. Once the puff primitive is sufficiently translated from the source, the next puff taken off the queue undergoes a similar set of transformations.

---

2. The smoke generator uses Briggs' equations while the other generators use equations derived from the Newtonian equation F=ma to calculate the amount of translation.

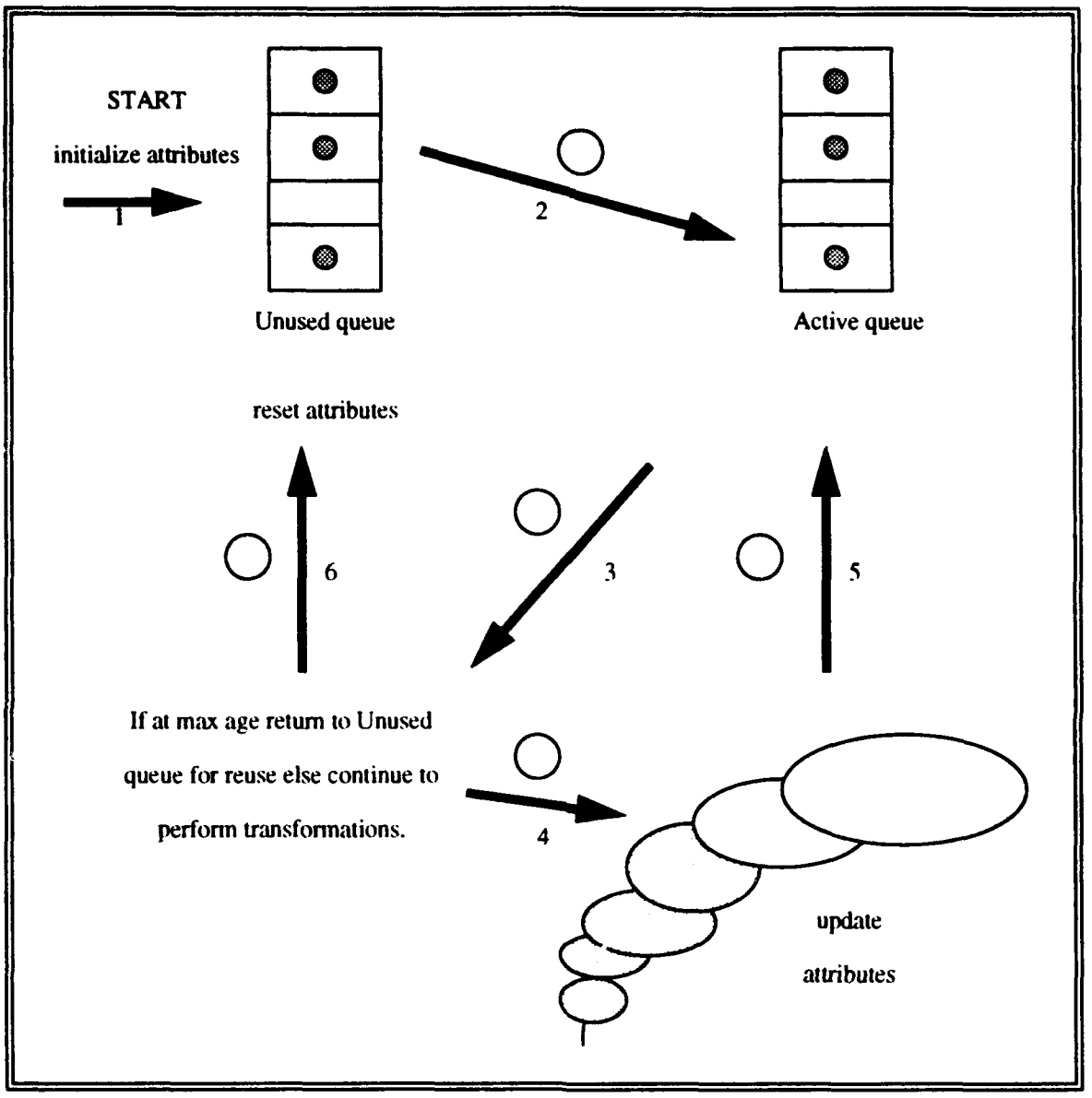

Figure 8. Life Cycle Of An Obscurant Puff

Each subsequent puff added to the active queue causes the length of the plume to increase. Once a puff's age exceeds the maximum life time defined for the plume, it is pushed onto the queue of unused puffs and reinitialized again for reuse.

The resetting of puff attributes allows the algorithm to run for any desired time with a finite number of puffs. It does however limit the maximum downwind distance that the obscurant can be translated. If this distance is not properly set for the available number of puffs, a discontinuity in the smoke generation at the source will be apparent during generation.

An obscurant plume is generated by simultaneously generating five columns using a footprint similar to the one suggested by Gardner in [GARD92] and shown in Figure 9. Each of the columns are independent so it generates and maintains the information for each plume and puffs in its own smoke structures.

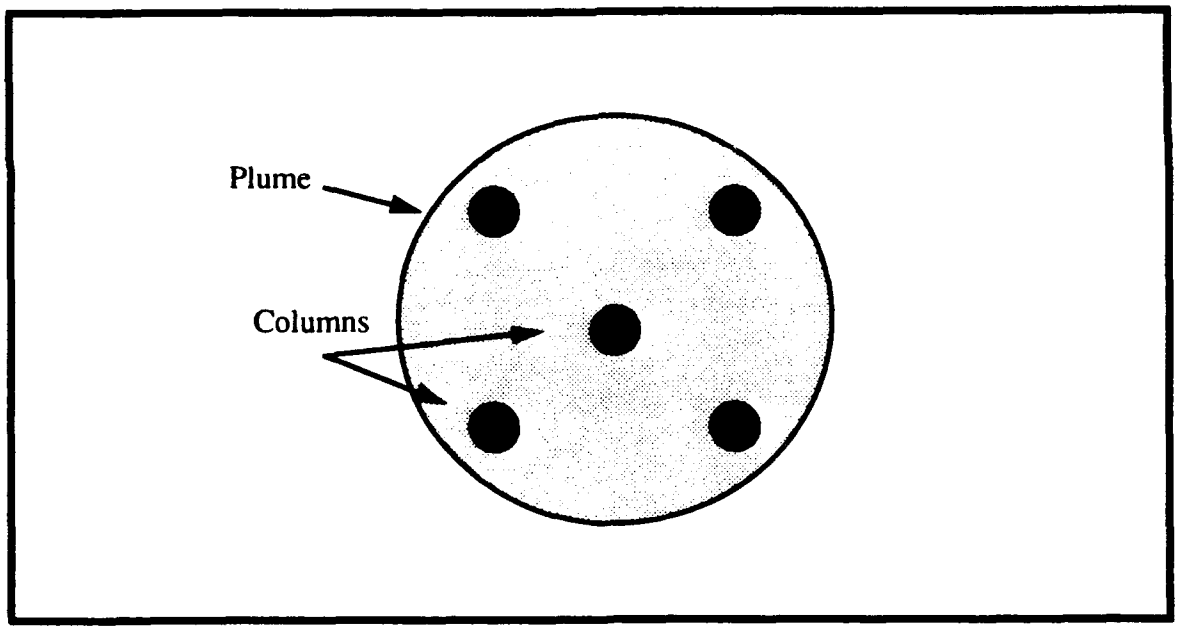

Figure 9. Obscurant Footprint

By rotating the footprint of the flame generator 45 degrees from that of the smoke generator, a fuller more robust flame smoke combination is achieved adding to the realism of the model. The shape of the plume is determined by the path of the puffs as they are

effected by the wind of the surrounding air mass and the smoke dispersion over time. Figure 10 illustrates a petroleum fire with a wind speed of 2.4 meters per second. Notice that the wind has more effect on the smoke as the distance from the source increases and the momentum caused by the initial updraft of the flame decreases. Figure 11 shows a vehicle dust generator being created by a vehicle traveling at 45 m.p.h. on light dusty terrain and demonstrates how the dust trail becomes noticeably less dense at the trailing edge.

Figure 10. Smoke And Flame Generator

Initially an attempt was made to change the alpha value of the underlying polygon as a function of the puff's chemical composition, but we found using a linear decrease in alpha with the age of the puff provided a sufficiently realistic result at a very small computational cost. The simplification resulted in a visually realistic plume without requiring the chemical concentration of each puff to be computed.

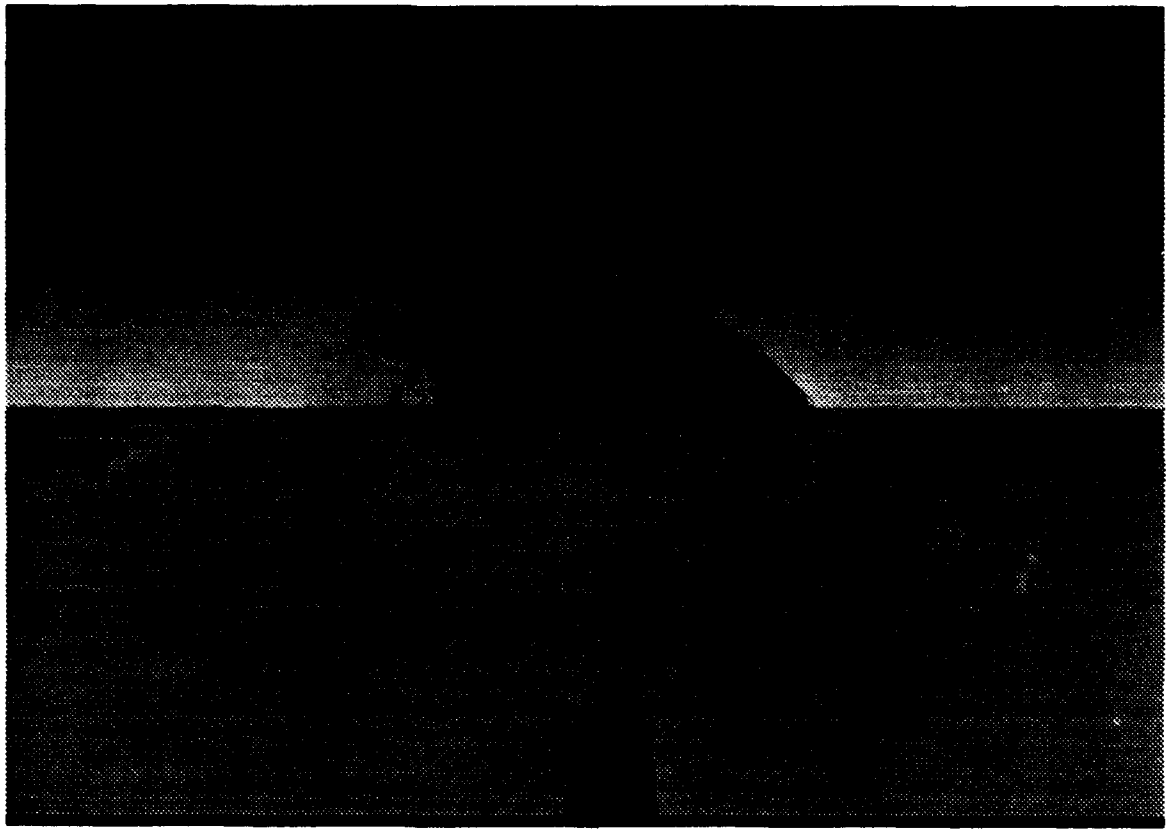

Figure 11. Vehicle Dust Trail

2. Passage of Time

The passage of time algorithm uses a set of functions derived from the equations for celestial bodies and daily solar events information discussed in Chapter IV. The purpose of the algorithm and supporting functions is to enable the user to position the sun and moon in their correct locations and determine the amount of available light in the scene and color of the horizon relative to the of line of sight, latitude and longitude of the observer.

The horizon is implemented by having a "horizon polygon" rendered as the first object in the graphics loop in 2-D orthographic mode. It is positioned at the farclipping plane of the observer. Since the viewer has the freedom to move in all three directions, the position of the horizon polygon is a function of the viewer's coordinates, reference coordinates and the dimensions of the view volume shown in Figure 12. The vertex locations of the horizon polygon are determined using the relationship between the height of the viewer, height of the reference point, and distance to the farclipping plane.

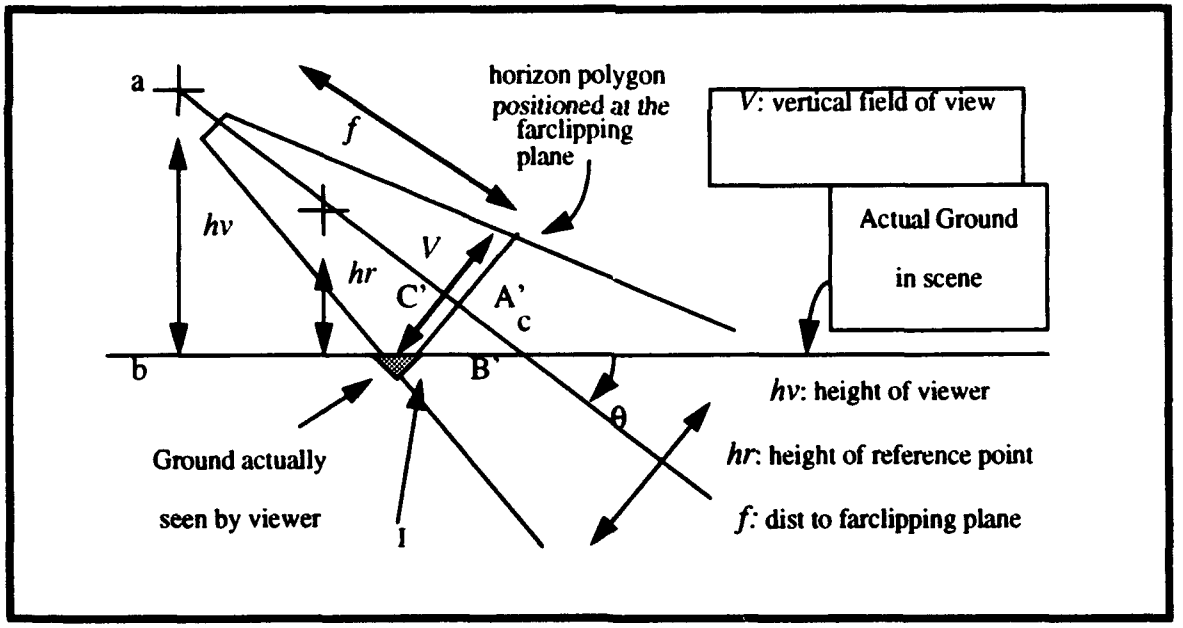

Figure 12. Relationship Of Ground In Scene To That In View Volume.

Using (Eq 27) and noting the similarity of triangles shown in Figure 13, we determine where the horizon meets the ground (shown as I in Figure 12) in screen

coordinates. The color of the horizon polygon is determined by assigning a color to each vertex of the horizon polygon and the resulting polygon color is determined by the Gouraud shading model. The color of the vertices is based on the probability the light of the blue band has a 16 times greater chance of being scattered than the red light in a same volume of air [SCHA81]. Our model calculates the number of relative atmospheres as a function of the inverse of the sine of the sun's altitude. The upper vertex color is determined by adjusting the color of the blue band as the day progresses and holding the red and green band constant at zero.

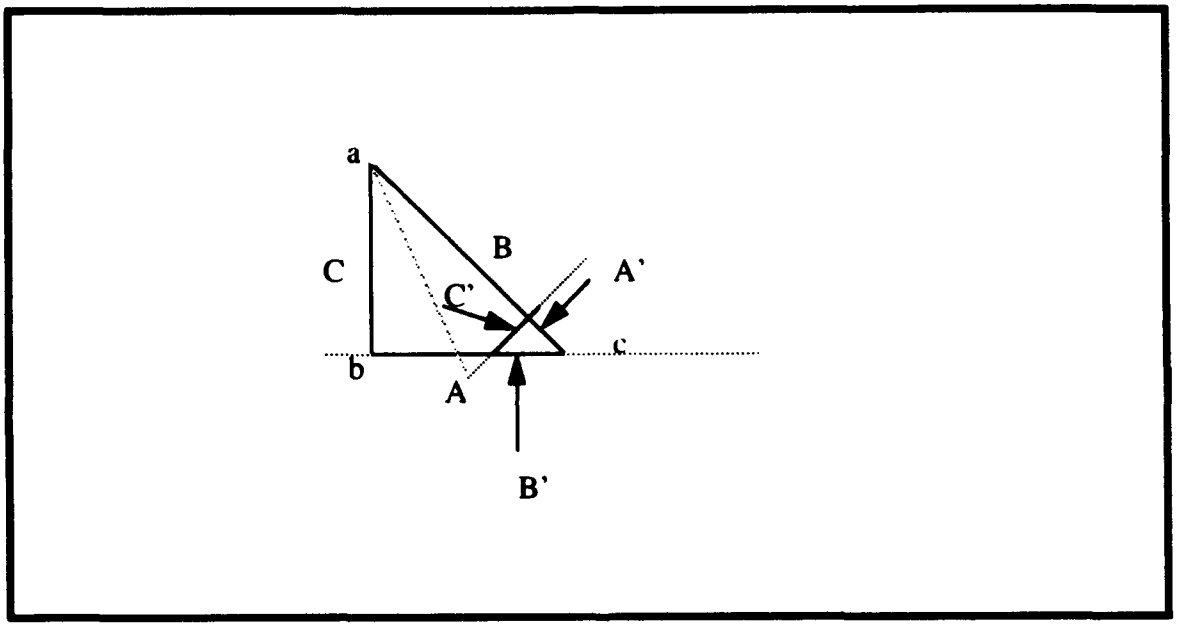

Figure 13. Similar Triangles ABC And A'B'C'

$$horizon-ground-intercept = \frac{V}{2} - \left(\left(\frac{1}{\cos(\theta)}\right) \times \left(\frac{f \times \tan(\theta)}{hv}\right)\right) \quad \text{(Eq 27)}$$

When the sun is directly overhead, no scattering of light occurs and the RGB values of the lower vertices is set to 255. However as the sun's altitude decreases, the value of the red band is decreased at a rate so it will have a zero value at nautical twilight.

Likewise, the green and blue band are decreased respectfully ten and sixteen times as fast for any given decrease in altitude. The gives the horizon a white scattered effect during mid day and falls off to a bright red and then dark sky at the end of twilight.

The amount of light in the scene is determined using the method discussed in Chapter IV. To use the resulting information in a meaningful way, we calculated the amount of light in the scene at the equator on June 21 at 1200 and arbitrarily set this value as the maximum ever possible. Then for any desired time, we set the intensity of our defined light to the ratio between the calculated light in the scene and the maximum possible available. This allows our model to have nights when there is enough illuminance from the moon so objects are visible.

When a light model is defined with a one-sided characteristic, the position of the light source determines which side of a polygon is colored and which is not. This gives the effect of having shadowed surfaces in the scene. As the position of the light source rotates around an object, the side that is colored will change. To model the changing position of the light source and illumination, we locate the defined light source by setting its position to the brightest of the sun or moon.

3. Night Observation Devices

The night observation devices (NOD) are modeled after the image intensification systems discussed in Chapter IV. When the NOD's are donned, black goggles are drawn over the scene to limit the field of view, electronic noise is simulated and the scene is given the distinct greenish glow present in the actual devices (see Figure 14).

To reduce the field of view, a mask is generated by drawing four rectangular and four triangular polygons in two-dimensions at the end of the graphics loop to produce the effect of wearing goggles. To simulate the electronic noise noticeable in the actual devices, we randomly place black and green points in the region of the screen where the goggles are drawn. By placing the points just in the eye piece rather than the entire screen area, we can achieve a high point density using a relatively low number of points. When the NOD's are

donned, the RGB values of the defined light is modified to obtain a greenish glow by setting the ambient component to (0, 0, 0) and the color component to (0,.68, 0). Since the horizon polygon has minimal significance during the night hours and the NOD's are unusable during daylight hours, it was decided the horizon polygon should not be rendered during use of the NOD because to do so requires constantly updating the color of the polygon when it is not viewed.

Figure 14. Night Observation Devices

The amount of light in the scene is used as an input to the algorithm to determine if the "light gain" of the NOD's is such that the image is washed-out because of an over abundance of ambient light. The "washout" is modeled by placing a polygon generated using a material with an alpha value directly proportional to the amount of illumination in the scene over the goggles. A lightning bolt causes the amount of light in the scene to reach a maximum for a short period of time and then quickly return to the amount defined by the illumination equations.

4. Lightning

The algorithm used to model lightning uses a recursive function to generate a tree of nodes and branches when the root node's location is given. The user defines the bounding coordinates of the world so the length of a bolt of lightning can be limited.

The angle any branch makes with the ground plane is randomly chosen from the set (225, 315) and the corresponding branch length is randomly chosen from the set (0, 100). The algorithm starts to back out of the recursion when a leaf node falls out of the bounding volume defined by the user or a node has no offspring. The user defines the probability a node will generate a left or right leaf which in turn determines the density of the generated lightning bolt. Figure 15 illustrates a typical set of lighting bolts with a density of 50 percent.

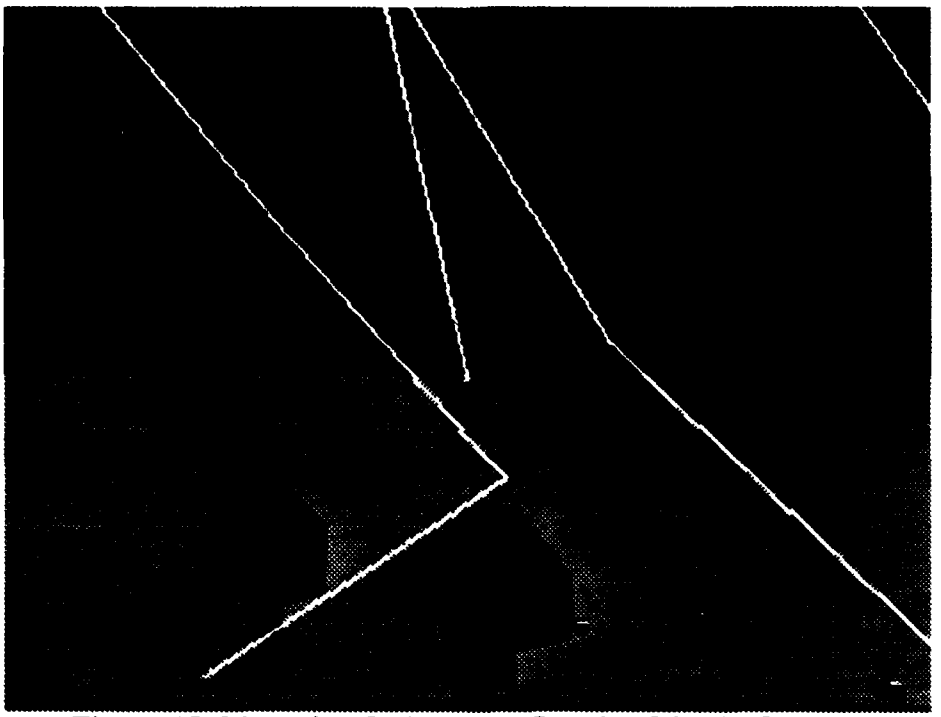

Figure 15. Lightning Strike With Density Of Fifty Percent

A call to the lightning algorithm causes the illumination in the scene to be momentarily set to the maximum amount possible and then is linearly decreased back to

the amount calculated by the illumination equations for the time of day. This causes the NOD's to "washout" during any lightning strike regardless of location of the wearer relative to the lightning bolt location.

B. IMPLEMENTATION USING THE IRIS PERFORMER TOOLKIT

IRIS Performer is an application development tool that combines a programming interface for creating visual simulation applications and a high-performance rendering library. It provides a powerful and general means of high levels of graphic performance from the IRIS workstation. It consists of two libraries: libpr.a and libpf.a. Libpr is a low level library that provides high speed rendering functions, state control and other machine oriented functions. Libpf is a rapid prototyping environment that uses libpr functions to create a multi-processing, automated database rendering system that takes advantage of IRIS multi-CPU hardware. [SGIB92]

Libpf provides a pipelined multiprocessing model for implementing graphics simulations. The three pipeline stages are (1) simulation, (2) traverse and cull and (3) draw. The simulation stage updates and queries the scene, the traverse and cull stage traverses the scene and adds geometry to the display list, which is then rendered by the draw stage. [SGIB92]

Performer uses a run-time database structure and hierarchy to maintain state information and geometry. This provides for a fast efficient method of object rendering. During execution, Performer examines the scene database, culls as necessary and then renders the geometry contained in the scene. The scene hierarchy describes how items in the scene database relate to each other in both the logical and spatial organization. The spatial organization of the database is used to increase the performance of certain operations such as drawing and determining the intersection of objects.

The basic element of the database hierarchy is the node and each of the nodes has a specific function. Table 2 lists the nodes pertinent to our work that are available and a brief

description of the functions they provide. For a complete description of their use see [SGIB92].

TABLE 2. PERTAINENT IRIS PERFORMER NODE TYPES

Node Type	Class	Description
pfNode	Abstract	Basic node type
pfGroup	Branch	Groups zero or more children
pfScene	Root	Contains the visual database
pfDCS	Branch	Dynamic Coordinate System
pfSwitch	Branch	Selects active children

Performer uses inheritance to allow nodes in the tree structure to share attributes with any leaf below it. The base node is called pfNode and all nodes inherit its attributes. This allows the children nodes to have properties that are not specifically designated by the user.

The Performer implementation of the smoke generator is a low fidelity version of the generator discussed previously. It is written in C++ and has only slight changes from the original GL version. To reduce the computational complexity, it uses a simple natural log function to translate each of the puffs in a single column. This method of transformation was found to be an order of magnitude faster than when using Briggs' Equations on the IRIS Indigo Elan. It uses the same queues and structures for maintaining the individual plume and puff information as discussed previously.

The differences in the GL and Performer version of the algorithms exists in the rendering phases. The main difference between GL NPSNET and NPSNET-IV using performer is that it requires the geometry to be passed from the simulation process to the

draw process by attaching the applicable plume nodes to the scene via a smoke group node before it is rendered.

To implement the smoke generator using Performer, in the smoke initialization function we first define and establish the smoke plumes' hierarchical tree structure consisting of all nodes from the plumeswitch to the puff image geometry and define a smoke group and attach it to pfScene as shown in Figure 16. To the smoke group, we attach a root node of the terrain database. The terrain is divided into subterrain blocks that allows for efficient spatial traversal of the database.

Individual plumes are made up of a pl_switch and pl_DCS node to which we attach as many puffswitch and puffDCS node combinations as there are puffs in each plume. The code fragments listed in Figure 17 and Figure 18 along with the following description provides an explanation of the implementation of Performer smoke functions. When smoke is required, the structure containing the information of all plumes is traversed to locate the next available smoke plume. After the next available plume is identified, the plume switch is set ON and the plume is transformed by applying the necessary translation values to the plume DCS so that the plume is moved to the appropriate position in the terrain database and attached to the sub-terrain node.

As each puff in the column is needed, it is taken off the active queue, its puff switch is turned ON and translated according to the translation and dispersion algorithm using the puff DCS. Once translated, the geometry is sent to the draw process for rendering. The puff continues to be taken off of and returned to the active queue as long as it remains alive. (using the same criteria as in the previously described smoke generator). Once the puff reaches the end of its useful life, the puff switch is turned OFF and the puff is placed back on the inactive queue of unused puffs. The plume will continue to generate puffs for the duration time set by the user. Once it has existed for the duration time, the plume switch is turned OFF and all the puff and plume parameters are reset so the plume will be available for reuse.

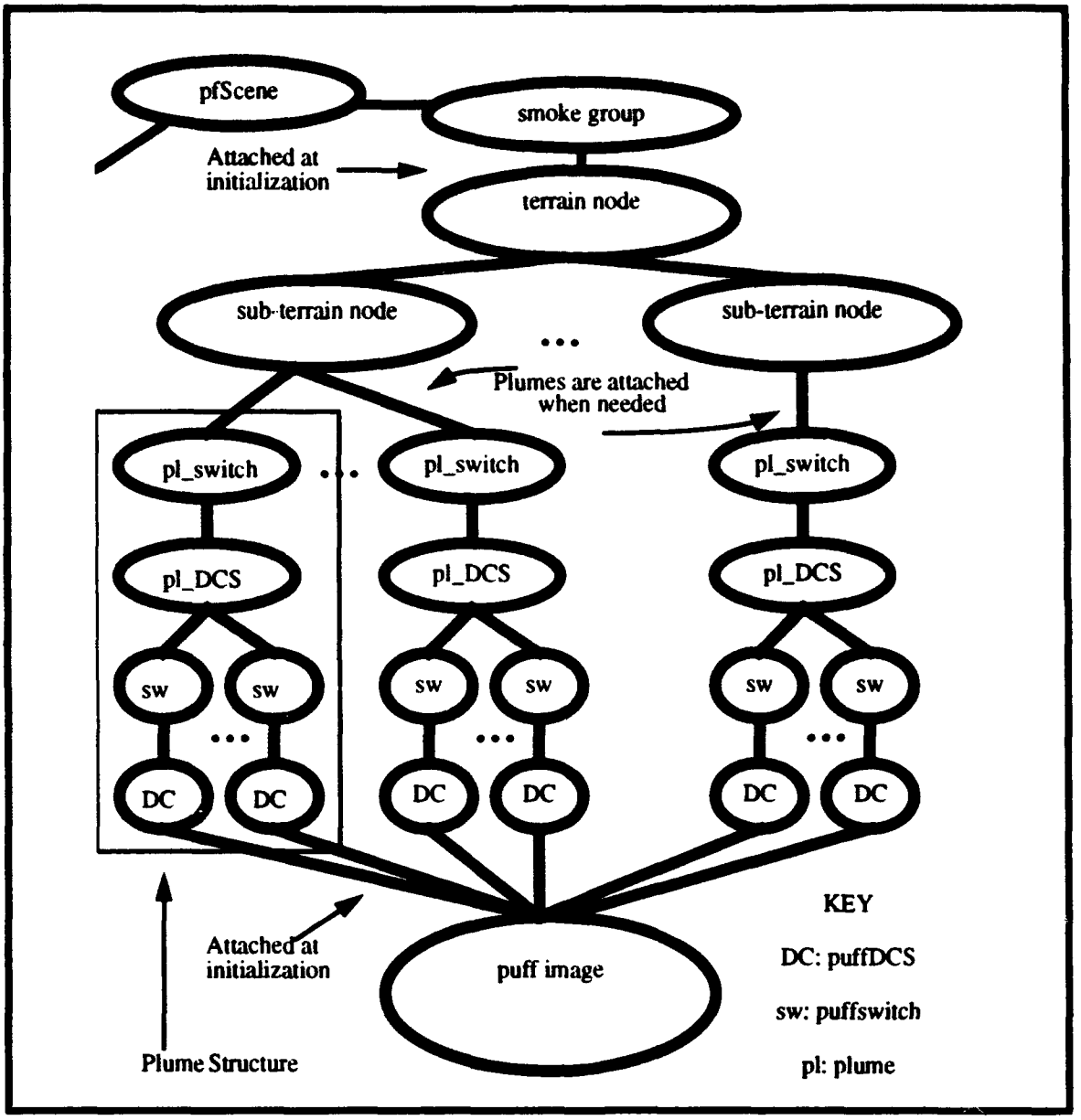

Figure 16. Performer Plume Hierarchy

```
main(){
smokeplume plume;
initialize_system();
initialize_all_variables();
initialize_fire(numberofplumes, winddirection, windspeed);
while(TRUE) loop{
   pfSync(); //get the processes working together
   pfChannel(); //setup the view point
   do_an_event requiring_smoke();
   usenextsmokeplume(x,y,z,duration);
   generateallsmoke();
   pfCull(); //find out what to draw
   pfDraw(); //draw a frame
}
```

Figure 17. Performer Implementation Of Smoke Generator

```
initialize_fire(int numberofplumes, float winddirection, float windspeed){
   set_pfSwitches_and_pfDCS();
   set_puff_attributes_of_all_plumes();
   build_plume_structure();
   build_trig_lookup_tables();
}//end initialize_fire

usenextsmokeplume(float x, float y, float z, int duration){
int plumenumberused
   findnextavailableplume(plumenumberused);
   addplumetoterrain(plumenumberused);
   return(plumenumberused)
}//end usenextplume

generateallsmoke(){
   loop through active queue{
   if(duration then stopandresetplume() and return());
       getnextpuff();
       translatepuff();
       if(puffisold then resetpuff())
       else
       putbackinactivequeue();
    }
}
```

Figure 18. Functions Used In Performer Implementation

The Performer based simulation was able to maintain 30 cycles per second with up to five plumes generating simultaneously when positioned close together. We ran the application with up to 15 plumes generating and the simulation maintained between 20 and 30 hz if all the plumes were not in the view area simultaneously. A typical smoke plume generated in the Performer environment is shown in Figure 19. It is a low fidelity version using one column of 15 textured puffs translated with a natural log function and scaled based on age. The puff object is textured using an image that is applied prior to compilation.

Figure 19. Low Fidelity Smoke Plume In Performer Environment

C. PERFORMANCE

The program used to test the performance of the GL obscurants typically operated between 30 to 40 cycles per second (cps) and was slowed to about 10 cps when three plumes of any type were generating in the field view close to the viewer. The lag decreased as the distance of the viewer to the plumes increased. The lightning function, because of its recursive nature, caused the movement of objects in the scene to stop while it was being generated. As long as the probability of having a left or right child was maintained below fifty percent, it was not disturbing to the viewer. The passage of time algorithm was only

called once per minute so its effects on performance is negligible. The program was able to maintained 30 to 40 cps when the NOD functions were being executed.

D. IMPLEMENTATION SUMMARY

The implementation of the obscurant functions are similar in their use of data structures and method of maintaining the state of each entity in both the GL and Performer versions. The lightning, Night Observation Devices and passage of time implementations are different from each and are designed so they can be used together or independently of each other, however if used together there is the interaction that causes the NOD's to wash-out in high light environments.

The environmental effects were implemented using techniques that allowed their use in a real-time application and often forego computationally expensive calculations for inexpensive yet visually accurate algorithms.

VI. NETWORKING

A. BACKGROUND

The large scale and complexity of real-time simulations are testing the limits of even the most powerful processors available. To overcome this limitation, large simulations can be broken down into smaller stand-alone workstation simulations systems that interact with each other over a high speed network [SCHM93]. This breaking down of the simulation requires each node to control and maintain certain aspects of the world. Additionally each node must inform all other players when actions it is controlling have changed. Although it introduces some redundancy in the information that is maintained by all nodes, it removes the requirement that there exist a super controller constantly communicating the state of the world to all players. With the removal of the requirement for a main node controlling the simulation, there is a decrease in network traffic since each node can proceed with the simulation while only requiring periodic updates.

A networking protocol that allows workstations to communicate the events of a simulation that has been "broken-down" into the necessary pieces so individual nodes can interact without a main controlling processor is called the Distributive Interactive Simulation (DIS) Protocol [IST91].

B. NETWORKING THE SIMULATION

The purpose of networking a simulation is to allow multi-player interaction from numerous nodes over long distances. With a networked simulation, numerous personnel can play from any location by just joining the simulation using an inexpensive workstation. A networked simulation provides a virtual representation of the warfare environment that is inexpensive enough to be available at most duty stations and allow frequent use by members of the forces. With the development of the DIS protocol, it is becoming increasing more feasible to generate large scale network simulations that can be executed on

inexpensive workstations that cost far less than the super computers that were once required.

The goal of NPSNET is to develop a basic virtual world shell that allows one to visit any area of the world for which a terrain database is available and to interact with other human or autonomous players found "in the system". Further, it is to construct a low-cost visual simulator/virtual world explorer interoperable with the DARPA SIMNET system and the follow-on DIS networking standard. [ZYDA93]

C. DIS 2.0.3 PROTOCOL

NPSNET-IV uses the DIS protocol to communicate with other nodes on the network during a simulation. The primary mission of DIS is to create synthetic, virtual representations of warfare environments by systematically connecting separate subcomponents of the simulation which reside at distributed, multiple locations [ISTA93].

The DIS protocol is a highly structured communication system designed as the follow-on to the SIMNET protocol. It has the capability to send pertinent information about warfighting units over a network.

The basic DIS concepts are first, to have a system in which there is no central computer for event scheduling. This removes the requirement of some simulations that there exists a high powered computer capable of maintaining the state of the world. Second, each node is responsible for maintaining the state of one or more entities and sending out information pertinent to other players in the simulation. Third, there is a standard protocol to communicate ground truth data that allows each node to determine whether another entity is visible or perceived by the host node. Fourth, nodes communicate only what changes in the state of the simulation and dead reckoning of dynamic objects is used to reduce traffic on the network [ISTB93]. These concepts allow dissimilar simulations to interact over a large area using inexpensive workstations. In developing the method of networking the environmental effects, consideration was given to all of these concepts. We had to determine how and when a node would perceive an effect, how to maintain its state, how

to communicate the change in an effect while attempting to reduce the amount of traffic necessary to relay the information.

The DIS method of sending data places all information into a packet called a Protocol Data Unit (PDU) and places the PDU on the network to transfer the information to all players taking part in the simulation. We will be dealing specifically with the entity state (ES) PDU so a discussion of that PDU is given. However for a complete discussion of DIS and its PDUs see [IST91], [ISTA93] and [ISTB93].

D. ENTITY STATE PROTOCOL DATA UNIT

One of the PDUs provided by the DIS protocol is called the entity state PDU. That PDU is used to communicate the information needed for a receiving host computer to represent an entity in its own version of the simulation. Since ninety-five percent of network data transmitted is entity state data [ISTA93], most of the packets placed on the network are ES PDUs. The PDU contains an entity's information such as position, type and capabilities and is used by a host node to determine how to render and react to a specific entity. The ES PDU contains the information found in TABLE 3 .

TABLE 3. DIS ENTITY STATE PDU INFORMATION

Field	Description
Header	Identifies the DIS version, exerciser ID and PDU type
Entity ID	ID of the issuing entity.
Force ID	ID of the force to which entity is assigned.
Entity Type	Issuing entities specific entity type.
Entity Type Alternate	Issuing entity's alternate entity type used when the function of guises is employed.
Time stamp	Time for which the PDU information is valid.
Location	The location in the simulated world.
Velocity	The linear velocity of the entity

TABLE 3. DIS ENTITY STATE PDU INFORMATION (Cont)

Field	Description
Orientation	The orientation of the entity in the simulated world.
DR	The parameters used as inputs into the dead reckoning algorithm.
Appearance	The appearance including smoke, dust and presence of articulated parts.
Capabilities	The capabilities of the entity such as logistic or repair.
Markings	Specifies the unique markings on an entity.
Articulation	The number of articulation parameters for the entities articulated parts.

E. IMPLEMENTATION

In implementing the environmental effects, we are concerned with the method of communicating the effects to other nodes interacting in the simulation while reducing the amount of information placed onto the network. We reduce the traffic placed on the network for several reasons. First, the probability of information loss on the network increases as the percentage of used network bandwidth increases and second, as the traffic increases, latency also increases because of the time required to process all the information placed onto the message queue.

1. Effects That Must Be Networked

The EEL contains a group of effects that need to be communicated to all other players in the simulation for realistic interaction because if one node is rendering the effect then it should have the capability of being perceived at every node in the network. This group of effects consists of the smoke, flames, vehicle dusttrails, and the effects of time passage. The Night Observation Devices have only a local effect so it is not necessary for other players to be notified they are in use.

To prove the concept of networking the environmental effects using the DIS protocols, we have implemented the smoke generator algorithm into NPSNET-IV. The method used for networking smoke and a description of an implementing method for the remaining effects is presented.

2. Networking Smoke And Flames

The method presented here deals specifically with smoke but it can be used to network flame generation. We tested our method of communicating the environmental effects to others players in the simulation by putting the smoke generator into NPSNET-IV. A smoke plume becomes an entity of the node generating it. The entity state PDU appearance field provides a method of communicating the generation of a smoke plume by setting the appropriate bits in the ES PDU of the generating entity. In our network model, we implemented a method that does not use the appearance bits but instead generates smoke at the location that it is perceived to be required, such as when a detonation occurs.

We based our concept on the idea each node is responsible for maintaining the state of its own world and must determine what is perceived at that node. With this in mind, our method causes a node to start the smoke generator when it perceives smoke is necessary. In NPSNET-IV this occurs when either a bomb or missile strikes a building, the ground or when a detonation PDU resulting from a ground or building impact is received over the network.

Our method has the advantage of not having to test the appearance bit each time the entity is rendered however, it has the limitation that if a smoke plume is created by an event not perceivable by other nodes, then all nodes will not know to generate the plume. This can occur when an event such as an overheating vehicle occurs. Another limitation is each node will be generating smoke plumes that are dissimilar to the plumes on all other machines. The difference in plumes are brought about by the randomness that is built into each algorithm. The first limitation can be overcome by using the appearance bits in the ES PDU but testing of the smoke bits for each entity will be required. The only method to solve

the dissimilar plume problem is to treat each plume puff as an entity of the node that caused its creation and send an ES PDU to communicate its location. This solution will dramatically increase the traffic on the network and is not a viable solution.

3. Networking Vehicle Dust Trails

The entity state PDU is used to communicate when an entity is generating a dust trail by sending one of four possible values in the dust cloud appearance field. The value defines the size of the dust plume being generated with 0 corresponding to no dust cloud and 3 corresponding to a large cloud.

The dust trail generator in the EEL requires the vehicle direction and terrain soil type to determine the size of the cloud generated for a given velocity. To use the dust cloud attribute of the ES PDU to network the dust trail, it will be necessary to create a lookup table that can associate the received dust cloud field value with the proper dust generator input parameters.

We present an alternate method of networking the dust trail that does not require the use of a table but uses values available in the ES PDU. To understand the method, we first explain how the vehicle movement status is communicated. An object controlled from another node is dead reckoned by the processor which is rendering it using the last acceleration, velocity and heading values received until the values are updated by the controlling node. Since the dust trail generator uses velocity information from the generating vehicle as inputs, we use the dead reckoned values as input parameters to the dust trail generator until updated information is available from the controlling node.

The dust trail algorithm is designed so that once a dust puff is generated, its translation is a function of the vehicle speed at generation and the existing wind. This method will present a noticeable gap in the dust trail when the updated position of the object generating the dust differs significantly from the dead reckoned position.

4. Networking The Passage Of Time

The algorithm used to calculate the effects of time passage requires a starting time to define the time of the simulation. The simulation world time (SWT) is the time of the events as perceived by an object in the simulation. The SWT is determined by adding the accumulated elapsed time to the starting time of the simulation. Each node calculates its own elapsed time using its own system clock. The elapsed time of unique nodes may differ because of system inaccuracies or the different clock resolutions across machines. This type of error should be insignificant since the clock drift is so much less than the resolution of the time of day algorithms.

There is however an "asynchronized player" problem since all nodes may not enter the simulation at exactly the same time. For example, the first node may start the simulation with a starting time of 1200, some time later a second node may enter with the starting time of 1200. This discrepancy will cause the two nodes to have different SWT's. We need to be able to synchronize players as they enter the simulation. The method presented here describes one technique that can be used to synchronize nodes as they join the exercise.

To overcome the "asynchronized player" problem we create an entity called the Time Control Monitor (TCM) which defines the SWT for all incoming nodes. The duties of the TCM are to inform all new nodes of the SWT and that the TCM already exists.

Each node, upon entering the simulation sends out a message claiming he is the TCM with his SWT. Unless the new node receives a message informing him a TCM already exists, he will start to fulfill the duties of the TCM and monitor the network for newcomers to the simulation. If however, the new node receives a message stating a TCM is already active then he sets his SWT to the received SWT and joins the simulation. In the event the acting TCM leaves the simulation, the next node to enter the simulation will assume the duties of the TCM when it does not receive a message stating one already exists.

The simulation may find itself without an acting TCM if: (1) The acting TCM leaves the simulation and no new node enters to replace it or (2) The first several nodes

entering the simulation do so at nearly the same time[1] and receive each other's message claiming to be the TCM and set their SWT to the received SWT. The first case presents no problem since the time drift caused by differing clock resolutions or the clock inaccuracies during a normal simulation duration should be less than one minute. In the second case, since both are starting at nearly the same time, even if both use the other's SWT, the time difference between any two nodes should be less than one minute so there should be not noticeable difference in time of day effects.

5. Networking Lightning

Our implementation of lightning is only for visual que and has no effect on objects in the scene so it is necessary to send only a message to the other nodes that lightning is occurring in the simulation. Each node in the simulation will, using a method similar to the smoke, have the responsibility of generating its own lighting flashes. As the simulation becomes more complex and the lightning effects communications, sensors and causes damage, it will be necessary to inform each node when the flash has occurred and the effects of the lightning flash.

1. The phrase "at nearly the same time" means the difference in starting times between the nodes is less than the time latency of the network.

VII. LIMITATIONS

The functions in the Environmental Effects Library use simplifying assumptions that reduce the computational complexity of the algorithms. Additionally, limitations are imposed by the amount of space that can be allocated to structures in the algorithms. Because of the simplifications made and the limitations imposed, the models presented cannot be used in applications requiring precise physical accuracy. But they all are "visually accurate" and suitable for use in DIS implementations. The major limitations are discussed below.

A. OBSCURANTS

The length of the plume of obscurant is limited by the number of puffs that are used to generate a column. As each puff reaches the end of its useful life, it must be reset so that it can be reused as a new puff at the puff source.

The method we chose to network the smoke generation allows each node to determine when and where smoke is required. This method reduces the traffic associated with smoke on the network but requires each node to generate its own version of the plume. Since the smoke generator algorithm is developed to present a random appearance, each plume will not be an exact duplication on every node of the network.

B. NIGHT OBSERVATION DEVICES.

The NOD's are modeled after a generic version of the Image Intensification systems available so the levels of light intensities that cause them to wash-out is arbitrarily determined. The actual amount of light causing the effect may be different depending upon the actual devices used.

There is a distinct blurring of bright objects when viewed with actual NODS. Our attempt at modeling this effect using the accumulation buffer slowed the performance to such an extreme that it was abandoned and the effect is absent from the model.

C. PASSAGE OF TIME

We based the numerical calculations for the sky color on a simplified probabilistic model, but to meet real-time constraints, we had to create an algorithm to determine sky color that is derived more from observations than the true physical models of light scattering. The resulting color of the sky is close to what you see on a typical day.

Although the amount of illumination in the scene considers the phase of the moon and time of year, the moon is always rendered as a full moon. Additionally no effort in determining and changing the observed colors of the celestial bodies is made.

D. CLOUDS

The cloud model presented is not effected by the movement of the ambient air mass surrounding it. Once the clouds are positioned they remain in that location throughout the simulation.

VIII. SUMMARY AND CONCLUSIONS

A. CONCLUSIONS

Developing a realistic model of natural phenomena with such as smoke, fire, clouds and dust plumes and sky color is computationally expensive. The resulting images of the complex models are realistic, however the time required to determine the transformations and characteristics of the object primitives prevents their use in interactive simulations. It was therefore, necessary to deviate substantially from the true physical model and rely more on user perception of the image to provide functions usable in real-time applications.

The intent of this thesis was to develop visually realistic environmental effects that could be integrated into the NPSNET battle simulator. Our research concentrated on the development of library of environmental effects usable in real-time graphics simulations. We developed a computationally simple algorithm for generating smoke, flames and dust trails, clouds, lightning and the effects of time passage as it relates to the sun and moon position and sky color. The limitations prevent the models provided from being used in scientific modeling, but they do provide the visually accurate effects usable in the NPSNET.

B. RECOMMENDATIONS

1. C ++

The complete EEL is written in C as a library of functions. The implementation in the Performer version is written as a library of functions that can be integrated into NPSNET-IV, but it does not use the object oriented programming (OOP) approach that makes C++ so powerful. It is recommended the library be changed so they are set up in classes and use the OOP paradigms.

2. Performer

The speed and high quality images achieved using the Performer environment can enhance the simulation. The effort needed to change from the GL to the Performer version is small once the Performer environment is understood. NPSNET can benefit by having the EEL changed to C++ and implemented using Performer.

C. FUTURE DEVELOPEMENT

1. Parallel Processing

The obscurant generators lend themselves to being placed onto multiprocessors by spawning a child to perform the calculations for each column to decrease elapsed time spent computing attribute values. Once parallelized, it will be possible to increase the physical reality of the plume since additional resources will be available for computation.

2. Other Environmental Effects

There are many effects that still remain to be modeled and integrated into NPSNET. They include rain, flooding, weather fronts and snow.

3. Lightning Effects Communications And Destroy Objects

The current lightning implementation gives a visual cue that there is an electrical storm occurring but it has no effect on objects in the world. It is desirable that a method be developed to have the lightning destroy objects and effect communication and other electronic gear when it strikes nearby.

APPENDIX

A. ENVIRONMENTAL EFFECTS LIBRARY MANUAL PAGES

The Environmental Effects Library (EEL) contains the procedures necessary to initialize and simulate smoke, fire, clouds vehicle dusttrails, the passage of time and Night Observation Devices (NOD). Appendix Figure 1 and Appendix Figure 2 give an example of how the functions of the EEL can be used in a simple program. The calls from the EEL have the prefix EEL_ for ease of identification.

```
#include "environ.h"
#include <allothers>
main{
  int number_of_vehicles = 10; //ten vehicles
  float starting_time = 1200.0; //start at noon
  boolean startlightning = false; //no lightning yet
  fov = 450; //field of view
  float lookfx, lookfy, lookfz, looktx, lookty, looktz wvel, wdir;
  initvariables();
  init_graphics();
  EEL_initializelmodel(); //set up EEL light model
  EEL_initializelights(); //set up EEL lights
  EEL_initializematerails(); //set up EEL materials
  EEL_initialize_ smoke(5); //set up the smoke puffs for 5 plumes
  EEL_initialize_flame(5); //set up the flames for 5 plumes
  EEL_initialize_dust(number_of_vehicles); //establish dusttarils for 10 vehicles
  EEL_initialize_cloud(); //set up random clouds
```

Appendix Figure 1. Initialization Portion Of Program Using EEL

The EEL is set up so the function calls can be called independently or in conjuction with any of the other EEL calls once the appropriate initialization routines are executed. The NVD, lightning and passage of time functions have the capability of interacting if used in conjunction with each other.

```
while(TRUE) { //main graphics loop
  getmenustuff() //user inputs
  perspective(fov, 1.25, NEARCLIPPING, FARCLIPPING); //set up perspective
  EEL_pass_the_time(head, &starting_time, 1.0, nod_is_on, lookfx, lookfy, lookfz, looktx,
   looktyl,looktz, FARCLIPPING, fov/10.0); //setup to pass the time
  EEL_generate_cloud(); //render the clouds
  dosomestuff(); //there is more than EEL
   if(startlightning) then
    //set up the lighning to start with root node at (0, 200, 0) with the first
    //angle to be 6.14 rads from the horizontal, any branch falling outside the
    //box with sides 2000 on a side will stop and there is a 99 percent chance
    //each node will have a left and right branch.
    EEL_do_lightning(0.0, 200.0, 0.0, 6.14, 0., 1000., 1000., .999);
  shootsomemissiles(missile);
  if(missilehitstank) then
    EEL_use_next_smoke(tank[x], tank[y], tank[z], 25.)//generate smoke at tank posit for25
      secs
    EEL_use_next_flame(tank[x], tank[y], tank[z], 25.)//generate flame at tank posit for 25
      secs
  pushmatrix();
    translate(tank.x, tank.y, tank.z);
    rot(tank.heading);
    display_this_object(tank);
  popmatrix();
  //generate dust behind tank using dustplume[1], that uses stoppingdust[1], //withwvel and
    // wdir, the tank is located at (tank.x, tank.y, tank.z)
    //traveling at tank.velocity, on a sandy road made of light dust.
  EEL_generate_dust(dustplume[1].dustcol1, &dustplume[1].stpuff, wvel, wdir, tank.x,
tank.y,
     tank.z, tank.velocity, 0.5, 1.);
  if(nods_are_on) then EEL_don_nvg(50) else EEL_doff_nvg(); //don the nods if you want /
//to
  EEL_generate_all_smoke(); //render the smoke
  EEL_generate_all_flame(); //render the flames
  switch_buffers(); //switch the drawing buffers
 }//end graphics
}//end main
```

Appendix Figure 2. Main Graphics Loop Of Program Using EEL

B. SMOKE GENERATOR

The smoke generator consists of functions used to create plumes of smoke puffs at the position and for the time duration inputted by the user. To generate smoke the user initializes the smoke generator, associates a smoke plume with a specific location and then renders the smoke.

NAME

initialize_smoke - Used to initialize the smoke generator.

use_next_smoke - Associates a plume of smoke with a location in the simulation.

generate_all_smoke - Generates all the plumes given a location and duration time.

C SPECIFICATION

#include "environ.h"

void **initialize_smoke(int number of plumes)**

void **use_next_smoke(float x, y, z, duration)**

void **generate_all_smoke()**

PARAMETERS

number_of_plumes	Number of plumes available at any one time
x, y, z	The location of the plume.
duration	Time in second the plume will exist.

DESCRIPTION

To initialize the smoke generator, the algorithm **initialize_smoke** must first be called to establish the number of smoke plumes needed, initialize the attribute values of the smoke puffs and identify the texture mapped to the individual smoke puffs.

A plume is associated with a given location by making a call to the function **use_next_smoke**. The function traverses the array of all plumes for the next available

unused plume and sets the status to "active" and the "duration" in seconds to the amount desired by the user.

The function **generate_all_smoke** is called after the locations have been defined for each of the plumes by **use_next_smoke** during each pass through the graphics loop. The function traverses the array of all columns and renders the puffs of the active plumes.

NOTES

If the next available plume is in use when this call is made, it will be stopped, reset and started at location x, y, z.

C. FLAME GENERATOR

The flame generator consists of functions needed to create plumes of flame puffs at the position and for the duration inputted by the user. To generate flames the user initializes the flame generator, associates a flame plume with a specific location and then renders the flame.

NAME

initialize_flame - Used to initialize the flame generator.

use_next_flame - Associates a plume of smoke with a location in the simulation.

generate_all_flame - Generates all the plumes given a location and duration time.

C SPECIFICATION

#include "environ.h"

void **initialize_flame**(int number of plumes)

void **use_next_flame**(float x, y, z, duration)

void **generate_all_flame**()

PARAMETERS

number_of_plumes	Number of plumes available at any one time
x, y, z	The location of the plume.
duration	Time in second the plume will exist.

DESCRIPTION

To initialize the flame generator, the algorithm **initialize_flame** must first be called to establish the number of flame plumes needed, initialize the attribute values of the flame puffs and identify the texture mapped to the individual flame puffs.

A plume is associated with a given location by making a call to the function **use_next_flame**. The function traverses the array of all plumes for the next available

unused plume and sets the status to "active" and the "duration" in seconds to the amount desired by the user.

The function **generate_all_smoke** is called after the locations have been defined for each of the plumes by **use_next_smoke** during each pass through the graphics loop. The function traverses the array of all columns and renders the puffs of the active plumes.

D. DUST GENERATOR

NAME

initialize_dust - Initializes the vehicle dust trail generator.

generate_dust - Generates the dust associated with a given vehicle.

C SPECIFICATION

#include "environ.h"

void **initialize_dust**(int number of trails)

void **generate_dust**(DUST dustcol, int *stoppingdust, float windvel, winddirection, posx, posy, posz, speed, dustindex, roadtype)

PARAMETERS

number of trails	Number available during the simulation at any one time.
dustcol	Identifies which column is associated with a vehicle.
stoppingdust	A counter associated with a specific plume.
windvel	The speed of the wind in meters per second.
winddirection	The compass direction in radians.
posx, posy, posz	The position of the vehicle.
speed	The speed of the associated vehicle
dustindex	The index of the soil from Appendix Table 1.
roadtype	The index of the road from Appendix Table 1.

DESCRIPTION

The algorithms to generate a vehicular dust trial are used by first initializing the generator with **initialize_dust** to establishes the number of trails to be used and set the attributes of the individual puffs used in creating the trails.

A specific trail is associated with a vehicle by making the call **generate_dust**. The parameters **dustcol** and **stoppingdust** are values associated with the individual trail to be used and are set by the generator algorithm. The **windvelocity** is the speed of the wind in

kilometers per minute, **winddirection** is the direction of the wind in radians with 0.0 representing true north. The parameters **posx, posy, posz, speed** are the position of the vehicle and its velocity in meters per second. The **dustindex** and **roadtype** indicate the type of soil and road type being traversed by the vehicle. Typical soil and road types are defined in Appendix Table 1.

APPENDIX TABLE 1. INDEX FOR DUST AND ROAD TYPES

dustindex	dust type	roadtype	road type
0.0	no dust	0.0	paved
0.5	dry clay	0.5	clay
1.0	light sand	1.0	sand

E. GENERATE CLOUDS

The cloud generator consists of an initialization routine that establishes the structures and defines the texture used in generating the clouds. A number of clouds are generated with random positions in the world and rendered in those positions by the **generate_cloud** function.

NAME

initialize_cloud - Used to initialize the cloud generator.

generate_cloud - Renders the clouds in a random position in the sky

C SPECIFICATION

#include "environ.h"

void **initialize_cloud**();

void **generate_cloud**();

DESCRIPTION

Initializing the cloud generator by calling **initialize_cloud** defines the cloud texture and sets the location of the clouds with random positions. Once the generator is initialized a call to **generate_cloud** renders the cloud puffs in the random positions set in the initialization routine.

F. LIGHTNING

NAME

do_lightning - Generates lightning bolts of varying densities at random locations.

C SPECIFICATION

#include "environ.h"

void **do_lightning**(double x, y, z, angle, ground elevation, x_boundary, z_boundary, density)

PARAMETERS

x, y, z	Position of the root node of the lightning bolt.
angle	initial angle the branch makes with the ground plane.
ground elevation	Elevation of the ground in world coordinates.
x_boundry, z_boundry	Absolute value of the bounding volume of the world.
density	Probability node will have a left or right child.

DESCRIPTION

Lightning is generated by calling the function **do_lightning**. The parameters **x**, **y**, and **z** represent the position of the root node of the lightning bolt. The **angle** parameter is any random value between 0 and 45. It is the angle the branch of each bolt makes with the normal to the ground. The **ground elevation**, **x boundary** and z **boundary** parameters define the volume the lightning is to be maintain in. The **density** parameter establishes the probability a node has a left or right leaf attached to it.

NOTES

When a call to **do_lightning** is made and there is a lightning flash the Night Vision Devices will be effected and will wash-out.

SEE ALSO

don_nvg

G. NIGHT OBSERVATION DEVICES

The night observation device functions are used to simulate an illumination intensification system. They can be used in conjunction with the passage of time function to enable the user to view items when the amount of light in the scene is low.

NAME

don_nvg - Used to simulate night observation devices.

doff_nvg - Used to remove the effects of the night vision goggles.

C SPECIFICATION

#include "environ.h"

void **don_nvg**(int density);

void **doff_nvg**();

PARAMETERS

density	Determines the amount of noise visible in the NOD's

DESCRIPTION

The night observation devices are turned on by making a call to the function **don_nvg**. The **density** parameter defines the amount of random noise rendered in the goggles of the NODs. The call changes the value of the defined light and must be placed just prior to switching the drawing buffer at the end of the graphics loop. To remove the NOD's and reestablish the ambient lighting conditions the call **doff_nvg** is made.

H. TIME

The passage of time functions are used to position the Sun and Moon and determine the amount of light in the scene. Additionally it determines the color of the horizon as a function of the time of day.

NAME

pass_the_time - Used to simulate the effects of time passage.

C SPECIFICATION

#include "environ.h"

void **pass_the_time**(float heading, double *startingtime, int nods_are_on, float minutesperminute, veiwx, viewy, viewz, refx, refy, refz, farclippingdistance, fieldofview)

PARAMETERS

heading	Direction the viewer is headed.
startingtime	The starting time of the simulation.
nods_are_on	Boolean used to determine if the NOD's are donned.
minutesperminute	Used to scale time in the simulation.
veiwx, viewy, viewz	The eye position of the viewer.
refx, refy, refz	The reference point of the viewer.
farclippingdistance	Distance from the viewer to the farclippingplane.
fieldofview	The viewers field of view.

DESCRIPTION

A call to **pass_the_time** updates the world time, positions the celestial bodies and renders the horizon polygon. The parameter **heading** represents the heading of the viewer in degrees. The parameter ***startingtime** represents the world time the play commences. The parameter **nods_are_on** is a variable defined in "environ.h" and is set as a function of nod use and it is sufficient to use the actual variable **nods_are_on**. The parameters **veiwx,**

viewy, viewz, refx, refy, refz, farclippingdistance, fieldofview refer to the viewing and reference coordinates of the viewer and the distance to the farclipping plane and the field of view of the viewing volume.

NOTES

If the NOD's are being used and the amount of light in the scene becomes high enough then the goggles will be washed-out until the light level falls below an appropriate level.

SEE ALSO

don_nvg

I. INITIALIZING LIGHTING AND MATERIALS

The environmental effects library uses a lightmodel, defined light and materials that are defined in environ.h and must be initiaized prior to their use. If another light is defined and used then the EEL function involving light may be effected.

NAME

initializelmodel - initializes the lightmodel used with the EEL.

initializelights - defines the light used in the EEL.

initializematerials - defines the materials used in the EEL.

C SPECIFICATION

#include "environ.h"

void **initializelmodel**()

void **initializelights**();

void **initializematerials**()

DESCRIPTION

The light model is defined and bound to the attributes maintained in the array MYMODEL and is created by making the call **initializelmodel**. The light is defined and bound to the lighting properties in LIGHT by making the call **initializelights**. The property arrays are defined in the file environ.c.

The materials used to define the underlying polygons for the obscurant puffs and the material used to wash out the NODs is defined by making the call to **initializematerials**. The material properties are maintained in the arrays lightmaterial and nodmaterial. The property arrays are defined in the file environ.c.

LIST OF REFERENCES

[ANDR93] Andrew, W., Baraff, D., "Differential *Equations Basics,*" paper presented at SIGGRPH93 20th International Conference on Computer Graphics and Interactive Techniques, Anaheim, California, 1 to 6 August 1993, pp. B1 - B8.

[BEYC79] Beychok, Milton, R., *"Fundamentals of Gas Stack Dispersion,"* Milton R. Beycock Consulting Engineer, 63 Oak Tree Lane, Irvine, California, February 1979, pp. I5 - IX8.

[BRIG69] Briggs, Gary, A., *"Plume Rise,"* Air Resources Atmospheric Turbulence and Diffusion Laboratory Environmental Science Services Administration, Oak Ridge National Laboratory, 1969, pp. 5 - 15.

[CAMP77] Campbell, Ian, M., *"Energy and the Atmosphere A Physical-Chemical Approach,"* 2nd Edition, John Wiley & Sons LTD, New York, 1977, pp. pp. 79 - 120.

[DARP89] Defense Advanced Research Projects Agency, Report No. 7102, *"The SIMNET Network and Protocols,"* by A. Pope, July 1989.

[GARD85] Gardner, Geoffrey, Y., "Visual Simulation Of Clouds," *Computer Graphics Proceeding*, Vol. 19, No. 3, 1985, pp. 297 - 303.

[GARD92] Gardner, Geoffrey, Y., *"Battlefield Obscurants Final Technical Report,"* Grumman Data Systems Corporation, Woodbury, New York, September 1992, pp. 8 - 31.

[IST91] Institute for Simulation & Training, "Military *Standard Protocol Data Units For Entity Information and Entity Interaction in a Distributive Interactive Simulation,"* Final Draft, Institute for Simulation & Training, Orlando, Florida, October 1991, pp. 47 - 50.

[ISTA93] University of Central Florida Institute for Simulation And Training Draft 2.2, *Distributed Interactive Simulation Operational Concept*, by B. McDonald, March 1993.

[ISTB93] Institute for Simulation & Training, "Proposed *IEEE Standard Draft for Information Technology- Protocols for Distributed Interactive Simulation Applications,"* Version 2.0 Third Draft, Institute for Simulation and Training, Orlando, Florida, May 1993.

[KLAS87] Klassen, V. R., "Modeling the Effects of the Atmosphere on Light," *acm Transaction on Graphics*, Vol. 6, No 3, July 1987, pp. 215-237.

[LESS93] Lesser, Roger, "Night Vision Technology Continues To Improve," *Defense Electronics*, January 1993, Vol. 25, No. 1, pp. 45 - 48.

[MILL78] Mills, H.R., *Positional Astronomy and Astro-Navigation Made Easy,"* John Wiley & Sons New York., 1978.

[NISH93] Nishita, Tomoyuki and others, "Display of The Earth Taking into Account Atmospheric Scattering," *Computer Graphics Proceedings*, Annual Conference Series 1993, ACM SIGGRAPH 1-6 August 1993, pp. 175-182.

[PRAT93] Pratt, David, R., *Construction and Management of Real-Time Virtual Worlds*, Doctoral Dissertation, Naval Postgraduate School, Monterey, California, June 1993.

[REEV83] Reeves, William T. "Particle Systems--A Technique for Modeling a Class of Fuzzy Objects," *acm Transaction On Graphics*, Vol. 2, No. 2, April 1983, pp. 359 - 376.

[SCHA81] Schaefer, Vincent J., Day, John, A., "A *Field Guide to the Atmosphere*," Houghton Mifflin Company, Boston, Massachusetts, 1981, pp. 155 - 173.

[SCHM93] Schmidt, D., *NPSNET: A Graphical Based Expert System to Model P-3 Aircraft Interaction With Submarines and Ships*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1993, p. 56.

[SGIA92] Silicon Graphics, Inc. Document Number 007-1210-040, *Graphics Library Programming Guide*, by P. McLendon, March 1992.

[SGIB92] Silicon Graphics, Inc. Document Number 007-1680-010, *IRIS Performer Programming Guide*, by P. McLendon, September 1992.

[THOR87] Thorp, Jack A., "The *New Technology of Large Scale Simulator Networking: Implications for Mastering the Art of Warfighting*," Proceedings of the 9th Interservice/Industry Training System Conference, November - December 1987.

[USNO87] U.S Naval Observatory, Washington D.C., "*Computer Programs for Sun and Moon Illuminance With Contingent Tables and Diagrams*," Circular No. 171, February 19, 1987, pp. 3 - 24.

[WEJC91] Wejchert, Jakub, Haumann, David, "Animation Aerodynamics," *Particle System Modeling, Animation and Physically Based Techniques Course Notes 16*, ACM SIGGRAPH'92, July 1992, pp. 2-12 - 2-21.

[ZYDA93] Zyda, Michael, J., Pratt, David, R., Falby, John, S., Lombardo, Chuck, Kelleher, Kristen, M., "The Software Required for the Computer Generation of Virtual Environments", *Presence*, Vol. 2, No. 1, 1994.

INITIAL DISTRIBUTION LIST

Defense Technical Information Center Cameron Station Alexandria, Va 22304-6145	2
Dudley Knox Library Code 052 Naval Postgraduate School Monterey, Ca 93943	2
Dr. David R, Pratt Code CS/PR Computer Science Department Naval Postgraduate School Monterey, Ca 93943	4
Dr. Michael Zyda, Code CS/ZK Computer Science Depatment Naval Postgraduate School Monterey, Ca 93943	5
Mr. Jeffery Turner Topographic Engineering Center Fort Belvoir, Va 22304	1
Mr. Stanley Goodman U.S. Army Simulation, Training and Instrumentation Command 12350 Research Parkway Orlando, Fl 32826-3276	1