1-1-1997

A Characterization Of Low Cost Simulator Image Generation Systems

Curtis Lisle

Michelle Sartor

INSTITUTE FOR SIMULATION & TRAINING

A Characterization of Low Cost Simulator Image Generation Systems

IST-CR-98-02

Prepared by Curt Lisle and Michelle Sartor

University of Central Florida

P14

INSTITUTE FOR SIMULATION AND TRAINING
UNIVERSITY OF CENTRAL FLORIDA

A CHARACTERIZATION OF LOW COST SIMULATOR IMAGE GENERATION SYSTEMS

IST-CR-98-02

Prepared by Curt Lisle and Michelle Sartor

September 1997

Table of Contents

1. INTRODUCTION

1.1 IDENTIFICATION

This report is submitted as a deliverable report under the DoD/STRICOM Contract No. N614339-97-K-0010.

1.2 SCOPE

This report identifies and briefly discusses the characteristics that should be considered in the evaluation, comparison, and selection of low cost computer image generation systems to be used for simulator applications. These characteristics are derived from features, capabilities, and performance typically required in military and commercial simulator applications.

As the commercial off-the-shelf (COTS) graphics cards for personal computers become increasingly powerful, they also become capable of adequately supporting an increased number of military and civilian simulation applications. Therefore, it is important to have a comparison and classification system for low cost graphics systems to determine the simulation applications for which they are suitable.

It is not the intent of this report to completely cover the operation of an image generation system but to identify characteristics of a generic image generator (IG) so that the reader understands the reasoning behind our selection of important characteristics. (For further details on many of the characteristics discussed in this report, consult [Foley92].)

Many of the characteristics listed in this report are capabilities (shading, depth buffering, sub-pixel resolution) of an image generation system and not performance metrics (update rate, pixel fill rate, polygon throughput). These capabilities will be enabled and disabled during benchmarking in order to test their impact on the image generation system performance. In some cases, a capability (anti-aliasing) may also be tested for it's correctness.

This report is a first attempt to characterize an image generation system with respect to simulation. The capabilities and performance parameters listed in this report are referred to as characteristics. Subsequent project work will focus on devising methods to make use of these characterisitcs in a benchmarking operation. Upon completion of the first iteration of testing, it is likely that some of the characteristics will be deleted from our list. The deletion of a characteristic may be due to lack of significant impact on system performance, lack of availability on low cost image generation systems, or lack of an efficient test procedure.

2. FACTORS TO BE CONSIDERED

There are four categories of factors that should be considered when evaluating the performance of a computer image generation system for its suitability to a particular application or in comparing it with other products.

2.1 SYSTEM PERFORMANCE

Performance metrics help to answer the question "How fast can it do it?" for the device under consideration. The performance of an image generation system is the set of measurements that quantitatively characterizes how fast the IG can carry out its various functions.

Image generation systems are frequently measured and benchmarked through measures of their graphics performance, often in terms of polygon throughput. While these graphical benchmarks are useful (if measured in comparable terms), they are not the only performance metrics that must be considered in evaluating image generation systems.

2.2 SYSTEM CAPABILITIES AND FEATURES

The identification of a device's capabilities and features helps answer the question "What can it do?" The primary purpose of image generation systems is to produce synthetic imagery based upon a digital representation of the environment. However, there are many functions that are used to carry out this task to varying degrees of fidelity such as priority resolution and texture processing as well as other functions such as special effects that are related to, but not an integral part of,its main purpose.

2.3 LIFE CYCLE AVAILABILITY

Issues related to availability of a device or its components (for repair and spares purposes) help to answer the question "For how long can it be purchased?" Particularly for defense applications, these issues can be as critical as performance or features, since systems that are soon obsolete may be difficult to deploy effectively.

2.4 LIFE CYCLE COST

Finally, the identification of life cycle costs of a device helps answer the question "How much will it cost?" This goes far beyond the acquisition cost of the device itself, but must also consider costs for integration, training, repair, spares, and maintenance.

3. DETERMINATION OF IMPORTANT IMAGE GENERATION CHARACTERISTICS

This report presents a list of image generation characteristics and provides insight on their relevance to simulation. The list was obtained from an analysis of several sources including the IMAGE Society's Annual IG Survey, the Close Combat Tactical Trainer (CCTT) image generation specification, and COTS vendor websites. (Refer to the Methodology section of [IST97] for the rationale behind selection of these sources.)

In order to develop benchmarks for comparing the performance of low cost image generation systems, it is necessary to create a list of image generation characteristics. Once a list of characteristics is established, tests for each characteristic can be developed. Initial testing of each characteristic will help determine the importance of a particular characteristic in distinguishing the image generation system's performance over another system. These initial test results along with an analysis of the impact of each characteristic on real-time simulation, will allow for the establishment of a list of minimum characteristics for use in benchmarking.

In the conclusion section of this document, the relevance of each IG characteristic to various aspects of the synthetic environment is summarized. Results obtained from low cost IG testing will be used to refine this initial list to a set of minimal characteristics that adequately quantifies the performance of low cost IGs in a real-time simulation application.

An IG system can be thought of as consisting of system level characteristics, geometry processing characteristics, texture processing characteristics, pixel processing characteristics, video characteristics, special features, and operating environment. The image generation characteristics are presented in this following sections with the categorization scheme laid out above.

4. SYSTEM CHARACTERISTICS

The characteristics identified in this section are those which cannot be easily allocated to a single subsystem in an image generation (IG) system. Instead, they tend to place the IG into a broad category of capability that is further refined by examining characteristics of individual subsystems (which are presented later in this report).

4.1 UPDATE RATE (UPDATE TIME)

The update rate (also sometimes known as frame rate) is the rate at which a new, complete image is drawn into the frame buffer to be refreshed onto the display device. The specification of an update rate assumes that the update time is relatively constant and does not typically vary from frame to frame.

The update time for IGs used in simulators is most often an integral multiple of the video refresh time, which minimizes the transport delay (discussed below). Thus, for a refresh rate of 60 Hz (16.67 ms), expected update rates would normally be 60 Hz, 30 Hz (33.33 ms), and 15 Hz (66.67 ms). The update rate for graphics systems is often expressed in nominal terms as these types of systems tend to operate open loop at the fastest rate possible.

The maximum update rate is dependent upon the image generation system itself, while the actual update rate depends upon the application, the visual database, and other performance requirements. A rule of thumb is that IG throughput (e.g., polygons and pixels) is doubled when the update rate is halved.

4.2 TRANSPORT DELAY

The transport delay of an image generation system is defined as the latency between a positional update command being sent to the IG and a new image rendered on the video output. Therefore, this latency is the sum of the time between when the command is sent and the time that it is parsed, the time required to render the new image (as modified by the host command) into the frame buffer, and the time required to refresh the video signal from the frame buffer.

The first of these times will depend upon whether the IG is synchronized with the host computer. If the IG is synchronized with the host computer, this latency will be a small value (approaching zero) since the host will send commands just in time for the IG to begin parsing them. If the two devices are running asynchronously, the latency will vary from frame to frame, the maximum time being one cycle of the command parser (which is often the same as the update time). It is this value that isused as a worst-case latency.

The time required to render the image into the frame buffer isat least the update time, since this is the time between frame buffer updates. Pipelined architectures have rendering times that are greater than this (and frequently integer multiples) since the rendering process is broken into tasks that execute simultaneously. This technique is often used to improve the update rate while sacrificing transport delay.

The video refresh time is most often simply the inverse of the video refresh rate. This accounts for the time required to get the information from the frame buffer to the display device where the observer can view the scene as modified by the host command. Though the change may conceivably only affect the last pixel of the image, the worst case of the entire refresh time must be used for this latency.

4.3 DYNAMIC COORDINATE SYSTEMS

Dynamic coordinate systems (DCSs) allow objects to move relative to the environment and to one another. Motion may be either independent of or articulated to other DCSs. These coordinate systems typically have six degrees of freedom (x, y, z, heading, pitch, and roll) but fewer may be used to improve performance.

Performance criteria for an image generation system's DCSs are the maximum number (both DCSs and total degrees of freedom) that can be managed simultaneously, the maximum number that can be simultaneously in the scene, and the maximum number of levels of articulation. Features that should be considered are whether articulation is performed by the image generation system and the type of extrapolation is performed on the position and/or orientation of DCSs.

4.4 EYEPOINTS

An eyepoint is a location from the perspective of which the IG renders an image. An eyepoint is not only used in the geometric transformation calculations but is also used for the purposes of paging portions of a large visual database into and out of memory.

An eyepoint is typically bound to a DCS, although the two may be offset by some distance. Furthermore, multiple eyepoints may be bound to the same DCS (usually called dependent eyepoints) or different ones (called independent eyepoints). An example of this is a model of a tank that is being simulated (also called the ownvehicle). Separate eyepoints for each of the visual assets of the tank (such as one for the gunner's sight and one for the commander's cupola periscopes) can be bound to the DCS responsible for moving the tank. The origin of the DCS might be the center of gravity of the tank, and each eyepoint would then have a specific offset from the DCS origin to the location of the eyepoint with respect to that origin.

The number of eyepoints can be used as a performance metric for IGs. However, widely spaced, independent eyepoints can often demand dedicated IGs because they can require different areas of a visual database to be paged into environment memory (see 0).

4.5 VIEW VOLUMES / FIELD OF VIEW

Each independent or dependent eyepoint has an associated *view volume* that specifies the mapping applied to transform vertex coordinates from the synthetic environment into display coordinates. The view volume and field-of-view of the eyepoint also define the associated perspective projections used to give depth cues to the rendered scene. Once a view volume is determined, the volume can be used to *cull* the database during a traversal – eliminating all areas of the database that are not potentially visible from the eyepoint. The resulting set of potentially visible polygons must then be *clipped* to conform to the exact dimensions of the view volume. The clipping process truncates polygon edges so that all vertex coordinates are contained within the view volume.

Database culling and clipping are generally performed during the database traversal and geometry processing operations in the IG. In low cost systems, this portion of the rendering process is often accomplished by the host computer due to the expense of dedicated geometry processing hardware.

A further performance consideration associated with system field-of-view is that wide field-of-view displays function like wide-angle lens on cameras can result in a large amount of the synthetic environment being potentially viewable. This means that database culling and clipping will be less successful in reducing the polygon count of the scene that must be rendered by the IG. Large field-of-views and/or far distance clipping planes in the view can result in substantially larger polygon processing requirements for the IG.

For systems where image generation is performed by a COTS workstation with graphics accelerator cards, performance measurements should focus on the host CPU speed and the software API architecture since geometry processing is likely to be the system bottleneck. Such systems will generally be restricted to smaller field-of-view displays or closer clipping planes to reduce the polygon count of the scene during geometry processing

4.6 DISTORTION CORRECTION

Some display devices require a non-linear mapping of the synthetic imagery to the display surface. This is the case for dome displays and for helmet-mounted displays with pixels concentrated in the center (or any area) of the field of view. Image generation systems accommodate this request by implementing *distortion correction* that performs a linear or non-linear map of the originally rectangular raster image into a shape that fits the display system requirements.

This feature is hardware intensive and generally not supported in low cost visual systems. Simple distortions can be implemented through an address lookup table to the video memory (so pixel locations are remapped). However, complex distortion requirements impact the entire image generation pipeline.

The measurements in this category could compare visual systems with regard to whether they support no distortion, linear distortion, or non-linear distortion maps.

4.7 AREA PROCESSING

Originally developed for high-performance IGs in the early 1980's, *area processing* refers to a particular pipelining technique used during pixel processing. With area processing, all priority, coloring, and shading calculations for each rectangular region of the screen (often called a *span*) are performed contiguously before the IG moves on to a different region. The size of a span is architecture dependent, but it is generally a square region encompassing a small number of pixels (e.g., 4x4 pixels, 8x8 pixels).

The area processing technique has several advantages over earlier rendering approaches. Area processing can be implemented in dedicated hardware that can be re-used for each region during a rendering cycle. Area processing completes regions of the screen earlier than the accumulating approach, where the entire scene is written into a frame buffer, thus making the output of the IG more suitable for a pipelined post-processing system to provide sensor effects or other types of image filtering. This early completion of screen areas allows for a shorter transport delay in pipelined post-processing.

Area processing is becoming available as COTS technology for low cost graphics and image processing systems. For example Talisman, currently under development by Microsoft, employs area processing -- referring to their display spans as *chunks* [Torborg96]. Oak Technology's new Warp5 (Windows Accelerator and Rendering Processor) chip refers to this technique as region-based processing. As this technique becomes more widely available for low cost systems, it should bring with it a noticeable improvement in rendering quality and performance.

4.8 SYSTEM LEVEL LOAD MANAGEMENT

Load management refers to the technique used by an IG to "gracefully degrade" its video quality if the scene complexity becomes too great to successfully render at the desired update rate. In high-end systems, load management is generally triggered by a signal from overloaded rendering hardware. The host computer system must react and lighten the load to avoid obvious rendering anomalies such as missing scene objects due to polygon overloading.

Load management problems generally occur for geometric or pixel processing overloads, and therefore are restricted to those subsystems. However, it is easy to show that a system-wide solution must be devised to guarantee graceful degradation while balancing the load in different processing areas of the IG.

The existence and completeness of load management schemes are an important issue in the comparison of image generation systems. Load management issues in the geometric and pixel processing subsystems are further discussed in section 0 and 0, respectively.

5. GEOMETRIC PROCESSING CHARACTERISTICS

5.1 POLYGON THROUGHPUT

The number of polygons that can be processed by an IG is a common means of comparing such systems. In this case, polygon processing includes the traversal of the online portions of the visual database or display list, culling of objects based upon distance or inclusion in the current field(s) of view, level of detail calculations, geometric transformations from the modeled coordinate space into that of the eyepoint being processed, and generation of shading and texture parameters used during pixel processing.

Unfortunately, the way in which this important performance metric is measured varies radically from vendor to vendor, from product to product, and from benchmark to benchmark. Common inconsistencies are the number of vertices (although triangles are the most widely accepted), the degree to which they are meshed (i.e., the number of vertices shared by a group of polygons), the number of polygons actually visible versus the number removed by various tests, the type of shading calculation performed (e.g., flat versus Gouraud), the amount of non-polygonal structure being processed (such as groups, objects, and culling nodes), and even whether the polygons are two-dimensional versus three-dimensional. Also important is whether the number specified is a theoretical maximum, an optimized test case, or a real-world value based upon a specific application. Many of these features and parameters (e.g., shading) are also of interest to measure, individually, in simulation applications of graphics systems.

The latter case mentioned above regarding whether the polygon throughput is based on a theoretical maximum, an optimization, or an application are the most difficult to measure objectively, but it is also the most useful for comparison. It is more useful if the application under test is similar to the one for which the device is being considered. For these reasons, an application-specific benchmark offers the best means for evaluating and comparing the polygon throughput of IGs.

5.2 STAMPS

A feature of some IGs is the ability to allow certain polygons to rotate about one or more axes such that the polygon faces the eyepoint being processed. These types of polygons are typically called stamps or billboards.

Stamps are frequently used to model trees, bushes, and other objects of a roughly symmetric shape in order to provide a computationally inexpensive way of depicting a large number of features. Image generation systems that support stamps (particularly as a low level of detail for an object) can often render images that appear more dense and rich than those IGs that use other techniques. While stamps are not appropriate in all cases (they can even cause problems in some applications), this capability can be an important feature for an IG. This capability is not generally found in low cost image generation systems but can be indirectly implemented through a higher layer software API that steers the stamp polygon as the eyepoint moves.

5.3 LIGHTS

Many simulated environments for training contain aircraft landing lights, navigation beacons, or other single-point light sources which must be appropriately rendered by an image generation system. Many systems, particularly the high performance ones, have separate rendering pipelines dedicated to high-intensity point lights.

Lights generally have a three-dimensional position in the database and an associated color. Depending on the system, light intensity may be subject to distance from the eyepoint and environmental conditions (fog, haze, and clouds). To faithfully represent lights in the scene, database occulting and fading issues should be addressed in the image generation system.

Lights are difficult to represent faithfully on a raster-based display because the small area of the light is often aliased by the limited display resolution. Lights represented as bright pixels on the screen will flash or scintillate during dynamic sequences. As a cure for this problem, some image generation systems support *calligraphic lights*. With calligraphic lights, a separate electron source similar to the trace on an oscilloscope is used to illuminate the screen directly at the light positions. Calligraphic lights are much crisper than raster lights, but require special video displays and image generation hardware not generally available in low cost image generation systems.

In addition to database support for lights, an image generation system provides a certain amount of controlled behavior for lights and light strings. All lights can be disabled during daytime simulation or addressing and control could be available for individual lights to enable strobes and flashing under host computer control. When comparing IGs, it is important to determine the amount of control provided for light effects.

5.4 GEOMETRIC LEVEL OF DETAIL

Because of their limited polygon throughput, image generation systems must achieve the highest realism possible on a fixed polygon budget. Level of detail (LOD) transitions are a common technique thatallows moving models and terrain skin to be represented with fine resolution (referred to as high LOD) when close to the eyepoint and coarse resolution (referred to as low LOD) far from the eyepoint. Detail transitions are appropriate because distant objects generally occupy a small display area and fine details are not visible because of limited display resolution. Several types of LOD switching are described below.

5.4.1 Model Level of Detail

Three-dimensional cultural features (e.g., water towers, power poles, or buildings) and moving models (e.g., armored vehicles, aircraft, or munitions) are modeled using polygonal representations as part of the off-line database development process. Generally three to five distinct levels of detail (LODs) are constructed for each object. All LODs are compiled into the run-time database so that the IG can choose the appropriate LOD for optimal use of the fixed polygon budget. The choice is based on the *slant range* calculated between the eyepoint and the object. Depending on the system, individual models may include separate transition distances programmable during database development. Model LOD transitions are generally instantaneous because the high LOD is maintained in the view until it is too small for the viewer to notice a difference between LODs.

5.4.2 Terrain Level of Detail

The synthetic terrain skin used in training applications will often be seen from both close and far ranges, such as when a ground-based vehicle is scanning from immediately in front of itself to the horizon or when an aircraft descends from cruising altitude to a landing. In such cases, the amount of synthetic ground area covered by a fixed display resolution varies widely. To address this requirement, terrain databases are generally arranged into a mosaic of fixed-size *load modules* or *area blocks*. During a simulation, the IG can dynamically choose between a set of LODs for each area block currently in its viewing area. The edges of the different LODs for a single area block must be aligned to avoid visual anomalies such as cracks appearing in the terrain skin.

Since terrain features cover large areas of the display even in the distance, instantaneous transitions are usually visible and distracting unless *transparency blending* is employed. During a blending operation, the current terrain LOD for an area block is gradually modified from opaque to transparent while the opposite operation is done for the new LOD. A smoother transition is accomplished with this technique, but the polygon counts must be adjusted to allow for two LODs of area blocks at the transition distances to be simultaneously displayed. Transparency blending requires the rendering pipeline to support polygon transparency, a feature that is not available on some low cost systems.

5.4.3 Continuous Level of Detail

To further reduce the visible LOD transitions in a rendered scene, continuous level of detail can be employed for terrain and/or models. With continuous LOD, the polygon count is gradually reduced or increased as individual polygons merge or split, respectively, under algorithmic control. Various algorithms exist for decimating triangular meshes, but most are non-real-time, such as [Turk92], [Schroeder94], and [Renze95]. Most techniques follow the process of re-meshing an arbitrary polygonal representation into a form with substantial geometric regularity and then exploiting the regularity to re-polygonize after the removal of some vertices to reduce polygon count. However, the recent uses of the wavelet transform has led to new wavelet-based alternative techniques [Eck95]. In the simulation community, the multi-resolution irregular triangulated network approaches use preprocessing to construct run-time structures that can provide a seamless, continuous LOD of terrain skin. This effect, while present in several high-end IGs, is generally too expensive for low cost systems.

5.5 COLOR / TRANSPARENCY

IGs can be viewed as systems that transform input geometry into rendered images by applying functions on the input data attributes according to high-performance hardware or low-level software algorithms. Colors attached to the model and terrain polygons of a synthetic environment database serve as a good example. The final hues in the rendered scene are a function of the color depth of the database pixels and the accuracy and precision of the transformations used during the rendering process.

Color values are generally represented in either RGB (red, green, blue) or HSV (hue, saturation, and value). For further reading on color representation and accuracy, please refer to [Lindbloom89]. Using either color system, a fixed resolution (specified by the number of bits allocated) is employed and preserved as much as possible during the rendering operations. Twenty-four bit (also referred to as true) color yields 2^{24} or 16.7 million colors. However, color resolution is an area often compromised to reduce hardware costs for IG systems. It is expensive to maintain twenty-four bit wide color multiplier paths to preserve the database colors. Furthermore, if polygon transparency (called alpha in the OpenGL literature) is supported, then the number of bits required for a polygon's base color is usually extended to include the transparency percentage. For example, thirty-two bits are required for twenty-four bit RGB and eight-bit (256 levels) transparency.

Low-cost systems generally compromise with sixteen, twelve, or even eight bit color representations. Limited transparency levels require sparse use of transparent features in the database modeling process since opacity occurs quickly with low-resolution alpha computations. For better resolution finer granularity (i.e., more bits per pixel) of alpha is required. For example, looking through two 25% opaque cockpit windows results in 50% opacity when viewed from outside the aircraft

5.6 SHADING

Shading is an algorithm used to modify the native colors on the polygons of the synthetic environment database. It is well known from physics and computer graphics theory that the angles of light incidence and reflectance off an object are the same. Therefore, the angle between the light source (the sun for outdoor, daylight scenes) and a polygon in the database is enough information to calculate an intensity modifier for shading the polygon according to light incident from the source. The common algorithms for shading are *flat* shading, *Gouraud* shading, and *Phong* shading. Flat shading maintains a single multiplier intensity across the entire polygon while the other two algorithms, variations of *smooth* shading, perform interpolations across the polygonal surface.

Issues for comparison of IG systems in the area of shading include the various shading algorithmssupported the resolution of the angle computations and interpolations if smooth shading is used.

5.7 ILLUMINATION

Illumination effects are useful for simulation of night scenes and the effects of ownship lights, moving model lights, and flares. For truly accurate modeling of light effects, incident light reflections must be calculated via ray-tracing techniques, but this cannot be accomplished in real-time and is considered unnecessary for most applications.

More affordable and still remarkably realistic effects can be achieved through intensity modification algorithms based on the pixel location on the display. For example, in a driving vehicle, the headlights generally illuminate ground area in two elliptical or conical paths with their origins at the bottom of the field of view (where the headlights are positioned). It is quite effective to generate the scene without regard to the headlights during rendering except that a two-dimensional intensity matrix (with elements corresponding to screen locations) is used as an "illumination multiplier map" for every pixel of the image. This yields the lobes for landing lights on the runway or the headlight views we expect to see out of our windows. This technique is limited to illumination areas that are fixed on the screen unless the map can be dynamically updated based on the movement of light sources in the scene.

A second type of illumination occurs for the special case of windows or doorways in night scenes. These objects are generally given their "glow" by special attributes on window and door polygons that make them immune to color shading and color fading calculations. When the majority of the scene is dark because of the lack of sun illumination, these un-darkened polygons will remain bright, giving the appearance of light sources inside the buildings.

To compare low cost systems, study the resolution (in bits) of the illumination maps, whether they can be dynamically updated, and whether remote lights (i.e., flares) are supported. Remote lights are more difficult because knowledge of database features (and correct occulting) is usually required for realistic images. The IG should be able to mark individual polygons as being shadable or unshadable to allow for windows and doors in buildings at night.

5.8 ENVIRONMENT MEMORY

For quick retrieval and real-time rendering requirements, image generation systems generally have dedicated environment memory that contains the polygonal database representation in a hardware-specific format conducive to efficient traversal by the hardware or low-level software.

In such a system, the environment memory contains the run-time IG format information for all terrain, features, and models that are currently or potentially visible from the eyepoint. The environment memory is generally directly scanned by dedicated database traversal hardware that serves as the beginning of the geometry processing pipeline. If the eyepoint moves substantially or a different simulated environment is chosen, the memory must be updated to contain the newly requested database. This update process is discussed further in section 0.

Systems that do not include local environment memory, such as common PC-based graphics cards, are subject to competing for access to main system memory if they directly address it. Otherwise, the graphics card handles only the final rendering tasks after the main processor performs the bulk of the geometry processing work in software. This category includes many of the COTS graphics cards at the time of this writing. Since these systems do not have local memory, an image generation system using them must emulate the function of the environment memory or use a design that does not require the environment memory.

5.9 PAGING RATE

Database paging is a necessary feature for IGs used in vehicle training systems that must function during simulations where the eyepoint moves a considerable distance or the vehicle interacts with remote entities in the synthetic environment. Current training databases can easily reach or exceed 100 square kilometers. It is therefore necessary for the IG to support run-time paging of portions of the database depending on the location of the eyepoint. For example, the SIMNET IG systems keep a 3 x 3 mosaic of load modules in the IG at all times. The center module contains the ownship vehicle. This results in a 1500 meter view in any direction without paging additional database content.

Important factors for simulation performance include the granularity (size) of a load module, the rate at which new load modules can be loaded, and the total on-line capacity of the IG (in load modules). High performance image generation systems include an established method for storage, organization of the database and on-line retrieval of load modules on demand (such as a dedicated database disk subsystem). However, low cost graphics display hardware generally does not address this need. It is important to assure that graphics hardware is compatible with a dynamic update strategy and that a strategy is actually included in the simulation software (via hardware API, software API, or application program).

5.10 GEOMETRY PROCESSING LOAD MANAGEMENT

Geometric processing overload conditions usually occur when there are too many potentially-viewable polygons in a scene and the geometry transformation hardware is overloaded.

Load management solutions for geometry processing typically involve a combination of several techniques:

- The most obvious overload technique is to slow the update rate of the rendering. This allows more time for both geometry and pixel processing. Often, the fixed update time is doubled (i.e., a 60Hz system reverts to a 30Hz system). The same rendering quality is preserved, but high speed eyepoint or moving model motion will not be rendered optimally.
- The level of detail switching distances for models and/or terrain can be reduced so that the polygon count for distant objects is reduced. This technique is of limited value since it is immune to the problem of overloading close to the eyepoint (too many moving models in the new view). Switching distances can be adjusted easily if the IG reads database switching distances for all objects and multiplies the values by an "inverse load factor" to generate the switching distance used during the current load conditions.

6. TEXTURE CHARACTERISTICS

In the following sections, the attributes common to texture features in low cost image generation system are presented

In early image generation systems, texture patterns were randomly generated noise patterns used to artificially modulate the faces of selected polygons in a rendered scene. While this technique is still feasible, it has been almost completely replaced by photo-based texture mapping where two-dimensional images are mapped across the surfaces of textured polygons. For the remainder of this section, we will assume a photographic texture approach.

6.1 FORMAT

Texture patterns are stored in the image generation system in two-dimensional arrays called *texture maps*. Offsets into the texture map are addressed by two Cartesian coordinates (called *texel coordinates*). The depth of a texture map refers to the number of bits of resolution provided for each texel. The texel width can be used as a black and white intensity modifier for the base polygon color or it can be treated as a color value that supersedes any polygon color values.

If the texel contains a color value, it can be encoded RGB or HSV depending on the hardware implementation. The texel width determines the number of discrete hues or intensity values available to the IG. For example, good results on previous systems have been achieved using 8-bit RGB maps with three, three, and two bits for red, green, and blue, respectively.

Texture maps are generally square with linear dimensions in a power of two to facilitate address manipulation into texture memory. Common sizes range from 128x128 to 512x512.

Characterization of the texture format of low cost systems should focus on the depth of the texels and the various formats the texel can support. For example, can a system support both intensity and color texture maps?

6.2 TEXTURE LEVEL OF DETAIL

Textured polygons in a synthetic environment can occupy a small display space (1 pixel or several pixels square) or cover the entire display depending on eyepoint motion. Throughout this range of magnification, the texture should gradually display more information without such anomalies as crawling, sparkling, or blurring. A solution to this requirement, called *MIP mapping*, was developed initially for high-end simulation systems and is now implemented in low cost graphics hardware as well.

The acronym MIP comes from the Latin phrase "multum in parvo," or "many things in a small place." [Cosman94] refers to the technique of having a hierarchy of texture maps that are successive 2x2 down-sampled copies of the same image (i.e., 128x128, 64x64, 32x32, 16x16, etc.). Each down-sampled map has been spatially filtered to remove high-frequency texel information that cannot be represented in the smaller map size. The amount of filtering performed on each down-sampled map done by the database modeler during off-line database development time. During a real-time simulation, the IG selects between different resolution maps according to the distance between the eyepoint and the textured polygon.

6.3 TEXTURE MEMORY

Local memory for texture patterns is generally provided in the IG and preloaded before commencement of an interactive simulation. Therefore, the IG must have adequate *texture memory* to contain all MIP-format texture maps that will be used during a simulation unless the texture memory can be dynamically updated during a simulation while being simultaneously accessed for rendering. When calculating texture map space, a MIP-format map requires 1.33 x the space of its highest LOD map to hold all the down-sampled copies [Cosman94].

A finite texture memory is not a difficult limitation if *generic texture* is used, but it is a strict limitation when *geospecific texture* is required. In generic texture, patterns of general-purpose trees, buildings, and ground patterns are stored in texture maps and used repeatedly across the database. The generic map technique can yield high quality results, but lacks the ability to exactly mimic a particular geographic location (such as particular military compound or a specific location in the world). Geospecific databases are more demanding on texture memory size because texture maps of specific objects (e.g., the Empire State Building or the terrain photograph from a satellite) can be used only once in a database. Because of the enormous texture memory requirements and the corresponding database development costs, geospecific scene simulation is currently not practical for low cost image generation systems.

6.4 MICROTEXTURE

Microtexture or *detail texture* is a feature in an image generation system that gives textured polygons an additional high-detail pattern when they are close to the eyepoint for improved depth cueing. Microtexture was developed as a result of limitations encountered when IGs were used for rotary wing vehicle landing trainers. With traditional, MIP mapping and tri-linear interpolation techniques (see 0), textured ground polygons were rendered too smoothly when the vehicle eyepoint came close to the ground – depriving the pilot of height-above-terrain cues available during an actual landing exercise.

The microtexture solution to this problem provides the IG with a set of small texture maps (generally loaded with randomized or noisy patterns). The microtexture patterns are black & white modulation maps that add detail to the polygon surface for use in close range viewing. Individual polygons can specify a microtexture index to select the best map and the IG is programmed with a fading curve that blends the microtexture pattern out exponentially so its contribution quickly fades with distance.

Microtexture is most commonly used to provide grass texture for low-level landing operations but it can also be used whenever it is appropriate in the database. For example, high-detail patterns on the walls of buildings can be used for individual combatant visuals in urban scenarios.

In general, dedicated microtexture capability is not available on low cost image generation systems today. However, a workaround can be implemented if a system supports multiple texture maps per polygon and allows different fading ranges for each texture map. In this case, a second ordinary texture map can be used for microtexture.

6.5 PERSPECTIVE CORRECT TEXTURE MAPPING

To provide convincing visual cues for scene depth and relative shapes for objects, it is necessary for an IG to warp and sample texture maps to generate perspective correct, textured scenes. For example, the texture map of a brick pattern is perspective warped until the brick lines disappear into the vanishing point when the texture is applied to a wall that fades into the distance.

Because a texture pattern can be used for different polygons at different scales, it is important for the IG hardware to offer texture map *wrapping* (where the pattern repeats over and over across a polygon). Image generation requirement specifications can sometimes be written to further specify that single texture patterns must be able to span multiple polygons (like a satellite photo on an elevation mesh). However, this is solely a database modeling issue. If the IG has the ability to tie a texel location in a texture map to a polygon vertex, then it is up to the modeling system to project the texture onto the polygonal mesh and determine the appropriate texture coordinates to attach to the vertices for later use during a real-time simulation. Texture smoothing and interpolation functions and their interaction with perspective projections are discussed further in section 0.

When comparing low cost image generation systems, study the ability to load texture maps with orthogonal (unwarped) images that are warped according to the underlying polygon's position in real- time. Texture wrapping ability simplifies database development and the use of the texture stored in the texture memory.

7. PIXEL PROCESSING CHARACTERISTICS

7.1 PIXEL FILL RATE / DEPTH COMPLEXITY

In order to correctly render a synthetic environment, an IG must process the contribution each polygon in the scene makes to the final color of some or all of the display pixels. (The display pixels affected depend on the object size, the position of the eyepoint, and its field-of-view.)

Processing a polygon contribution can take either a fixed or variable amount of time depending on the IG design. *Depth complexity* is a performance metric related to pixel processing that quantifies the average number of times an IG can afford to *visit a pixel* during the rendering process. In this context, visiting a pixel means determining the contribution of a single polygon to the color of this pixel. The visits per pixel value quantifies the scene complexity an IG can handle without resulting in an overload situation.

The depth complexity metric is an important value for judging the scene density that an image generation system will be able to deliver. Additionally, for some rendering approaches, texture is much more expensive than ordinary shaded polygons. Therefore, it is recommended that depth complexity performance be measured for both textured and non-textured scenes to uncover the relative costs of each type of polygon rendering. It is likely that textured polygons will be much more expensive than untextured polygons on low cost IGs. In contrast, high-performance IGs have been designed with the expectation that a reasonably high percentage of the polygons in a scene will be textured.

The depth complexity can be used to determine the content of scene that can be supported without an IG overload. Depth complexity is calculated using the following formula:

$$DepthComplexity = \frac{PixelFillRate}{NumberOfDisplayPixels * UpdateRate}$$

As an example, the basic Real 3D PRO-1000 has a pixel fill rate of 50 million pixels / second. If the display is 1024x1024 at 30 Hz, the average depth complexity is 50M/((1024x1024)*30) = 1.6. This means the entire screen can be painted once (by sky/ground background) and approximately 60% of the pixels can be painted by an additional polygon (from a cultural feature or model).

7.2 PRIORITY

A scene with even moderate complexity will almost always have overlapping objects that must be rendered correctly by the image generation system. In such systems, occulting issues are resolved according to a *priority scheme*. High priority objects should always occult lower priority objects in a scene. Object priority can be roughly determined by the *slant range* between the eyepoint and the object (closer objects generally occult far objects). In fact, early image generation systems used the slant range to the object centroid as the priority value for render ordering of objects. However, it can be easily shown that slant range alone cannot distinguish objects in all cases. For example, viewed from the ground, the centroid of a long runway could easily be closer than the centroid of a plane on top of the runway but the plane should have priority over the runway. Obviously, as the rendering eyepoint moves about the scene, the relative priorities of objects in the scene must change to allow correct rendering. The priority problem gets even more difficult if interpenetration is allowed between objects (e.g., tank guns jutting out from a tree that occults the remainder of the vehicle.)

Priority algorithms can be further classified as *fixed priority* or *dynamic priority.* A priority scheme is fixed when relative object priorities are decided by pre-processing the database. In contrast, dynamic priority algorithms use temporally varying database state to recalculate priorities during a simulation. In the following sections, several of the common priority algorithms will be discussed.

Evaluation of the priority scheme used in a low cost image generation system should focus on which scheme (or combination of schemes) is. Particular attention should be given to how the priority of moving models and special effects are determined.

7.2.1 Fixed List Priority

The simplest priority scheme is *fixed-list priority* where all of the polygons in a scene are arranged in a list according to the order they should be rendered for a correct result. For scenes without interpenetration, if polygons are rendered starting with the farthest polygon in the scene and finishing with the closest, the resulting render is correct. This approach is called the *painter's algorithm* since it models the layering done by artists during oil painting.

While its simplicity is attractive, fixed list priority is generally unacceptable for image generation because of several limitations including an eyepoint change necessitates a complete reordering of the polygon list, and no polygon interpenetration is allowed since the entire surface of a polygon is always rendered. This approach is appropriate only for cases where the eyepoint and objects are stationary or changes are so minor that they can be handled by list insertion during the simulation. Due to these limitations, more powerful priority schemes were developed.

7.2.2 Binary Separating Planes

The Binary Separating Plane (BSP) [Fuchs80] approach was the first priority scheme to be widely implemented in image generation systems. As a preprocessing step, the environment database is augmented by a subdivision process that places separating planes between objects. A set of planes is added such that the relative priority of any two objects can be decided by comparing their location and the location of the eyepoint with respect to the separating planes. This algorithm is traditionally classified as a *fixed priority* scheme because all relative object priority is determined off-line during the plane calculation process. However, prototype rendering systems with dynamic priority based on separating plane technology have been recently demonstrated [Naylor95].

Separating plane algorithms are efficient, requiring a relatively moderate amount of hardware in the image generation system; however they do not gracefully handle moving models and special effects. As models move around in the scene, their relative priority changes with respect to the environment. In a pure BSP approach, the address of each model relative to all separating planes must be updated after any moves. Even worse, if a model straddles a separating plane boundary, it must be "carved up" into separate models, each of which lie entirely within a separating plane region. These algorithms are prohibitively expensive for hardware implementation and in most image generation systems these dynamic priority cases have been resolved using a different priority approach such as depth buffering.

7.2.3 Depth Buffering

Fixed-list and BSP approaches are both *polygon-based priority* schemes were the granularity of priority calculations is restricted to the polygon level. No polygon interpenetration is allowed in polygon-based priority. However, as scene density and the need to support interpenetration increases, it is advantageous to shift to *area priority* schemes where polygon priority can be resolved differently for various areas of the screen. The previous example where a tank gun emerges from the leaves of a tree is a good illustration. The gun tube is composed of long polygons that have priority over the leaves in some areas while behind the leaves (with lower priority) in other areas.

The most common area priority approach is known as the *depth buffer* or the *Z-buffer* where the depth of the closest object encountered so far is kept for each area of the scene during rendering. In a traditional Z-buffer, the transformed database coordinates are used as input to the depth buffer. Depth is determined by finding the distance between the database vertex and the eyepoint and measuring the length of the projection of this vector along the viewing direction. This works well for objects in the center of the field-of-view but non-optimally for objects at the extremes of the view volume. The *R-buffer*, a modification of the Z-buffer directly compares the slant range from the eyepoint to the database vertex for each object instead of projecting the distance vectors perpendicular to the viewing plane. The R-buffer is more expensive to compute but generally provides superior results.

When comparing depth buffer implementations, consider the following factors:

- Is the depth value derived from the Cartesian coordinate of the polygon (a Z-buffer) or is it the per-area slant range (as in an R-buffer) calculated between the eyepoint and the polygon? The R-buffer generally gives superior results.
- Does the approach require a full-screen depth memory plane (expensive for large resolutions) or is it pipelined in a fashion that eliminates the need for a large depth buffer? Pipelined approaches can continue texture processing earlier because priority decisions for each area are completed before the pipeline progresses. Full-screen depth buffers do not have final priority decision until the entire scene has been processed (this leads to higher latency or greater hardware expense to avoid a delay.)

7.2.4 Polygon Layers / Decals

Special features like logos painted on vehicles or windows and doors on buildings must themselves be polygons rendered with correct priority relative to the rest of the scene. This is accomplished by a special priority algorithm called *decaling* or *layering*. Decal polygons carry special attributes that tell the rendering hardware to give the decal priority over the co-planar polygon it adorns. Image generation systems generally support a finite number (between one and sixteen) of relative priority layers for use. In some implementation techniques, these polygons can be relatively expensive to render. However, since there are relatively few decal polygons in a scene at a time, this cost can usually be ignored.

Although it seems to be a minor feature, an IG without layering capability is at a distinct disadvantage in rendering large environments because it makes difficult demands on database development (avoiding co-planar faces) and it is prone to visible priority anomalies such as decal flickering.

Instead of augmenting the priority algorithms with decals, some low cost systems use a *stencil buffer* to provide layering ability. The term refers to the technique used to render the decals and underlying polygon using multiple writes to the accumulation buffer.

Decal support is an especially important feature for low cost IG systems since priority resolution and texturing (alternate way to implement decals) are generally not done with the same fidelity in low cost systems. Systems without decal support will be adequate for training applications that do not require decals on polygonal models.

7.3 ANTI-ALIASING (GEOMETRY, DEPTH, TEXTURE)

The image quality of temporal phenomena that arise during a simulation is largely determined by the anti-aliasing approaches taken by the image generation system. When comparing image generation systems, look for the existence and the implementation fidelity of these anti-aliasing algorithms.

7.3.1 Geometry Anti-Aliasing

Because of its raster-based nature, an IG induces quantization and sampling error on the scene it renders. Since the pixels on the display device are the smallest features in the final image, an algorithm is required to determine the contribution to pixel colors by each relevant polygon.

Due to its finite resolution, a display pixel represents the appearance of a non-trivial volume from the synthetic environment. The exact volume represented is a function of screen resolution, the eyepoint field-of-view, and the maximum far rendering distance. Geometric aliasing occurs in the *edge-mask generation* used during pixel coloring. A polygon's edge mask represents the percentage of a pixel covered by the polygon.

The anti-aliasing approach that yields the best results is to directly minimize quantization error by rendering the entire scene at a resolution higher than the final display resolution. This technique is called rendering at *sub-pixel resolution*. Rendering resolution is chosen as an integer multiple of the final display pixel (e.g., 2x2 subpixels or 4x4 subpixels per pixel.) The contribution of each polygon to the final pixel then corresponds to the number and pattern of subpixels occulted by the polygon. The exact definition of what causes a subpixel to be included in the mask is architecture specific, but the general case is shown in the figure below.

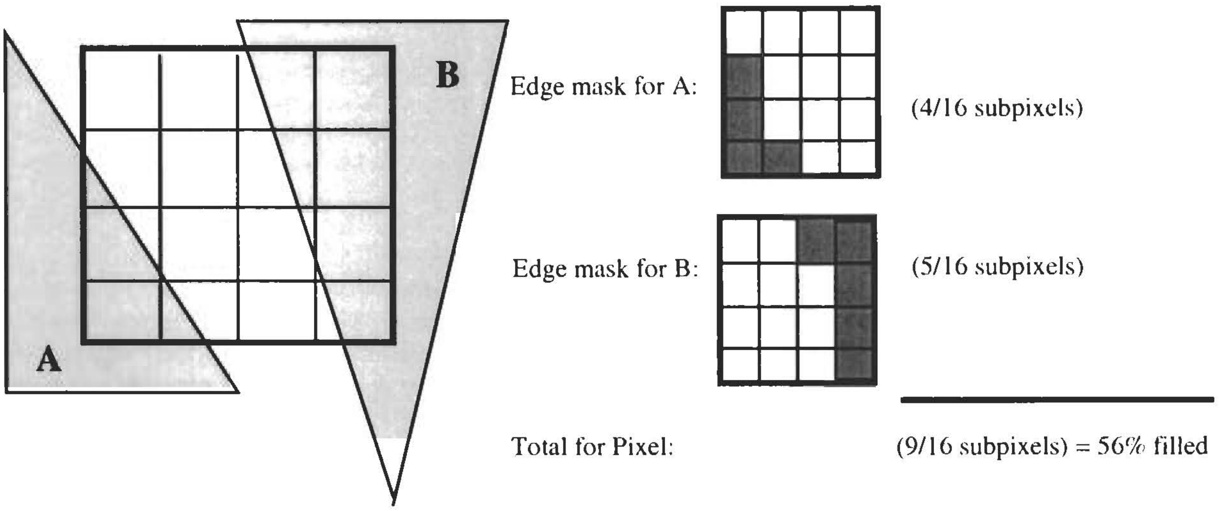

Figure 1 - Edge Mask Generation

Given the pattern or *edge-mask* of subpixels occulted by a particular polygon, its contribution can be either a direct function of the percentage of the subpixels covered, or the output of an *area-weighted* function that assigns different weights to each subpixel according to its position in the mask.

Rendering at subpixel resolution increases the amount of processing linearly with the number of subpixels per pixel. Therefore, in low cost systems, a small number of subpixels may be used (e.g., 2x2) and combined with a weighting scheme to still yield adequate edge smoothing.

7.3.2 Depth Anti-Aliasing

For any dynamic priority scheme, such as a depth buffer, the IG implementation of the algorithm has finite precision. This results in finite spatial resolution for rendering virtual environments during a simulation. If inadequate resolution is used, features that are close together will exhibit increasing priority anomalies as their distance from the eyepoint increases. Priority anomalies generally manifest themselves as occulting errors ("The tree is in front of the house, now it's behind, now it's in front...") In the following paragraphs, three factors that determine the depth accuracy of an image generation system are considered - limited spatial resolution, depth sampling accuracy and the interpolation of adjacent depth values.

Limited Spatial Resolution - Low-cost image generation systems which employ a hardware depth buffer generally use fixed point depth values in an effort to save the substantial hardware cost of floating-point depth calculations. However, if fixed-point mathematics is used, it is important to realize that spatial resolution is limited by the number of bits of depth resolution and the maximum viewable distance. Spatial resolution for an N-bit fixed-point depth calculation using linear depth mapping is given by the formula:

$$Spatial\,\mathrm{Re}\,solution = \frac{(MaxDepth - MinDepth)}{2^N}$$

Consider a gunnery training application where targets must be visible from immediately in front of the eyepoint to 2 kilometers. If a 16-bit fixed-point depth buffer is used, then the depth resolution available is (2000 meters) / 65536 values = 3 centimeters – resulting in very accurate depth calculations. However, if an 8-bit depth buffer is used, the resolution drops to (2000 meters)/256 values = 7.8 meters – an unacceptably high granularity for depth comparisons. Remember that priority resolution of nearby objects in a scene is limited to the accuracy of the depth calculations. Therefore, any objects closer to each other than the spatial resolution of the IG system can exhibit priority flashing during a simulation.

If floating-point depth calculations are performed, spatial resolution is still limited by the number of bits of numerical resolution. However, due to their exponential multiplier, floating-point representations avoid the need for the linear mapping of distance values as shown in the earlier formula. Therefore, floating-point calculations more gracefully handle the juxtaposition of close and distant objects since the range between maximum and minimum depth can be much greater than in a fixed-point system.

Depth Sampling Accuracy - The previous paragraphs demonstrated that depth calculation schemes have inherently limited resolutions in the comparison of two depth values. The frequency of performing these depth calculations is every pixel, every subpixel using the same subpixels as the edge-masks, or every subpixel but using a larger subpixel than the geometry calculations. If the sampling is performed at too coarse a resolution, then pixel coloring will be adversely affected. Coloring is degraded because object priorities will be incorrectly resolved for some portion of the depth sampling interval since the depth will be constant throughout the interval.

Consider the illustration below as an example of depth sampling accuracy. In this figure, a gun turret is pointing through foliage with the end of the tube visible from the front view. As shown in the top view, the gun should have priority over the foliage for two subpixels, but since the depth subpixels are larger than the edge mask subpixels, the distance used for the gun in the depth buffer put it behind the front edge of the foliage for one of the two subpixels. The color contribution of the gun to the final rendering is only 50% of what it should be because of limited depth sampling.

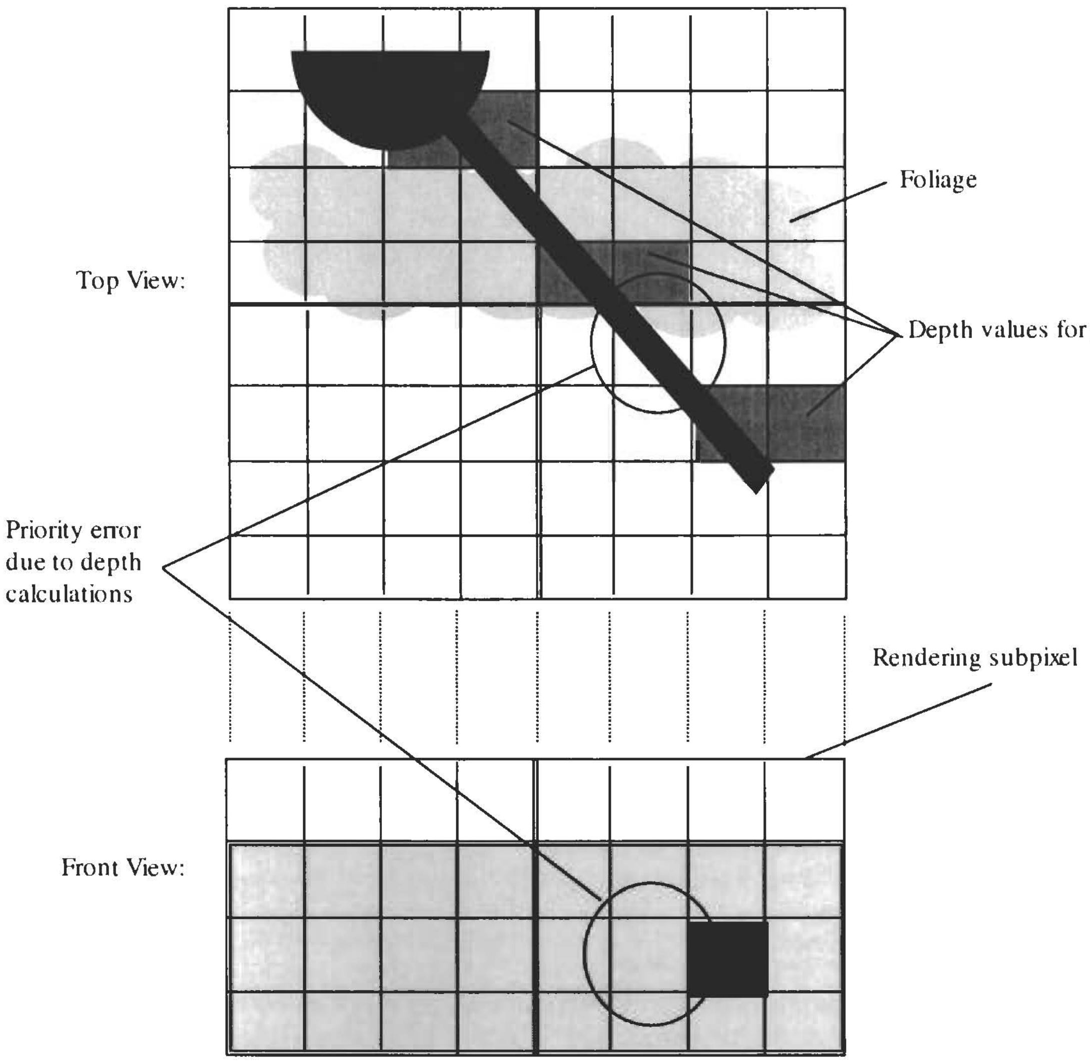

Figure 2 - Aliasing of Depth Calculations - Top and Front Views

<u>Interpolation of Adjacent Depth Values</u> - To save on the number of computations, most depth algorithms calculate the polygon depth at a subpixel by interpolating from the depth of the adjacent (and just previously calculated) subpixel. This eliminates the need to go back to the original transformed database vertex coordinates for each depth sample, but it opens the door for error accumulation as a function of the number of sequential approximations allowed.

Each image generation system addresses the depth calculation issues discussed above during design. Characterization tests to assess an IG for use with a particular application must attempt to detect the rendering differences resulting from the design tradeoffs in the context of what training applications the system will be used for. When characterizing the performance of a low cost image generation system with regard to depth anti-aliasing, the following issues should be addressed:

- Are the depth values in fixed or floating point representation? Floating point is generally superior because it avoids the scaling resolution problems used in fixed-point systems. Also, most database generation systems use floating-point representations, so it will be easier to develop databases for a floating-point depth system.

- What is the depth resolution in screen-coordinates (2x2 pixel spans, pixels, or sub-pixels)? Is there a mismatch in the size between the edge masks and the depth calculations?
- How many sequential depth approximations are made based on previous data and what is the resolution of the interpolation function (i.e., how much error is introduced by the interpolation?)

7.3.3 Texture Anti-Aliasing

In the MIP map approach, a series of maps with different spatial resolution is available. The IG generally blends between any two of these maps to smooth the patterns on textured polygons. Such a process uses *tri-linear or 3D interpolation* : the first two dimensions (which we refer to as alpha and beta) are along the two axes of each texture map while the third dimension is the blending value between the two maps used to generate pixel color. The three axes of interpolation for a textured polygon's pixel are illustrated in the figure below. The interpolation is performed by finding alpha and beta values corresponding to each of the four corners of the pixel to be rendered. Eight texture values are read from the maps (four corner values from each map). Then the tri-linear interpolation between the eight values is performed across the pixel surface.

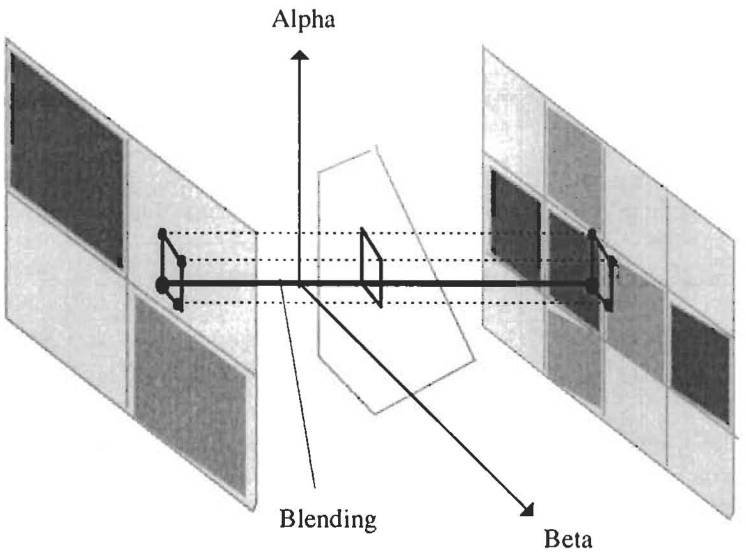

Figure 3 - Tri-Linear Texture Interpolation

Tri-linear interpolation produces excellent texture results and is the method of choice for medium and most high-performance IGs today. However, this process can produce artifacts as the polygon becomes perpendicular to the eyepoint. In this case, texture becomes artificially smooth until it almost completely "blurs out" as the polygon becomes perpendicular. To address this problem on high-end systems, an alternate texturing approach called *anisotropic texturing* is employed which remains sharp at all angles. However, anisotropic texturing is more expensive to implement than tri-linear interpolation.

At the time of this writing, neither tri-linear nor anisotropic texturing is widely available on low cost systems. Less expensive algorithms (bilinear and point-sampling) exist but they generally produce lower quality results. To compare lower-cost systems look for the following artifacts:

- Crawling or shifting of texture with respect to the polygon as the polygon is moved around on the screen. This indicates poor texel address mapping functions.

- Blurring or fuzzing as the polygon becomes "edge-on" (perpendicular to the eyepoint). This happens when too many texels are averaged together into the final pixel color. This effect can be somewhat minimized by creative texturing in the database modeling process. The only true cure is an anisotropic texturing algorithm.
- Level of detail popping as a polygon moves from a far distance to a quite close distance. Popping is seen if there are not enough levels of MIP maps or if the texture process does not smooth between texture layers. If no smoothing is done, visual aliasing will result in scintillation during polygon movement.

7.4 TRANSPARENCY

Transparency in computer generated imagery can generally be created by two means: through *polygon transparency* or through *texture transparency*. Polygon transparency, previously mentioned in section 0, refers to assigning alpha (opacity) values to individual polygons.

In IG systems that use subpixel anti-aliasing, when it comes time to calculate the pixel color contribution from a partially transparent polygon, the polygon alpha is used to construct a *transparency mask* similar to the edge mask used in geometric anti-aliasing (see section 0.) Transparency subpixel masks are generally uniform patterns constrained to the percentage of opaque subpixels equal to the alpha value of the polygon. In this approach, the resolution of the subpixels directly affects the potential levels of transparency since transparency is handled through subpixel accumulation at the same time that edge masks are evaluated. This approach results in beautiful imagery, but is generally used in moderate to high-cost systems because of its hardware complexity.

Texture transparency is implemented by supporting a special *transparency flag* for a texture map texel that indicates the texture (and therefore the polygon as well) is transparent at this location.

7.5 PIXEL PROCESSING LOAD MANAGEMENT

Pixel processing overload conditions can occur when there are too many polygons for the priority algorithms to successfully function. How much pixel processing load the system can handle and what load management options are available is system dependent.

Systems that use a depth buffer for priority resolution are at a distinct disadvantage for load management since correct priority is unknown until all visible polygons in a scene have been processed. Earlier static priority schemes (such as separating planes) can generally traverse the database in close-to-far order that minimizes visual anomalies in overload conditions.

When an overload condition exists in only the pixel processing subsystem, a status message is generally sent back to the host computer indicating the situation so that future frames can have an adjusted load. Increasing the fog or haze in the scene to obscure rendering problems in far-away objects is about the only useful technique that can be performed directly during pixel processing. This technique is only effective if the IG processes objects in a close-to-far order.

Most low cost graphics systems do not address load management during pixel processing since they commonly use the "render until done" processing method without the strict time limitations associated with fixed update rate simulation applications. In particular, there is generally no way to provide feedback from the pixel processing hardware to indicate overload when it occurs. In such systems, it is important to assure that pixel processing overload never occurs since it cannot be handled gracefully by the hardware.

7.6 AMBIENT VISIBILITY / DEPTH CUEING

Viewing conditions such as haze, fading to a uniform background color, and fog are generally handled by the pixel coloring algorithms in the pixel processing subsystem. As previously described, a polygon's contribution to pixel colors is a function of the edge and depth masks that determine the percentage of the pixel occulted by this polygon. In addition to these earlier factors, a *fading function* is generally used to gradually blend a polygon's color into the background color as the distance between the eyepoint and the polygon increases. Pixel color is calculated according to the relationship:

$$Color = FadingFunc(R) * BackgroundColor + FadingFunc(1/R) * PolygonColor$$

where R is the slant range between the polygon and the eyepoint. The fading function can be linear, exponential, or whatever is appropriate for the individual training application. Fading function designs vary between IG. They can be fixed table lookups that require hardware modifications to update or they can be dynamically loadable tables that can be initialized before or during a simulation.

Hardware supported fading is expensive since the above equation must be repeated for each pixel in the scene and for each of the three colors per pixel (assuming an RGB color scheme). Because of its expense, this capability is not always present in low cost systems. For example, the Matrox Mystique, a current low cost PCI graphics board offers hardware acceleration for pixel processing but does not implement fading functions.

8. VIDEO CHARACTERISTICS

8.1 DIGITAL TO ANALOG (D/A) RESOLUTION

The number of colors that can be directly rendered by a graphics subsystem is limited by the pixel depth in the frame buffer memory and the resolution of the VDACs (video digital to analog converters) which create the video signals. Most medium and high-performance image generation systems use 24-bit color depth in an RGB format (eight bits for each color) to yield 16.7 million potential colors.

For some training applications, it may be acceptable to have less color resolution (fewer colors). In this case, eight or sixteen bit frame buffers using a *dithering algorithm* and *color lookup tables* may provide adequate resolution. Eight and sixteen bit buffers can display a maximum of 256 and 65536 colors simultaneously, respectively. Dithering creates the appearance of more colors by creating patterns of differently colored pixels interspersed so the eye sees the "average" of the colors used. Color lookup tables provide a final color-to-color mapping and allow the fixed number of colors available (based on pixel depth) to be allocated to any set of hues desired by the database modeler. For example, in a night-vision goggle scenario everything is rendered in shades of green due to the goggles. A color lookup table would enable an 8-bit frame buffer to provide 256 shades of green – providing enough resolution for most applications.

Most COTS graphics cards at the time of this writing are capable of supporting one channel of 8-bit deep video for sizes 1280x1024 or smaller with 2 megabytes of video RAM. With 4 megabytes of VRAM, most cards can deliver pixel depths of 24-bits for the same display size.

8.2 OUTPUTS

This characteristic refers to the number of separate video channels that can be generated by the IG or graphics card. Separate video channels are useful when a training system requires multiple out-the-window displays and for embedded applications where the vision blocks of existing vehicles are driven with synthetic imagery.

Most dedicated image generation systems can scale the number of video channels to be generated through the addition of extra video memory cards. However, the depth complexity and polygon processing capacities are shared between all video outputs so a larger number of generated pixels results in sparser scene detail as dedicated by the IG depth complexity limitations. Refer to section 0 for details on depth complexity.

For low cost graphics cards installed in COTS workstations, another consideration is whether the graphics display can render within a window on the main console or whether it requires a separate video monitor. For the majority of dedicated training systems, a separate monitor installed in the training mock-up is probably the best choice, but in-window rendering may offer some advantages.

8.3 FORMATS

This characteristic refers to the number and type of standard video signals that can be produced by the IG. Standard analog formats include RS-170 (separate R,G,B signals with the sync pulse generally on the green.), NTSC composite video, Betacam, SuperVGA, and S-VHS. Standard digital video formats include D-1 and VGA computer monitor video. The majority of low cost graphics cards generate SuperVGA and/or NTSC composite video to drive COTS monitors. For an overview of video memory and video generation see [Artwick84].

If the application includes potential interaction with European video equipment, the PAL or SECAM formats may be necessary. However, external video format conversion hardware is commercially available to convert most standard input types into whatever other standard is needed. Resolution and video quality may suffer during the conversion process (which often involves A/D, resampling, and then D/A conversion).

Since most low cost systems use SuperVGA and/or NTSC video, this capability should be easily available in COTS graphics boards. For example, the Matrox Mystique directly generates SVGA video and a daughtercard option is available to produce NTSC video as well.

8.4 GENLOCK

When the IG is part of a larger video system or several IGs are used together in a simulation application, it can become necessary to synchronize all video frame updates to occur simultaneously. This is accomplished by running a master synchronization signal (called the *master clock* or *master sync)* into the *genlock* inputs of all video systems. A video system with genlock ability is capable of slaving its timing circuitry to an external video signal.

This feature is necessary for any applications where synthetic targets or special effects are generated by the IG and overlaid onto another video signal. The background signal and the IG signal would be genlocked for synchronization and fed into an external video mixer.

This feature is not generally expensive to implement and should soon be more widely available on any video or graphics systems designed to address more than driving a single, dedicated video monitor.

8.5 POST-RENDER FILTERING

This characteristic of a low cost IG refers to whether access is provided to the video or frame buffer memory in a way that facilitates post-processing of the synthetic imagery. For example, a reasonable level of geometric anti-aliasing can be accomplished by down sampling and smoothing the image generated by the IG. If this smoothing is enough for the desired training application, then there is no need for more expensive subpixel edge mask and depth mask generation. This technique was employed successfully on the SIMNET system for the video used in the tank vision blocks.

Simply stated, does the IG provide access and enough control signals to easily support either video or direct memory access-based post processing options?

9. FEATURES

Many applications requiring image generation technology call for special effects or specific interactions with the synthetic environment that have been addressed in the IG design. This section describes several of the most important of these features.

9.1 HEIGHT ABOVE TERRAIN / SOIL TYPE

In some simulation systems, the IG is the only part of the system with a complete terrain database. In others, it is the only subsystem with the full-resolution database used to generate imagery. For each case, queries about terrain elevation and slope must be resolved by the IG since it is the sole possessor of this information.

For full query functionality, an IG should be able to provide the elevation value, slope vector, and maybe also surface material type for any location in the database referenced by the host computer. This would be used by a host computer for its *terrain following algorithm*. In a typical terrain following application, the host computer will query a set of four to sixteen points around the perimeter of the vehicle and use the elevations of these points to position the hull and wheels, tracks, or other articulated parts of the vehicle on the synthetic terrain.

To quantify this characteristic in a low cost system, determine whether a query ability is supported at all. If it is supported, determine the number of query points allowed per frame and the transport delay between the request and the answer. This feature is particular to image generation systems and will not be supported on most low cost COTS graphics cards.

9.2 COLLISION DETECTION / SURFACE CONTACT

Detecting collisions between objects in the environment or between the own ship and the environment are a primary requirement for many simulation applications. Simple implementations will detect the overlap of bounding boxes between objects while high-fidelity solutions will support hierarchical, articulated bounding boxes for moving models and 3D cultural features. In vehicle simulation applications, collision detection between the own ship and the environment is detected by the IG and used as a signal to affect the vehicle dynamics simulation running on the host computer.

To quantify this characteristic in a low cost system, determine whether a collision detection ability is supported at all. If it is supported, determine the number of collision volume tests allowed per frame and the extent to which articulation is supported (i.e., the number of levels allowed.) Also determine whether collision detection takes texture transparency into account. For example, a gun turret should not collide with the transparent part of a tree polygon because the transparent area is not supposed to be occupied by the tree. Collision detection is a problem particular to simulation and virtual reality systems and is not likely to be supported on most low cost COTS graphics cards.

9.3 LINE OF SIGHT

For gunnery or target acquisition applications, it is necessary to determine the distance between the eye point and any point in the synthetic environment. The environment point could be on a 3D cultural object attached to the terrain or it could be a moving model on top of the terrain. Depending on the system design, these calculations could be done on the host computer or assigned to the image generation system.

Early, high-performance systems relegated line of sight (LOS) testing to the IG since the IG had the complete database and many simulation applications were centered on driving or fighting performance of own ships in a relatively sparse environment. IG LOS testing is particular effective for range finding applications in gunnery trainers or target acquisition trainers.

Line of sight testing is usually done by identifying a pixel in display coordinates. The IG then returns the slant range to the polygonal object closest to the eyepoint visible in that display pixel. Texture transparency should be handled correctly so that the range to a complete transparent feature is not incorrectly returned.

As DIS has developed and simulations involve more moving entities of all types and purposes, it is less likely for the IG to handle the LOS testing because of its knowledge of the entities in the environment is restricted to their polygonal profiles.

Because of the uniqueness of this algorithm to military simulation applications, few if any COTS low cost graphics cards will address it. However, that is less of a limitation for current and future applications since the host computer often has a redundant database for analytical purposes now (LOS, tracking DIS entities, etc.)

9.4 WEATHER EFFECTS

Weather effects which can occur in simulation scenarios include haze, cloud decks with scud (irregular bottom or top elevations), rain, and lightning. All of these effects can be handled well using combinations of texture, illumination, and fading functions under either local control by the IG or external control by the host computer. Cloud decks can be simulated by two randomly textured polygons placed parallel to the ground above the eyepoint. When the eye point nears the elevation of the cloud deck, the fading distance can be changed to cause quick fading to simulate the opacity of the cloud interior. The irregular edges of scud above or below a cloud deck can be simulated by bringing the fading distance in and out suddenly when the eyepoint is near the location of the cloud deck. Lightning can be simulated by dynamic changes to the illumination pattern in the display pixels. A sudden bright flash (possible by changing the illumination map on the screen) is effective.

From the above descriptions, it follows that a low cost system with texture transparency, fading, and illumination ability can implement weather effects within the IG subsystem or under host control. If a system does not have a fading function, then a partially transparent polygon can be placed to completely cover the field of view to simulate the opacity of a cloudbank. Completely covering the field of view adds considerable depth complexity to the scene (one visit/per pixel just for the transparent polygon), so this technique is generally not practical for low cost systems.

Full support for weather effects by an image generation system implies the temporal changes in fading and illumination would be handled automatically as a result of setting the weather conditions once from the host computer. As a rule, only moderate to high-performance IGs provide this feature.

9.5 SENSOR REPRESENTATIONS

Real-time true sensor modeling is extremely difficult with today's technology for even high-performance image generation systems. Radar, infrared, and night-vision sensors each exhibit unique, irregular signal-to-noise ratios and technology-specific visual artifacts. Exact reproduction requires a detailed sensor simulation.

One previously employed technique assigns reflectivity and ambient heat values to all polygons in the scene. These values are input to color lookup tables in the rendering hardware which function as a rudimentary sensor model. A second (and more powerful) technique uses the post-render filtering approach described previously. With this approach, a dedicated additional video output board that includes filters for sensor effects is added to the IG. Better simulation of AC and DC signal-to-noise artifacts can be accomplished using the post-processing effect.

Because of its obvious cost, true sensor simulation is generally not possible in low cost IG systems, but either of the above techniques can be employed if the hardware includes color lookup table or post-render filtering capability.

9.6 TIME OF DAY

Time of day cues are most commonly associated with two cues: sun angle and shadows. Sun angle is easy to implement in the polygon processing subsystem of an IG because it is simply a polygon shading algorithm. Sun angle requires flat or smooth shading and an algorithm that takes the sun angle into account when shading each polygon in the scene. This capability should be readily available on most low cost visual systems.

Real-time shadows are a more difficult problem. Shadows require traversal and collision detection on the database since the shadows of objects must be projected onto the terrain and potentially other objects. For further reading on a modification of the binary separating plane priority approach that calculates and incorporates shadows, refer to [Chin89].

10. OPERATING ENVIRONMENT

Since many of the low cost systems available as commercial products are graphics accelerators for standard workstations, they are designed toward use with commercial software standard Application Programmer's Interfaces (APIs) and standard operating systems. In general, this increases the software development necessary to successfully equip these systems for use in interactive trainers.

Much of what separates an IG from a high-performance graphics accelerator is the IG's ability to perform run-time load management and scene content management in fixed update rate applications. Before COTS hardware is chosen for a low cost visual system application, the amount of software development necessary should be carefully estimated. With a COTS hardware platform, software development will include re-inventing much of what the simulationcommunity expects from a visual system. To support our position, we further discuss some of the software integration issues in this section.

10.1 GRAPHICS API OPTIONS

For Intel or Intel-compatible workstations, the two most-widely used API's available are OpenGL and Direct3D. Both standards are designed for general-purpose graphics applications in contrast to the specific needs of the interactive simulation community. Therefore, many of the characteristics described previously in this paper and generally expected in an image generation system will have to be implemented using OpenGL, Direct3D or higher level API calls as part of the host computer software.

OpenGL is an industry standard rendering library interface that evolved from the proprietary graphics library (GL) developed by Silicon Graphics. The OpenGL standard defines a virtual rendering machine with state variables accessible through library calls. When any OpenGL state variables are changed, the effects are visible on the next rendered frame. Limited support for queries of the database (such as line of sight) is available through OpenGL's *feedback* mode. Basic fading functions are also supported.

Direct3D is an interactive graphics API proposed by Microsoft as part of its DirectX interface standard for multimedia applications. Because of its orientation directly to the consumer market, it is the most popular interface for low cost graphics hardware for the Microsoft Windows-compatible hardware platforms. The Direct3D interface is separated into two levels, Immediate Mode and Retained Mode. Retained Mode is a high-level programming layer that provides support for features such as hierarchical objects, animations, lighting, and fading effects. In contrast, the Immediate Mode interface is a low-level, hardware independent data format designed to be directly implemented by 3D acceleration hardware. The Retained Mode API layer calls are implemented using the Immediate Mode layer calls. When compared to OpenGL, the Retained Mode is a slightly higher-level interface while the Immediate mode is considerably more detailed and close to the hardware. Support is available in Direct3D for fading functions, texture transparency and model articulation among other features.

In contrast to OpenGL and Direct3D, the OpenGVS API is designed specifically for interactive simulation. Designed to simulate a visual system, OpenGVS provides an interface that supports weather effects, articulated models attached to DCSs, and overload management in a way that harnesses COTS graphics hardware and performs much like a dedicated image generation system. For many simulation applications where it provides adequate performance, OpenGVS might substantially reduce the custom software development necessary to integrate visuals into the remainder of the training system.

10.2 SUPPORTED OPERATING SYSTEMS

By definition, a real-time visual system must be capable of supporting a fixed update rate under light and heavy load conditions. Load conditions vary with the database and the eye point direction, and it is up to the visual system to successfully deal with the temporally varying load. This has resulted in the development of custom run-time software environments in today's high-performance IGs.

Since the situation is similar for Intel, Motorola, and other non-Intel hardware platforms, the remainder of this section will focus on the Intel architecture without loss of its generality to other platforms. Low-cost, commercial hardware has been designed to work with industry standard operating systems such as DOS, Windows, or Windows NT. While they are not as widely used, several commercial and non-commercial versions of the Unix operating system (i.e., Solaris-x86, Linux, SCO Unix, NeXTStep) are also available on Intel platforms.

To address real-time needs, an operating system must include:

- Access to a hardware-based real-time clock to serve as a timing standard on which the fixed update rate system relies. The clock will serve as a reference to determine how much time remains before processing must begin on the next video frame.
- Sophisticated software and hardware interrupt support. The operating system will use software interrupts to signal the completing of events such as database tree traversal as well as managing the flow of incoming control data from input devices or from a computer network.
- A task scheduler that balances many simultaneous tasks and preferably provides a mechanism for supporting task overruns. This may take the form of sending software signals to the processing indicating an overrun has occurred and managing the fast clean-up of the current frame before beginning the next frame.

Unix and Windows NT (because it has a Unix-like kernel) have the most powerful support for these features, but pseudo-real-time programming can be conducted in any of the commercial operating systems. However, Unix currently offers the most stable platform that is compatible with legacy workstation development environments.

When comparing low cost visual hardware for use in simulation systems, life cycle costs may be substantially lower for hardware supported by multiple operating systems. This removes the problem of being locked into a particular software environment because of the hardware investment. Because of their adherence to hardware and software standards, most commercial hardware can successfully host multiple operating systems.

11. SUMMARY

An image generation system consists of various characteristics that distinguish its ability to produce synthetic imagery based upon a digital representation of the environment. These characteristics were presented in the sections above. Higher end systems not only exhibit better performance of these characteristics but make available more of the characteristics. Consequently, they generally carry a higher price tag. A simulation may not necessarily need, or be able to afford, a high end image generation system. A lower cost image generation system may be adequate for serving its purpose. It is the intent of the benchmarking process to assess the image generation performance of a system (or card within a designated system) that meets the requirements for a particular type of simulation. Understanding the simulation's synthetic environment is crucial to developing benchmarking performance metrics.

In an effort to describe the synthetic environment used by a simulation, functional categories were derived. A synthetic environment can be thought of as consisting of terrain, 2D culture (e.g., roads, rivers), 3D culture (e.g., buildings, trees), moving models (e.g., vehicles, humans), and special effects (e.g., weather, dust clouds, engine exhaust). The influence of each characteristic on the synthetic environment aspects listed above was analyzed for future use in narrowing the list of image generation characteristics that will be tested during a benchmarking operation. The Appendix on the following pages includes a table of all of the image generation characteristics described in this report and their influence on the synthetic environment.

APPENDIX - Image Generation Characteristics and Their Impact on the Synthetic Environment

“All” appears once on a row for characteristics that effect all aspects of the synthetic environment.

BIBLIOGRAPHY

[Artwick84] B. Artwick, Applied Concepts in Microcomputer Graphics, Prentice-Hall, Englewwod Cliffs, NJ 07632, 1984, pp.60-125

[Chin89] N. Chin and S. Feiner, "Near Real-Time Shadow Generation Using BSP Trees", Proceedings of SIGGRAPH '89, (Boston, Mass, July 31-August 4, 1989) in *computer Graphics 23,4*, August 1989, ACM SIGGRAPH, New York, NY, pp.99-106.

[Cosman94] M. Cosman, "Gobal Terrain Texture: Lowering the Cost", Proceedings of the 1994 IMAGE VII Conference, The IMAGE Society, Tucson, AZ, 12-17 June 1994, pp.53-64.

[Eck95] M. Eck, et al., "Multiresolution Analysis of Arbitrary Meshes", Proceedings of SIGGRAPH 95 (Los Angeles, CA, August 6-11, 1995). In *Computer Graphics* Proceedings, Annual Conference Series, 1995, ACM SIGGRAPH, pp. 173-182.

[Foley92] Foley, vanDam, Feiner, Hughes, Computer Graphics: Principles and Practice, Second Edition, Addison Wesley, 1992.

[Fuchs80] H. Fuchs, Z. Kedem, and B. Naylor, "On Visible Surface Generation by a Priori Tree Structures", Computer Graphics V. 14(3), pp. 124-133, June 1980.

[IST97] Institute for Simulation and Training, "A Comparison of Military and Commercial Specifications for Visual Systems", August 1997.

[Lindbloom89] B. Lindbloom, "Accurate Color Reproduction for Computer Graphics Applications", Proceedings of SIGGRAPH '89, (Boston, Mass, July 31-August 4, 1989) in *computer Graphics 23,4*, August 1989, ACM SIGGRAPH, New York, NY, pp.117-126.

[Naylor95] B. Naylor, "What Trees Can Do For Large Synthetic Environments", Proceedings of the 1st Workshop on Simulation and Interaction in Virtual Environments: SIVE'95, Iowa City, IA, July 13-15, 1995.

[OpenGL92] OpenGL reference manual: the official reference document for OpenGL, release 1, Addison-Wesley Publishing Company, Reading, Massachusetts, 1992.

[Renze95] K. Renze and J. Oliver, "Generalized Surface and Volume Decimation for Unstructured Tessellated Domains", Proceedings of the 1st Workshop on Simulation and Interaction in Virtual Environments: SIVE'95, Iowa City, IA, July 13-15, 1995.

[Schroeder94] W. Schroeder, B. Yamrom, "A Compact Cell Structure for Scientific Visualization", Course Notes: Advanced Techniques for Scientific Visualization, 21st International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'94), Association for Computing Machinery, July 24-29, 1994, pp. I47-I51.

[Torborg96] J. Torborg, J. Kajiya, "Talisman: Commodity Realtime 3D Graphics for the PC", Proceedings of SIGGRAPH '96.

[Turk92] G. Turk, "Re-Tiling Polygonal Surfaces", Proceedings of SIGGRAPH'92, Computer Graphics, Vol. 26 Number 2, July 1992, pp.55-64.

	Aspects of Synthetic Environment				
3D Graphics Characteristic	**Terrain**	**2D Culture**	**3D Culture**	**Moving Models**	**Special Effects**
System Characteristics					
Update Rate	All				
Transport Delay	All				
Number of Dynamic Coordinate Systems / Levels of Articulation					
Eyepoints	Distant eyepoints may cause database paging problems - need to page two terrain areas				
View Volumes / Field of View	FOV drives the polygon count of scenes				
Distortion Correction	All				
Area Processing				Priority against environment decided per area	
System Level Load Management	Distant terrain can contribute to overload conditions				
Geometric Processing Characteristics					
Polygon Throughput					
Stamps					
Lights					
Geometric Level Of Detail	Can have multiple LODs or continuous LOD	Should have the same number of LODs as underlying terrain	Advantageous to have multiple LODs to save on polygon budget	Generally 4 or 5 LODs of models	Generally only one LOD since effects are transient
Color / Transparency	Polygon transparency used to blend terrain LODs		Polygon transparency - windows; texture transparency - trees	Transparency on vehicle windows	Texture transparency used on explosions, exhaust flames
Shading	Usually flat or smooth shaded by sun angle		Usually flat or smooth shaded by sun angle	Can have dynamic shading or fixed angle shading	
Illumination	Illumination by night flares or by ownship lights	Illumination by night flares or by ownship lights	Illumination by night flares or by ownship lights	Illumination by night flares or by ownship lights	
Environment Memory	Bulk of memory capacity used by terrain. Memory sized for terrain	Generally attached to a terrain patch	Generally attached by model reference from the terrain	One copy of model def'n stored here and instanced when needed	

	Aspects of Synthetic Environment				
3D Graphics Characteristic	**Terrain**	**2D Culture**	**3D Culture**	**Moving Models**	**Special Effects**
Paging Rate	Rate effects number of load modules which can be paged in and size of viewable area				
Geometry Processing Load Management	Terrain often 50% of polygon cost. Overload could cause closer far clipping plane or simpler terrain LODs		Overload could cause display of lower LODs	Except around 30% of polygon cost for ground applications. Overload causes use of lower LODs.	Transient nature keeps effects from causing overload (not for long, anyway)
Texture Characteristics					
Format	Geospecific imagery or generic imagery used for texture maps			Generally only generic texture used on models	Explosions, exhaust, smoke are textured images with transparent borders
Texture Level Of Detail / MIP mapping	Many terrain LODs of imagery needed in aerial vehicle applications		Imagery of geospecific buildings or targets supported using MIP maps		
Texture Memory	Memory update required for large geospecific missions because of size of terrain imagery		Geospecific targets are expensive and may require dynamic update during simulation.		
Microtexture		micro patterns of grass texture and rubber on runways / roads			
Perspective Correct Texture Mapping					
Pixel Processing Characteristics					
Pixel Fill Rate / Depth Complexity	Sky and ground contribute to at least 1 visit/pixel for any scene		Can substantially contribute to depth complexity if many individual trees or similar objects are used	Close-up views of models can be very expensive and must be considered in the system performance estimates	
Priority (Fixed List, BSPs, Depth Buffering, Polygon Layers)	Static features can be handled by BSP or fixed priority list	Static features can be handled by BSP or fixed priority list	Static features can be handled by BSP or fixed priority list	Requires dynamic priority wrt each other and the surrounding environment; Z-buffer is popular choice	Requires dynamic priority wrt each other and the surrounding environment; Z-buffer is popular choice

	Aspects of Synthetic Environment				
3D Graphics Characteristic	**Terrain**	**2D Culture**	**3D Culture**	**Moving Models**	**Special Effects**
Anti-Aliasing (Geometry, Depth, Texture)	Anti-aliasing very important for overall image quality	Anti-aliasing very important for overall image quality	Anti-aliasing very important for overall image quality	Look for quality of model vs. terrain priority and geometry conflict resolution	
Transparency			Used in windows; bridges; fences		Used extensively for good special effects
Pixel Processing Load Management	All				
Ambient Visibility (Haze)/Depth Cueing/Fogging	Lower visibility allows for closer terrain clipping plane and lower polygon count				
Video Characteristics					
Resolution	All				
Outputs					
Formats					
Genlock					
Post-Render Filtering					Filtering for sensor effects proven useful in ESIG2000
Features					
Height Above Terrain / Soil Type	Terrain database queried by IG for points requested	May affect the soil type return value from the IG			
Collision Detection / Surface Contact			collision volumes must be included for 3D culture (buildings, bridges, berms, etc.)	Collision volumes must be articulated to match vehicle specifications	
Line of Sight	Must detect terrain polygons		Must detect 3D culture polygons (potential targets); Must NOT detect transparent areas of tree polygons	Must detect moving models (potential targets)	Must not detect transparent parts of smoke or explosions
Weather Effects				Cloud decks can be implemented with moving models	
Sensor Representations	May require special attributes on the polygons	May require special attributes on the polygons	May require special attributes on the polygons	May require special attributes on the polygons	May require special attributes on the polygons

	Aspects of Synthetic Environment				
3D Graphics Characteristic	**Terrain**	**2D Culture**	**3D Culture**	**Moving Models**	**Special Effects**
Time of Day	Sun angle shading performed on environment polygons	Sun angle shading performed on environment polygons	Sun angle shading performed on environment polygons		
Operating Environment					
Operating System	All				
3D Graphics API	All				

0000207