In *Computer & Graphics*, Vol. 17, No. 1, pp. 65-69.

NPSNET: Hierarchical Data Structures for Real-Time Three-Dimensional Visual Simulation

John S. Falby, Michael J. Zyda†, David R. Pratt and Randy L. Mackey
Naval Postgraduate School
Code CSZk, Dept. of Computer Science
Monterey, California 93943-5100
Email: zyda @ trouble.cs.nps.navy.mil

Abstract

3D visual simulation systems must present a world, including terrain, cultural features and 3D icons, in real-time at a level of detail which supports the use for which the system is intended. A “simple” world lends itself to blasting all the polygons through the workstation’s existing graphics pipeline. However, a “simple” world is not very realistic and/or does not operate in real-time. For complex worlds, such as that modeled in NPSNET, providing high fidelity in real-time requires the use of hierarchical data structures. We explore the implementation of such a structure on the world modeled by NPSNET utilizing quadtrees.

Introduction

Many factors determine the quality of a real-time 3D visual simulation. Two of these are the fidelity of the scenes rendered and the frame rate of the simulation, factors which are at odds with each other. High fidelity scenes provide a greater degree of realism, enhancing the effectiveness of the simulator as a training device or visualization tool, but slowing frame rate. High frame rates, in excess of 15 frames per second, provide smooth motion and improve interaction between the user and simulator, but allow less time to render individual frames.

The challenge is to add as many features that enhance realism as are required for the intended use of the simulator while keeping the frame rate above the threshold at which the human eye is able to detect individual frames. Specifically, we seek to expand the terrain area rendered in the simulation while reducing the polygon flow.

---

† Contact author

NPSNET: Overview

The researchers of the Graphics and Video Laboratory of the Department of Computer Science at the Naval Postgraduate School have focused for years on the production of prototype 3D visual simulation systems on commercially available graphics workstations, specifically, on the Silicon Graphics, Inc. IRIS workstation in all its incarnations (Personal IRIS, GT, GTX, VGX...) [Zyda 1990a] [Zyda and Pratt 1990e-f] [Zyda et. al. 1988, 1990b-d] [Nizolak et. al. 1990]. These visual simulation systems have supported communication between workstations using an Ethernet local area network. Our current visual simulation efforts are focused on the NPSNET system, a real-time, workstation-based, 3D visual simulation system capable of displaying vehicle movement over the ground or in the air utilizing SIMNET databases and networking formats [Thorpe 1987] [Zyda and Pratt 1991].

History of Real Time 3D Visual Simulators at NPS

Over the past six years the researchers at NPS have developed a series of real time visual simulators. We examine these focusing on how terrain paging, multiple resolution and polygon culling were accomplished.

The first effort was the Fiber Optically Guided Missile (FOGM) simulator [Smith and Streyle 1987]. It used a 35km x 35km terrain area with only a 10km x 10km active area in memory at any time. All vertices for all terrain polygons were stored in an array. Terrain paging and multiple resolution were not supported. Polygons outside a rectangular area encompassing the field of view were culled, but some polygons outside the field of view were still rendered.

The second simulator developed was the VEH Vehicle Simulator [Oliver and Stahl 1987]. It used the same terrain dataset and data structures as the FOGM and also did not support terrain paging or multiple resolutions. Although the polygon culling algorithm was an improvement, some polygons not within the field of view were still rendered, but less than in the FOGM.

The Moving Platform Simulator (MPS) series was next: MPS [Fichten and Jennings 1988], MPS II [Winn and Strong 1989] and MPS III [Cheeseman 1990]. Like their predecessors, the

MPS series did not support terrain paging, but again required the user to explicitly choose a new active area. All offered three levels of resolution, but the algorithm was too closely tied to the rectangular terrain structure and range of each resolution level is not independent of field of view direction. Polygon culling is accomplished by drawing a bounding box at each resolution which encompasses the field of view at that resolution level. However, some polygons outside the field of view are still rendered.

The Command and Control Workstation of the Future (CCWF) [Weeks and Phillips 1989] was developed during the same time frame as MPS II. CCWF does not provide terrain paging, but does provide three resolution levels. Polygon culling is accomplished by dividing the 360 degree panoramic view into 72 five degree sectors and displaying only as many as are required to cover the current field of view. At the default 45 degree field of view nine sectors would be displayed. Resolution boundaries are fixed within each sector.

The ongoing Underwater Autonomous Vehicle (AUV) Simulator project [Jurewicz 1990] [Brutzman 1992] does not provide terrain paging at this juncture but does provide multiple resolutions, allowing the user to select the number of resolutions and how far out terrain is rendered. The currently unused resolution terrain definition was retained in memory to avoid a hysteresis effect, chugging back and forth between high/low resolution. In addition, the simulator has a dynamic mode in which these resolution and depth of view parameters are automatically adjusted in an attempt to keep the frame rate and quality of terrain rendered within predefined limits. If the frame rate falls below a certain threshold, resolution boundaries are adjusted and more terrain is automatically rendered at lower resolution levels. Conversely, if the frame rate rises above a certain threshold, more terrain is automatically rendered at a higher resolution. Only polygons outside a bounding box encompassing the field of view are culled.

Expanding the Terrain Area

NPSNET uses data from the SIMNET Database Interchange Specification (SDIS) for an actual 50km x 50km terrain area of Fort Hunter-Liggett, California. The dataset has a resolution

of one data point for every 125 meters [Lang and Wever 1990]. In order to increase performance, the dataset has been divided into 2500 text files based on the one kilometer standard of the military “grid square” with each file containing data for one square kilometer. These were preprocessed into binary format and three additional lower resolutions generated (250, 500 and 1000 meter), together with fill polygons for each level. The final form of the dataset is 2500 binary files, each containing a multiple-resolution (4 level) description of the terrain for one square km, stored as a heap-sorted quadtree [Mackey 1991]. The final dataset contains 79 megabytes of data in binary form for the original polygon descriptions alone. When three additional lower resolution descriptions are generated, the size increases to 147 megabytes. As this is too much to store in main memory at one time, and we do not wish to limit the simulation to some smaller area (as in previous versions), paging terrain data through a dynamic algorithm is required.

A 16km x 16km active area was chosen based on considerations for memory size of available workstations, frame rates, required field of view and desired range of view. This amount of terrain data is in main memory at any given time and available for rendering. Sixteen kilometers allows a seven kilometer field of view in all directions for immediate rendering with one kilometer acting as a buffer to ensure terrain is fully paged in before attempting to render it (Figure 1). On multiprocessor workstations, the simulator does not wait for additional terrain to be paged in. Instead, the additional CPUs are used to page in the terrain in parallel.

Terrain Paging Algorithm

When the simulator is initialized, the user/vehicle is centered on a 16 x 16 active area. The indices of the center one kilometer square containing the user/vehicle become the notional center. Data is loaded into the appropriate elements of a 50 x 50 array, and a bounding box is established around the user/vehicle (centered on the index of the center square). When the user/vehicle reaches the bounding box in any direction, memory space is freed in the direction opposite of travel, terrain is paged in the direction of travel, and the bounding box moves. If the bounding box were 1000 x 1000 meters (equal to data division), edges of the new and old bounding boxes would

coincide. The user/vehicle could continually move back and forth across the boundary resulting in thrashing. Use of a 1200 x 1200 meter box requires the user/vehicle to change direction and travel a minimum of 200 meters before the terrain just paged in would be paged out (Figure 2). The size of the bounding box can be adjusted as required by vehicle speed/turn rate characteristics. Terrain paging is independent of the hierarchical data structure implemented.

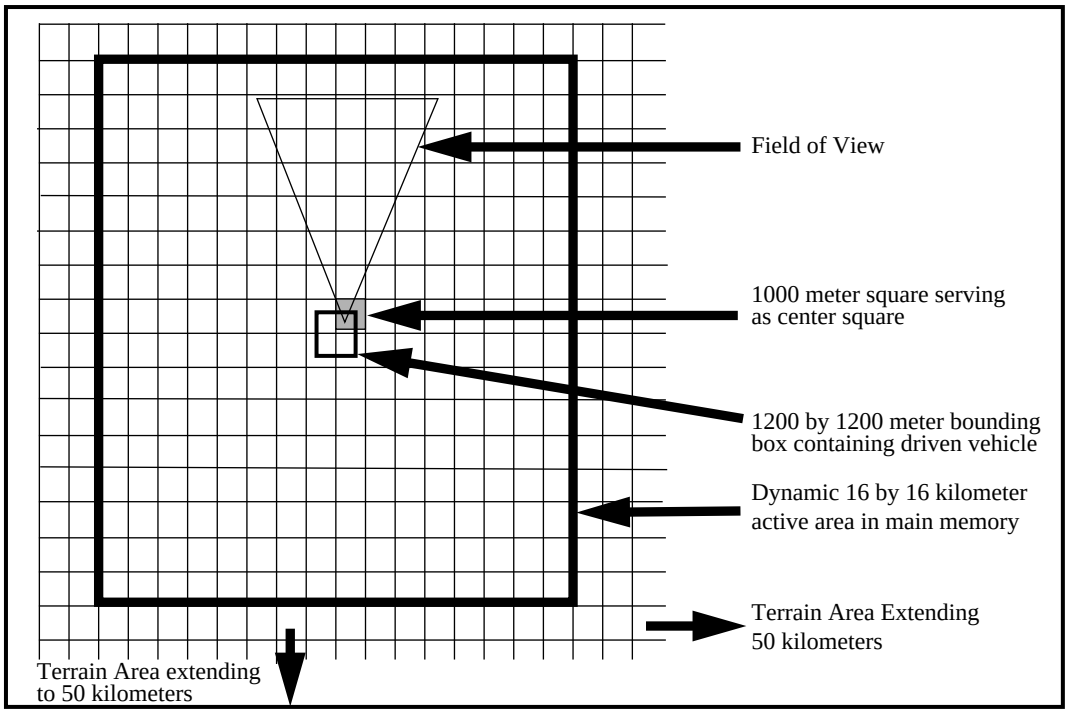

Figure 1. Active Area of Terrain in NPSNET

Implementation of Quadtrees

Polygon flow is reduced by implementation of a hierarchical data structure that supports rapid culling of polygons not within the field of view (whose 3D aspect is a pyramid) and multiple resolutions of terrain. NPSNET utilizes an array of quadtrees. Implementation starts with preprocessing the SDIS dataset to generate three additional versions, or levels, of the 125 meter data in successively lower resolutions. We do this by combining four higher-resolution area descriptions to form one lower-resolution area description. Descriptions of fill polygons for each level are generated and on ground features are added. Color is simply the majority color of the enclosed high-

est resolution polygons, except in the case of shorelines, where polygon corner elevations are compared and, if not within a certain tolerance of each other, the color of the triangle set to soil to avoid “sloping” water. (While the majority color of underlying polygons is water, the corners of the triangle formed for the lower resolution vary greatly, resulting in a “slope” of water.)

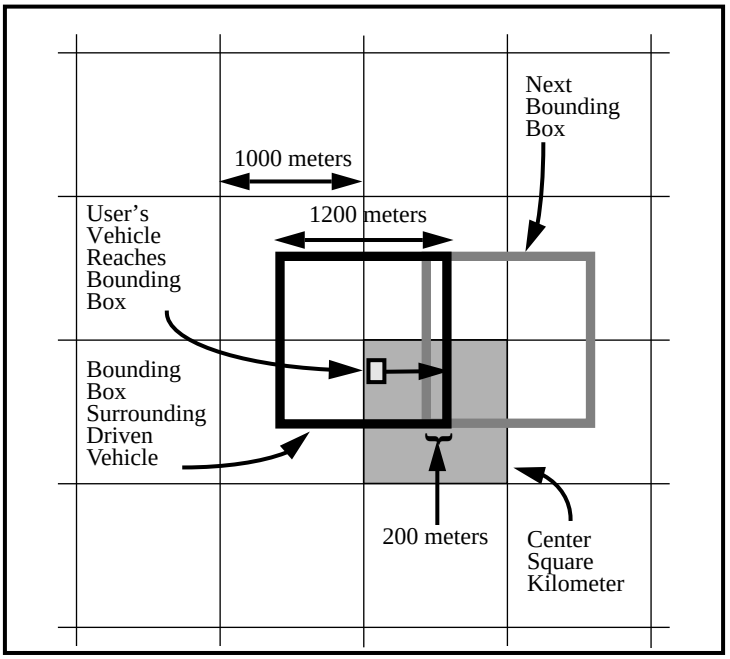

Figure 2. Boundaries for Terrain Paging

The final format for the binary terrain data files is designed for fast access using the C function *fread()*. All polygon descriptions are stored in memory-image format, therefore, no data conversion has to be done during paging. The 2500 files resulting from preprocessing contain:

1. Count of polygons in each node of full four level quadtree (85 total).
2. Total polygon descriptions in the file.
3. Multi-resolution description of terrain in this square kilometer stored in quadtree heap-sort order, lower resolutions first (Figure 3).

The terrain data structure is an array of quadtrees. The base array is 50 by 50 and corresponds to the area of the entire 2500 file dataset. Each element of the array contains all the data for one square kilometer; however, as previously discussed, only data for the 16 x 16 active area is actually in memory at any given time. An array element contains a structure consisting of two arrays. One is an array of 85 integers which holds the count of polygons at each quadtree node. The other

is an array holding pointers to the polygon descriptions for each node. Both arrays are stored in quadtree heap-sort order, lowest resolution first. The structure is depicted in Figure 4.

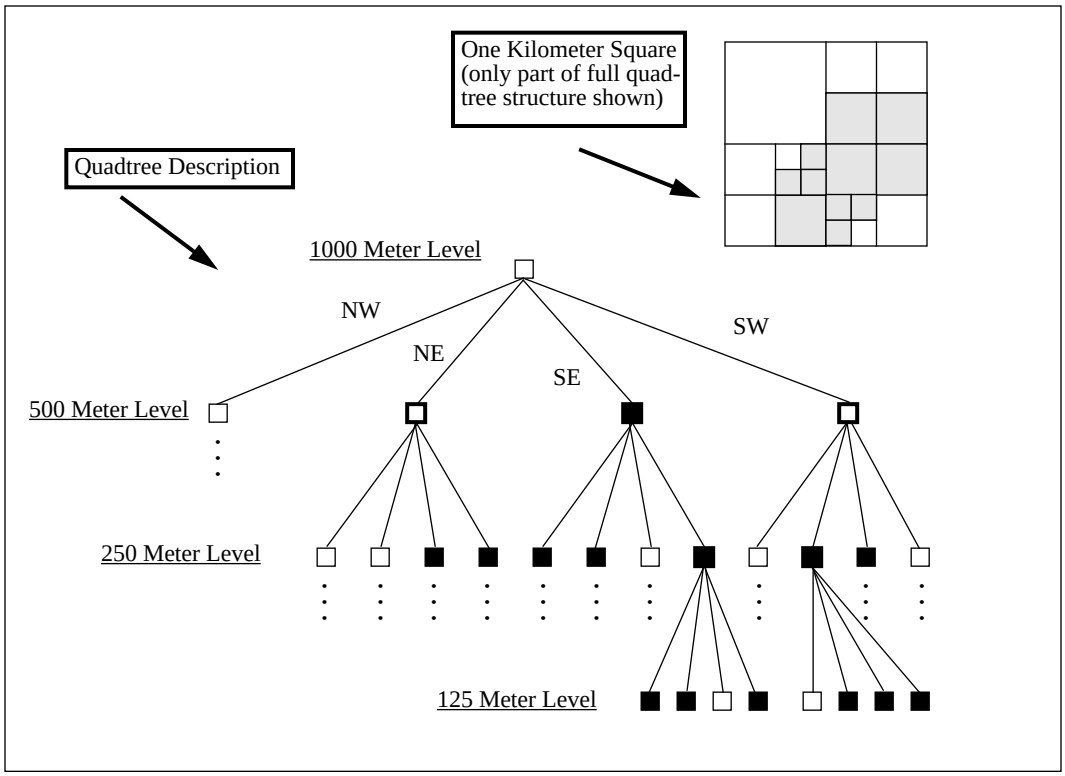

Figure 3 - Multiple Resolution Quadtree

Initially 256 files are read to fill the 16 x 16 active area. During terrain paging, files corresponding to a 16 x 1 km strip of terrain are consecutively block-read into the data structure (after the “opposite” strip is freed) in order to update the active area of the simulator (Figure 5).

Terrain Rendering

Terrain rendering involves several steps in NPSNET:

1. Determine which 1000m x 1000m squares are actually in the field of view.
2. Determine resolution within each 1000m x 1000m square (there may be at most two resolutions), including which fill polygons are needed.
3. Render the terrain.

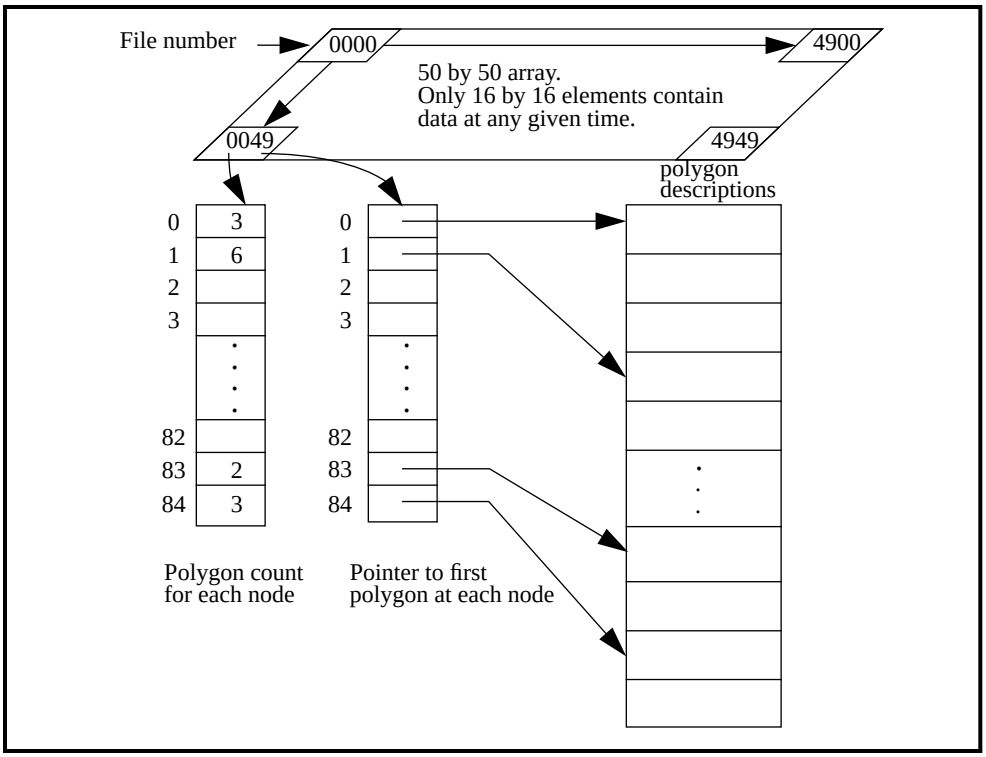

Figure 4 - Terrain Data Structure in NPSNET.

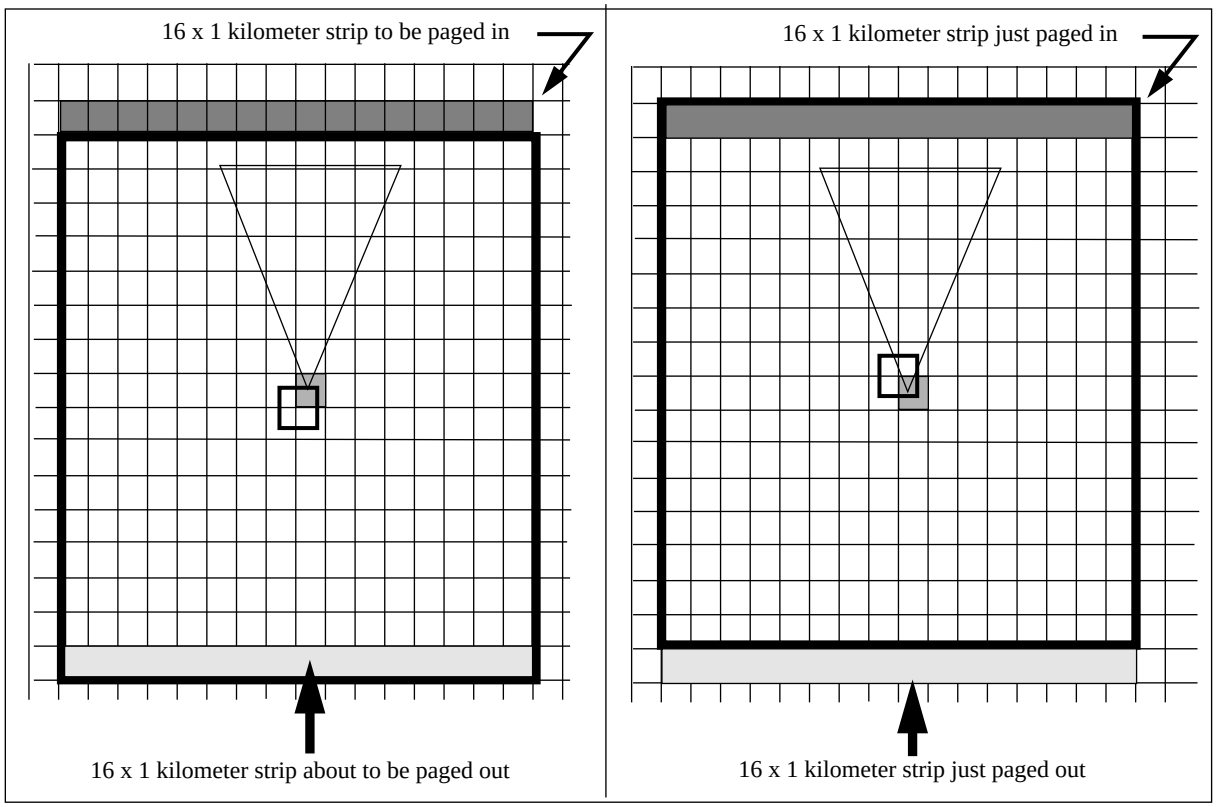

Figure 5 - Terrain Paging in NPSNET

Two algorithms are involved. One checks to see if a polygon is within the field of view by calling a procedure that checks for the intersection of a point (each point of the polygon) and a polygon (the triangle composing the field of view) [Fichten and Jennings 1988]. The other determines the resolution, essentially which nodes of the quadtree to render, by checking the intersection of nodes with concentric circles corresponding to ranges of the resolutions [Shaffer 1990]. The circle-rectangle and point-polygon intersection algorithms are applied repetitively to render only terrain within the field of view and at the appropriate resolution levels. Figure 6 depicts multi-resolution within the field of view. NPSNET is graphics bound. Therefore, the computational expense of the above algorithms outweighs the cost of rendering terrain not actually in the field of view.

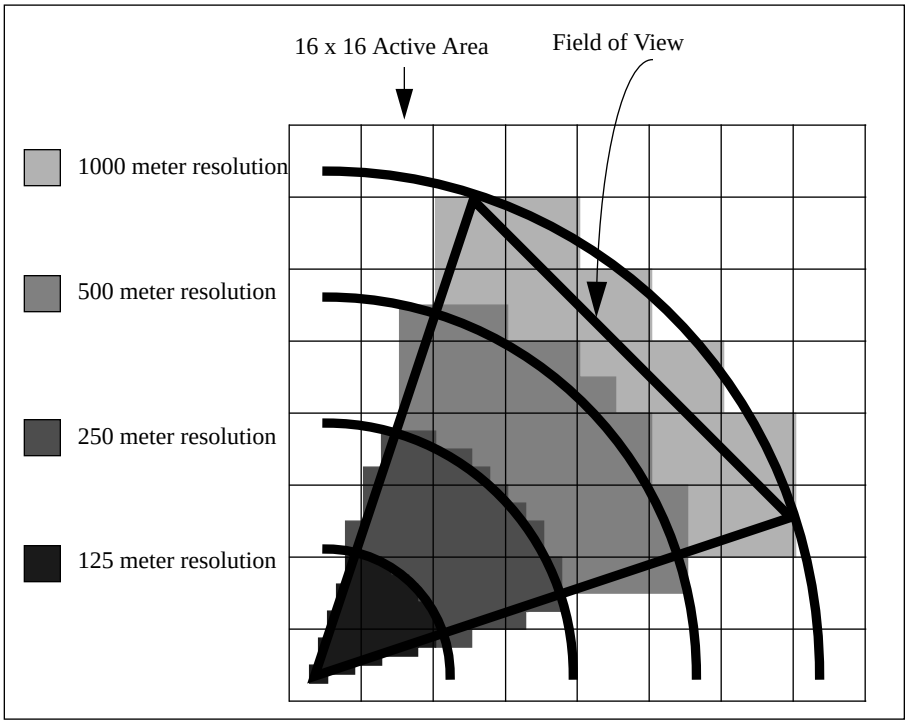

Figure 6 - Multi-Resolution Rendering

NPSNET: Current Performance

Implementation of the above has resulted in a doubling of the performance of the simulation over rendering at high resolution alone. Figure 7 summarizes the results of rendering the terrain

from the same vantage point and direction of view at various distances and resolutions with a few objects on the terrain. We realized a 30% increase in frame rate by rendering out to 1000 meters in high resolution and then to 2500 meters in Medium High resolution as compared to rendering just in one resolution. Rendering in two resolutions out to 6000 meters was 47.8% slower than rendering the same distance in four resolutions. However, performance when large numbers of objects (trees, vehicles) are present in the viewing area does not change. This is due, in part, to the increased number of polygons requiring rendering. Not only are there more polygons due to the increased number of objects, but the objects are rendered at a constant resolution regardless of distance from the user/vehicle.

Resolution Configuration	Distance (meters)	Frames per Second
High	0 - 2500	6.3
High Med High	0 - 1000 1000 - 2500	8.2
High Med High	0 - 3000 3000 - 6000	4.4
High Med High Med Low	0 - 2000 2000 - 4000 4000 - 6000	5.4
High Med High Med Low Low	0 - 1500 1500 - 3000 3000 - 4500 4500 - 6000	6.5

Figure 7 - Comparison of Frame Rates

Future Work

We need to integrate multiple resolution 3D icons into the same or similar data structure in order to realize greater performance improvements when large numbers of icons are in the viewing area. Multi-resolution rendering of mobile vehicles requires further research.

We also need to handle high-flying vehicles better by defining some lower resolutions and extending the active area to include an additional 19 km of terrain data (at the lowest resolution only) to increase range of view, based on altitude, to a maximum of 26 km, the generally accepted limit of view from an aircraft [Schachter 1983].

Acknowledgments

We wish to acknowledge the sponsors of our efforts, in particular Major David Neyland, USAF, DARPA/ASTO, George Lukes of the USA Engineer Topographic Laboratories, Michael Tedeschi of the USA Test and Experimentation Command, Carl Driskell of USA PMTRADE, Orlando, Florida, John Maynard and Duane Gomez of the Naval Ocean Systems Center, San Diego and Major Dennis Rochette, USA of the Headquarters Department of Army AI Center, Washington, D.C.

References

1. Brutzman, Donald P. (1992), *NPS AUV Integrated Simulator*, M.S. Thesis, Naval Postgraduate School, Monterey, California, MArch 1992.

2. Cheeseman, Curtis P. (1990), *Moving Platform Simulator III: An Enhanced High-Performance Real-Time Simulator with Multiple Resolution Display and Lighting*, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1990.

3. Fichten, Mark A. and Jennings, David H. (1988), *Meaningful Real-Time Graphics Workstation Performance Measurements*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1988.

4. Jurewicz, Thomas A. (1990), *A Real-Time Autonomous Underwater Vehicle Dynamic Simulator*, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1990.

5. Lang, Eric and Wever, Peters (1990), *SDIA Version 3.0 User's Guide*, BBN Systems and Technologies, Bellevue, Washington, August 1990.

6. Mackey, Randall L. (1991), *NPSNET: Hierarchical Data Structures for Real-Time Three-Dimensional Visual Simulation*, M.S. Thesis, Naval Postgraduate School, Monterey, California, September 1991.

7. Nizolak, Joseph P. Jr., Drummond, William T. Jr., and Zyda, Michael J. (1990), "FOST: Innovative Training for Tomorrow's Battlefield," Field Artillery, HQDA PB 6-90-1, February 1990, pp. 46-51.

8. Oliver, Michael R. and Stahl, David J. (1987), *Interactive, Networked*, Moving Platform Simulators, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1987.

9. Schachter, Bruce J. (1983), *Computer Image Generation*, John Wiley & Sons, New York, 1983, p 75.

10. Shaffer, Clifford (1990), "Fast Circle-Rectangle Intersection", *Graphics Gems*, Ed. Andrew Glassner, Academic Press, Boston, 1990, pp. 51-53.

11. Thorpe, Jack A. (1987), "The New Technology of Large Scale Simulator Networking: Implications for Mastering the Art of Warfighting," *Proceedings of the Ninth Interservice Industry Training Systems Conference*, November 1987.

12. Weeks, Gordon K. and Phillips, Charles E. Jr. (1989), *The Command and Control Workstation of the Future*, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1989.

13. Winn, Michael C. and Strong, Randolph P. (1989), *Moving Platform Simulator II: A Networked Real-Time Simulator with Intervisibility Displays*, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1989.

14. Zyda, Michael J., McGhee, Robert B., Ross, Ron S., Smith, Doug B., and Streyle, Dale G. (1988), "Flight Simulators for Under $100,000," *IEEE Computer Graphics & Applications*, Vol. 8, No. 1, January 1988, pp. 19-27.

15. Zyda, Michael J. (1990a), "3D Visual Simulation for Graphics Performance Characterization," *NCGA '90 Conference Proceedings*, Vol I, 22 March 1990, pp. 705-714.

16. Zyda, Michael J., Fichten, Mark A. and Jennings, David H. (1990b), "Meaningful Graphics Workstation Performance Measurements," *Computers & Graphics*, Vol. 14, No. 3, 1990, Great Britain: Pergamon Press, pp. 519-526.

17. Zyda, Michael J., McGhee, Robert B., Kwak, S., Nordman, D. B., Rogers, R. C. and Marco, D. (1990c), "3D Visualization of Mission Planning and Control for the NPS Autonomous Underwater Vehicle," *IEEE Journal of Oceanic Engineering*, Vol. 15, No. 3, July 1990, pp. 217-221.

18. Zyda, Michael J., McGhee, Robert B., McConkel, Corinne M., Nelson, Andrew H. and Ross, Ron S. (1990d), "A Real-Time, Three-Dimensional Moving Platform Visualization Tool," *Computers & Graphics*, Vol 14, No. 2, 1990, Great Britain: Pergamon Press, pp. 321-333.

19. Zyda, Michael J. and Pratt, David (1990e), "Zydaville," on ACM SIGGRAPH Video Review, Vol. 60, August 1990, entitled "HDTV & The Quest for Virtual Reality." The video segment shows our NPSNET system and a brief interview of Professor Zyda.

20. Zyda, Michael J., Pratt, David (1990f), "3D Visual Simulation as Workstation Exhaustion," *Proceedings of Ausgraph '90*, Melbourne, Australia, 10-14 September 1990, pp. 313-32

21. Zyda, Michael J. and Pratt, David (1991), "NPSNET: A 3D Visual Simulator for Virtual World Exploration and Experimentation," *1991 SID International Symposium Digest of Technical Papers*, May 1991, pp. 361-364.