

AFRL-RI-RS-TR-2008-206
In House Interim Technical Report
July 2008



OUT-OF-CORE DIGITAL TERRAIN ELEVATION DATA (DTED) VISUALIZATION

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Air Force Research Laboratory Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2008-206 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/
JULIE BRICHACEK, Chief
Agile Information Concepts
Information Systems Division

/s/
JAMES W. CUSACK, Chief
Information Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE*Form Approved*
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**1. REPORT DATE (DD-MM-YYYY)**
JULY 2008**2. REPORT TYPE**
In House Interim**3. DATES COVERED (From - To)**
May 06 – May 08**4. TITLE AND SUBTITLE**OUT-OF-CORE DIGITAL TERRAIN ELEVATION DATA (DTED)
VISUALIZATION**5a. CONTRACT NUMBER****5b. GRANT NUMBER****5c. PROGRAM ELEMENT NUMBER**
62702F**6. AUTHOR(S)**

Jason Moore (AFRL), Aaron McVay (CACI, Inc.)

5d. PROJECT NUMBER
558S**5e. TASK NUMBER**
AV**5f. WORK UNIT NUMBER**
IH**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**AFRL/RISF
525 Brooks Road
Rome, NY 13441-4505**8. PERFORMING ORGANIZATION
REPORT NUMBER****9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**AFRL/RISF
525 Brooks Rd
Rome NY 13441-4505**10. SPONSOR/MONITOR'S ACRONYM(S)****11. SPONSORING/MONITORING
AGENCY REPORT NUMBER**
AFRL-RI-RS-TR-2008-206**12. DISTRIBUTION AVAILABILITY STATEMENT**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# WPAFB 08-4197

13. SUPPLEMENTARY NOTES**14. ABSTRACT**

This interim technical report discusses a project under the Advanced Visualization and Interactive Displays (AVID) in-house program to develop techniques to load and display voluminous high-resolution terrain data with sufficient visual quality and interactivity. The AVID program objectives are to develop, evaluate, and exploit new concepts in information visualization, display technology, and human-computer interaction (HCI) that provide airmen with a tailored information environment. The result of this project was the successful design and implementation of a data structure for storing, manipulating, and processing terrain data vertex information, and an algorithm that intelligently processes the information contained within the data structure.

15. SUBJECT TERMS

3D Continuous Level of Detail, 3d CLOD, Massive dataset visualization, Digital Terrain Elevation Data visualization, DTED visualization.

16. SECURITY CLASSIFICATION OF:**a. REPORT**
U**b. ABSTRACT**
U**c. THIS PAGE**
U**17. LIMITATION OF
ABSTRACT**

UU

**18. NUMBER
OF PAGES**

17

19a. NAME OF RESPONSIBLE PERSON

Peter A. Jedrysik

19b. TELEPHONE NUMBER (Include area code)

315-330-2150

Table of Contents

1	Introduction	1
2	Approach	1
2.1	Data Structure	2
2.2	Algorithm	4
3	Conclusions	11
	References.....	13

Table of Figures

Figure 1	Wireframe depiction of stitched bands	3
Figure 2	TreeNode structure	3
Figure 3	Non-textured OpenGL lit DTED	4
Figure 4	Top Level Example – Two Top Level TreeNodes Share IndexNodes ..	5
Figure 5a	Refinement Process – Initial State through Step 2	6
Figure 5b	Refinement Process – Step 2 through Step 5.....	7
Figure 6	Data structure after one refinement process iteration	8
Figure 7a	Decimation Process – Initial State	8
Figure 7b	Decimation Process – Step 1 through Step 3	9
Figure 8	Display Process Flow Chart.....	10
Figure 9	Applying the display process to the data structure.....	11

1 Introduction

3D Continuous Level of Detail (CLOD) systems for massive datasets often sacrifice the raw data format and accuracy for speed, removing undesirable features in the source data, or to eliminate inherent differences in resolution. We present a novel approach which does not have any of these limitations and is focused on displaying the source data from the National Geospatial Intelligence Agency (NGA) Digital Terrain Elevation Data (DTED) [1] without modification. The implementation ensures accurate and timely presentation of the terrain data that is critical in a Command and Control environment and addresses the in-house research objective to develop novel information visualization concepts.

This implementation addresses the need to display high resolution terrain data (e.g. DTED) from global to high-resolution scales. This type of task has seen proliferation among modern computers due to the free distribution of applications like Google Earth[®]. This approach addresses specifically the methodology for loading and managing gridded datasets of the ilk available from the NGA and the US Geological Survey (USGS). Since even the best modern hardware cannot use brute force methods for loading or displaying such voluminous data at the global level, the only way to achieve the visual quality and the interactive speed desired is to use real-time algorithms that vary the quality of the displayed content.

The ability to view terrains with such a large number of vertices is an important research area, where several in-core and out-of-core techniques have been developed. However, these techniques require an expensive pre-processing stage which decimates, or re-grids the data, changing the statistical accuracy of the original incoming data. This problem is still open and challenging as the amount and resolution of data is growing faster than that of our ability to visualize such data.

This report describes the implementation to load and manage large terrain datasets. This includes a data structure for storing, manipulating, and processing of vertex information, and an algorithm that intelligently processes the information contained within the data structure.

2 Approach

The approach provides a data structure and algorithm for loading and/or displaying gridded datasets by adaptively loading data from an external datastore

and providing this information to a modern graphic processing unit in an acceptable form. In particular, this approach uses independent view refinement criteria for determining the usefulness of a particular vertex of the gridded data, retrieves that data from some backing store and inserts that data into the Circular Linked List of Geometrical Relationship (CLLGR) data structure. When all necessary refinements have been completed, it dispatches the vertex information as a list of triangle fans for visualization. These refinements can be adaptive, such that more data may be fetched to satisfy the view refinement criteria, but a redraw can occur due to the desire to provide the user with a more responsive system. The data may be procedurally generated as is the case for fractally generated terrain or displaying of the World Geodetic 1984 (WGS84) reference ellipsoid, or can be read from memory if the desired data has already been provided to a cache, or can be loaded from some other persistent storage device like a modern hard disk drive, or can be fetched from some web service or other network capable infrastructure.

2.1 Data Structure

The algorithm is implemented using two different data structures; a forest of Quadtrees comprised of `TreeNode`s and a shared pool of elevation data called `IndexNodes`. Each `TreeNode` is a collection of linked `IndexNodes` that encode some spatial relationship, and potentially children `TreeNode`s. Each `IndexNode` maintains unique mappings to other `IndexNodes` and has a vertex for its Cartesian based information. The algorithm relies on a forest consisting of 282 `TreeNode`s, and 1500 `IndexNodes`. The top level `TreeNode`s form the coarsest representation of the Earth. Refinement is currently based on approximated projected screen area of the `TreeNode`; however, other previously published techniques for refinement can be easily integrated [2].

The number of initial `TreeNode`s is derived from segmenting the globe into latitudinal bands where each `TreeNode` in a band covers the same area as its neighbor. The bands are: 90°N to 82°N, 82°N to 80°N, 80°N to 75°N, 75°N to 70°N, 70°N to 50°N, 50°N to 30°N, 30°N to 0°N. This is repeated symmetrically in the southern hemisphere. Each band is made up of a different number of `TreeNode`s, but always a number that evenly divides the DTED elevation information. This prevents any top level `TreeNode` from cutting through the changes in data resolution of the underlying six DTED bands: 82°S to 75°S, 75°S to 50°S, 50°S to 0°N, 0°N to 50°N, 50°N to 75°N, and 75°N to 82°N.

Each `TreeNode` is comprised of nine `IndexNodes` that map directly to elevation posts found in the source data. `IndexNodes` store the native elevation data found in the DTED files along with four links to their adjacent neighboring `IndexNodes` which are unaware of the `TreeNode` topology. A critical component of this algorithm is that `TreeNode`s share `IndexNodes`. The `IndexNodes` on the right

edge of the “orange” triangle fan are the same as the IndexNodes on the left edge of the “blue” triangle fan (Figure 1).

During refinement of a TreeNode, the new data point is inserted by loading the new elevation from the source data then walking the appropriate edge to find its insertion point. If an IndexNode already exists with the same latitude and longitude, the new node is discarded. A distinct advantage is that a single TreeNode never needs access to its neighbor and that insertions can happen in non-uniform placements. This makes stitching between various resolutions possible.

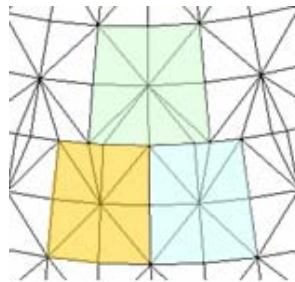


Figure 1 Wireframe depiction of stitched bands

For this example, and our current embodiment, it resembles a traditional Quadtree structure. Where each IndexNode has a “top”, “bottom”, “left”, and “right” pointer to another IndexNode, and each tree node consists of a center IndexNode (denoted as V0) and eight immediate neighbors, see Figure 2. Each TreeNode has zero or four children.

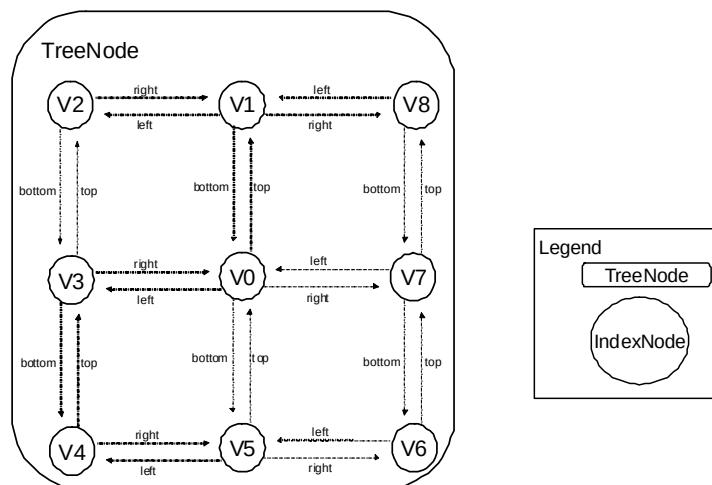


Figure 2 TreeNode structure

2.2 Algorithm

Our patent pending algorithm implements a particular view refinement process and has an out-of-core mechanism for the loading of NGA's DTED [3]. The algorithm uses a screen projected node width approximation for ascertaining whether refinements, or simplifications, of the CLLGR are required. The loaded information is also then projected into a user specified coordinate system that converts the source latitude and longitude tuples into geocentric vertex information. The three available projections at this time are a flat earth projection, a WGS84 projection, or a Lambert Conformal projection; however there is no limitation to the types of projections that are possible.

A significant advantage to the algorithm is it eliminates the need for conversions from the native formats allowing it to respond quickly to new data, preserve source data statistical accuracy, and allow for a controlled lighting environment. By stitching between various resolutions of data instead of drawing shelves between resolutions as is the case with NASA World Wind [4], Chunked LOD [5], and JCanyon [6], our algorithm can calculate normals and use OpenGL based lighting instead of only relying on pre-lit imagery. This is critical when displaying non-photographic imagery, road maps, or the raw terrain data since lighting provides the only visual cue of terrain undulations (Figure 3).



Figure 3 Non-textured OpenGL lit DTED

The procedure to define and process the data structure in preparation for rendering is as follows:

1. Create one or more top level TreeNodes - If more than one TreeNode is created then link neighboring TreeNodes where desired. For example, a cylinder could have two top level Tree Nodes, where traversing an IndexNodes right links would create a cycle (Figure 4). In this example the TreeNodes share edges of the finished geometry by sharing the IndexNodes labeled A, D, G, C, F, and I. This creates the lowest quality version for this case. Each TreeNode directly accesses nine IndexNodes, but due to the sharing of some indexNodes, there are only twelve unique IndexNodes.

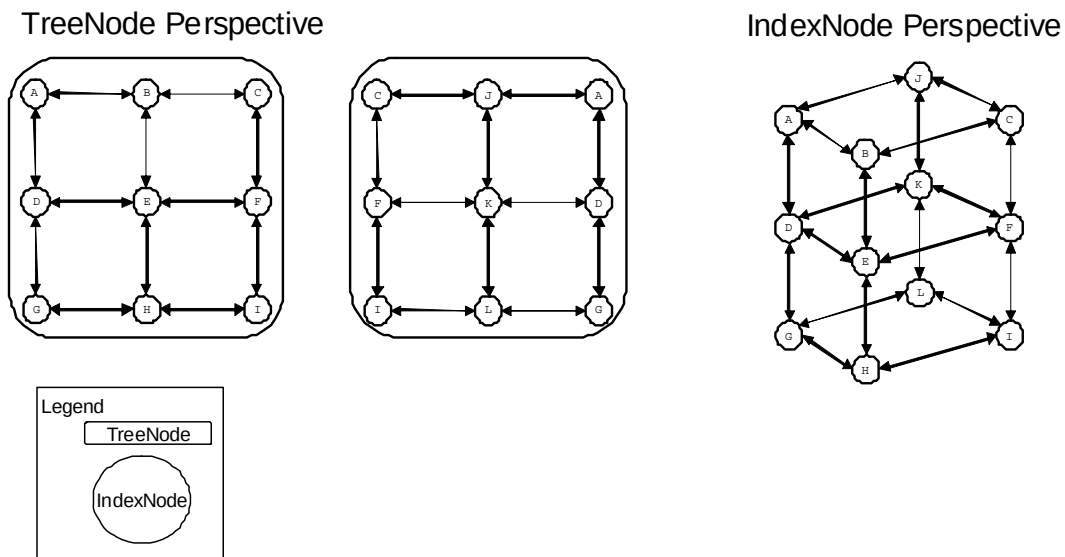


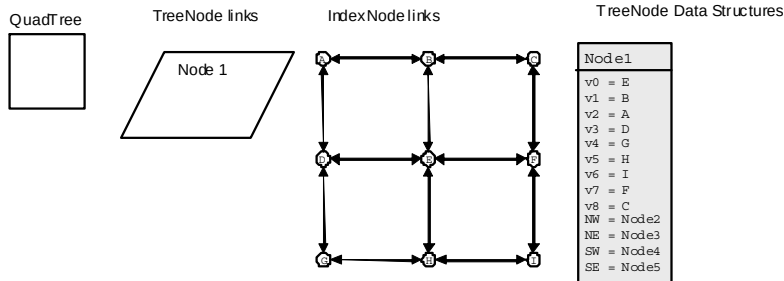
Figure 4 Top Level Example – Two Top Level TreeNodes Share IndexNodes

Before displaying a particular level of refinement, a view refinement step can be taken. This step essentially determines whether the current representation as stored by the tree of TreeNodes and IndexNodes satisfies the criteria. For this case, assume one TreeNode failed the check and needed refinement.

2. Test each TreeNode based on the defined quality measurement - If the current representation is less than the desired goal, the node is refined. Refer to the description for the refinement process below (Figure 5). Step 1 is to create four TreeNode children. Step 2 is to create up to sixteen new IndexNodes. If an IndexNode already exists that satisfies the geometric relationship, then the previously created IndexNode must be shared by these new IndexNodes. This is

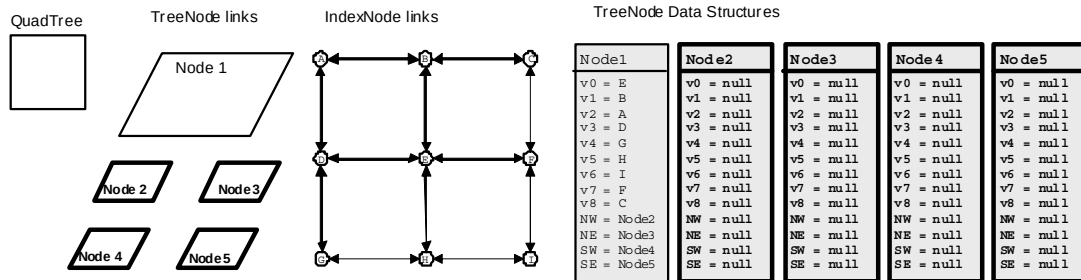
identical to the requirements for the TreeNodes as specified in Figure 4. In this example the assumption is that no previously created IndexNodes satisfy the geometric relationship so sixteen new IndexNodes are created. Step 3 is to assign the newly created IndexNodes to their respective TreeNodes. Notice that the sharing of IndexNodes is pervasive through this algorithm and is the basis for it's efficiency and effectiveness. Step 4 is to assign the four newly created TreeNodes to the TreeNode that failed the view criteria. Step 5 is to then link in the new IndexNodes.

Initial State



* bold indicates changes from previous state

Step 1) Create 4 new Tree Nodes



Step 2) Create 16 new IndexNodes

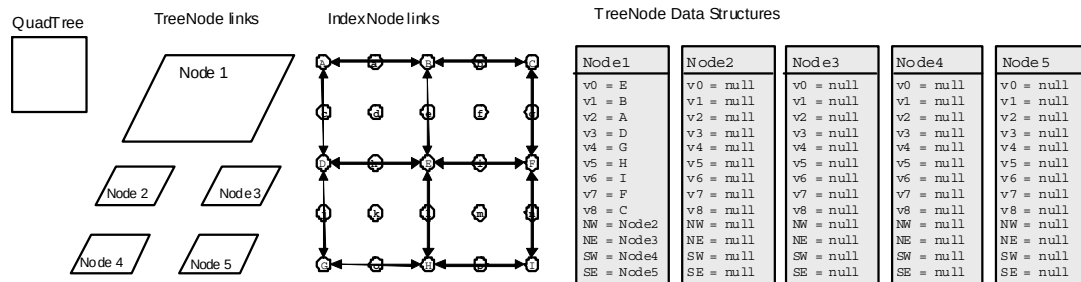
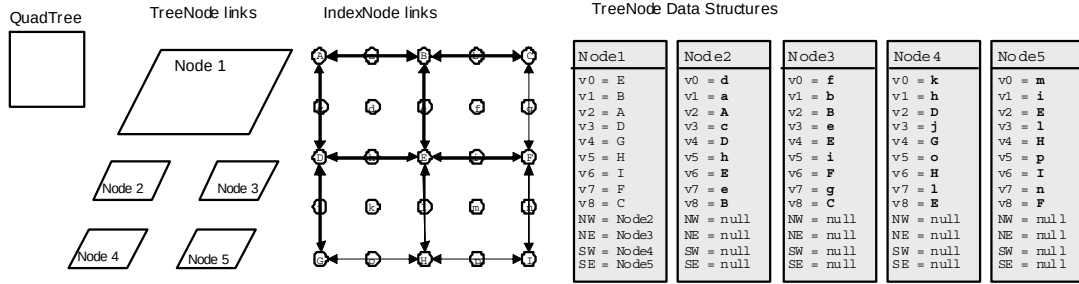
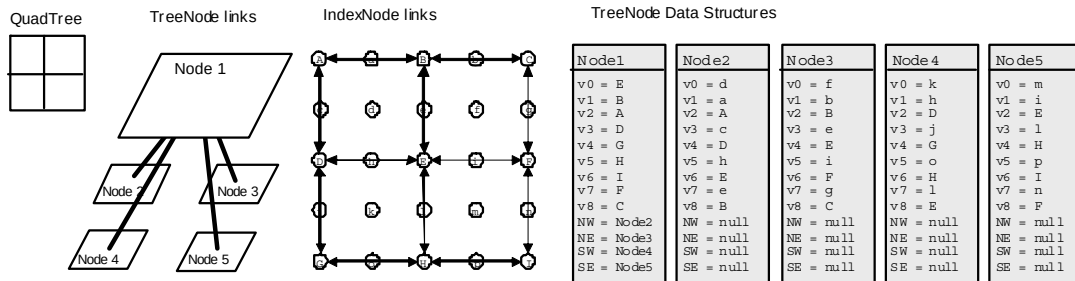


Figure 5a Refinement Process – Initial State through Step 2

Step 3) Assign new IndexNodes



Step 4) Assign Children Tree Nodes



Step 5) Link Index Nodes together

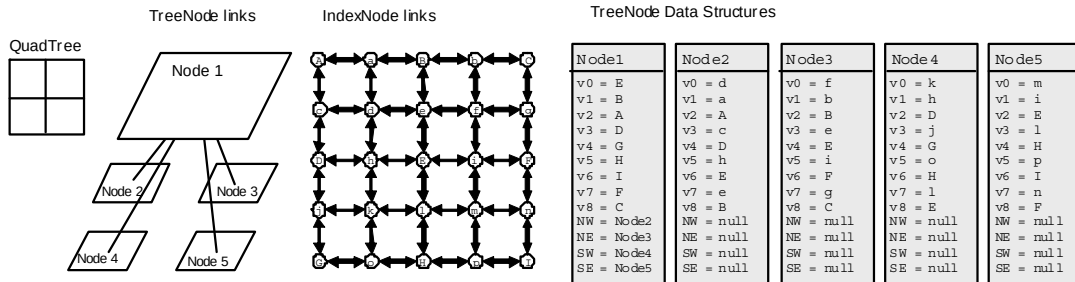


Figure 5b Refinement Process – Step 3 through Step 5

After these connections are created the ability to create a cycle by walk around the IndexNodes is preserved. This creates the depiction in Figure 6.

TreeNode Perspective

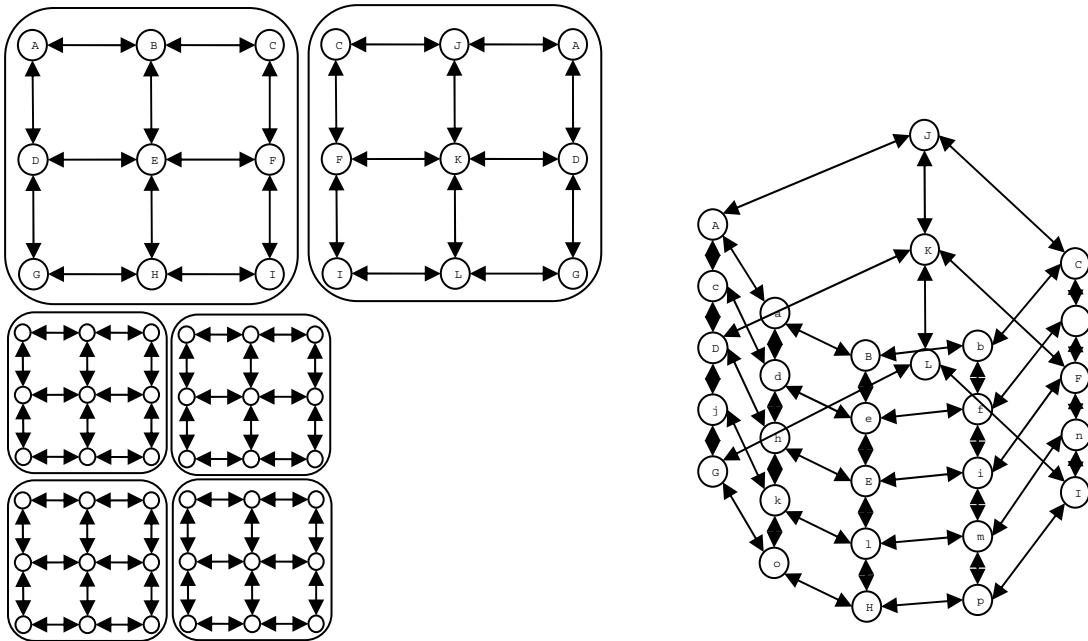


Figure 6 Data structure after one refinement process iteration

If the current representation exceeds the desired goal, the node is decimated. Refer to the description for the decimation process below (Figure 7).

Initial State

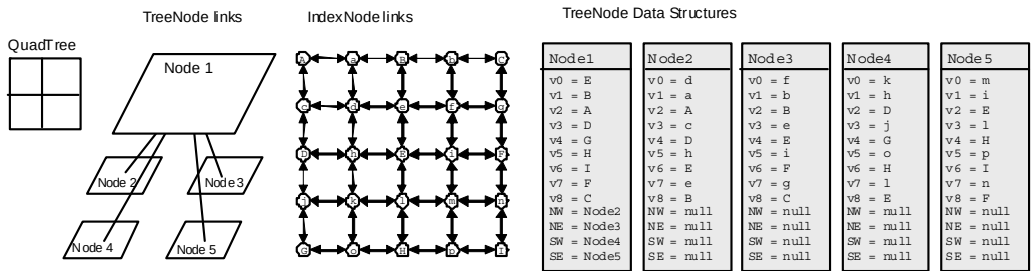
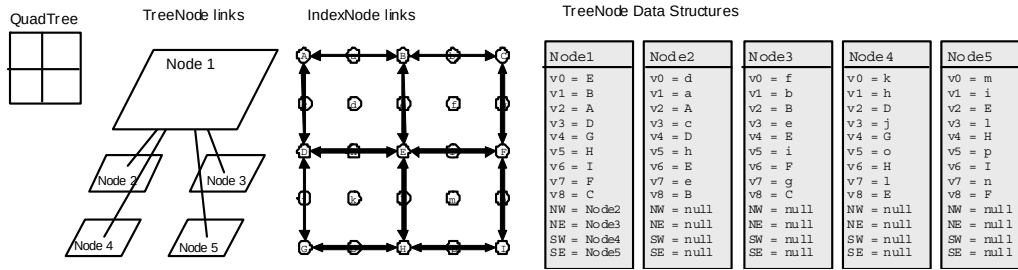
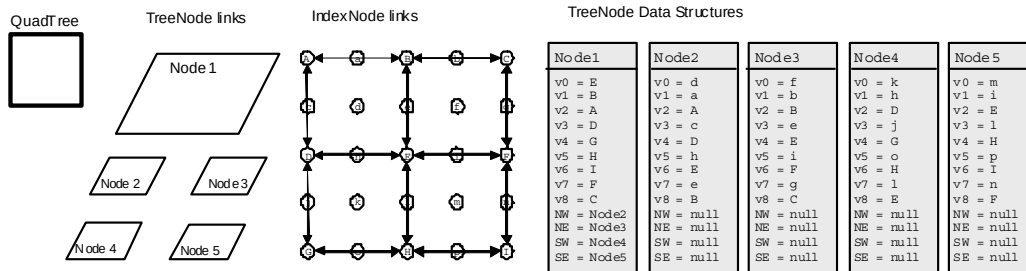


Figure 7a Decimation Process – Initial State

Step 1) Unlink 16 IndexNodes



Step 2) Disconnect Children Tree Nodes



Step 3) Remove children TreeNodes

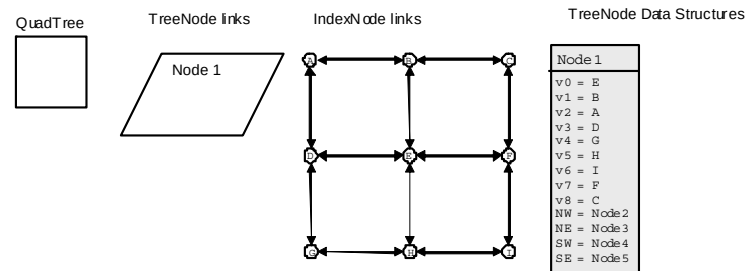


Figure 7b Decimation Process – Step 1 through Step 3

3. Generate triangle fans from the TreeNodes for rendering - Referring to Figure 8, after all necessary refinements are made, the display process of creating the triangle fans to be sent to the graphics card begins. The process is applied at all TreeNodes that do not have children. For each TreeNode, transform the data in the IndexNode into the desired coordinate system and add it to a list of vertices, referred to as the vertex list, for the triangle fan. Follow that IndexNode's top link. Transform that new IndexNode's data into the desired coordinate system and add it to the vertex list. Follow that IndexNode's left link.

Transform and add that new IndexNode's data to the vertex list. If this node has a bottom link, take it; otherwise follow the IndexNode's left link. Continue this operation until all directly accessible IndexNodes for this TreeNode have been traversed. This may result in more than nine vertices being stored, since a neighbor may have refined.

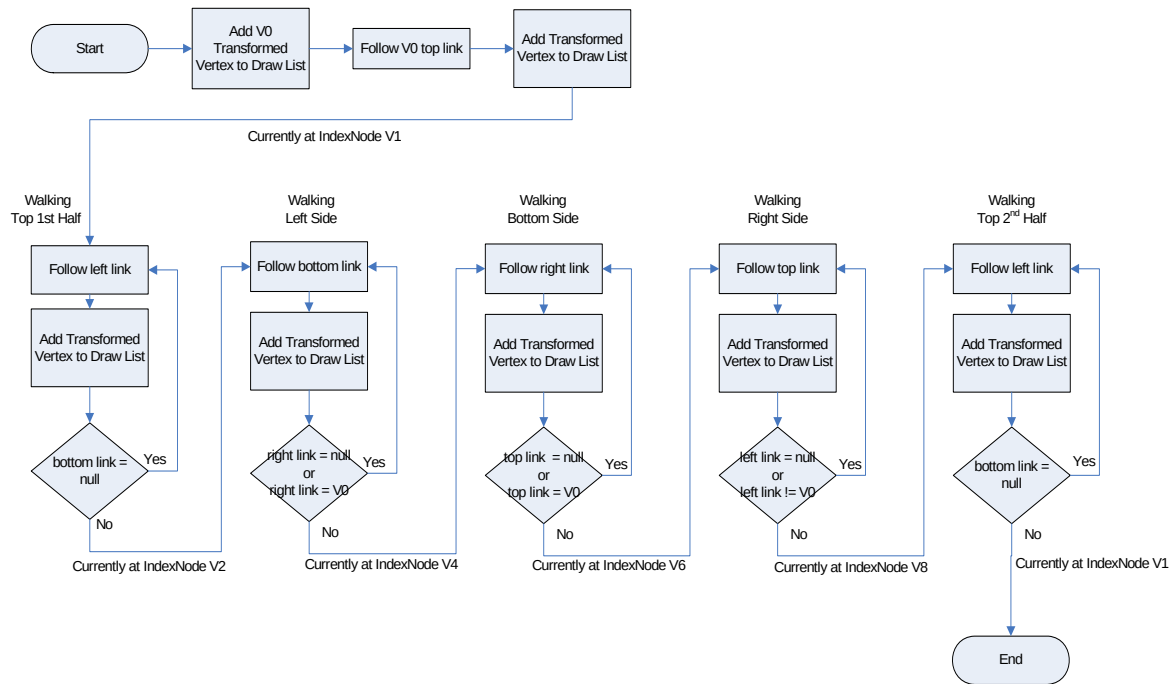


Figure 8 Display Process Flow Chart

Referring to Figure 9, after applying the display process explained in Figure 8 to the data structure depicted in Figure 4, the resultant triangle fans would be generated. Each individual fan is shaded with a different pattern. Since there are five TreeNodes that do not have children, 5 triangle fans are created. The triangle fans avoid T-junctions by the fact that the IndexNodes are shared and thus neighbors that require additional levels of refinement are effectively inserting IndexNodes for multiple TreeNodes without the need for direct knowledge of their siblings. The final step is to send the triangle fan list generated as efficiently as possible to your particular graphics hardware.

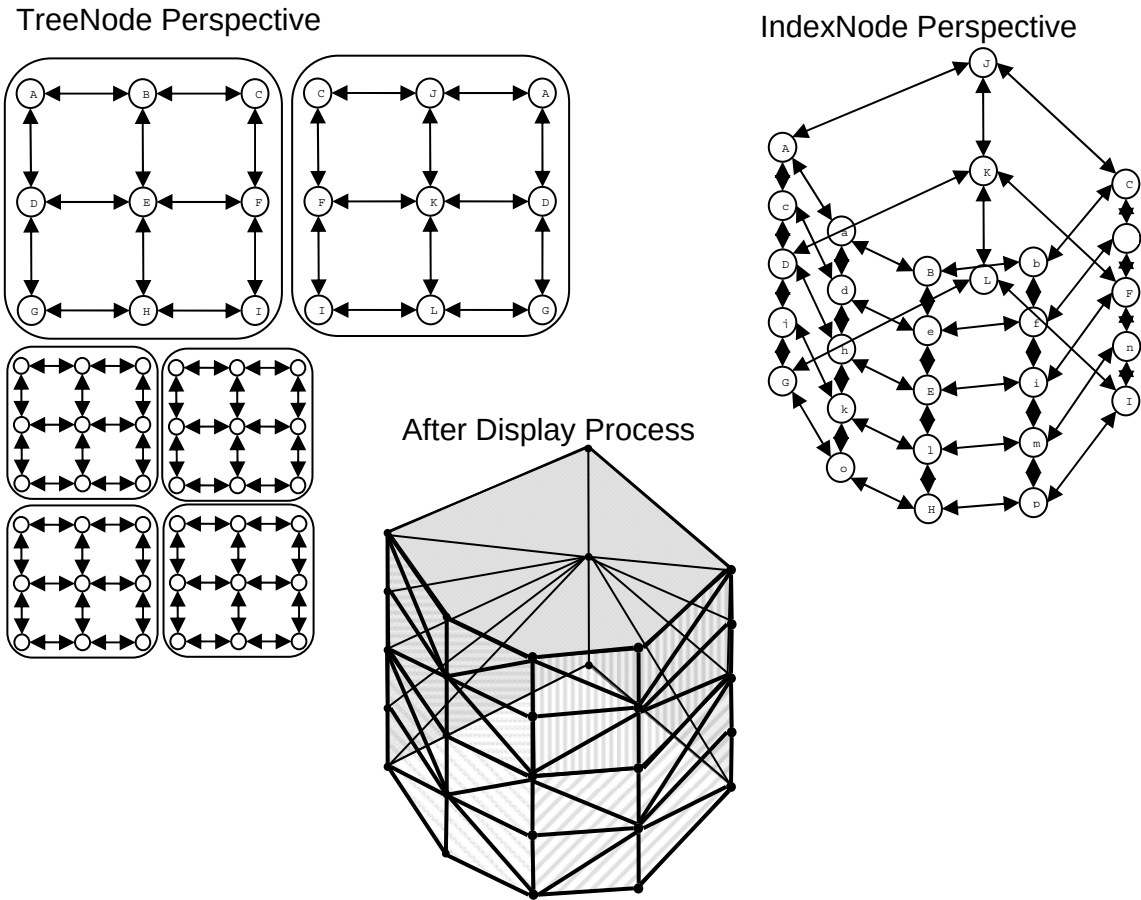


Figure 9 Applying the display process to the data structure

The display process can be easily modified eliminating the need to transform or to store the entire transformed vertex data into other more effective data structures for particular hardware or graphics library.

This algorithm could easily be modified to use non-uniformly gridded data, as through the use of currently published works that perform a gridding operation. Non quad tree representations could also be used, allowing for different refinement processes. Non-view dependent criteria can also be used.

3 Conclusions

Timely presentation of massive terrain datasets is critical in a Command and Control environment with a delicate balance between speed and accuracy. It is extremely important to preserve the raw data for precise mission planning, but

just as important is fast access to the information to make timely decisions. We have developed a novel approach for displaying massive datasets without sacrificing the raw data format and accuracy for speed.

This implementation addresses the need to display high resolution terrain data (e.g. DTED) from global to high-resolution scales. This approach addresses specifically the methodology for loading and managing gridded datasets. Since even the best modern hardware cannot use brute force methods for loading or displaying such voluminous data at the global level, the only way to achieve the visual quality and the interactive speed desired is to use real-time algorithms that vary the quality of the displayed content.

The algorithm described here provides a novel approach to visualization of unmodified source data where data precision is important but varies along the surface. This technique isn't constrained to geospatial visualization and can be used to display other models or mathematical functions.

REFERENCES

- [1] National Imagery and Mapping Agency, "Digital Terrain Elevation Data (DTED), Tech. Rep. MILPRF-89020B", 2000.
- [2] Duchaineau, M., Wolinsky, M. Sigeti, D., Miller, M., Aldrich, C., Mineev-Weinstein, M., "ROAMing Terrain: Real-time Optimally Adapting Meshes", 1997.
- [3] Moore, J., and McVay, A., "Method for Loading And Displaying Massive Gridded Datasets Patent #60/879,211", 2007.
- [4] National Aeronautics and Space Administration, "World Wind", <http://worldwind.arc.nasa.gov>, 2006.
- [5] Ulrich, T., "Super-Size It! Scaling Up to Massive Virtual Worlds", ACM Siggraph 2002.
- [6] Russell, K., "JCanyon: Grand Canyon for Java", SUN Microsystems, 2001.