



## **High-Speed Real-Time 3D Graphics for Two Dimensional People**

**Alexander Derbes, Case Western Reserve University**

**Principal Investigators: Mike McCann  
Summer 1999**

**Keywords: Computer Graphics, Virtual Reality, 3D, QuadTree,  
VRML, data visualization, ROV**

### **ABSTRACT**

A quad tree is a tree-like data structure with nodes and branches. Using a dynamic quad tree to represent a 3D terrain mesh allows the software to vary the resolution of small sections of the terrain. By using deeper branching, the system can increase resolution where it is beneficial to the user and decreasing the resolution where higher resolution would not be beneficial to the user. This project examines the use of the VRML language and a quad tree data structure to provide interactive Virtual Reality high-resolution (10 Million+) polygon terrain visualization on mid range desktop computers. The paper concludes with an evaluation of the current and future suitability of using VRML to view high-resolution terrains such as the 20Meter grid of Monterey Canyon.

### **INTRODUCTION**

The complexity of the rendered objects and the speed of the computer limit what can be performed in real-time with computer graphics. For example, computers commonly available today are capable of rendering perhaps a million polygons per second. At a reasonable frame rate (10 frames

per second) this limits the complexity of all rendered objects to something on the order of 100,000 polygons<sup>1</sup>. For each additional polygon, the computer must perform additional calculations to determine where on the two dimensional screen to draw the polygon<sup>2</sup>.

MBARI is interested in providing high speed, interactive computer graphics, because using interactive 3D visualization, it is easier for scientists and others to assimilate data. It is advantageous both for science and educational outreach if MBARI were able to distribute scientific data of Monterey Canyon in a high-resolution interactive Virtual Reality world.

However, using standard computer graphics rendering techniques, over 10 Million polygons<sup>3</sup> are required to display the high resolution model of Monterey Canyon, and more are required to add models of scientific data, ROV's, Ships and other objects. One solution is to provide the rendering system with knowledge of the world, so that the rendering system can choose not to draw polygons that will not make a perceivable difference to the user.

## **MATERIAALS AND METHODS**

One method to reduce the number of polygons is the addition of intelligence to the rendering engine, so the rendering engine knows more specifically what polygons to draw. One way to do this is a quad tree<sup>4</sup>. A quad tree is a tree-like data structure with nodes and branches. A node is a leaf if it has no children. Branches connect a node to its child nodes. Each node represents a section of the terrain we call a tile. Each tile contains a section of

---

<sup>1</sup> As an interesting note, it has been estimated that to produce life like three-dimensional interactive scenes of most places in the real world requires around 100,000,000 polygons/s.

<sup>2</sup> Specifically, each time the viewpoint changes, the computer must re-calculate the position of the polygons vertexes on the computer monitor. This is known as geometry transformation. The process is usually done using matrix multiplication, the 1x4 coordinate marris is multiplied by the 4x4-transform matrix (the transform specifies rotation around all three axis as well as a scale transform on each axis). For more information about basic computer graphics the book: Computer Graphics Principals and Practices published by Addison Wesley.

<sup>3</sup> Although it is possible for some systems available today to render at this speed, it is not possible using web distributable technologies such as VRML97.

<sup>4</sup> W.J. Schroeder, J.A. Zarge, and W.E. Lorensen, "Decimation of Triangular Meshes," Proc. Siggraph92, Addison Wesley, Reading, Mass., 1992

terrain at a specific resolution. Each tile that is not a leaf has four children. A tile's children are created by dividing the parent node in the center of both the X-axis and Y-axis (the Z-axis is elevation), creating four children. If the number of points in the parent and the child are the same, then the child tiles will have double the linear resolution of the parent but cover one quarter the area.

The number of levels in a quad tree is dictated by the desired maximum resolution, the number of points per tile (determined by the speed of the renderer) and the size of the data set. In the case of the 20-meter grid of Monterey Canyon, to represent the entire data set using 64x64 tiles, 5 levels of detail are required, yielding a maximum resolution of 2048x2048 elevation points or about 8.2 Million polygons. This task is well beyond the capability of current mid-range computer graphics systems.

<b>System</b>	<b>Price</b>	<b>Polygons/s</b>	<b>Hardware Texture</b>	<b>Hardware Shading</b>
<b>Silicon Graphics O2</b>	\$3,500	1,000,000	Y	Y
<b>Silicon Graphics OYNX RM</b>	\$250,000	210,000,000	Y	Y
<b>3DLabs Oxygen</b>	\$7,500	3,300,000	Y	Y
<b>Sony Playstation</b>	\$200	360,000	Y	Y
<b>Sony Playstation II</b>	\$400	75,000,000	Y	Y
<b>Sega Dreamcast</b>	\$450	3,000,000	Y	Y
<b>TNT2 Card</b>	\$1500	6,000,000	Y	N

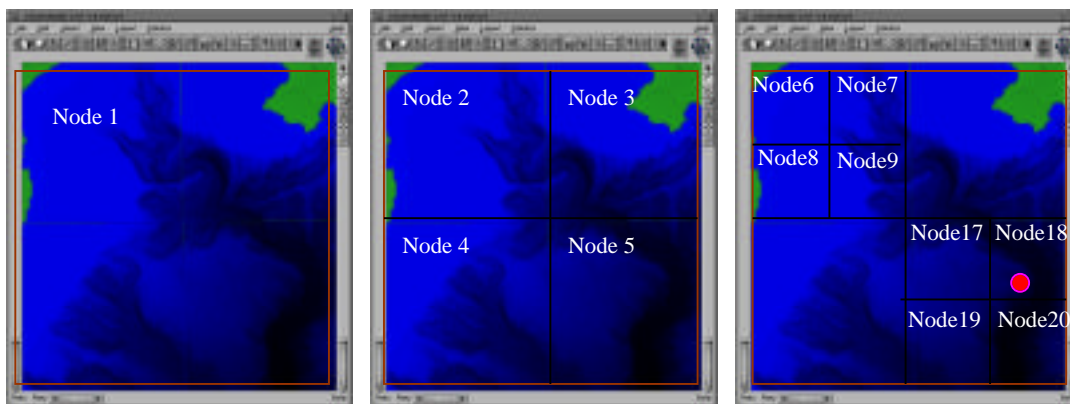
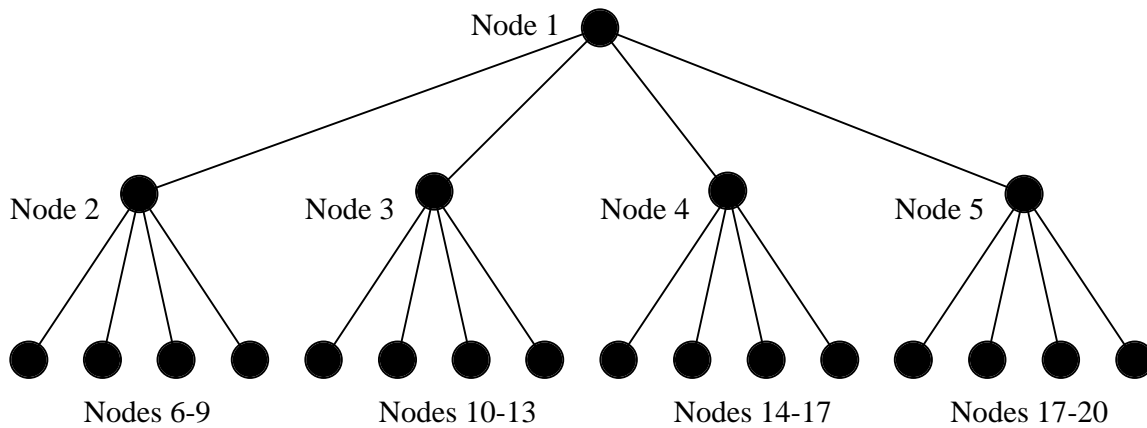
**Figure 1:** Graphics rendering power, options and price for an assortment of machines. Please note statistics reported by the marketing departments of graphics companies are almost never true.

The quad-tree data structure, allows the software to vary the resolution of small sections of the terrain, increasing resolution where it is beneficial and decreasing resolution where higher resolution would not be beneficial. The next step is to inform the rendering engine what how to choose resolution. The information needed is the user's current viewpoint, the ROV location on the current dive and the disparity of each terrain tile.

Level or detail pruning relies on the users current viewpoint to calculate which tiles to display in high-resolution. Level of detail pruning loads and unloads tiles based on how close the user's current viewpoint is to that tile. For example, if the user's viewpoint is close the top left of the terrain, the tile that covers the top left will be unloaded and replaced with its four higher

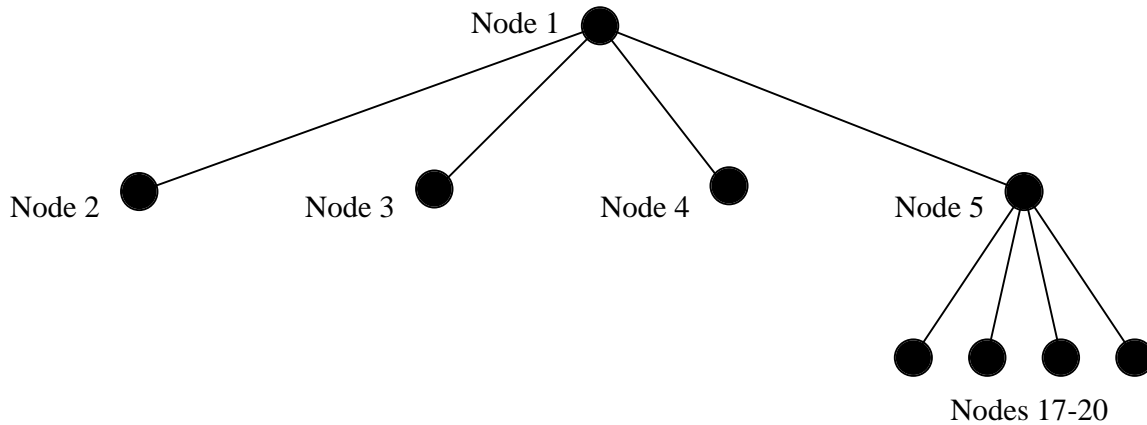
resolution children, effectively doubling the resolution of the area covered by the tile. If the user's viewpoint is even closer to the tile in the top left, then the child that the user is closest to they are will also be unloaded and replaced with its higher resolution children. This process is continued until the bottom of the tree is reached and the highest resolution is achieved. At the same time that the higher resolution tiles are loaded, tiles that are not visible because the viewpoint is too focused are not rendered ensuring that the number of polygons rendered at any given time remains reasonably constant.

The second method employed is area of interest pruning. Area of interest pruning tells the computer not



**Figure 1.1:** QuadTree diagram described as nodes and links between nodes with corresponding node locations in the terrain system.

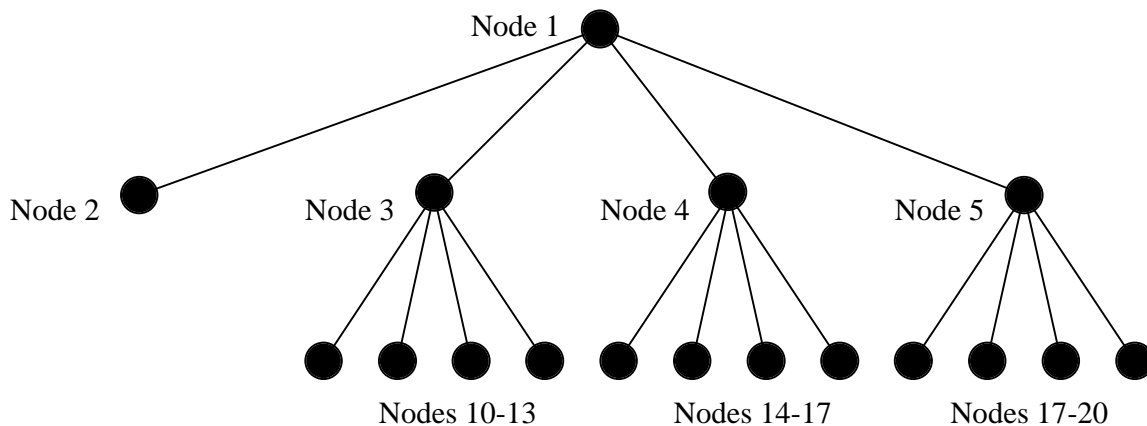
to load high resolution tiles for areas the ROV did not go near, for example, if the ROV was only near the top part of the canyon then high resolution data would only be loaded for the top portion of the canyon. Even if the user were to look directly at a location at the bottom part of the



**Figure 1.2:** In this example, assume the ROV location to be the red dot in tile 18. Using area of interest pruning, nodes 6-17 would be pruned leaving only high-resolution data for the area directly surrounding the ROV's location.

canyon, the computer would not load the high-resolution data.

The third method used is disparity pruning. Using disparity pruning, the computer will load high-resolution



**Figure 1.3:** In this example, the first quad tree has been pruned using disparity pruning, you can see that nodes 6,7,8 and 9 have been removed because that area of the canyon is flat, high resolution is not necessary to correctly model it.

data when the area will benefit from the increased resolution. For example, high-resolution data is not required to accurately describe a flat plane, but high-resolution data would be desired to describe a jagged surface.

## **Choices**

There were many design considerations for this project. Perhaps the decision that most effected the project was the desire to use for the display language rather than a competing technology. The decision to use VRML97 was made because at the inception of the project, VRML97 was the only language available with the ability to distribute 3D worlds over the Internet and have those 3D worlds viewable on a number of different computer platforms. However, there are a number of problems with VRML97, it is by no means mature technology. VRML97 is slow and the available VRML97 rendering engines do not display the highest quality output. None of the rendering engines support all VRML97 features and it is difficult to create a world that will play in different VRML97 browsers.

The second design decision was to use custom software written in the C programming language to perform tile division, resolution reduction and edge-merging. This decision was made because other languages considered, such as matlab, perl and python, were too slow<sup>5</sup>.

## **Data Structures**

There are three files for each node: tilename.wrl, tilename-tree.wrl and tilename.gif. The tilename.wrl is a VRML97 file that contains the DEM data for the tile. Tilename-tree.wrl is a VRML97 file with the decision logic for the node. Tilename.gif file is a GIF format image containing texture for the node. The tilename-tree.wrl file decides either to pass the decision on to its four

---

<sup>5</sup> The software written in the C programming language was more than ten times faster than the equivalent perl software.

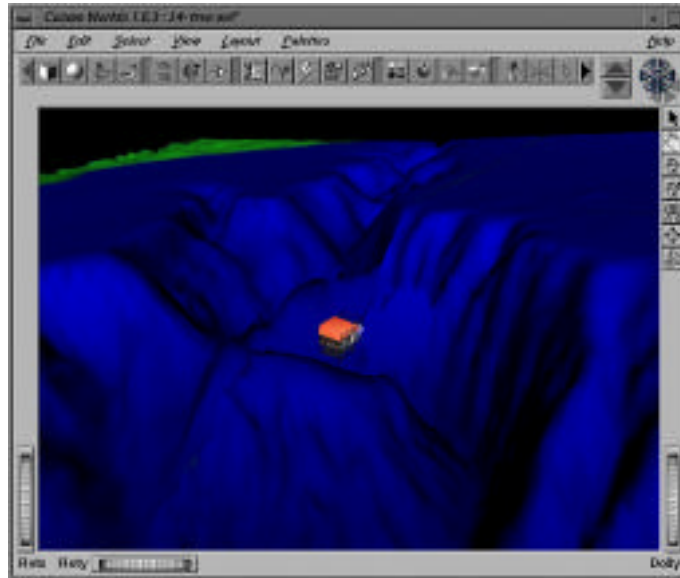
children or to pass its tilename.wrl file to the renderer based on the user's viewpoint location as reported by the renderer.

### **Programs and execution order:**

The software used to break up the terrain file into a quad tree consists of 10 small programs each with limited functionality. They are tied together with the use of pipes and temporary files. A list of programs and limited information about each is given in figure.

The initial input to the system is an ASCII digital elevation map (DEM) as exported by **ArcView**. Using the **asc2bin** tool, the system parses the ASCII DEM and creates out a custom binary data format (header followed by the DEM as a series of 32bit floating point numbers). The system then calls the **divide-all.pl** program which recursively calls the **divide** program generating the tiles in the quad tree. After the tiles have been generated, they are reduced to 64x64, using the **reduce-all.pl** program. The edges of the tiles must now be matched with their brothers and sisters using the **edge-fix-all.pl** program. Now the files can be converted to VRML97 using the **bin2vrml** program. At this point in execution, the system has created a large number of files, two per tile (the digital elevation map and the texture for the elevation map). The last step is to link these tiles together, creating the tilename-tree.wrl files that contain the level of detail pruning, area of interest pruning and disparity pruning. The product being a single VRML97 tree that will vary the resolution of the terrain based on the user's viewpoint.

## RESULTS



**Figure 2:** ROV Ventana Examining the Monterey Canyon resolution 256x256

The project developed software to produce high-resolution quad tree based terrain files in the VRML97 language. These files are available for download from the MBARI web site as part of the ROV data visualization project.

The solution implemented for this project allows the user to view the high-resolution Monterey Canyon terrain on a mid range graphics computer such as an SGI O2 (3,500\$). The method reduces the number of polygons the software needs to render from over ten million to a about 200,000 and further reduces the number of polygons that the graphics hardware needs to render at any given time from over ten million to around sixteen thousand, a significant reduction.

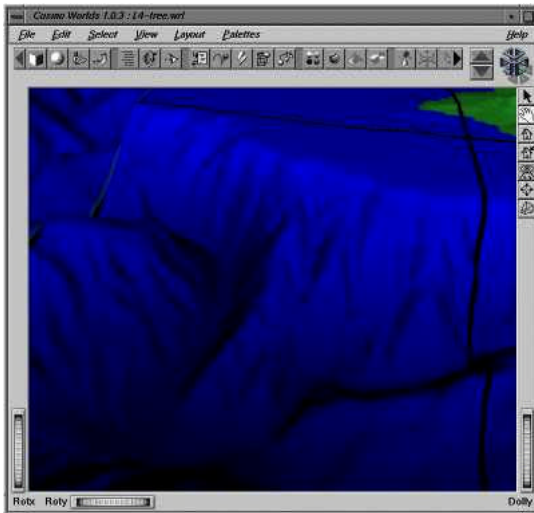


Figure 3.1 256x256 resolution

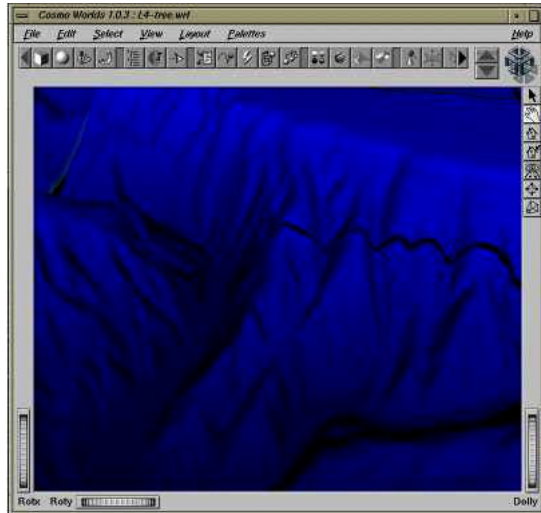


Figure 3.2 521x521 resolution

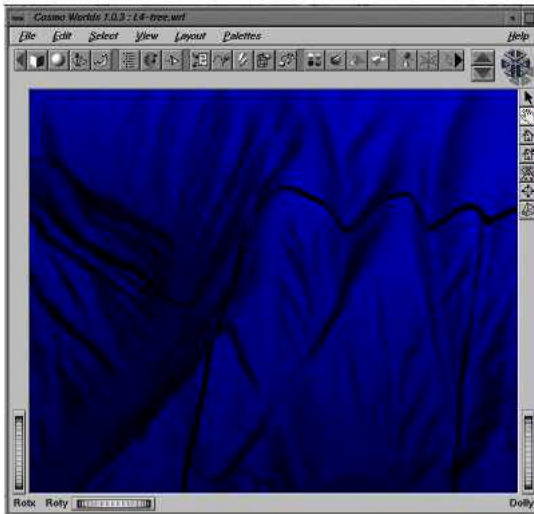


Figure 3.3 1024x1024 resolution

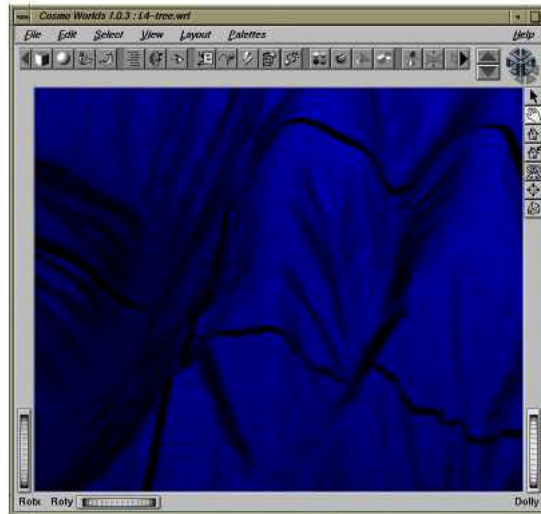


Figure 3.4 2048x2048 resolution

Figure 3.x: The same location in Monterey Canyon at different resolutions

Using these methods, a user can visualize data on a mid to low end computer that would otherwise be possible only on the most expensive high-end graphics computers. Likewise, a user can visualize data on a expensive high-end graphics computers that it would simply not be possible to visualize otherwise<sup>6</sup>.

The time it takes the rendering engine to load the world using area of interest and disparity pruning is  $1/6^{\text{th}}$

---

<sup>6</sup> An example of this is the TerraVisionII project at SRI International. The TerraVisionII visualizes a project loads multi terra-byte terrain databases with Billions of polygons. The Terravision project uses High-end SGI Reality Monster computers, custom databases and a custom rendering engine.

the time required to load the world without area of interest pruning (90 seconds vs. almost 10 minutes<sup>7</sup>).

Another measured advantage is the memory foot print of the application, using area of interest pruning and disparity pruning the memory footprint is reduced from 140 Megabytes of ram to a mere 40 Megabytes of ram (evaluated using Cosmo Worlds and the memusage ).

## DISCUSSION

There are a number of areas for improvement in the design of the current software. The software currently requires all tiles to be loaded into memory even if they will not be rendered. The time and memory requirements to load and display the world would be reduced significantly if the tilename-tree.wrl files also contained logic to load and unload files from memory. Another possible improvement for future direction is automatically generating the terrain files from a single terrain file when they are requested over the network. A third area of improvement would be utilizing an improved texture algorithm, the texture algorithm could be changed to use a better looking sun shading algorithm.

Also, if the technology were matured, area of interest and disparity pruning would not be necessary. Using this version of VRML it is not possible; at least it is not possible on multiple platforms to dynamically load and unload files from memory. If this were possible a dynamic world could be created in which wherever the user went, the highest It would be possible to have the best of both worlds if not for the limitations of the VRML programming language. The problem, is that short of developing a custom solution or shifting away from PC based distribution there are no other choices to be had. Numerous solutions exist for low polygon count graphics<sup>8</sup>, however, none of them except VRML offer any serious support for large polygon count worlds.

---

<sup>7</sup> Tests done using CosmoWorlds on an SGI O2 with 256 megs of ram and a MIPS R5000 CPU

<sup>8</sup> examples of low polygon count dynamic content distributions systems are: [www.blaxxsun.com](http://www.blaxxsun.com) , [www.hypercad.com](http://www.hypercad.com), [www.apple.com/quicktimeVR](http://www.apple.com/quicktimeVR)

## RECCOMENDATIONS

VRML97's immaturity caused a number of problems. The VRML97 language although almost 2 years old, still does not have a strong reference implementation nor advanced development tools. Because of this, we were forced to develop on a low level and encountered many problems with differing behavior between browsers.

The size of the terrain files made internet distribution, it is also difficult to create Virtual Reality worlds small enough to be easily distributed, thus It is the recommendation of this research that the choice of for distributed 3D graphics be re-evaluated.

These two problems may eventually be solved by the advent of faster computers and higher bandwidth networks. However, as computer speed increases so will the amount of data MBARI is collecting. This is happening now with the addition of the Oddessy AUV's to the MBARI fleet. One possible solution this and the issue of platform stability is the new generation of home video game systems such as the Sony PlayStation II and the Sega Dreamcast. These home game systems offer astonishing graphics performance up to 10 times that of current mid-range personal computers at a fraction of the cost (under 500\$). They also have the potential for easy data dissemination, with built in network connections and CD-ROM or DVD-ROM drives. Another options for the future work is the emerging X3D technology. X3D is in many ways, what for practical purposes stillborn VRML standard will become.

In closing, the conclusion of this research is the vision, to use 3D graphics and Virtual Reality technology to display science data in a meaningful way both to scientists and the general public, is correct. However, available technology is not quite up to the task. Many very smart people are working on the problem the technology is expanding faster now than ever before, "You have got to have faith".

APPENDIX #1

<b>Program Name</b>	<b>Programm ing Language</b>	<b>Input arguments</b>	<b>output</b>	<b>depends on</b>	<b>operation performed</b>
<b>asc2bin</b>	C	an arcview digital elevation map file	binary format file (STDOUT)	none	converts the input argument from a ascii arcview file to a binary custom format file
<b>resize</b>	C	a binary file	binary format file (STDOUT)	none	chops the input file to a square format by cropping the larger dimension
<b>print_header</b>	C	a binary file	text (STDOUT)	none	prints the header information for the binary file
<b>bin2vrml</b>	C	a binary file	text (STDOUT)	none	prints the VRML for the given DEM to STDOUT
<b>divide</b>	C	a binary file, and the base output name	the four children of the input file, saved as base_output_name_1.bin, base_output_name_2.bin etc.	none	divides the file into its four children
<b>filter</b>	C	a binary file	binary format file (STDOUT)	none	filters the file, removing non valid values
<b>reduce</b>	C	a binary file and the size to reduce the maximum dimension to	a binary file	none	reduces the input file to the dimension specified as the argument, maintaining proportion
<b>fix-edges</b>	C	a binary format file	a binary format file	none	fixes the edges of the file by matching them with the file's brothers
<b>convert_all.pl</b>	Perl	a directory name	a modified copy of each input file in the out/directory of the argument directory	bin2vrml	converts all .bin files in the input directory to vrml and saves them in the /out directory of the argument directory
<b>divide_all.pl</b>	Perl	a file name and the maximum level of detail	binary format files for each nodes DEM in the LOD	divide	starts with the root node and divides each file until the maximum level of detail is reached
<b>edges.pl</b>	Perl	a file name and the maximum level of detail	re-writes each file in the tree with mended edges	fix-edges	starts with the root node, fixing the edges on each file in the tree

<b>make_stitched_files.pl</b>	Perl	root .wrl file name, a maximum depth and the area of interest specified in eastings and northings	VRML files for the -tree.wrl nodes in the tree	none	writes the edge stitch aware -tree.wrl files for each node in the tree
<b>reduce_all.pl</b>	Perl	a directory name and a maximum dimension size	reduced files	reduce	reduces all the files in the tree to the specified dimension

## REFERENCES

A. Derbes, J. Ota., "Mars Pathfinder Robotics Visualization Applied to Sub-Marine Archaeology," *Proc. Underwater Intervention, New Orleans, LA. January 1999*

T.M. Rhyne, "Going Virtual With Geographic Information and Scientific Visualization," *Computers and Geosciences, Vol. 23, No. 4, 1997, pp. 489-491*

M. Reddy et al., "TerraVision II: Visualizing Massive Terrain Databases in VRML," *IEEE Computer Graphics and Applications, March/April 1999, pp. 30-38*

P.Lindstorm et al., "Real-Time Redinement of Progressive Meshes," *Proc. Siggraph 92, Addison Wesley Longman, Reading, Mass., 1997, pp. 189-198*

M. Reddy et al., "Modeling the Digital Earth in VRML," *AIC Technical Report No. 559. SRI International, Menlo Park, CA. 9 November 1998*

J. Washburn., "A Feasibility Study of VRML for Visualizing ROV Dives," *MBARI 1997 Intern Papers. MBARI, Moss Landing CA. August 1997*

N. Stepheson., Snow Crash. Spectra Books; ISBN 0552562614, Reissue edition May 1993.

