REAL-TIME PHIGS: APPLYING GRAPHICS STANDARDS TO SIMULATION

Brian S. Heaney
Star Technologies, Incorporated
Graphicon Products Division
Research Triangle Park, NC

ABSTRACT

The increasing cost of simulation software development and visual database generation is threatening to offset the dramatic cost reductions of image generation hardware. Sharing software and databases between simulator programs, through the use of a standardized graphics language, clearly reduces program cost and development time. This paper describes applying the PHIGS standardized graphics language to simulator database and software development. PHIGS (Programmer's Hierarchical Interactive Graphics System) is a well-architected graphics language which provides a consistent framework for database creation, editing, and archival. Key features include database hierarchy, model instantiation, a rich set of primitives and attributes, powerful database editing, and standardized archival methods. PHIGS does not, however, support some important real-time simulator features such as level of detail, overload management, texturing, and mission functions. The Real-Time PHIGS (RTP) implementation described in this paper supports real-time simulator applications through several extensions to the PHIGS standard.

INTRODUCTION

This paper describes an approach to reducing both database and software development costs associated with visual simulators through the application of the PHIGS standardized graphics language. PHIGS (Programmer's Hierarchical Interactive Graphics System) is a general-purpose 3D graphics system designed for mainstream applications (e.g., CAD/CAM). As a result, PHIGS does not have some of the important features required for real-time simulators and trainers. Star Technologies' Real-Time PHIGS (RTP) implementation on the Graphicon 2000 image generator has significant enhancements that extend the PHIGS framework into the simulator domain.

An overview of PHIGS is presented, after providing background information on the need for standards. The details of the Real-Time PHIGS extensions are then described.

BACKGROUND

In this era of tight government budgets, program cost reduction has become a major objective of government procurement agencies. The image-generator industry, where the price/performance ratio of IG *hardware* has fallen dramatically in the past ten years [4], has responded to the challenge.

But IG hardware is only one cost component in a training simulator. Other significant cost items include:

- crewstation and instructor/operator console
- database development
- real-time software
- training software
- program management and documentation

There have not been equivalent cost reductions for these items. In fact, since software productivity has not kept pace with increases in the size and complexity of databases and real-time software, database and software development costs have *risen* in many programs.

Sharing software and databases between simulator programs through the use of a standardized graphics language would clearly reduce program cost and development time. Unfortunately, each image-generator vendor has unique software and database formats, and most programs have unique database and software requirements.

In an attempt to promote database sharing and reuse, the government has recently initiated two important database projects:

Project 2851, whose mission is to develop *standard simulator databases* (SSDB's) that can be transformed to run on many image generators.

Rapidly Reconfigurable Data Base, whose mission is to provide geographically specific mission rehearsal within 72 hours.

Project 2851 is developing database modeling tools and the database transformation programs that are to be used by future training programs. These programs, however, do not address development costs associated with the real-time software that manipulates the database and controls the IG.

The commercial computer-graphics industry, faced with the same problems of database incompatibility and software obsolescence, has been aggressively pursuing standards. X-windows is becoming the industry standard for windowing and user-interface systems.

PHIGS is rapidly emerging as the 3D graphics standard of choice, having been adopted by Sun Microsystems, Hewlett-Packard, Digital Equipment Corporation, Tecktronix, and Apollo. On the horizon is PEX (PHIGS Extensions to X), which offers the promise of network-transparent 3D graphics in an integrated windowed environment.

PHIGS is the only approved standard that exists for hierarchical 3D graphics, having been approved by ANSI (American National Standards Institute) and ISO (International Standards Organization). Lighting and shading extensions to PHIGS, called PHIGS PLUS, are currently winding its way through the standardization process, and should be approved next year. PHIGS is particularly well-suited for database modeling applications.

This standardization is resulting in the following trends in the graphics industry:

- an increase in the number of vendors supporting the standard language
- an increase in third-party software packages supporting the standard language
- increased sharing of software and databases
- the training of a pool of software developers skilled in the language

The result is the reduction of graphics equipment cost and an increase in productivity as software and databases are reused.

The government, recognizing this trend, is beginning to require standards such as PHIGS for mission planning and command-control-communication-intelligence (C^3I) programs. To date, however, there has not been a bridge between inexpensive PHIGS modeling stations and real-time simulators.

PHIGS

The Programmer's Hierarchical Interactive Graphics System provides a set of functions for:

- definition, display, and modification of 3D graphical data
- definition, display, and manipulation of geometrically related objects
- modification of graphics data and the relationships between graphics data [2]

These functions provide a standard interface between the application program and a graphics device. The layered model shown in Figure 1 illustrates the role of the PHIGS subroutine library in a graphics system [2].

In order to support a wide variety of hardware platforms while providing standard features, the PHIGS database model has two components: a device-independent *centralized structure store* (CSS), and

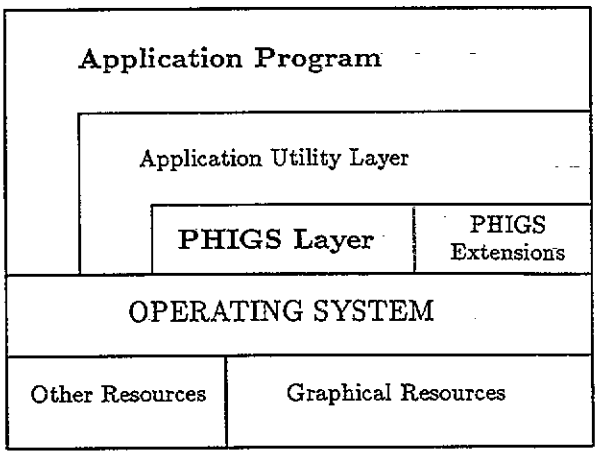

Figure 1: **Layer Model of PHIGS.**

device-dependent *workstations*. The CSS is the *display program* containing the 3D graphical and hierarchical definition of the graphics database. Workstations provide a display surface, a set of device-dependent display characteristics, and a mechanism for mapping the 3D graphical data in the CSS to the 2D display. Graphical output on a workstation is produced by traversing and interpreting the CSS display program.

Centralized Structure Store

PHIGS supports the storage and manipulation of data in a centralized hierarchical data structure, known as the *centralized structure store* (CSS). The fundamental entity of data is a *structure element* and these are grouped together into units called *structures*. Structures are organized as acyclic directed graphs called *structure networks*. Functions are provided to support searching and inquiry of the content and topology of structure networks and the CSS in general. The graphical output generated by PHIGS is built up from two groups of basic elements called *output primitives* and *primitive attributes*.

The following subsections describe the details of the PHIGS functions provided for creating and manipulating the centralized structure store, including: structure editing, structure hierarchy, output primitives, primitive attributes, modeling transformations and structure archival.

Structure Editing. One of the most attractive and powerful features of PHIGS is the *structure editor*. The PHIGS editing model is analogous to the familiar text editor (e.g., EDT, VI, EMACS). In this analogy, a PHIGS structure is analogous to a file and PHIGS structure elements are analogous to words. PHIGS provides an *'element pointer'* (cursor), where all edits

to a structure (file) are relative to the location of the 'element pointer' (cursor). Editing functions are available to insert new structure elements (words), replace elements with new structure elements, delete structure elements, navigate within a structure and inquire structure element content.

Figure 2 illustrates a simple PHIGS editing example. The application program first changes the line color, replacing the "RED line color" element with a "GREEN line color" element. Then the application inserts a transformation matrix to rotate the object around the X-axis by 30 degrees.

Structure Hierarchy. As shown in Figure 3, structures may contain invocations of other structures, resulting in a directed acyclic graph. The invocation of a structure is achieved using the *"execute structure"* element, which is analogous to a subroutine call. Structure networks can be edited, manipulated, and deleted, providing the application program significant flexibility in manipulating the topology of the structure network.

When a structure contains invocations of other structures, it a is *parent* of those structures. Likewise, each of the invoked structures a *child* of that parent. In Figure 3, structure 2 is the parent of structures 4, 5 and 6; structures 2 and 3 are children of structure 1. Also, structure 1 and 2 are *ancestors* of structures 4, 5 and 6; while structure 6 is a *descendant* of structures 1, 2 and 3.

Output Primitives. PHIGS provides a set of basic output primitives that are displayed on a workstation during structure traversal. The PHIGS primitives include [2]:

POLYLINE: A set of connected lines defined by a point sequence.

POLYMARKER: Symbols of one type centered at positions defined by a point sequence.

TEXT: A character string at a given position in modeling coordinate space.

FILL AREA (polygon): A single polygonal area which may be filled with a uniform color, pattern or hatch style.

FILL AREA SET: A set of polygonal areas. This allows for specifying areas with holes or disjoint regions.

PHIGS PLUS has significantly enhanced the output primitives supported by PHIGS, adding support for lighting, shading, tessellated surfaces, and parametric surfaces [1]. The PHIGS PLUS primitives include:

FILL AREA WITH DATA: The polygon definition can contain any combination of vertex colors, vertex normals, a facet color and a facet normal.

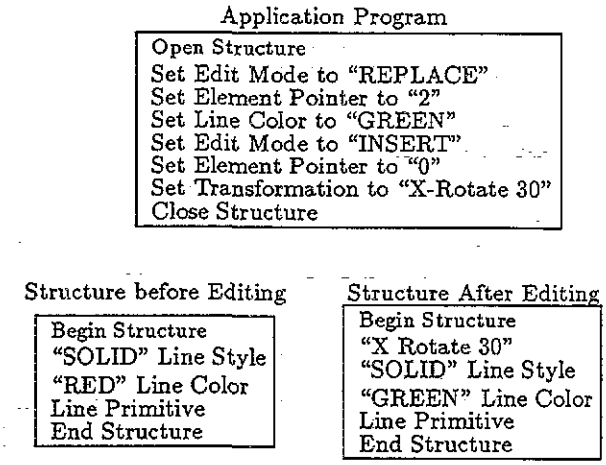

Figure 2: **PHIGS Editing Example.**

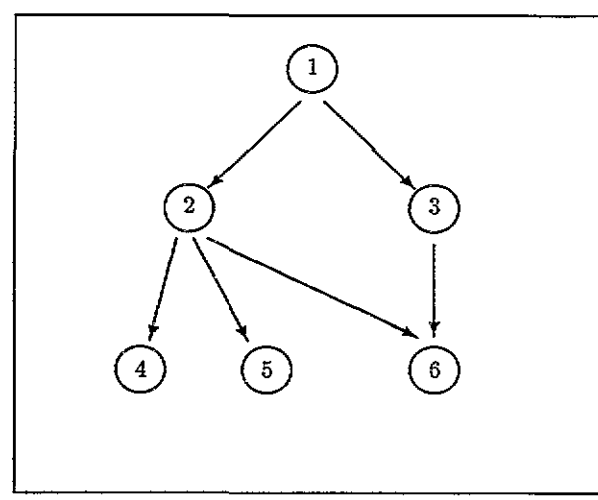

Figure 3: **Hierarchical Structure Network.**

TRIANGLE STRIP WITH DATA: A sequence of $N-2$ triangles from a list of N vertices. The primitive definition can contain any combination of vertex colors, vertex normals, facet colors and facet normals.

QUADRILATERAL MESH WITH DATA: A sequence of $(M-1) \times (N-1)$ quadrilaterals from a two-dimensional array of $M \times N$ vertices. The primitive definition can contain any combination of vertex colors, vertex normals, facet colors and facet normals.

POLYHEDRON WITH DATA: A group of facets generated from a list of indices into a single vertex list. The primitive definition can contain any combination of vertex colors, vertex normals, facet colors and facet normals.

NON-UNIFORM B-SPLINE SURFACE: A non-uniform B-spline surface of two specified orders in two independent parameters based on a list of knots for each parameter and a grid of control points. Either a rational or a non-rational B-spline can be selected.

PARAMETRIC POLYNOMIAL SURFACE: A uniform parametric polynomial surface of two specified orders in two independent parameters.

Primitive Attributes. As shown in Table 1, PHIGS provides a rich set of primitive attributes. Each primitive attribute has an associated function which results in a single structure element. These attributes are bound to the corresponding primitives at traversal time.

During traversal, the current values of these attributes are saved (pushed onto the stack) when a structure is entered and are restored (popped off the stack) when the traversal of that structure completes. A child structure therefore *inherits* attributes from its parent, but cannot modify the attributes of the parent.

Modeling Transformations. In PHIGS, the application programmer can compose a graphical picture from separate parts, each of which can be defined within its own Modeling Coordinate (MC) system. The relative positioning of the separate parts is achieved by having a single World Coordinate (WC) space onto which all the defined modeling coordinates are mapped [2].

The composite modeling transformation is formed during traversal from the hierarchy of the encountered transformation matrices. The composite modeling transformation is implicitly saved and restored when a child structure is executed. As a result, the composite transformation is inherited by descendant structures and child structures cannot modify the parent's composite transformation.

Structure Archival. A key feature of PHIGS is the standardization of database archival (storage). PHIGS provides functions for archiving from the centralized structure store to an archive file, retrieving from an archive file to the CSS, or deleting structures from an archive file.

PRIMITIVES	ATTRIBUTES
Polyline:	Line Type Line Width Scale Factor Polyline Color Line Shading Method
Polymarker:	Marker Type Marker Size Scale Factor Marker Color
Fill Area:	Interior Style Interior Color Edge Flag Edge Type Edge Width Scale Factor Edge Color
PHIGS PLUS Facets:	Interior Style Interior Color Interior Shading Method Ambient Reflection Coeff Diffuse Reflection Coeff Specular Reflection Coeff Specular Color Specular Exponent Transparency Coefficient Reflectance Equation Edge Flag Edge Type Edge Width Scale Factor Edge Color

Table 1. **Primitive Attributes.**

By having a standardized archival mechanism, databases can be easily transferred from one hardware platform to the next. A PHIGS-based database modeling lab could have many hardware platforms and all would be able to access each other's databases through the PHIGS archive files. This would clearly promote database sharing since databases would not have to be painstakingly translated from one vendor's format to anothers.

Workstations

PHIGS workstations are conceptually analogous to an image generator's "channel." The PHIGS workstations are the abstract interface through which the application program controls physical devices. An application may have more than one workstation (channel) open and may open more that one workstation type at a given time. Data in the centralized structure store may be displayed on any open workstation. An

abstract PHIGS workstation has the following capabilities:

- one rectangular display surface of fixed resolution;
- supports several line types, text fonts, character sets, colors, etc., to allow output primitives to be drawn with different attributes;
- has access to the centralized structure store and can traverse and display structure networks when directed to do so; and
- permits the definition of several transformations which support 3D viewing operations.

The application can determine the characteristics and capabilities of a particular workstation and adapt its behavior accordingly. For example, a simulator application running on a workstation (channel) that supports real-time display of 3000 textured polygons per frame would load a database with textured polygons with the appropriate scene density. That *same* application running on another workstation (channel) that supports only 1000 non-textured polygons would load a simpler, non-textured database. This clearly illustrates how PHIGS (and RTP) facilitate the sharing of code, since the same simulator application can be run on many different workstations.

PHIGS PLUS Lighting and Shading

PHIGS PLUS adds the key features of lighting and shading to the PHIGS rendering model. As with other PHIGS attributes, lighting and shading are independently selectable.

Lighting is applied on a primitive-by-primitive basis; no interactions between objects such as shadows and reflections are defined [1]. Each workstation has a light source table, where each entry in the table defines a single light source. The *"set light source state" structure element* in the CSS determines which light source(s) are active. The defined light sources and their characteristics are:

AMBIENT: Color that illuminates polygons independent of their orientation and position.

DIRECTIONAL: Color and direction that illuminate polygons based on the primitive's orientation, but independent of their orientation.

POSITIONAL: Color, position and direction that illuminate polygons based on the primitive's position and orientation.

SPOT: Color, position, direction, attenuation coefficient, concentration exponent and spread angle. A point that lies outside the spot light's cone of influence is not illuminated by the light source.

The shading portion of the rendering pipeline takes a primitive's geometric and color information and produces interpolated geometric and color information across the primitive. The shading method, selected by the *"set shading method"* structure element, specifies which aspects of a primitive are interpolated. The defined types of polygon shading methods are:

NONE: A lighting calculation is performed for each facet to produce a reflected color per facet. Also referred to as *constant* or *flat* shading.

COLOR: A lighting calculation is performed at each vertex of a primitive, using the primitive's color and vertex normals. The resulting colors are interpreted across the primitive. Also referred to as *smooth* or *Gouraud* shading.

DOT: The dot product between the vertex normal and light source are computed at each vertex. The dot product and diffuse colors are interpolated across the primitive. The lighting calculation is performed at each pixel using the interpolated dot product and color.

NORMAL: The vertex normal and diffuse color are interpolated across the primitive. The lighting calculation is performed at each pixel based on the interpolated normal and color. Also referred to as *Phong* shading.

REAL-TIME PHIGS EXTENSIONS

As indicated in the introduction, PHIGS is flexible and general-purpose graphics standard oriented towards the mainstream of 3D graphics applications. As a result, PHIGS (with the PHIGS PLUS additions) is particularly well-suited for database modeling. PHIGS, however, does not provide some of the functions required in special-purpose applications such as real-time simulators and trainers. The PHIGS committee realized that PHIGS could not be "all things to all people" and made provisions for extensions. Guidelines have been established for allowing vendors to add non-standard features in a standardized manner. These guidelines promote the portability of extensions and ensure that fundamental PHIGS concepts are not violated.

Star Technologies, with the Real-Time Phigs (RTP) graphics library running on the Graphicon 2000, has developed a PHIGS implementation with several key extensions suited for real-time simulator applications. These real-time extensions are fully consistent with the PHIGS framework. Therefore, databases generated on any PHIGS-based modeling station could easily be ported to the Real-Time Phigs environment and used for training and simulation applications. Table 2 outlines some of the attribute extensions provided by RTP.

RTP takes full advantage of the hardware features provided by the Graphicon 2000 image generator. Described in [3], the G2000 IG is a modular board-set architecture that nominally renders 3000 textured, anti-aliased, smooth-shaded polygons at a 30Hz frame rate.

PRIMITIVES	ATTRIBUTES
Polyline:	Line Anti-Alias Method
Light Points:	Point Anti-Alias Method Light Fade Method Light Blink Rate
RTP Facets:	Texture Map *Texture-Coord. Matrix* Texture Folding Method Texture Transparency Edge Anti-Alias Method Haze Method Spot-Light Mode Edge-On Fading Method

Table 2. **RTP Extended Primitive Attributes.**

The following sections describe the key extensions provided by the RTP language. The functions provided by these extensions and how they fit within the PHIGS framework is presented in detail.

Texturing

The Graphicon 2000 IG supports photo-derived texturing with up to 96 texture maps, each at multiple levels of detail. Each texture map can contain both color and transparency information. RTP provides four categories of texture functions: map loading, map selection, textured primitives and texture-coordinate transformation.

Texture-Map Loading. The texture map is loaded into the workstation (channel) specified by the application. Therefore, different maps can be loaded into different channels in a multiple-channel application. A texture map has attributes that include the map identifier, the map size and a transparency flag indicating the existence of transparency information.

Texture-Map Selection. RTP provides structure elements for selecting a texture map and for determining whether the map should fold when the map repeats. The database modeler has the ability to change texture maps at any level of granularity that best suits the application (e.g., between each polygon, between each structure, etc).

Textured Primitives. A textured polygon is distinguished from a non-textured polygon by the existence of texture-map <u,v> coordinates at each vertex. RTP provides for flat-shaded, smooth-shaded and user-shaded textured polygons.

Texture-Coordinate Transformation. RTP provides a 3x3 matrix used to transform a primitive's <u,v> coordinates. This texture transformation can be hierarchically modified with structure elements in the CSS (database) in a manner similar to modifying the modeling transformation. Effects such as flowing water, rotor wash and billowing smoke can be simulated by dynamically updating this transformation.

Anti-aliasing

The Graphicon 2000 provides anti-aliasing using a variable-width filter and a 4x4 subpixel mask. Both polygon edges and vectors (for heads-up displays) can be anti-aliased. RTP provides a structure element allowing the modeler to choose the anti-alias filter appropriate for the application.

Atmospheric Haze

The G2000 performs haze calculations on a per-vertex basis after lighting calculations have been performed. This haze-attenuated color is linearly interpolated across the face. RTP provides two haze models: one for surface-based applications and one for airborne applications. In either case, the haze model coefficients are specified on a per-workstation (channel) basis, allowing different haze characteristics in different channels. The surface-based haze uses a piece-wise linear approximation of a exponential atmospheric model. The more complex airborne haze function is based on a multi-layer atmospheric model. The air is modeled with variable-height density layers, where each layer has a programmable altitude and visibility range.

Level-of-Detail Blending

RTP supports the use multiple levels of model detail as its primary mechanism for maximizing scene content. It is important to transition from one level of detail to the next in a smooth manner, avoiding any distracting popping effect. This smooth blending during LOD transitions is supported by the RTP through the specification of transition-zone range rings. The mechanism provided is the *"LOD execute structure"* element, which is an extension of the PHIGS "execute structure" construct.

Point Lights

RTP has extended the PHIGS primitive set to include point light primitives, which are required for night-time out-the-window imagery. These point light primitives include:

- light strings, with pseudo-random geometric perturbations
- rotating and strobe lights, with programmable periods
- sequenced lights
- directional lights (VASI's, PAPI's, REIL's, etc.)

- self-luminous polygons (windows)

Each of these lights is accurately attenuated by the atmosphere using the selected haze function.

Landing Lights and Headlights

Landing lights and headlights are implemented using the PHIGS PLUS *spot* light source, which is characterized by color, position, direction, attenuation coefficient, concentration exponent and spread angle. Only vertices within the light's cone of influence will be illuminated.

This mechanism, by itself, is not sufficiently accurate for realistic head light effects. Runways and terrain faces are usually very large polygons, where the per-vertex spot-light calculations would result in vaguely-defined highlights. This is unacceptable in a training environment, where the shape of the light's "edge" is an important visual cue and must be well defined.

As an extension to PHIGS, the RTP provides a structure element used to enable or disable the "special spot-light processing" mode. Normally, only runways or terrain polygons would be so marked. Marked polygons whose illumination gradient exceed a programmable threshold are recursively split into a set of self-similar triangles called *micropolygons*. The spot-light illumination calculation is then performed at each created vertex, resulting in accurate landing-light visual effects.

Moving Models

PHIGS does not provide any explicit mechanism for moving models, which is a serious shortcoming in simulator applications. As an extension to PHIGS, the RTP maintains a *moving-model state list (MMSL)*. Each entry in the MMSL is defined by the model's structure id, world-space position and world-space orientation. Moving models are defined as standard PHIGS structures and structure networks, having all the characteristics of other structures in the database. Each workstation (channel) has a list of active moving models. In this scheme, the application program updates the position and orientation of a moving model once in the MMSL and all workstations (channels) will reflect that change when the next frame is displayed. Based on this low-overhead approach, the G2000 can support up to 100 moving models in real time.

Gridded Terrain

PHIGS implicitly supports terrain through the use of fill areas or fill area sets. This is not adequate for training applications, where gridded terrain must be a specialized data structure to efficiently support culling, priority determination, moving model assignment, and height-above-terrain feedback.

RTP supports two specialized terrain structure elements: regularly gridded terrain and irregularly gridded terrain. The *"regularly gridded terrain"* element is defined by a reference point, a grid size, a list of elevation points and grid children. These child structures contain definitions of the culture and objects on a given terrain face. The *"irregularly gridded terrain"* element is defined by a list of contiguous, axis-aligned rectangles and associated child structures.

Mission Functions

RTP has extended PHIGS inquiry functions to support a comprehensive set of specialized mission functions, including:

- Collision detection
- Hierarchical collision detection
- Missile impact detection
- Height above terrain
- Terrain attitude
- Laser rangefinding
- Line-of-sight threat occulting
- Texture coordinate feedback
- Annotation-data feedback (mud, snow, ice, etc.)

The Graphicon 2000 is capable of providing this feedback data in real time. The simulation host computer can then use the information for vehicle dynamics, scoring, explosion-effect processing, or intelligent-threat processing.

Repeating Animation Sequences

RTP provides the mechanism whereby the modeler can construct pre-programmed animation sequences in the database. These repeating sequences provide the mechanism for having significant on-screen activity with no host-computer overhead. With this construct, the modeler can create an arbitrary number of "frames" (child structures) that will be cycled through, where the cycle rate is user defined. RTP also supports the ability to blend (in a manner similar to LOD blending) between frames, resulting in smooth transitions during the animation sequence.

For example, moving water could be represented by a repeating cycle of 20 frames spanning a 2 second interval. Each frame (child structure) would be identical except for the values in the *"translate texture"* element. The visual result is that the water texture pattern would scroll across the water polygon every one-tenth of a second.

Non-Repeating Animation Sequences

The *non-repeating* animation sequences are similar to the repeating sequences except that:

- the host computer must explicitly enable the sequence (with a single command); and
- the sequence is not restarted once it completes.

For example, a tank explosion sequence might consist of five separate versions of the tank. The modeler would create the versions as five separate PHIGS structures. When the tank is hit by a missile the host enables the animation sequence. Based upon the time relative to the initiation of the sequence, the Graphicon would determine which of the five versions to draw. RTP also supports the ability to blend between versions, resulting in smooth transitions during the animation sequence.

Overload Management

An RTP workstation (channel) can be instructed to run in real-time at a desired update rate. This will cause the G2000 to perform overload management when the actual display time approaches the desired frame time. The G2000 overload manager smoothly reduces scene complexity without causing abrupt changes to the visual scene. Specifically, the techniques employed during overload are:

- reduce object level of detail
- increase haze, reducing the visibility range
- eliminate distant targets
- reduce special processing, such landing-light tessellation

Objects that are near light sources (in night scenes) and objects that have been tagged by the database modeler as *areas of interest* will not have their LOD reduced by the overload manager.

CONCLUSION

Reducing the database and software development associated with visual simulators is an important objective of both the government and the simulator industry. A primary mechanism for achieving this objective would be the use of a standardized graphics system for building and manipulating the visual database. The PHIGS standard (Programmer's Hierarchical Interactive Graphics System) is the direction being pursued by the general-purpose graphics community. As presented in this paper, PHIGS has now been extended into the real-time simulator domain with the Real-Time PHIGS (RTP) implementation. RTP has incorporated the significant simulator features required for real-time image generation into the standardized PHIGS framework.

References

[1] American National Standards for Information Systems. *Programmer's Hierarchical Interactive Graphics System Plus Lumiere und Surfaces (PHIGS PLUS).* April, 1989.

[2] American National Standards for Information Systems. *Programmer's Hierarchical Interactive Graphics System (PHIGS) Functional Description.* September, 1988.

[3] Rich, Henry H. "Tradeoffs in Creating a Low-Cost Visual Simulator." *Proceedings of the Eleventh Interservice/Industry Training Systems Conference,* (November, 1989).

[4] Wyckoff, Brad. "Managing Cost/Performance Tradeoffs for Successful Visual Training." *Proceedings of the Eleventh Interservice/Industry Training Systems Conference,* (November, 1989).

ABOUT THE AUTHOR

MR. BRIAN HEANEY is the Graphicon 2000 Project Engineer with responsibilities for system, algorithm and software development. He joined GE's Simulation and Control System Department in 1983 as a software developer on the US Army Conduct-of-Fire Trainer (COFT) tank simulator program. Brian came to Graphicon in January 1985 as a software designer on the Graphicon 700 and Graphicon 1700 graphics accelerators. He was a key contributor to the system and algorithm development for the Graphicon 1700S image generator. Brian graduated Tau Beta Pi and Phi Beta Kappa from Swarthmore College in 1983 with a BS in Engineering and a BA in Economics. He received his MS in Electrical Engineering from Duke University in 1986 through the General Electric Advanced Course in Engineering Program.