

Virtual Planetary Exploration: A Very Large Virtual Environment

Lewis E. Hitchner
hitchner@riacs.edu

Research Institute for Advanced Computer Science (RIACS)
NASA Ames Research Center
Moffett Field, CA

1. Introduction

Virtual Planetary Exploration (VPE) is a NASA Virtual Environment (VE) project. It is a "research tool for investigating concepts, methods, and user-interaction strategies that may prove useful for the design of planetary exploration workstations based on the virtual reality paradigm."¹ The goal of this application is to support "virtual exploration" of planetary terrain that exhibits geomorphological structure and detail usable by planetary geologists and rover vehicle engineers. VPE will be used to perform scientific observation and analysis and to assist mission planning and operations. This VE must not only be immersive and highly realistic but also scientifically accurate. VPE is expected to demonstrate that both unmanned and manned space exploration can benefit from "virtual exploration" capabilities.

The initial application of VPE is visualization of the surface of Mars. The data used in our geometric model has been measured, not synthesized, and it represents the entire surface of the planet. We use digital terrain data derived from images that were transmitted from NASA's Viking orbiter satellites and processed by the U. S. Geological Survey. VPE can also be operated with any other digital terrain data such as that currently available from Venus and Earth, or higher resolution data to be returned from the Moon and Mars in the next few years.

VPE project requirements are derived from two primary sources:

- NASA operational experience in planetary exploration and visualization of planetary terrain data
- investigation of and field research on the users' needs.

For example, we have studied methods used for 3D terrain visualization on Surveyor Moon missions and Mars Viking orbiter and lander missions. McGreevy and Stoker² have observed planetary geologists doing their work in the field on terrestrial sites that are used as planetary terrain analogs. This research has shown that our requirements are not only very application driven but also are quite different from most other Virtual Environment systems. First, our virtual scenes and interactions must have scientific validity, fidelity, and measurability as well as the capability to provide data enhancement via scientific visualization techniques. Secondly, our users' data needs encompass very large data sets and very large ranges of data resolution. Ideally, our virtual environment should support planetary surface exploration from the scale of satellite orbital imaging (for landing site selection and large scale geology study) to the scale of making a rover traversal or simulating an astronaut-geologist walking amongst boulder fields -- as well as all scales in between the two! Today's planetary terrain data sets are low resolution with surface sampling measured in kilometers or a few hundreds of meters. Such data sets are on the order of a few gigabytes in size. Tomorrow's planetary terrain data will be measured in meters or possibly centimeters. Thus, tomorrow's data sets will be larger by a couple of orders of magnitude -- squared!†

†Resolution is measured in linear distance. Since terrain data is collected over an area, an increase in resolution increases data set size by the square of that increase.

Obviously, such requirements present a problem to a VE system, where data visualization and virtual world interaction is desired at near real-time rates. VE technology today is hard pressed to meet our project's requirements. We believe in the future our requirements are likely to grow as fast or faster than VE technology growth. Therefore, our solution to meeting VPE project requirements is to make engineering trade-offs tuned to our users' needs. Our experience to date in designing and implementing the VPE system has demonstrated promising results.

2. VPE Project Overview

We first describe some of the history that led to the VPE project. The hardware and software systems are summarized next. This is followed by a description of the Mars digital terrain data we use for our terrain models, and our method for modeling and rendering terrains. Finally, descriptions of the two primary virtual environment operating modes of our system are presented.

2.1. History

The VPE project was initiated by Dr. Michael W. McGreevy of the Aerospace Human Factors Division at NASA's Ames Research Center in 1988. VPE has evolved from the pioneering virtual environment work done at NASA Ames by McGreevy and others beginning in the mid 1980's.³⁻⁷ The project has had a technical staff varying in size between one and four full-time and part-time persons since 1989. A working prototype version of a virtual planetary terrain exploration system has been running since mid 1990. Since then hundreds of visitors -- school children and teachers, astronauts and pilots, satellite engineers and researchers, geologists and oceanographers, video and photo journalists, artists and science fiction writers, senators and bureaucrats, colonels and sergeants, graphics gurus and computer geeks, and, of course, Mom and Dad -- have visited Mars (virtually) in our lab.

2.2. System Architecture

2.2.1. Hardware

VPE hardware consists of a 3D graphics workstation, a head-mounted display (HMD), a six degree-of-freedom (DOF) position tracker, and a six DOF joystick,

Through Spring 1992 the project used a Stardent GS2000 graphics workstation for interactive 3D graphics image synthesis. The GS2000 has a dual video channel full color high resolution display system: 1280 x 1024 24+8 bits/pixel. 3D graphics rendering hardware can generate a peak rate of 150,000 Gouraud shaded z-buffered triangles/sec. with a sustained 30 frames/sec. for 2200 polygons (one graphics pipeline for both video display systems). Our experience has shown those vendor supplied figures are highly optimistic. Z-buffering uses main system memory and is allocated 32 bits/pixel. The video memory refresh hardware can update full screen display memory via pixel copying from main memory at video refresh rates (74 Hz). System processors include four 5 MIPS CPU's and an 80 MFLOP single precision (40 MFLOP double precision) vector floating point processor. Our system has 128 MBytes main memory and four 760 MByte disk drives. Note, we consider all the above system resources and performances, except for the video memory pixel copy rate, insufficient for meeting our project's requirements. As a performance upgrade the VPE project is installing in Spring of 1992 a Silicon Graphics 4D/440 SkyWriter.

Our primary head-mounted display is a Virtual Research Flight Helmet. A second HMD is a VPL Model I EyePhone. Both systems use two color LCD's with 360 x 240 resolution containing 28,800 color triads. Both HMD systems include a wide field of view Leep optics system that presents approx. 100 degrees visual arc (total for both eyes) with roughly half of each eye's field overlapping for stereo viewing. The high resolution workstation monitor images are scan converted to NTSC composite video signals for input to the HMD by a pair of Folsom Systems scan converters.

Our interactive input devices include a six DOF Polhemus Isotrak magnetic field tracker for measuring head position and orientation. We use a Spatial Systems, Ltd. six DOF Spaceball joystick, and the Stardent workstation mouse and keyboard input devices. We have purchased and are in the process of integrating a pair of Exos Exoskeleton hand joint measurement systems into our VE system.

2.2.2. Software

Our system uses the following vendor supplied system software: Unix operating system (Stardent's Stellix, a primarily System V version of Unix), X Window System (Version 3), Stardent's proprietary (and minimal) X Toolkit, and Stardent's XFDDI 3D extension of X. We have not used any commercial vendor's virtual environment software, but instead have written about 25,000 lines of our own C code. As of May 1992 we were investigating commercial software products to use in collaboration with other NASA Ames VE research projects after our change to the new vendor's graphic workstation.

2.3. Planetary Terrain Data Sets

The VPE project uses 2D and 3D digital terrain data that was derived from NASA mission imagery and processed by the U. S. Geological Survey, Astrogeology Branch, Flagstaff, AZ. Our Mars data is from the Viking missions of 1976 through 1979. USGS has processed the orbiter images into two forms. Digital elevation data, called DTM (Digital Terrain Model), is available for the whole planet at a grid sampling resolution of 1/64th degree of latitude, which is approximately 925 meters. This data has been geodetically controlled and registered to a Sinusoidal Equal Area map projection. The elevation datum values are measured in meters, though the error in the data may be as much as two orders of magnitude greater than that. This data is provided as a 2D array of 16 bit integers and, thus, may be used as a height field to generate a 3D polygon mesh model of a terrain surface for rendering in a 3D graphics system.

In addition to elevation data, orbital satellite imagery for the whole planet surface has been mosaiced and geodetically controlled into digital image sets called MDIM's (Mosaiced Digital Image Models). These data are also projected into Sinusoidal Equal Area maps (except for North and Sole Pole regions), but they are available at a higher spatial sampling resolution of 1/256th degree (approx. 230 meters). Although these are monochrome data only (8 bits/pixel) they may be used for texture mapping on the polygon mesh surface derived from the DTM data. A non-linear transformation is required to map the MDIM texture map coordinates to each DTM polygon mesh vertex coordinate, however. Some color imagery of Mars is available, though none that has been geodetically controlled and registered is publicly available.

We also have obtained from USGS Flagstaff 8 bits/pixel monochrome digital data sets of images scanned by the panoramic cameras on board the Viking 1 and Viking 2 lander spacecraft. The panoramic cameras could scan up to 340 degrees field of view. Each lander has two cameras and, thus, we were able to obtain several pairs of stereo images. The USGS panoramic images were mosaiced from several originals in order to obtain optimal gray level exposures for all regions of each panorama scene.

The Mars DTM and MDIM data sets were obtained from the National Space Science Data Center (NSSDC)⁸ at NASA Goddard Research Center, Greenbelt, MD. Our original copies of these data sets were on 9 track magnetic tape. As of early 1992 the complete MDIM data set is available on 6 CDROM diskettes in ISO 9600 format.[†] By the summer of 1992 a seventh CDROM will be available that includes the Mars DTM elevation data. Each MDIM image is available in a file containing a 1280 by 1272 array of pixels and also in a file containing reduced resolution 1/64th degree pixels in a 320 by 318 array. The files cover about 1200 by 1200 km., or about 5 degrees of latitude and longitude (at the Equator). Each CDROM diskette contains about 650 MBytes of data.[‡] Software for viewing the images

[†]This data may be requested (free of charge to non-profit organizations) from:

National Space Science Data Center
Code 633.4
NASA Goddard Space Flight Center
Greenbelt, MD 20771
301-286-6695

NASA's vast space science database may be queried online through the NASA Master Directory by using telnet to login to host 128.183.36.23 with username "NSSDC" (i.e., NSSDC@NSSDCA.GSFC.NASA.GOV).

[‡]NASA Ames Research Center provides anonymous ftp access to the MDIM CDROM's (only one of them is mounted at any one time). The Mars MDIM's are on the Internet host ames.arc.nasa.gov (128.102.18.3) in the directory pub/SPACE/CDROM2.

on MacIntosh, IBM PC, DEC VAX, and Sun workstation computers is included on the CDROM's.

2.4. Terrain Modeling and Rendering

A VPE Mars terrain is modeled as a height field that is used to generate a 2D polygon mesh surface composed of triangles. Mars DTM data provide the altitude (Z) values. The base plane X and Y coordinates may be derived from each DTM's Sinusoidal Equal Area mapping projection parameters. Since nearly all our Mars data is large scale 1 km/pixel (or greater), our height fields cover large portions of the planet, e.g., tens of degrees of latitude and longitude. VPE terrain modeling includes planet surface curvature, and, thus, we map the height field onto the surface of a sphere. For simplicity, we assume a spherical planet with a radius of 3397.2 km.† As a program option a user may specify a vertical scale exaggeration factor that is used when the 3D terrain model is built from the DTM data. Interactive modification of vertical scaling is not currently supported.

VPE supports two methods for coloring our terrain surfaces. One is polygon vertex coloring that is used for Gouraud shading. Vertex colors may either be computed or read from a file. If colors are to be computed, the user may specify one of three pseudo-coloring methods and the HSV (hue, saturation, value) parameters for a linear color ramp as program options. The HSV pseudo-color mapping values may be modified interactively. The three methods are coloring by vertex elevation, by vertex slope angle, and by vertex slope aspect direction. "Vertex slope" is computed from the "vertex normal", i.e., the polygon vertex normal averaged from the surface normals of all common polygons sharing that vertex. In pseudo-color mode polygons may also be lighted with a directional light source. Vertex colors are modified by the lighting calculation (done in the graphics hardware) and incorporated into the hardware Gouraud shading.

The second polygon coloring method is texture mapping using the Mars MDIM satellite images. Since Mars MDIM's do not contain color information, we convert each MDIM pixel value into an HSV triple by generating hue and saturation with one of the three coloring methods above and then combining those with the MDIM gray scale intensity value. Each vertex in our height field polygon mesh must be mapped to a coordinate position in the texture map space. Even though both the DTM and MDIM pixel arrays conform to Sinusoidal Equal Area coordinates systems, the vertex mapping requires a trigonometric computation since the data sets usually have different centers of mapping projection.

VPE renders its virtual world scenes on a workstation with special purpose 3D graphics hardware. Visible surface calculation (a.k.a. hidden surface removal) is done by with a hardware z-buffer. VPE supports a number of alternate polygon rendering modes: wireframe, 2D colored markers (at each vertex), colored with no lighting, lighted with no coloring, lighted with vertex coloring, and texture mapped. In the three shaded polygon modes, either flat or Gouraud shading may be specified. Lighting options include one directional light source to represent the Sun plus ambient lighting. Ambient and diffuse coefficients as well as lighting direction may be modified interactively. We do not support shadow computation. In texture mapped mode no synthetic lighting is supported since shadows and lighting are incorporated in the MDIM satellite images used for our texture maps.

To generate the 3D terrain model a polygon mesh data structure is computed from a DTM data set during program initialization. For each vertex of the polygon mesh, this requires 3 floating point vertex coordinates, 3 floating point vertex normal coordinates, a 32 bit rgb color value, and two floating point texture map coordinate space values. If all these options are requested, each polygon vertex requires 36 bytes of program data memory. Our standard demo data set, the middle 1000 km by 800 km section of Valles Marineris, is a 400 by 500 grid that generates 200,000 vertices. Thus, it requires more than 7 MBytes for vertex data storage plus 15.5 MBytes if the texture mapping option is requested. The largest polygon mesh terrain model we have attempted contains 900,000 vertices.

VPE implements a multi-resolution or pyramidal data structure for 3D terrain surface models. Also, terrain models are spatially sub-divided using a quadtree structure. These techniques are discussed in detail below.

†Mars is actually an oblate spheroid with an Equatorial radius 17.7 km. greater than its Polar radius.

2.5. Operating Characteristics

2.5.1. 6 Degree of Freedom 3D Environment Mode

VPE supports two modes for virtual exploration. The first is 6 DOF 3D environment mode. In this mode our graphics system renders perspective views of the 3D terrain model as the user moves a virtual platform and camera about the terrain. This mode generates a number of X windows on our workstation screens: one or two 3D scene rendering windows (for monocular or stereo views), a 2D map or plan view window, a push-button control input window, and a status window. A marker in the map window indicates camera orientation and field of view. The push-button control window can be used to input exact coordinate values. The status window provides text display of position and orientation in the VE plus various program control modes. A user can use VPE sitting in front of our workstation in non-VE mode, or can wear an HMD and operate in immersive VE mode with each 3D scene window providing the video source for stereo viewing. Although the 2D map view window can just barely be resolved with the LCD displays in the HMD, text windows are unreadable. Thus, immersive mode requires an assistant to the HMD wearer to help operate motion controls and set program modes that require the mouse and keyboard input.

Users have six degrees of freedom for motion control. The 3D platform position can be translated over the terrain using three alternate input methods: Spaceball x,y,z translation (i.e., using the Spaceball as a 3 axis throttle), mouse point and click in the 2D map view window, and mouse input in the push-button control window. Since VPE terrains can be hundreds of km in extent and our terrain models include planet curvature, platform motion conforming to great circle routes is also supported.† The virtual eye or camera orientation can be rolled, pitched, and yawed via four input methods: Polhemus tracker (i.e., HMD wearer's head orientation), Spaceball x,y,z rotation, mouse point and click in the 2D map window, and mouse click in the control window. In addition to the above motion controls, we support two non-interactive inputs: playback of scripted commands from a text file and interpolation ("in-betweening") of 6 DOF waypoints that can be interactively selected and edited in the 2D map view window or read from a text file. All motion control interactions can be optionally logged to a text file, which is also usable as a playback script input file. Interactive input motion controls can be operated simultaneously while non-interactive scripted playback or waypoint interpolation is occurring. For example, while "flying" a scripted path, a user can "look around" in the HMD using the Polhemus tracker to alter the camera direction controls in the script.

A variety of program options are available that may be interactively set or initialized either via command line options or X Windows resources. Some of these include perspective transformation field of view, polygon rendering mode, pseudo-coloring HSV values, gamma coefficient for displays, various terrain modeling parameters, window attributes, input device selection and modes, and many more.

Theoretically one can explore our Mars terrain data set to go anywhere on the whole planet in VPE's 6 DOF 3D environment mode. However, DTM and MDIM data files often require merge or extract operations to make them fit the user's selected region of exploration. Also, we have found that it is desirable to apply contrast enhancement to the MDIM texture map images. Due to the size of these data files, rather than doing these file operations at the start of a VPE exploration session, we have chosen to do them in pre-processing steps prior to execution. Thus, a virtual exploration session requires offline preparation, and somewhat restricts the freedom to arbitrarily explore any place on Mars. We plan to overcome this limitation in a future upgrade of our user interface and terrain database management system. Our standard demo exploration data set, the middle 1000 km by 800 km section of Valles Marineris, requires a 400 KByte DTM input file and a 3.9 MByte MDIM image file for texture mapping.‡

†Such motion is, thus, actually a rotation about the planet center rather than a translation.

‡We do not use full resolution data in our standard demo. The file sizes would be 4 times larger if we use the highest resolution data.

2.5.2. 2 Degree of Freedom 2D Panorama Mode

The large size of VPE's planetary terrain data sets limit the interactive performance in our 6 DOF 3D mode. Depending on the size of the terrain data selected for viewing, animation update rates can be as slow as several seconds per frame. A second VPE viewing mode provides a close to real-time frame updating rate with no limitation on the amount of scene complexity, but with only 2 degrees of freedom for motion control. This mode is used to present panoramic views of up to 360 degrees. Only yaw and pitch rotations of the HMD wearer's head are tracked and responded to in this mode.

Since viewpoint position changes are not measured or simulated in this mode, the user's view of a 3D planetary terrain can be equated to that of a static 2D image mapped onto a cylinder surrounding the HMD wearer. The user looks at the virtual world from inside the cylinder by rotating his or her head in pitch and yaw angles (we provide a swivel chair to minimize neck twisting). The image presented for viewing in this mode is a precomputed 2D image of the 3D terrain scene. Of course, due to the limited motion control and static image, perspective, occlusion, and distance to objects in the scene are all constant. But, since the panoramic image is precomputed, there is no need to perform great amounts of graphics computing in response to changes in the wearer's head orientation. The display computer merely shifts the 2D image left and right or up and down on the screen in response to the HMD yaw and pitch rotations. Head roll is not supported. By using specialized "virtual pixel map" hardware in our Stardent GS2000 computer, we can present and shift very large (e.g., 16 MegaPixels) 2D images in the graphics video memory at close to video refresh rates. This enables users to experience a strong sense of viewing a realistic terrain as if they were looking around at it from a stationary position. We call this VPE mode by various names: panorama mode, cylindrical frame buffer mode, or the "Mars National Park Scenic Overlook" mode.

To view a virtual panorama of Mars, we must generate panorama image scenes from our terrain data. This is done by rendering and saving to a disk file a panorama view of up to 360 degrees that was computed while running VPE in 6 DOF 3D mode. For full 360 degree panorama scenes, the display presentation wraps around so the HMD yaw angle can be unlimited (as long as you don't get completely wrapped up in video and tracker cables). We generate a panorama as a mosaic formed from several planar projections at various yaw angles such that the image is equivalent to that formed by projecting the 3D virtual world onto the surface of a faceted cylinder.† Considerable distortion is noticeable with this projection method if the number of facets is too small or if there are high contrast linear features in the scene foreground. However, we have found that with as few as eight facets typical panorama images are not noticeably distorted. The amount of time required for the panorama image's generation varies depending on the size and complexity of the terrain data set selected. For typical 360 degree panoramas that are 6,000 by 2,000 pixels, scene generation time is about 10 minutes on our Stardent GS2000 computer. Of course, there is no limit to the amount of detail, scene complexity, and graphics synthesis realism that one might choose to render into such images.‡

An advantage of the VPE panorama mode is that the 2D image need not be a synthetic, computer generated image. Alternatively, it can be a digitized panoramic photograph of a real terrain scene. Thus, we can use the Viking 2 Lander Mars panorama images in this mode. The HMD wearer's viewpoint then becomes the same as that of Viking's camera. As he or she turns and tilts the HMD, exquisitely detailed views of the rocks and the plains of Utopia Planitia are seen (as well as portions of the spacecraft in the foreground). On each Viking lander there are two cameras located about one meter apart. Stereo pair images from these cameras can be viewed in VPE, though the fixed camera location means that strong stereo fusion only occurs in the direction perpendicular to the two cameras' base line and diminishes as one turns to the side. Of course, synthetically generated stereo pair

†An alternate implementation which we plan with our SGI Skywriter is to use the images computed as a projection onto each facet as environment maps⁹ for the polygon panels of a faceted cylinder, rather than mosaicing them into one single rectangular panorama image for use with "virtual pixel map" display. This method could use a faceted sphere rather than a cylinder since there would no longer be a requirement for a single rectangular source image. Head roll could be supported, as could a limited amount of head translation to simulate translation in the virtual environment, though perspective would not change correctly as it should for foreground objects. Such a mapping is similar to that used for IMAX and OMNIMAX projection theaters.¹⁰

‡A summer student implemented ray traced 360 degree Mars terrain panoramas.

panorama images can be computed from 3D terrain models and viewed in this mode.

3. Methods for Modeling and Rendering Very Large Virtual Environments

As stated above, we have made several engineering trade-offs to meet VPE's project requirements for modeling and rendering very large terrain data sets. Below we present a description of our efforts in three areas: visibility analysis and culling, level of detail management, and use of 2D imagery in 3D environments.

Although our methods and their implementations are intended for application to very large terrain VE's, many of them apply to other VE databases as well. We note that below where appropriate.

3.1. A "Suffer the Consequences" Solution

Given the performance rates of 2,000 to 5,000 polygons/frame at 30 frames/sec. for 3D graphics workstations in the under \$300,000 price range,^{11,12} very large polygon databases, on the order of hundreds of thousands of polygons, result in extremely low frame update rates. Two solutions to this problem come immediately to mind: use models with fewer polygons or accept a lower frame rate. In VPE we have tried both out of necessity, but are satisfied with neither.

Reducing the number of polygons rendered in our VE was easily implemented. As described below in the section on level of detail management, VPE maintains several geometric models of the same terrain surface. Each model is a polygon mesh with a different spatial sampling resolution and, thus, a different number of polygons. We can interactively select which model to render. Thus, lower resolution models may be rendered with higher frame update rates. Unfortunately, the amount of information usually is too low to do much useful visualization.

Alternately, we have found using low frame update rates with high complexity terrains (i.e., the highest level of detail models) does permit worthwhile visualization. We call this a "look and wait" strategy. After an initial orientation period in low level of detail mode, a user in the HMD can follow the strategy of looking towards an area of interest, and then waiting while the graphics system renders the scene. Our subjective observations of many users who have tried this have shown, surprisingly, that many people can adapt to this strategy fairly quickly. This view of virtual space is sort of like a very slow strobe lit view of the world, however, and there is little hope of a realistic immersive impression in this mode. Also, many people easily get disoriented by the update lag, and some even quickly experience simulator motion sickness.

3.2. Visibility Analysis and Culling

There are better alternatives besides these two, however. In any visual simulation system with a very large scene database a high percentage of the time is spent viewing a small portion of the database. For example, portions of the scene behind the viewpoint or hidden behind a large hillside can't be seen and ought not be considered for rendering. A naive approach to rendering the scene by passing the complete model or display list through the hardware graphics pipeline results in much unnecessary processing. In very large VE systems such as VPE, situations where only 5% or 10% or less of the database are in view can occur frequently. Thus, 90% or 95% of the graphics pipeline operations would result in no visible output. For small databases and fast hardware, this inefficiency may not be a bottleneck. In large VE databases, no matter how fast the hardware, a bottleneck will always occur.

In the naive approach to rendering a VE update rates will be limited by the rate at which the whole scene model can be processed through the graphics pipeline stages for matrix transformation and for clipping. The obvious solution is to adhere to the principle,

Do not attempt to render data that is not visible in the current view.

In other words, cull as much as possible of the non-visible portions of the database to eliminate pipeline processing of individual polygons. This is an efficient solution provided the cost (i.e., processing time) of the culling analysis (which usually must be done in software on 3D graphics workstations) is less than the cost of the graphics computations eliminated from hardware pipeline processing. Fortunately, several computationally inexpensive methods exist.

3.2.1. View Volume Object Culling

The first application of database culling is avoiding processing portions of the scene that fall outside the viewing volume as defined by the clipping planes. This technique has a very high ratio of the reduction in rendering cost to culling analysis cost, and it should probably be the first technique applied for VE efficiency improvement.

3.2.1.1. Hierarchical Spatial Sub-Division

A first step in visibility culling for VE databases is to spatially sub-divide the data. Since most terrain databases, such as the Mars DTM elevation data, are arranged in a regular grid as a 2D array of vertices, the obvious sub-division scheme is to use inherent grid line boundaries within the data set. The smaller sub-divided regions are, the more precision to be gained from the culling methods. However, too much precision can generate excessive overhead due to additional storage for data structuring and additional processing for accessing sub-regions. So, sub-region sizes must be such that a balance is achieved. In VPE we limit sub-divisions to a minimum of 25 grid points (i.e., 32 triangles). The NPSNET system¹³ at the Naval Postgraduate School uses fixed size 64 point sub-divisions. The renderer used in the Mars Navigator project^{14,15} (which was not designed for interactive, real-time rendering) sub-divides Mars terrain into 257 by 257 regions.

Since any method applied for culling sub-regions can also be applied to collections of sub-regions, it is logical and cost effective to organize sub-regions into a hierarchy. In VPE, as in most other systems,¹³ we use a quadtree data structure^{16,17} though a bintree would be suitable as well.

In VPE quadtree leaf nodes (level N of the tree) contain terrain data at full resolution. The leaf nodes partition and completely cover the database. Both interior nodes and leaf nodes of our tree are populated with data, and they contain pointers to polygon mesh information for vertices, surface normals, vertex colors, and texture map (u,v) coordinates. Quadtree terrain nodes higher up the tree represent the same region of terrain as that of their four children, but reduced by a factor of two in resolution of its terrain grid. The root node (level 1 of the tree) represents the whole terrain, but its polygon mesh data is reduced in resolution by a factor of $1/2^{*(N-1)}$ that of the full resolution terrain. For our Mars databases N is usually a maximum of 6 or 7.

Hierarchical spatial sub-division can also be applied to other virtual world objects besides gridded terrains. Many complex 3D models of man-made objects have a component structure that can be used to obtain spatial sub-division boundaries. One example of this is an architectural database such as used in building walkthrough VE's. Buildings have inherent hierarchical structure of rooms and hallways, floors, and the building itself. Intermediate hierarchy clustering can be easily added. Scene sub-division can be extended outside the building to cityscapes structured upon street boundaries (city blocks) and larger man-made boundary lines. For an excellent description of research in the area of building walkthrough visualization the reader is referred to the works done by Prof. Sequin and his students at U. C. Berkeley^{18,19} and by Prof. Brooks and his students at the Univ. of North Carolina.²⁰

Spatial sub-division can also be applied to scenes composed of arbitrarily shaped 3D objects that may not contain natural sub-division boundaries. Although, in the worst case, some polygon splitting may need to be done, methods such as the binary splitting tree algorithm²¹ could be used to automatically subdivide arbitrary polygonal objects. It should be noted that although sub-division by structural or functional components using part hierarchies as is done in most 3D solid modeling and CSG systems might seem useful, such sub-division may not be an efficient form of spatial sub-division. The VPE project experienced this problem when we attempted to interactively visualize the Hubble Space Telescope using a hierarchical model based upon component structure.

3.2.1.2. Object Extents: Bounding Boxes and Bounding Spheres

Visibility culling eliminates a terrain region or a 3D object from graphics pipeline processing by determining if it is totally outside the clipping volume. Given regular sub-divisions of a terrain database, it is sufficient to process the region's bounding box to determine if it can be culled. For nearly rectangular terrain regions, a bounding box is very efficient. For arbitrarily shaped 3D polygonal objects, an alternate is a bounding sphere. To determine whether a terrain region can be culled, one

must clip its bounding box and see if any portion is visible within the viewing volume. Since the bounding box is tested against the clipping volume but not actually drawn, a function is needed to perform the test. Some systems, such as the Stardent GS2000, have such a graphics function in their 3D library, while other graphics systems do not. If no such function is available, the current viewing transformation matrix must be accessed and then used to clip the bounding box. However, even this overhead is well worth the effort.

Culling efficiency can be greatly increased when a hierarchical sub-division is used. Bounding box clipping can be recursively applied to nodes of the tree starting at the root and terminating the descent down each branch when a culled node is reached. For example, after only a single bounding box clip test at level 2, it's possible that one quarter of the database could be culled immediately.

In VPE we compute bounding boxes for each terrain sub-region and store them in the quadtree data structure. Bounding boxes are computed in model coordinates (i.e., World Coordinates) and are transformed by the current viewing transformation before clip checking. Our bounding boxes are not merely the bordering grid lines of our terrain sub-regions since there is a slight warping of grid points when mapping from the DTM Sinusoidal Equal Area coordinates to 3D planet surface coordinates. This can result in quite a bit of overlapping of bounding boxes and a slight loss of efficiency.

3.2.1.3. Database Paging

The most direct way to avoid processing non-visible portions of a database is to not include those portions in the database to begin with. However, since visibility is dynamic depending on the viewpoint, so is the decision as to what to include in the database. Thus, one can partition the database into an active portion and an inactive portion, where the active portion covers (at a minimum) the current viewing volume extent and a buffer zone around each side. This approach has been employed for a number of years in CIG systems, i.e., Computer Image Generator a.k.a. flight simulator systems.^{22,23} A typical method is to sub-divide the database into large regular regions. In VE systems using hierarchical sub-division, these regions correspond to nodes a few levels above the bottom of the hierarchy. These large regions of the database are stored in secondary storage. When a region is determined to be in part of the active portion of the scene, it is loaded into main system memory, and when a region is determined to no longer be in the active set, it is removed from memory. Thus, database memory can be managed and paged as in virtual memory systems. However, assuming no changes were made to the terrain, writing pages back to disk is unnecessary since database pages can't ever become "dirty". Files should be stored in a form ready to merge into the online terrain model so that model building pre-processing is not needed at run time. Due to the speed limitations for accessing disks systems, most such systems do anticipatory database loading. In commercial CIG systems, this processing is typically handled by a separate database management processor.

The same method is applicable with current high performance engineering workstations, esp. ones with multiple CPU processors. This has not been implemented in VPE, though it is planned for the future. The reader is referred to NPSNET¹³ at the Naval Postgraduate School for an example.

3.2.2. Occluded Object Culling

The second application of database culling is avoiding processing portions of the scene that are occluded from view by other nearer objects in the scene. One might naively assume that hardware z-buffering obviates the need for doing this. But again, in a very large VE system the numbers of polygons and the high payoff for such culling easily prove such an assumption invalid. With a modest amount of computation, much of which can be done statically prior to run time, it's possible that fifty or a hundred thousand polygons hidden behind a mountain or a building wall can be totally eliminated from matrix transformation, clipping, and z-buffer processing.

3.2.2.1. Hierarchical Spatial Sub-division

To accomplish culling based upon occlusion, we also must utilize a hierarchical spatial sub-division. With respect to terrain sub-division, see the discussion above in the view culling section. A different sub-division requirement must be met for culling based upon occlusion, however. It will be necessary to obtain opaque bounding polygons to use in occlusion analysis (see below).

3.2.2.2. Occlusion Analysis and Processing

Culling VE objects based upon occlusion by other objects in the current view is done similarly as in culling based upon view volume extent. However, rather than merely checking object bounding boxes against the clipping planes, one must check object bounding boxes against the opaque boundaries of the spatial sub-division. Although this is considerably more complex than view volume clip checking, a great deal of the computation is based upon static analysis of the environment geometry. It can be done ahead of actual run time and stored as part of the database. For an excellent discussion of this, the reader is referred to the work by Prof. Sequin et al at U.C. Berkeley.^{18,19} Of course, non-static parts of the database (moving objects) require run time occlusion analysis.

Although we have not implemented culling based upon occlusion for VPE terrains, the following describes how some extensions of the building walkthrough work might be applied for terrain occlusion analysis. Architectural visualization systems rely heavily upon the visibility clipping planes imposed by opaque boundaries such as building walls and floors. Terrain environments have only one such structure (excluding man made objects), the terrain surface itself, and it is not nearly as geometrically simple as building components. We believe that terrain surfaces can be abstracted into simpler geometric structures, similar to the walls of building, that will permit similar visibility analysis as for architectural structures.

In a building, room walls are used to divide it into cells and the walls serve as the visibility planes for clipping. Doors and windows are considered portals through which visibility extends from one cell to the next. Visibility analysis is based upon lines of sight from one cell to the next through a portal.^{18,19} One could impose cell structure, opaque visibility boundaries, and portals upon a terrain surface by relaxing the goodness of fit of geometric modeling used for the surface itself. For example, a simple block-like abstraction could be developed within each hierarchical sub-division the terrain surface with a bounding rectangle that lies below all points of the terrain surface and side skirt polygons that are added around each of the four sides. The "walls" and the "portals" of this abstraction would be defined by the abutment, and lack of abutment, of the side skirts at sub-division boundaries. This abstraction is a much less tight geometric representation than that available in architectural systems, so the expected amount of culling will probably be less than optimal. On the other hand, it seems likely that the analysis of cell to cell visibility through portals could be considerably simpler for terrains than for buildings.

We anticipate great payoff from such visibility culling for our Mars terrain in the VPE system. Given the vast valley systems and towering volcanos of Mars where valley walls and mountain slopes can occlude huge regions of the database, we expect to find significant potential for rendering efficiency improvement.

3.3. Level Of Detail (LOD) Management

A second principle of efficient VE rendering is,

Don't model and render objects at a resolution greater than necessary.

For example, rendering a 20,000 polygon object model into a few screen pixels ought to be avoided!

Below we present methods to avoid wasteful resolution by using level of detail (LOD) management. LOD management is a method that stores terrain and other scene object models in multiple representations, with varying amounts of detail as measured in the number of polygons. The model selected for display at a given time should balance the rendering time with the amount of detail desired or needed. As viewpoint and other conditions change, an object's level of detail can be changed so that resolution is only as great as needed and rendering speed can be maintained at the highest possible rate. The principle of LOD management has been used for many years in CIG simulator systems.^{22,24}

We consider three domains for LOD management, each of which manage level of detail based upon a different factor or parameter. The domains and factors are:

1. Object Space: the user's level of interest based upon application requirements
2. Viewing Coordinate Space: distance from the viewpoint

3. Screen Coordinate Space: distance from the center of visual field

In VPE LOD is designated by an integer number, ranging from 1 for the lowest LOD, to N for the highest. Our LOD levels, thus, correspond to levels in our terrain quadtree. Leaf nodes of the quadtree contain terrain data at full resolution with LOD value N. Nodes higher up the tree correspond to lower LOD levels with one step higher up the tree representing a factor of two reduction in resolution of the terrain grid and a decrease by one in the LOD level. The root node has LOD level 1, and represents the whole terrain reduced in resolution by a factor of $1/2^{*(N-1)}$. For our Mars databases N is usually a maximum of 6 or 7. We display current LOD values in our status window and also show the equivalent resolution per terrain grid point spacing in metric units.

3.3.1. Object Space

The visual complexity or degree of shape variation in an object is not necessarily proportional to a user's need to see detail in the object. For example, one planetary geologist may have a very high interest in detail on the escarpment of Mars' Olympus Mons volcano, but could care less about the detail of its caldera, while another geologist may have opposite interests. Thus, even though both features have complex surface shapes, for each geologist, our VE system would be doing wasted work if it rendered both the escarpment and the caldera at the same level of detail.

In VPE we have experimented with allowing a user to specify regions of higher than normal and lower than normal interest based upon the application needs. We permit a user to point with the mouse inside our map view window to select higher and lower regions of interest. The result is that selected terrain quadtree nodes are flagged for rendering at LOD's increased or decreased by one relative to what their LOD would be due to other controls. Our implementation is only in the preliminary stages of testing, and we do not have an elegant user interface nor extensive options for controlling how this LOD management is varied. But, even this simple technique has shown itself to be effective, and we plan to extend it in the future.

3.3.2. Viewing Coordinate Space

The most widely used method for LOD control is based upon distance of scene objects from the viewpoint. This is a primary technique in CIG systems,^{22,24} for terrain surfaces as well as for man made, or cultural, objects within a scene. The current generation Evans & Sutherland ESIG-4000 system, for example, supports up to 12 levels of detail for terrain resolution.²⁵ Workstation based terrain renderers have applied the same principle.^{13,15} Typically, the decrease in LOD is based upon a function of object size as measured by the number of pixels it projects onto screen space. For gridded terrain surfaces the grid resolution can be reduced as a function of its distance from the viewpoint. This is equivalent to computing LOD level as a function proportional to $1/\text{distance}$ from the viewpoint since the perspective transformation used in most 3D graphics systems scales object sizes by that factor.

In VPE we have applied this to quadtree node selection using a function of the form:

LOD level = (int) ((scaling coefficient / distance from viewpoint) + offset).

Alternately, one could use a small lookup table that pairs the distance from viewpoint with each LOD level. In our quadtree terrain model, the LOD value is used to select the node with the appropriate resolution for each region of terrain. The scaling and offset coefficients can be adjusted for performance tuning (see below).

3.3.3. Screen Space

A third method for LOD control we have tested is based upon the assumption that the desire for scene detail is dependent on distance from the center of field of view. That is, objects viewed in the center of the HMD screen (or foveal region of the eye) must be shown at high resolution, whereas objects closer to the periphery need less resolution. This matches with the actual retinal resolution of the human visual system.

Some high performance flight simulator systems that use projection screens (rather than CRT displays) support a high resolution small inset display that is projected onto the display screen surrounded by a lower resolution background. The positioning of the inset view is driven by an eye

tracking system so that the high res inset appears exactly where the observer is looking. The inset region is modeled at the same resolution as the background, but it is displayed with a higher screen pixel resolution.

We make the assumption that when using head mounted displays the location where a terrain region appears on the HMD screen indicates the HMD wearer's interest in that region. That is, the user will always look in the direction towards the terrain region of highest interest, rather than turning the head one direction while turning the eyes in another. If the user were to become more interested in something in the periphery, he or she will turn the HMD in that direction so that what was peripheral becomes centered in the field of view. Thus, in VPE we have used the function:

$$\text{LOD level} = (\text{int}) ((\text{scaling coefficient} / \text{distance from center}) + \text{offset}).$$

The LOD value is then used, as in viewing distance LOD control, to select the quadtree node mesh model to use for each terrain region within the field of view.

3.3.4. Composition of LOD Control Functions

One of the main advantages of LOD management is that it can be used as a "knob" to turn for system performance tuning. For example, if frame update rate is chosen as the most important performance characteristic of a VE system, then LOD control parameters can be tuned to lower the number of polygons sufficiently that the desired update rate can be sustained. If, on the other hand, for a different application scene detail is deemed more important and lower update rates are acceptable, then LOD control parameters can be tuned to increase the number of polygons. Interactive tuning to allow LOD criteria to be modified during a visualization session would also be desirable. As an example of LOD control, one might set LOD values for terrain resolution by using a control function based upon distance from the viewpoint as exhibited above. To tune the performance, one would adjust the scale and offset "knobs".

Another important performance characteristic for visual scene simulators is graceful degradation. This simply means that as system workload increases, such as an increase in polygon count when the viewable scene content becomes more complex, the system's decline in performance should be gradual rather than catastrophic. Using LOD management it is possible to permit the system itself to automatically apply graceful degradation. In the tuning example above, automatic adjustment of the scale or offset coefficients could be done using a function dependent upon polygon count.

In the VPE system we are extending LOD control to include several different criteria. Thus, for performance tuning and for graceful degradation control it will be possible for users to choose among several "knobs" rather than just those that control LOD based upon distance from viewpoint. We have proposed, but not yet implemented, the use of a composite tuning function that would give users somewhat higher level control of LOD. Such control would remove users from some of the more technical details, and also give them control that is closer to application related requirements. Our concept is to simply define a multi-component LOD control function that is a weighted sum of the LOD control functions for the three categories discussed above. We plan to test this first with a simple linear function of the three, though non-linear combinations may merit investigation. With such a control function users could, for example, "turn up the control knob for field of view center weighted LOD fall-off" or "turn down the control knob for distance attenuated LOD fall-off". Similarly, users could specify preference weightings for a composite function used in graceful degradation control.

3.4. Using 2D Imagery

A third principle of efficient VE rendering is,

A picture is worth a thousand polygons.

It sometimes seems that many Virtual Environment systems are overly enamored with the need for 3D graphics modeling and rendering. In practice simple 2D imagery may not only be less costly and more efficient, but also may better meet the user requirements. We discuss two ways in which we use 2D imagery in the VPE system.

3.4.1. Texture Mapping

Because of the very large size of the terrain databases used in the VPE system and the scientific requirements of our application, we feel that geometric modeling and rendering at the highest resolution are usually necessary. Unfortunately the 3D polygon rendering performance of current systems is often insufficient for our application. However, some such systems can come much closer to satisfying our performance requirements when texture mapping is used. The reason this is true is that hardware texture mapping is relatively fast and cost effective compared with 3D polygon rendering. Furthermore, the increased visual complexity presented in the 2D texture imagery permits acceptable visualization with considerably lower level of detail modeling of the polygon mesh onto which the texture is mapped. In VPE we use photorealistic texture map imagery from the Mars MDIM's described above. We have compared terrain models rendered as Gouraud shaded polygon meshes at a given LOD with the same terrain rendered with texture mapping at LOD's as much as 2 or 3 levels lower, i. e. using 1/16 or 1/64 the number of polygons. Although some geometric distortions are obvious, particularly along silhouette edges (ridge lines), we generally consider the lower LOD geometry with texture mapping far superior to higher LOD geometry without. Planetary geologists who visit our lab show a very high preference for texture mapped terrains over non-texture mapped ones.

Thus, our goal and recommendation is to operate in texture mapping mode whenever possible. In VPE we have not always employed texture mapping in the past due to computer system performance problems, and due to the fact that we only recently obtained MDIM texture images for all of Mars. With our new 3D graphics computer system, the complete MDIM data set, and the anticipation of much higher resolution and better calibrated imagery from future space missions, in the future we expect to operate nearly 100% in texture mapped mode. There will continue to be a need in VPE for using shaded polygon modes with terrain data for which there is no texture data. Also, some terrain analysis can be done better with synthetic shading for enhancing features that may show up better without the surface clutter of the textured image.

3.4.2. Mixing 2D Environment Maps with 3D Rendering

Although we live in a three dimensional universe, the views of what we see around us are often limited to a 2D projection of space and the objects in it. To perceive three dimensionality our visual system relies upon such cues as binocular disparity, motion parallax, occlusion and changes in occlusion, and perspective change with distance. If these cues are absent we usually cannot perceive any significant difference between a 3D scene and a 2D planar projection of that scene. For example, a landscape that is distant from our viewpoint lacks binocular disparity depth cues and a view into a room through a doorway several meters away severely restricts our ability to use motion parallax. For both of these views, it is likely that a 2D image of the scene will be perceived nearly the same as the 3D scene.

Thus, it appears to us that virtual environment visualization has a great potential for higher performance and enhanced realism by integrating 2D images into VE scenes rendered from 3D models. Higher performance would be achieved since relatively expensive 3D rendering of large numbers of polygons could be replaced by relatively inexpensive texture mapping onto a few large polygons. Enhanced realism would be achieved since exquisite detail from digitized photography or from very realistic, expensive 3D rendering (performed offline) would replace the realism obtainable from limited 3D models that can be rendered at near real time animation rates.

We have proposed and plan to investigate extensive use of 2D images in our 3D planetary terrain virtual environments. These images will be used as texture maps, but they will not be mapped onto the surface of complexly shaped 3D object models composed of many, many polygons. Instead we propose to map them onto a few, large polygons as if they were images on a painter's canvas, a billboard, or an outdoor movie theater screen. This is similar to environment mapping^{9,26} except that we don't use imagery as reflection of the environment surrounding an object, but as a view of the environment mapped onto a window pane or a geodesic dome within the environment. This is very similar to environment mapping as used for Omnimax film projection.¹⁰

Many such images of environment views could be employed at different places within a VE. Of course, this strategy will also be performance limited due to the amount of memory required for

multiple, large environment map images. So, as the observer moves around through virtual space, we propose to offer viewing spots somewhat like scenic overlooks in a national park. As one moves throughout the 3D environment model, icons or kiosks will be discovered that indicate the availability of high resolution 2D images of that part of the environment. Although high performance will be required for loading images from secondary storage, anticipatory database paging can be used during the time a user is traversing through the virtual environment to the next scenic overlook.

We have demonstrated the viability of this technique with VPE's 2D panorama viewing mode ("Mars National Park Scenic Overlook" mode). The view of Utopia Planitia with its thousands of rocks, from pebble size to boulders, with clear evidence of crystal and pore structure in nearby rocks, and even grains of sand, will never be possible with real time 3D rendering systems. With 2D imagery integrated into a 3D virtual environment model as we have demonstrated, truly realistic virtual reality may someday become reality.

4. Acknowledgements

Dr. Michael W. McGreevy is acknowledged for his foresight, perception, enthusiasm, perseverance, budget manoeuvring, and ability to dream. Without his inspiration and hard work this project would never have succeeded nor even existed. David Koblas is acknowledged for his whiz bang programming skills. He wrote most of the C code for implementing the VPE virtual environment. Current and former VPE staff members, Amy Wu, Romy Bauer, and Bob Brown, are acknowledged for software development and moral support. All the folks at NASA Ames, virtual and real, who have worked over the years to make all this fun virtual environment stuff successful enough that NASA is willing to pay some of us to work on it full time, are acknowledged. And, the final acknowledgement is to Ivan Sutherland -- who started it all.

References

1. M. W. McGreevy, "Virtual Reality and Planetary Exploration," *Online*, vol. 13, no. 8, pp. 3-8, Computer Systems and Research Division, NASA Ames Research Center, Moffett Field, CA, August 1991.
2. M. W. McGreevy and C. R. Stoker, "The Presence of Field Geologists in Mars-like Terrain," , submitted for publication 1991.
3. M. W. McGreevy, "NASA Ames Virtual Environment Display: Applications and Requirements," internal technical document, Aerospace Human Factors Division, NASA Ames Research Center, Moffett Field, CA, 1984.
4. S. S. Fisher, M. McGreevy, J. Humphries, and W. Robinette, "Virtual Environment Display System," *ACM 1986 Workshop on 3D Interactive Graphics*, pp. 23-24, Chapel Hill, N.C., October 1986.
5. S. S. Fisher, E. M. Wenzel, C. Coler, and M. McGreevy, "Virtual Interface Environment Workstations," *Proceedings of the Human Factors Society 32nd Annual Meeting*, Anaheim, CA, October 24-28, 1988.
6. M. W. McGreevy, "The Virtual Environment Display System," *Proceedings of the 1st Technology 2000 Conference*, vol. 1, pp. 3-9, National Aeronautics and Space Administration, 1990.
7. M. W. McGreevy, "Virtual Reality and Planetary Exploration," in *Virtual Reality Applications: Softwhere*, ed. A. Wexelblat, Academic Press, Cambridge, MA, 1992.
8. "The National Space Science Data Center," NSSDC 88-26, NASA Goddard Space Flight Center, Greenbelt, MD 20771, January 1989.
9. N. Greene, "Environment Mapping and Other Applications of World Projections," *IEEE Computer Graphics and Applications*, vol. 6, no. 11, pp. 21-29, November 1986. An earlier version appeared in *Proceedings of Graphics Interface 86*, May 1986, pp. 108-114.
10. N. Max, "Computer Graphics Distortion for IMAX and OMNIMAX Projection," *Proceedings Nicograph 83*, pp. 137-159, December 1983.
11. Michael J. Zyda, Robert B. McGhee, Ron S. Ross, Douglas B. Smith, and Dale G. Streyle, "Flight Simulators for Under \$100,000," *IEEE Computer Graphics & Applications*, vol. 8, no. 1, pp. 19-27, Jan. 1988.
12. Michael J. Zyda, Mark A. Fitchen, and David H. Jennings, "Graphics Workstations and 3D Visual Simulation: Some Performance Expectations and Measurements," NPS52-89-046, Dept. of Computer Science, Naval Postgraduate School, Monterey, CA, July 1989.
13. M. J. Zyda, D. R. Pratt, J. G. Monahan, and K. P. Wilson, "NPSNET: Constructing a 3D Virtual World," *1992 Symposium on Interactive 3D Graphics, special issue of Computer Graphics*, pp. 147-156, ACM SIGGRAPH, Cambridge, MA, March, 1992.
14. P. Hughes, "Mars Navigator," Masters Thesis, University of California, Santa Cruz, Santa Cruz, CA, March 1991.
15. Peter Hughes, "Building a Terrain Renderer," *Computers in Physics*, pp. 434-437, July/August 1991.
16. H. Samet, *Applications of Spatial Data Structures*, Addison Wesley, 1990.
17. H. Samet, "Quadtree Tutorial Transparencies," *SIGGRAPH '85 Tutorial Course Notes*, vol. 25, Quadtrees, Octrees and Related Hierarchical Data Structures, July 1985.
18. S. J. Teller and C. H. Sequin, "Visibility Processing For Interactive Walkthroughs," *Computer Graphics*, vol. 25, no. 4, pp. 461-69, July 1991. *Proceedings of SIGGRAPH '91*, held in Las Vegas, NV, 28 July-2 August, 1991
19. T. A. Funkhouser, C. H. Sequin, and S. J. Teller, "Management of Large Amounts of Data in Interactive Building Walkthroughs," *1992 Symposium on Interactive 3D Graphics, special issue of Computer Graphics*, pp. 11-20, ACM SIGGRAPH, Cambridge, MA, March, 1992.

20. J. M. Airey, J. H. Rohlfs, and F. P. Brooks, Jr., "Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments," *1990 Symposium on Interactive 3D Graphics, special issue of Computer Graphics*, vol. 24, no. 2, pp. 41-50, ACM SIGGRAPH, Snowbird, Utah, March, 1990.
21. H. Fuchs, Z. M. Kedem, and B. F. Naylor, "On Visible Surface Generation by A Priori Tree Structures," *Computer Graphics*, vol. 14, no. 3, pp. 124-133, July 1980. Proceedings of SIGGRAPH '80
22. B. J. Schachter, "Computer Image Generation for Flight Simulation," *IEEE Computer Graphics and Applications*, vol. 1, no. 4, pp. 29-68, October 1981.
23. Robert A. Schumacker, "A New Visual Systems Architecture," in *Proceedings of the 2nd Annual Interservice/Industry Training Equipment Conference and Exhibition*, pp. 94-101, National Security Industrial Association, November 18-20, 1980.
24. Evans & Sutherland Computer Corp., *Technical Overview ESIG-4000*, Salt Lake City, Utah, 1990.
25. M. Cosman, personal communication, Evans & Sutherland Computer Corp., 1990.
26. J. F. Blinn and M. Newell, "Texture and Reflection in Computer Generated Images," *Communications of the ACM*, vol. 19, no. 10, pp. 542-547, October 1976.