DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/88D-18

A SOFTWARE SYSTEM TO CREATE A HIERARCHICAL, MULTIPLE LEVEL OF DETAIL TERRAIN MODEL

THESIS

LeeAnne Roberts
Lieutenant, NOAA

AFIT/GCS/ENG/88D-18

AFIT/GCS/ENG/88D-18

A SOFTWARE SYSTEM TO CREATE A HIERARCHICAL, MULTIPLE LEVEL OF DETAIL TERRAIN MODEL

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Systems

LeeAnne Roberts, B.A., M.A.

Lieutenant, NOAA

December 1988

Preface

The goal of this effort was to design and implement a software system that will generate a hierarchical database composed of more than one level of detail of a terrain model and cultural objects from digital elevation and cultural data files.

I would like to thank my advisor, Major Phil Amburn, for his support and encouragement during this effort. I would also like to thank my readers Major John Stibravy and Captain Donlan for their guidance and timely suggestions. I would also like to express my gratitude to Armstrong Aeromedical Research Laboratory and the National Oceanic and Atmospheric Administration for their sponsorship of this effort.

LeeAnne Roberts

Table of Contents

List of Figures

List of Tables

Abstract

The purpose of this effort was to design and implement a software system that will generate a hierarchical database composed of more than one level of detail of a terrain model and cultural objects from digital elevation and cultural data files.

The software system designed and implemented in this effort was written in the C language and implemented on a Digital Equipment GPX Workstation and on a Silicon Graphics Iris 3130 Workstation.

Although the primary source of elevation data used in this effort was the Defense Mapping Agencies' (DMA) Digital Terrain Elevation Data (DTED), a method was developed to build a terrain model from non-gridded elevation data. In addition to allowing the use of non-DMA data, this method also makes it possible to build a terrain model from a non-gridded subset of a DMA DTED data file.

Two database formats were designed and implemented in this effort. The first database (which was used to generate the terrain model) contains lists of the vertices, exterior edges, interior edges, and polygons (and their interrelations) that form the terrain model. The second database contains polygons (that form the terrain model) and cultural features (such as radio towers) at one or more levels of detail, and a search tree (so that data in a particular area may be retrieved quickly).

The list database may be used to generate contours, or non-real-time images of terrain. The hierarchical database may be used in a simulator environment.

A SOFTWARE SYSTEM TO CREATE A HIERARCHICAL, MULTIPLE LEVEL OF DETAIL TERRAIN MODEL

I. Introduction

1.1 Background

Armstrong Aerospace Medical Research Laboratory (AAMRL) has developed a head mounted stereoscopic display for use in a simulator environment. Currently, the display of the terrain model used by the simulator is limited to a monochrome grid of lines, such as that in Figure 1, and was created using non-automatic methods. The reality of the simulated environment would be enhanced by displaying the terrain as a colored surface instead of a grid of lines. Work is currently underway to update the head mounted display hardware and simulator software to be capable of displaying color surfaces. A software system that will automatically generate a terrain model from elevation data would provide a variety of realistic terrain models that could be used by the updated simulator.

Elevation data and terrain features are available in digital form from several agencies including the Defense Mapping Agency (DMA), and the National Oceanic and Atmospheric Administration (NOAA) (for more information see Appendices C and D).

DMA's terrain file (Digital Terrain Elevation Data or DTED) contains elevation data, and its cultural file (Digital Feature Analysis Data or DFAD) contains surface details such as areas covered by forest, and locations of some man-made features such as dams, bridges, and pipelines.

NOAA's nautical data files contain data for nautical charts such as depth data, aids to navigation (buoys and lighthouses), and dangers to navigation (rocks and reefs). NOAA's aeronautical data files contain data for aeronautical charts such as aids to navigation, and dangers to navigation (radio towers, tall buildings, and other obstructions).

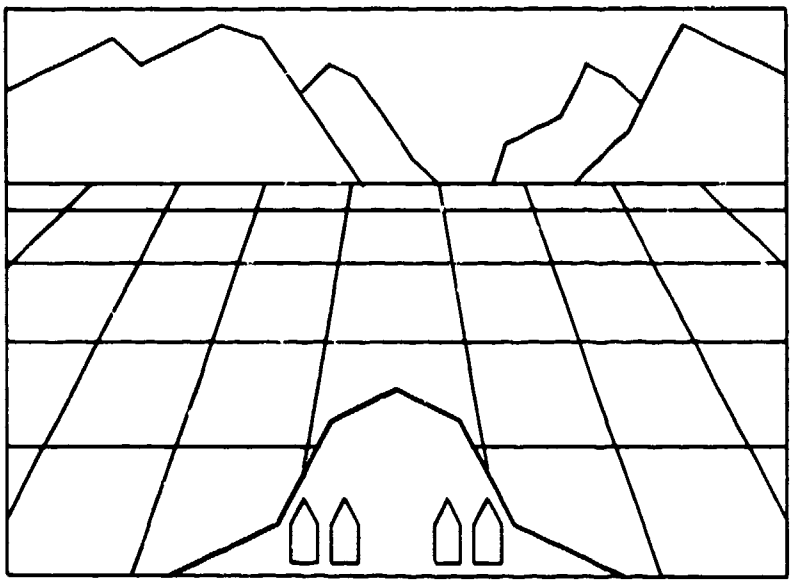


Figure 1. A typical display of a HMD system
(11:4)

1.2 Problem Statement

The goal of this effort is to design and implement a software system that will generate a hierarchical database composed of more than one level of detail of a terrain model and cultural objects from digital elevation and cultural data files.

1.3 Scope

It is well known that in real-time systems the time spent on generating the image of an object should be proportional to the area that the image will occupy on the display screen. Objects that are farther away from the viewer, will occupy a smaller area of the display screen than objects that are closer to the viewer, and need not be represented by as detailed a model. Because this terrain model will be used in a simulator environment, however, an object might not remain far away from the viewer. The terrain model should, therefore, include more than one level of detail for the terrain and cultural objects.

1.4 Limitations

This effort will be limited to the development and implementation of a software system that will generate a terrain database from elevation data and cultural data, and a library of procedures that may be used by an application program to access the data. Existing software will be used to display the data for testing purposes.

Because of the timing constraints imposed by simulator display hardware and software, the number of polygons that may be displayed without degrading the simulated motion of the user through the terrain model is limited. This number of polygons, or polygon budget, will depend on the display hardware and software. In order to remain within the polygon budget, the display software should be able to access only the data that will be displayed.

1.5 Assumptions

The simplest data structure for storing a terrain model in a database is a sequential file. As the terrain model grows in size and complexity, however, the use of a sequential file becomes inadequate because the time taken to process the terrain model and then display the visible sections of the model becomes longer than the minimum acceptable screen refresh time. The display software should be able to retrieve a subset of the terrain database without being required to read through all of the model. This will be accomplished in this effort by the use of hierarchical data structures. The display software should also provide each section of the terrain model at a level of detail that is consistent with the screen area it will occupy. This will be accomplished in this effort by the use of multiple levels of detail.

The basic surface element of the terrain model will be a triangle. Many display algorithms require that the surface elements or polygons be planar. One way to insure that the polygons are planar is to use triangles.

1.6 *General Approach*

The design effort will be divided into two areas: generating the terrain model, and selecting an appropriate data structure for storing the terrain model in a database.

The design effort for the terrain model generator will be concentrated in two areas: reducing the volume of the data, and generating a terrain model from elevations that might not be arranged in a rectangular grid.

A one-degree by one-degree cell of DMA elevation data may contain more than a million elevations. If most of those elevations describe a flat plane, then most of the elevations may be discarded. Similarly, a mountain seen on the horizon can be described by a less detailed set of polygons. An algorithm must be designed to determine which elevations will be used for the different levels of detail.

Elevations from DMA DTED files are located on a rectangular grid. Data from other sources might not be arranged on a rectangular grid. (see Fig 2) The software system that generates the terrain model must, therefore, be able to generate a terrain model from non-gridded data.

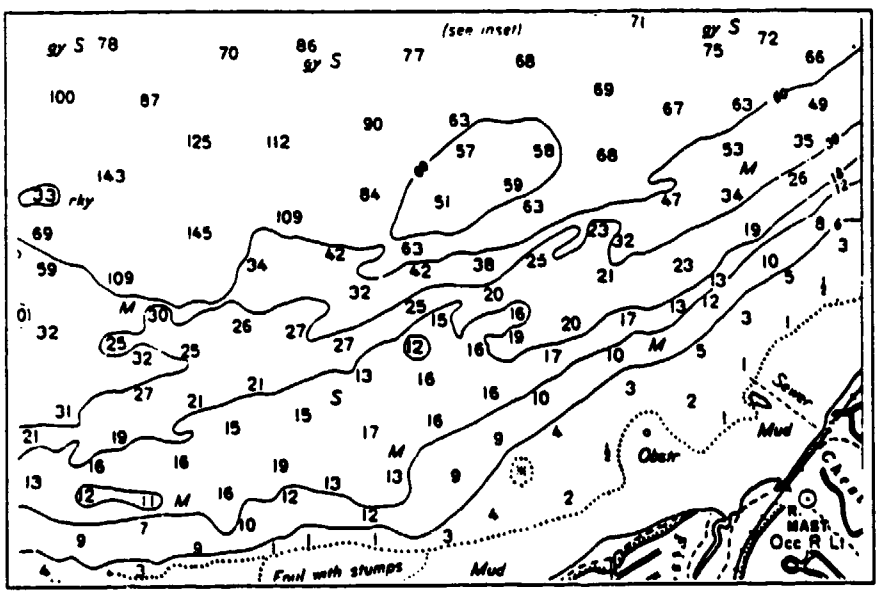


Figure 2. Non-Gridded Data
(8)

The database data structure must be selected so that the simulator display software may retrieve a subset of the terrain model. This will be accomplished by using hierarchical data structures, and by providing more than one level of detail of the data to be displayed. The display software will be able to retrieve only the data that is required for the display, at the level of detail that is required.

1.7 Presentation

This thesis is made up of four chapters and four appendices. Chapter II presents a survey of the literature in the areas of terrain model generation, hierarchical data structures, and databases containing more than one level of detail. Chapter III covers the design and implementation of the system. Chapter IV covers the results of this effort. Appendix A contains the user manual, Appendix B contains a more detailed coverage of the algorithms used in this effort, Appendix C contains information about DMA data, and Appendix D contains information about NOAA data.

II. Literature Review

2.1 Overview

The goal of this effort was to design and implement a software system that will generate a hierarchical database composed of more than one level of detail of a terrain model and cultural objects from digital elevation and cultural data files. Research in the areas of terrain model generation, hierarchical data structures, and databases containing more than one level of detail had to be accomplished before the design effort could begin.

2.2 Terrain Model Generation

DMA's DTED data (elevation data) are arranged in a rectangular grid. Most terrain model generators that use DMA's digital data make use of this aspect of the data file by making two triangles from the four corner points of each grid cell (4) (9) (12) (see Fig 3). Using this technique of model generation, the quantity of data may be reduced or filtered by selecting every Nth elevation, by selecting an elevation in the area near a grid location that is representative of the elevations in that area (highest, lowest, ...) and moving it to the grid location, or by interpolating a value for the grid location from the adjacent elevations.

Mirante and Weingarten presented an algorithm for building an irregular network of triangles "by radiating triangles from the node closest to the centroid" (7:12). The data must be sorted by heading and distance from the node (point) closest to the centroid. When all of the points have been added to the network, new triangles are added to make the boundary of the network convex. Finally, the shapes of the triangles are corrected (to get rid of long, thin triangles) (see Fig 4):

> ... each triangle is tested against each neighbor to determine whether their indices should be switched. ... If the distance between the two common nodes is greater than the distance between the two unique nodes, the triangle indices are switched ... This process is repeated until an entire pass through the data base produces no changes. (7:13)

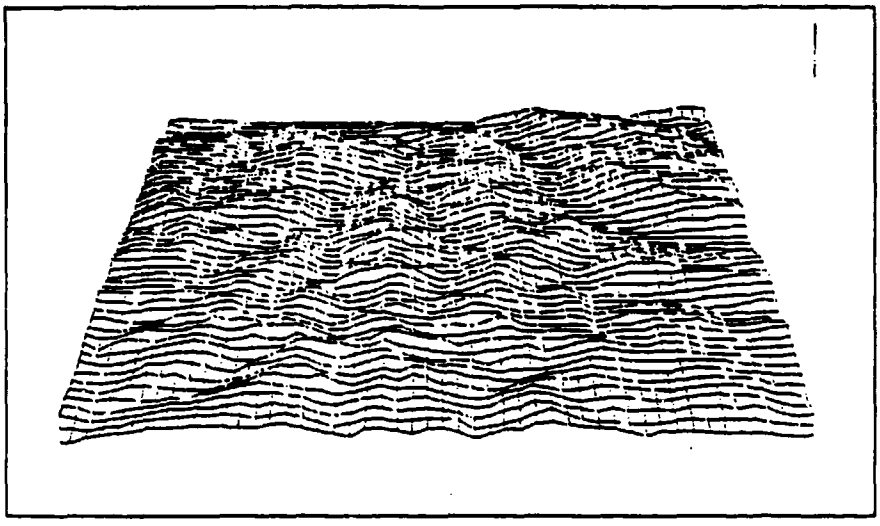


Figure 3. Terrain Model Built From Elevations On A Rectangular Grid (12:46)

2.3 Hierarchical Data Structures

Elevations that are arranged in a rectangular grid may be stored in an array and can be accessed by computing the array indexes. Elevations that are not arranged in a rectangular grid must be stored and accessed using a more complex data structure. In this effort, the terrain model will be stored using hierarchical data structures. Three such data structures are Quadtrees, kd-Trees, and spatial kd-Trees.

2.3.1 Quadtrees A Quadtree is a four-way tree used to encode two dimensional data. An area is divided into four subareas if it contains more than one object. Quadtrees may be subdivided based on many parameters such as texture or color. For example (see Fig 5), a digital image composed of an array of pixels (spots of color) may be stored using a Quadtree. If the image contains more than one color, it is divided into four subareas. If any of the subareas contain more than one color, the subarea is in turn divided into four subareas. This process is continued until all of the "leaf" subareas consist of one color (13).

2.3.2 kd-Trees A kd-tree is a variation of a k-dimensional homogeneous binary search tree. In a normal kd-tree (see Fig 6), a node has left and right pointers (lp,rp), a

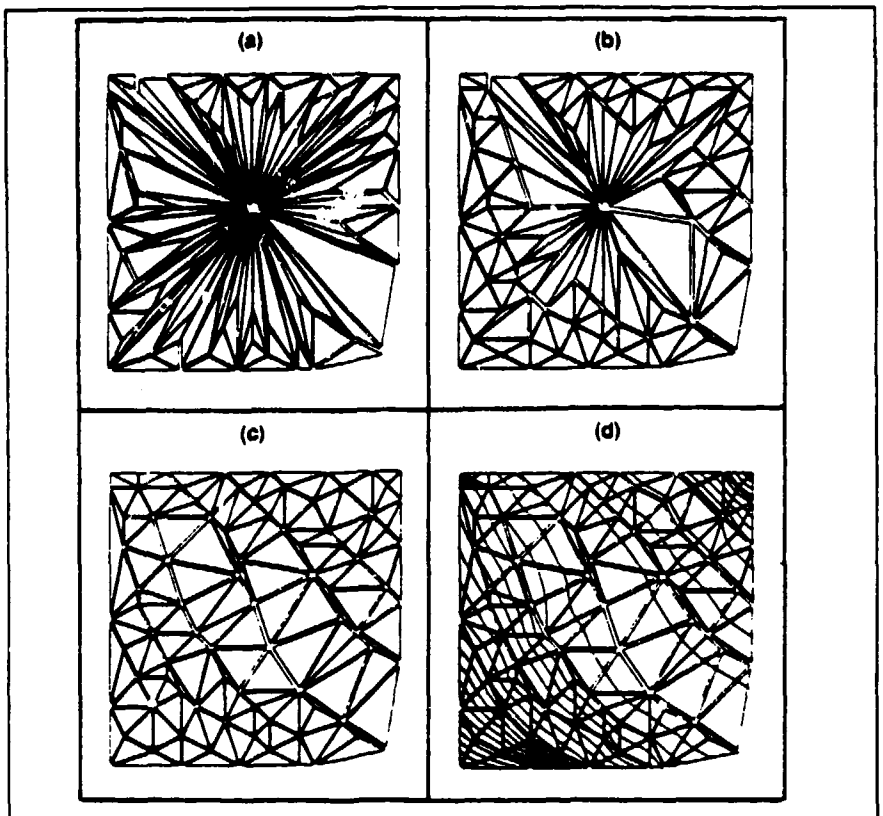


Figure 4. Triangle switching and contouring: (a) mesh before first switching pass, (b) after first switching pass, (c) after all switching passes, and (d) with contour cuts.

Figure 4. Radial Sweep Algorithm
(7:13)

discriminator (d) (which determines which of the k dimensions will be used to indicate the direction of the search), a value of that d dimension (d0), and a representation of the data. All of the data in the subspace indexed by the left pointer have a value in the d dimension that is less that d0, and all of the data in the subspace indexed by the right pointer have a value in the d dimension that is greater than d0. The kd-tree works well for point data, but objects of non-zero size (lines, surfaces, and solids) must be either split into smaller objects, or duplicated (have duplicate references) (10).

2.3.3 Spatial kd-Trees In a spatial kd-tree, a node has two additional values (see Fig 7). These values are the maximum value of the d dimension in the left subspace (lmax), and the minimum value of the d dimension in the right subspace (rmin). An object of non-

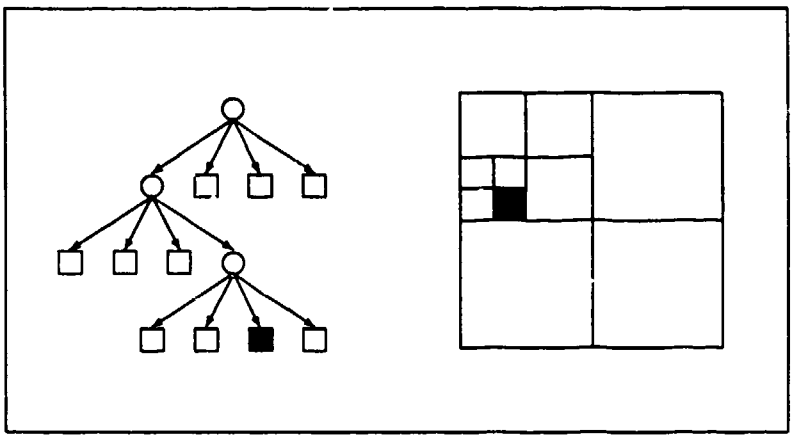


Figure 5. A Quadtree

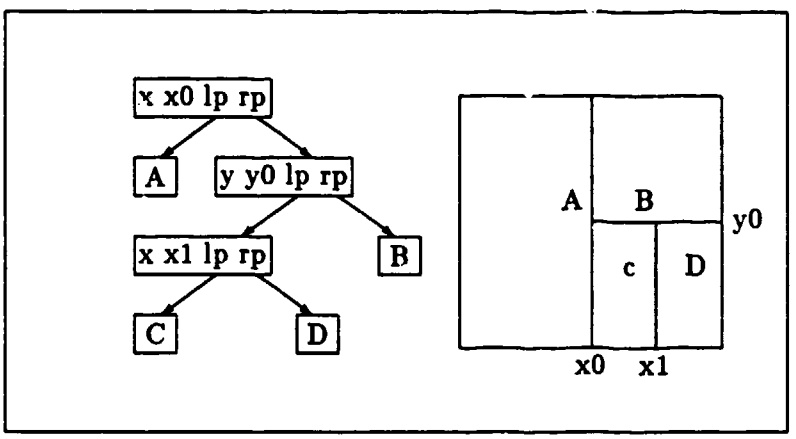


Figure 6. kd-Tree

zero size will be stored in the subspace that contains its centroid. If this objects' centroid had a value in the d dimension of d1 which was greater than d0, then it would be placed in the right subspace. If this object had a minimum value in the d dimension of d2 that was less than d0, then the value of rmin would be changed to min(d2,rmin). In Figure 7, the centroid of object B has a value in the x direction that is greater than x0. It is therefore placed in the right subspace. Object B has a minimum value in the x direction of x2 which is less than x0. Therefore, value of rmin is x2. In this way, non-zero sized objects can be stored in a spatial kd-tree without duplication of data or data indices (10).

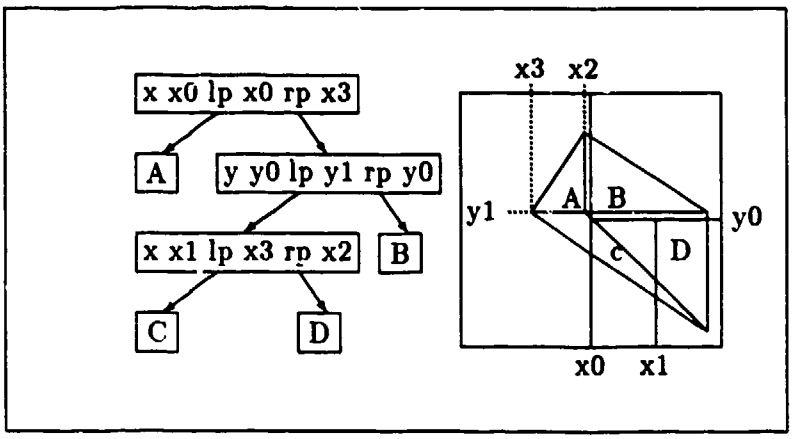


Figure 7. Spatial kd-Tree

2.4 Multiple Levels Of Detail

The time spent on generating the image of an object should be proportional to the area that the image will occupy on the display screen. This may be accomplished by providing descriptions of the object at varying levels of detail.

Paul MacDougal, in his PhD dissertation, applied "level of detail management" to a simulation of a submarine docking maneuver. A set of descriptions of each object in the simulation at different levels of detail was created. These descriptions were connected in a list, and the simulator software selected one description for each object depending on the distance at which a certain percentage of the objects' edges occupied an area smaller than or equal to the area of one pixel (6).

Instead of a set of fixed descriptions, objects may be displayed by instantiating procedural models (5:10-11). For example, a radio tower could be displayed by calling a procedure:

```
tower(Radio,Position,Height,RedFlashingLights,LevelOfDetail);
```

or several towers could be displayed by calling the procedure repeatedly:

```
tower(Radio,Position,Height,RedFlashingLights,LevelOfDetail);
tower(Control,Position,Height,NoAttribute,LevelOfDetail);
```

In addition to selecting the level of detail of an object by the display screen area that it occupies, Clark suggested:

> Since the center of attention of a scene is often its geometric center, one might effectively render the scene with a center-weighting of detail. In other words, the maximum area an object is allowed to cover before splitting it into its subobjects becomes larger towards the periphery of the field of view. This is somewhat analogous to the center-weighted metering systems of some cameras. Likewise, since moving objects are less resolved by both the human eye (because of saccadic suppression) and a camera (because of blurring), one can render them with an amount of detail that varies inversely with their speeds. (3:552)

Multiple levels of detail of an object can be provided to the display software by either storing multiple versions of the object in a database, or by instantiating the object at a selected level of detail with procedural modeling.

Chapter 3 will present the design and implementation of the software system that was developed in this effort.

III. System Design and Implementation

3.1 Overview

This chapter covers both the design and the implementation of the software system produced by this effort.

The software system designed and implemented in the effort was written in the C language and implemented on a Digital Equipment GPX Workstation (GPX) and on a Silicon Graphics Iris 3130 Workstation (IRIS).

The design effort was divided into two areas: generating the terrain model, and selecting an appropriate data structure for storing the terrain model in a database. The generation of the terrain model was broken into four tasks: acquiring the elevation data, processing the elevation data to form a hierarchical terrain model, acquiring the cultural feature data, and adding the cultural feature data to the hierarchical terrain model (see Fig 8).

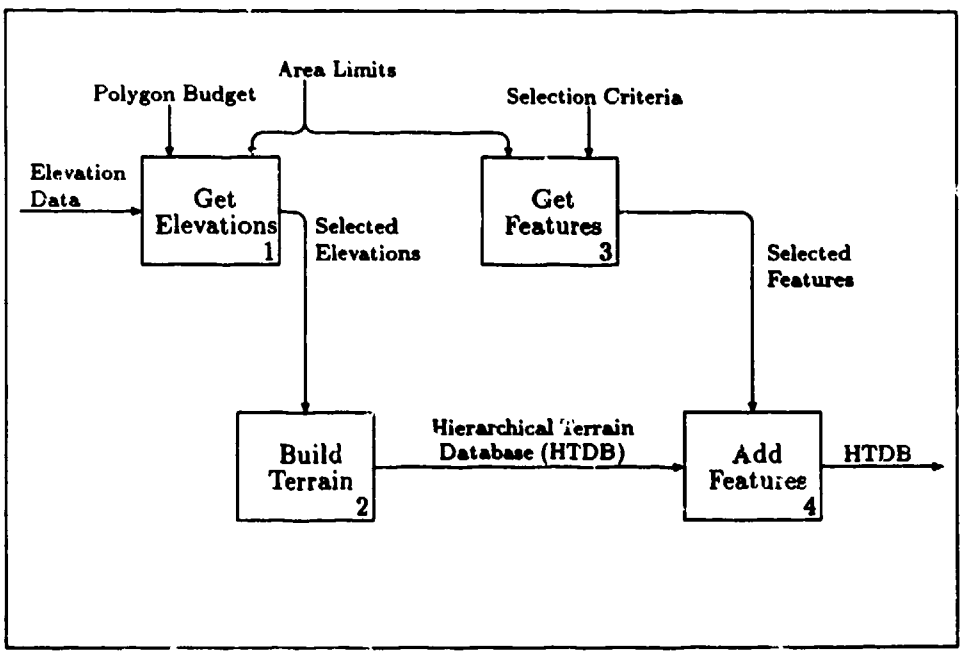


Figure 8. IDEF0 Diagram A0

3.2 Elevation Data Acquisition

The primary source of elevation data used in this effort was DMA's DTED data. The algorithms were designed so that they could be used with other sources of data such as NOAA's depth data. The problem with either DMA's elevation data, or NOAA's depth data was not acquiring the data, but having too much data.

As has been mentioned before, a one degree by one degree cell of DMA DTED data can contain more than a million elevations. Data acquisition was divided, therefore, into reading the elevations from a data file, and reducing the amount of data (or filtering the data) (see Fig 9).

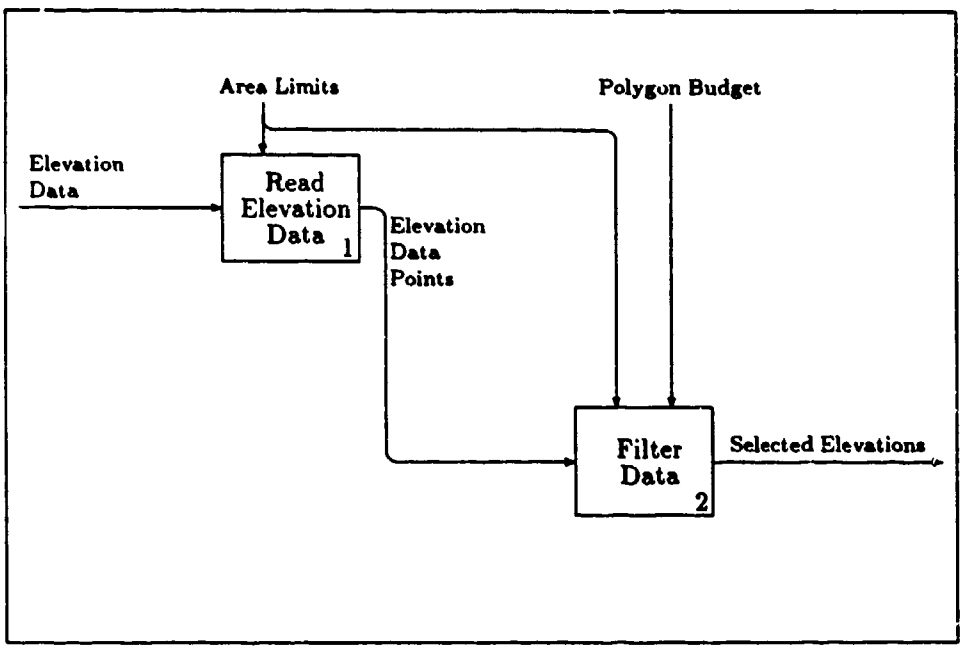


Figure 9. IDEF0 Diagram A1

3.3 Data Volume Reduction

The number of polygons (polygon budget) that may be displayed without degrading the simulated motion of the user through the terrain model is limited. In this effort, the user must select an appropriate polygon density in order to keep within this polygon budget. The polygon density will be application dependent. For example, if the area to be

displayed (at the highest level of detail) is 200 miles by 200 miles, and the polygon budget is 800 polygons, then the polygon density must not exceed 2 polygons per 100 square miles (or a distance between grid vertices of ten miles).

Two methods by which the data volume may be reduced are: (1) selecting every Nth vertex (if the data is gridded), or if the data is non-gridded, selecting the vertex that is closest to each grid position; or (2) dividing the data into rectangles of equal area and then selecting (by some selection criteria such as average height, highest elevation, lowest elevation ...) one or more vertices from each area. The easiest method is to select every Nth vertex. Because this method is easy to implement for data that is arranged in a rectangular grid, but not for non-gridded data, the software system will be tested with DMA DTED data and then, time permitting, with some non-gridded data.

Harry Ross, in his thesis *Analysis and Preparation of a Digital Terrain Data Base for Flight Simulator Use*, interpolated DMA DTED data using grid spacings varying from 50 meters to 500 meters (12). A data set with a grid spacing of 500 meters could be produced by selecting every fifth vertex from a DMA DTED file. This data set would be composed of 58,081 data points. The time needed to build the terrain model from the elevation data varies by the square of the number of elevations. In order to build a terrain model with a grid spacing of less than one mile, the area covered must be limited. The first method used to reduce the volume of data was to select every twentieth vertex from a DMA DTED file (giving a file with 3721 vertices and a grid spacing of approximately one mile) (see Fig 10). The second terrain model was built by selecting every fifth vertex from a DMA DTED file (a grid spacing of approximately 500 meters) and by dividing the file into sixteen sections. Figure 11 shows one of those sections.

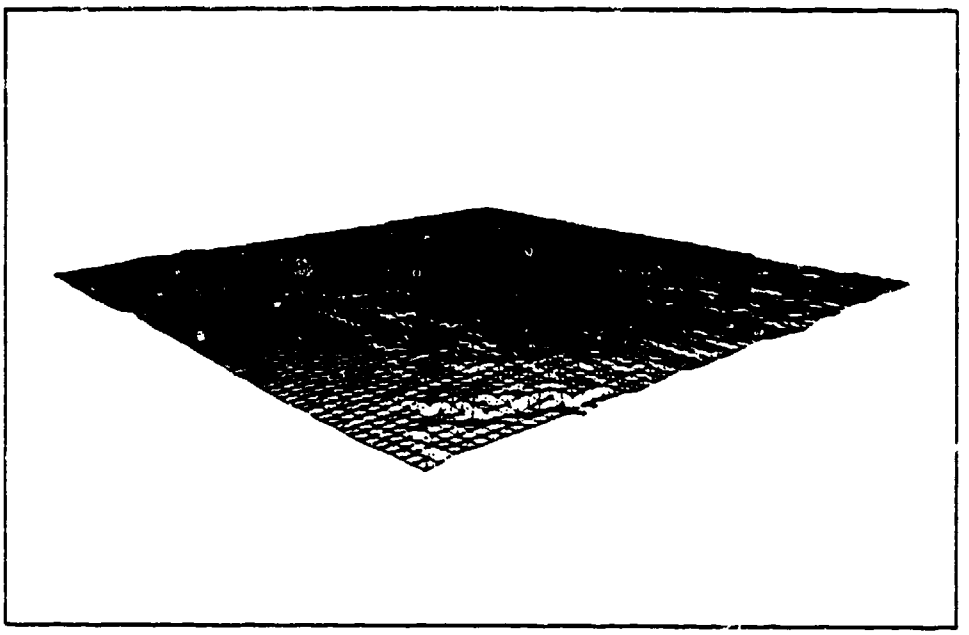


Figure 10. Terrain Model Built From Data With A One Mile Grid

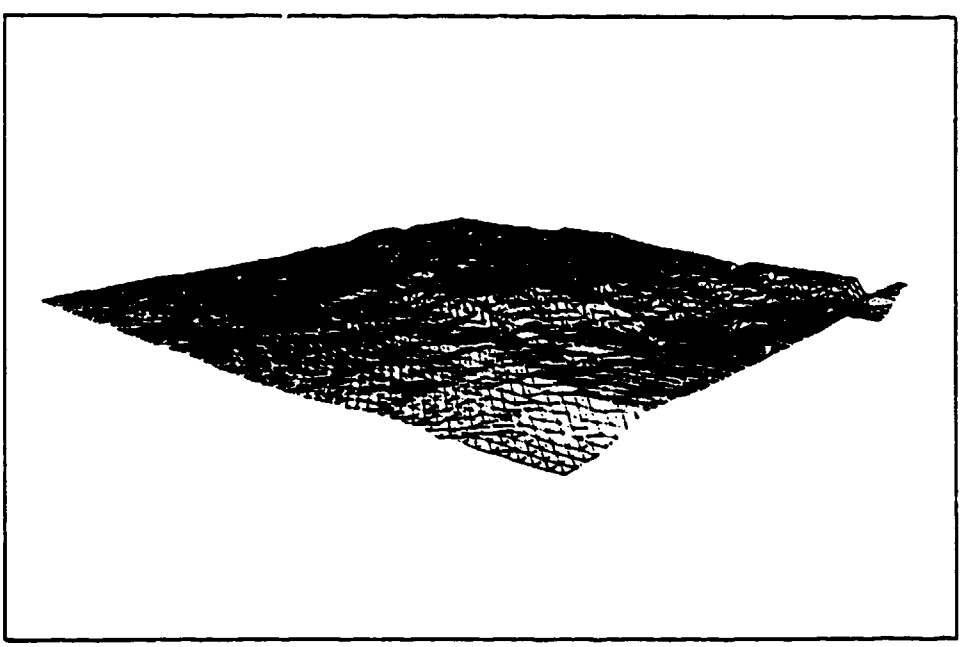


Figure 11. Terrain Model Built From Data With A 500 Meter Grid

3.4 *Building The Terrain Model*

The software system created in this effort was designed to use either DMA digital data, or NOAA digital data. Since NOAA depth data is not arranged on a rectangular grid, the process used to generate the terrain model had to be capable of using non-gridded elevations. The terrain model is built by dividing the data by area into smaller pieces, building a terrain patch (a small terrain model) from the data in each piece, and then connecting the patches to form the terrain model (see Fig 12). Each patch was formed by selecting three vertices not on a straight line, making a triangle from them, and then adding the rest of the vertices one at a time (for more information, see Appendix B). This method is similar to the way a patch-work quilt is made (see Figure 13). Each patch is connected to the patch to its right (East) until all of the patches in one row are connected to form a terrain strip. The patches in the row above the terrain strip are then connected to form another terrain strip. The two strips are connected to form the terrain model. Each succeeding row of patches is connected to the terrain model in the same fashion.

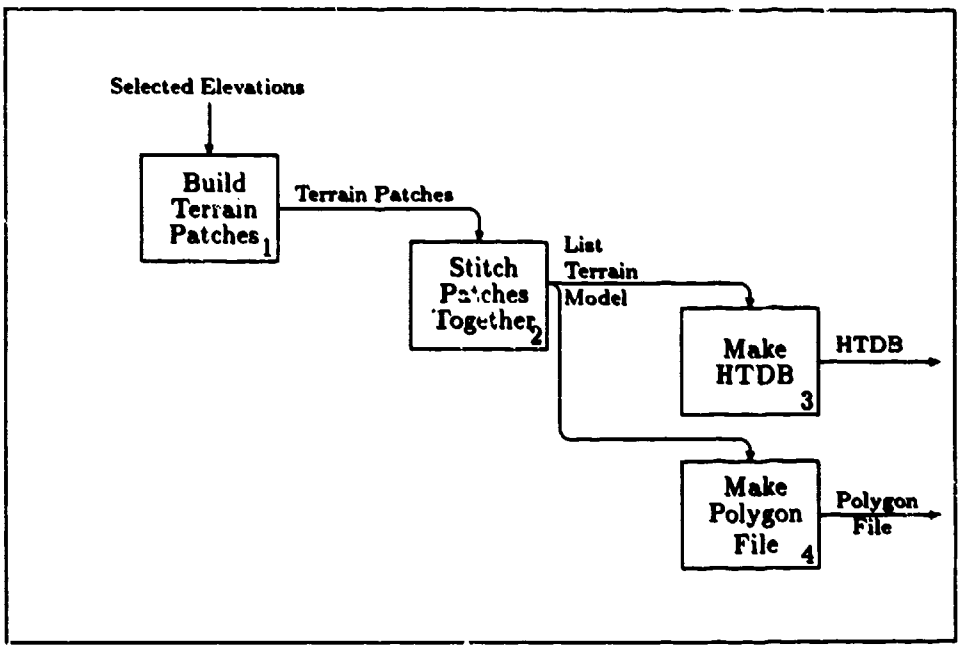


Figure 12. IDEF0 Diagram A2

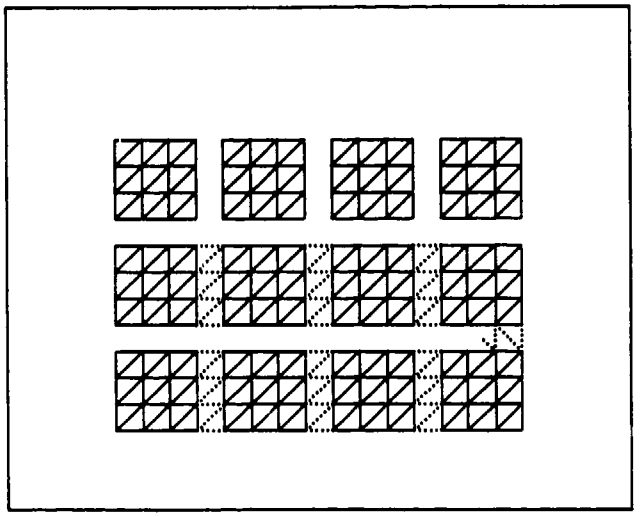


Figure 13. Connecting Terrain Patches

3.5 Terrain Model Database

The process of building the terrain model requires more information than is required to display the terrain model. Since the final form of the terrain model is designed to be accessed and displayed in real-time, it contains only the information necessary for display. An intermediate form of the terrain model, a file of lists, was used to build the terrain model. Multiple levels of detail of a terrain model are incorporated in the hierarchical terrain database by building two or more separate terrain list files, and then adding them to the hierarchical terrain database (for more information, see Appendix A).

3.5.1 List File In order to build a terrain model, the interelations between the vertices (elevations), the edges (lines between two vertices), and the polygons (in this case, the triangles) must be known. The list file, an intermediate form of the terrain model, was used to build the terrain model. This list file consists of a list of vertices, a ring (a circular list) of exterior edges, a list of interior edges, and a list of polygons. Each edge record contains references to the two vertices that form it, and to the one or two polygons that are adjacent to it (exterior edges are adjacent to only one polygon). Each polygon record contains references to the three vertices and the three edges that form it (see Fig 14).

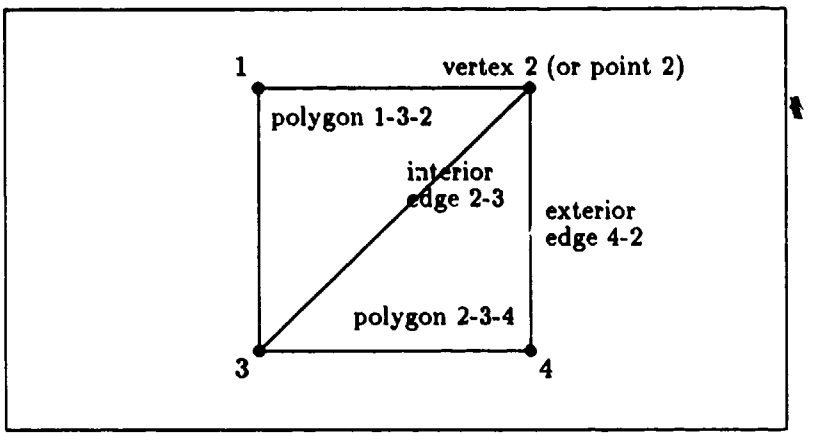


Figure 14. Terrain ...t Objects

The structure of the list file is similar to Bruce Baumgarts *Winged Edge Polyhedron Representation.* In the Winged Edge representation, an edge record contains references to the two vertices that form it, the two polygons that are adjacent to it, and the next and last edges that are part of a ring of edges for each of its two vertices (1) (2:15-18) (see Fig 15). The major differences between the Winged Edge representation and the structures used in this effort are the placement of the ring of edges and the addition of a ring of polygons. Each vertex in the list file contains, in addition to its three dimensional position, references to the ring of edges that meet at the vertex, and the ring of polygons that share the vertex.

3.5.2 Hierarchical, Multiple Level Of Detail File The final form of the terrain model had to allow the user to select a given area of the terrain model at a given level of detail. This was accomplished in this effort by using a combination of a Quadtree and a Spatial kd-tree (see Fig 16).

Trees may be balanced or not balanced. The trees shown in Figures 5, 6, and 7 are not balanced because their leaves are not at the same level. The Spatial Quadtree shown in Figure 16 is balanced so that the data may be accessed using more than one method. The primary method of accessing the data will be to read through the Spatial Quadtree starting at the top (or Root) node, and then at each level, determine which quadrant will hold data that is within the retrieval area. An alternate method of accessing the data would be to determine which leaves are within the retrieval area, compute their node addresses (since

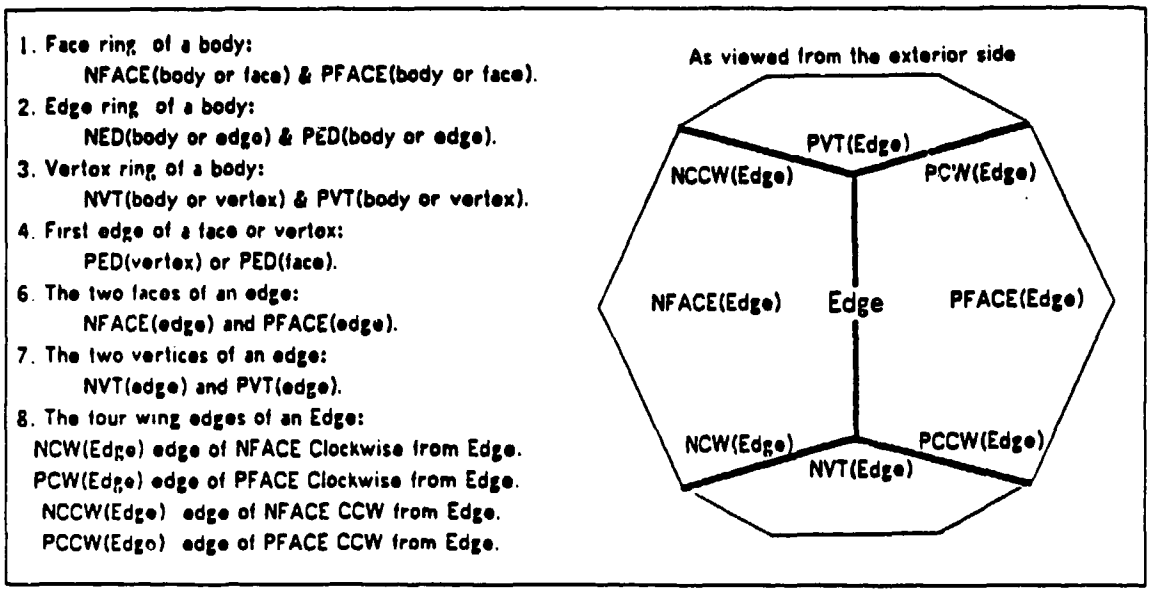


Figure 15. Winged Edge Polyhedron Representation
(1:591)

the tree is balanced, the number of nodes at each level is known and the nodes can be arranged as if they were in an array), and then retrieve the data. The alternate method is faster, but will not retrieve data that extends into the retrieval area but is stored in a nearby leaf (such as triangles B, C, and D in Fig 16, if only the NW quadrant were retrieved).

The hierarchical database is composed of two disk files. The first file contains the Spatial Quadtree, and the second file contains the data records. The Spatial Quadtree is contained in a separate file so that it may be expanded and still provide the alternate method of access that was described in the previous paragraph. It is common knowledge that, when using a disk file, the seek time (the time required to position the read head over a block of data) is greater than the read time for a small amount of data. Therefore, it is better to read a few large blocks of data than to read several small blocks of data. The polygon records are stored in blocks of 4096 bytes (a power of two is a common size for a block of data). A block of 4096 bytes was chosen because it will hold 31 polygon records (experience with the IRIS has shown that an acceptable screen refresh rate is obtained when the number of displayed polygons is less than 900: a screen complexity of 900 polygons would require reading 30 blocks of polygon data). The feature data is stored

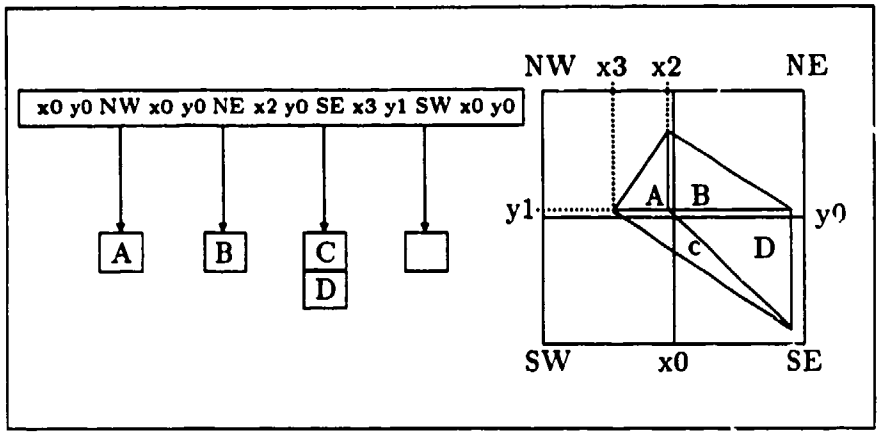


Figure 16. Spatial Quadtree

in blocks of 512 bytes (a block of this size will hold 15 feature records). If a leaf of the Spatial Quadtree contains more than one block of either kind of data, the data block will contain the address of the next data block.

3.6 Adding Cultural Feature Data

There are three categories of cultural features in DMA DFAD data files: point features such as radio towers, linear features such as roads, and area features such as an area covered by trees. The hierarchical database is designed to accommodate either point features or area features, but does not directly accommodate linear features. Area features can be added to the database by finding each polygon that is within the limits of the area feature, and then changing the terrain-type parameter of that polygon. Point features are added to the hierarchical database by simply adding them to the appropriate feature data block (see Fig 17). Two types of point features, radar reflectors and miscellaneous obstructions, were implemented.

3.7 Summary

A method was developed to build a terrain model from non-gridded elevation data. In addition to allowing the use of non-DMA data, this method also makes it possible to build a terrain model from a non-gridded subset of a DMA DTED data file.

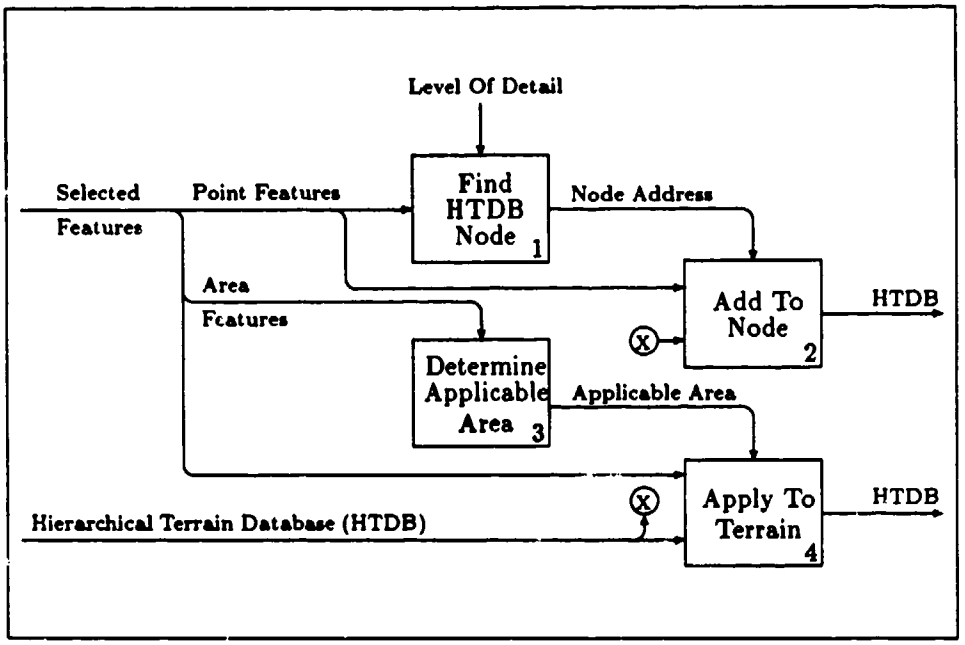


Figure 17. IDEF0 Diagram A4

A hierarchical database containing more than one level of detail of a terrain model and cultural features was developed. This database has not been fully tested because the development of a software system to display the data is beyond the scope of this effort.

The next chapter presents the results of this effort, some conclusions about this effort, and recommendations for further research.

IV. Conclusion

4.1 Results and Conclusions

The overall goal of this effort was to design and implement a software system that would generate a hierarchical database composed of more than one level of detail of a terrain model and cultural objects from digital elevation and cultural data files. This goal was achieved.

In support of this goal, two methods were developed to build a terrain model. The first method will build a terrain model from non-gridded elevation data (for more information, see Chapter 3, Section 3.4, and Appendix B). This method is computationally intensive (it involves a lot of floating point calculations). The accuracy of a floating point number, on the GPX, is only six or seven digits, and fractions are not represented exactly. For example, the number 0.50000 is stored in memory as 0.499983. Because of this problem, the elevations must not be spaced too closely, or errors may occur during the execution of the program. The second method of building a terrain model assumes that the data is gridded. Table 1 shows a comparison between the two methods of building a terrain patch using gridded data. The non-gridded method can be used to build a terrain model from gridded data, but it is not as efficient as the gridded method. Therefore, unless non-gridded data is being used, the simpler gridded method should be used to build the terrain model.

Table 1. Time Comparison

Method	Time (hh:mm)	# of points per patch	# patches
Non-gridded	8:10	64	64
Non-gridded	4:00	256	16
Non-gridded	2:21	1024	4
Gridded	0:45	3721	1

Table 1 also shows a comparison of the times required to build the same terrain model using different sized patches. The terrain patches are built in memory, whereas the connection of the patches is done using random access disk files. As can be seen from the data in Table 1, increasing the size of each patch, and thereby decreasing the number of patches that need to be connected, produces a significant reduction in the time spent to build the terrain model.

Due to time considerations, area and line features were not added to the hierarchical terrain database.

4.2 Recommendations

The software system developed in this effort could be used for a variety of purposes other than to display elevation data. There are also several improvements that could be made to the software, as well as some topics for future research.

4.2.1 Enhancements Currently, the user specifies the number of vertices in each terrain patch. Using this number, the program computes the number of terrain patches. The software system could be modified to compute these values from the total number of vertices and the total area covered by the data so as to optimize the time required to build and connect the terrain patches. The software system could also be enhanced by adding line features and implementing area features.

4.2.2 Current Applications In addition to building a terrain model for use in a simulator environment, the software system developed in this effort could be used to build other kinds of display models. For example, the software system could be used with gravity map data (gravity is not constant over the earths' surface, but varies according to elevation, latitude, and the composition of the material at and under the surface at any one point). The intermediate form of the terrain database (the List file) could be used to compute contour lines (lines of constant elevation).

4.2.3 Future Research There are several possibilities for future research:

- Improve the data reduction algorithm. One method would be to first build a terrain model from a grid of data, and then add the high and low data points.
- Use modern database management techniques. Instead of reading only the data that is to be displayed (from the disk each time), the simulator software (either the user's application software, or the database access routines) could read blocks of data that might be displayed (and store the data in working memory).
- Texture Mapping. Texture mapping could be used to increase the complexity of the display (the polygon records in the HTDB have texture mapping indices built-in for use by future applications).
- Parallel Processing. The data could be displayed using parallel processing. One processor could handle the selection of data to be displayed, while one or more other processors could handle the display processes.

4.3 Summary

A software system was designed and implemented that will generate a hierarchical database composed of more than one level of detail of a terrain model and cultural objects from digital elevation and cultural data files. Two methods were developed to generate a terrain model from elevation data. The first method will build a terrain model from non-gridded elevation data, and the second method will build a terrain model from gridded data. In order to build a hierarchical, multiple level of detail terrain model, two or more rrain models are generated (at different levels of detail), and are then combined.

Appendix A. *User's Manual*

A.1 Sequence of Programs

The following process will produce a hierarchical terrain database using the non-gridded method of building a terrain model:

- Read the data at the highest level-of-detail.
  - RDTAPE
  - RDDTED
  - RDNOAA
- Preprocess the data.
  - TERRAIN1
- Build the terrain patches.
  - TERRAIN2
- Connect the terrain patches.
  - TERRAIN3
- Link the point rings.
  - LINKRING
- Build the hierarchical terrain database.
  - MKTREE
- Read the data at a lower level-of-detail.
  - RDDTED
  - RDNOAA
- Preprocess the data.
  - TERRAIN1
- Build the terrain patches.
  - TERRAIN2

- Connect the terrain patches.
  - TERRAIN3
- Link the point rings.
  - LINKRING
- Add the lower level-of-detail terrain model to the hierarchical terrain database.
  - ADDTREE

The following process will produce a hierarchical terrain database using the gridded method of building a terrain model:

- Read the data at the highest level-of-detail.
  - RDTAPE
  - RDDTED
  - RDNOAA
- Build the terrain model.
  - MKGRID
- Link the point rings.
  - LINKRING
- Build the hierarchical terrain database.
  - MKTREE
- Read the data at a lower level-of-detail.
  - RDDTED
  - RDNOAA
- Build the terrain model.
  - MKGRID
- Link the point rings.
  - LINKRING
- Add the lower level-of-detail terrain model to the hierarchical terrain database.
  - ADDTREE

A.2 Read Data

Two programs are used to read DMA DTED data: RDTAPE and RDDTED. RDTAPE reads a DMA DTED tape and transfers the elevation data to a disk file. RDDTED reads the disk file, selects data, and writes the selected elevations to a disk file as point positions. The following sample run of RDTAPE will read the DMA DTED tape and write the elevation data to the file w115n37.

```
% rdtape > w115n37
```

The following sample run of RDDTED will read the data file $w115n37$ and write a grid of vertices (grid spacing = 10.0 minutes, or about 10.0 miles) to the file $w115n37.pts$. The vertices in the *.pts* file will be in the form of XYZ, where X and Y are in minutes of longitude and latitude relative to the origin 37.0°North, 115.0°West, and Z (the elevation) is in meters.

```
% rddted
enter input filename : w115n37
enter grid size in minutes (MM.mm) : 10.0
grid only (y,n) ? : y
enter output filename : w115n37.pts
enter origin Latitude   (DD.dddd) : 37.0
             Longitude (DDD.dddd) : -115.0
end of rddted
```

The program RDNOAA reads NOAA depth data, selects depth data by area, and writes the depth data to a disk file as point positions. The following sample run will read the data file *loihi*.1, select the data that lies within the area bounded by 11.0°N, 11.2°N, 177.2°W, 177.0°W, and then write the data to the file *loihi.pts*.

```
% rdnoaa
enter input filename: loihi.1

read all or select limits (a,s) ? : s
enter South Latitude   (DD.dddd): 11.0
enter West Longitude  (DDD.dddd): -177.2
enter North Latitude   (DD.dddd): 11.2
enter East Longitude  (DDD.dddd): -177.0

enter output filename (filename.pts): loihi1.pts
end of rdnoaa
```

A.3 Preprocess Elevation Data

The program TERRAIN1 reads a point file, eliminates any duplicate points, and then sorts the points into buckets in preparation for the next program TERRAIN2. The following sample run of TERRAIN1 will read data from the file w115n37.pts, sort the points into buckets with a maximum of 1024 points in each bucket, and then write the points to the file *w*115*n*37.*ptb*.

```
% terrain1
enter max points per terrain patch : 1024
enter point file name (filename.pts; $ to stop) : w115n37.pts
enter point file name (filename.pts; $ to stop) : $

enter point bucket file name (filename.ptb): w115n37.ptb
enter a line of text for the file header (end with $):
500 meter spacing, 115W,37N to 114.75W,37.25N
$
writing file

end of terrain1
```

A.4 Build Patches

The program TERRAIN2 reads the point bucket file produced by TERRAIN1, builds a terrain patch from the points in each point bucket, and produces a terrain list file. The following sample run of TERRAIN2 will read the file *w115n37.ptb*, build the terrain patches, and write the terrain patches to the file *w115n37.lst*.

```
% terrain2
enter name of point bucket file (filename.ptb): w115n37.ptb
enter output list file name (filename.lst): w115n37.lst
end of terrain2
```

A.5 Connect Patches

The program TERRAIN3 reads the list file produced by TERRAIN2 and connects the terrain patches to form one terrain patch. The following sample run of TERRAIN3 will read the file *w115n37.lst*, connect the terrain patches, and then update the file *w115n37.lst*.

```
% terrain3
enter terrain list file name (filename.lst): w115n37.lst
end of terrain3
```

A.6 Build Terrain From Gridded Data

The program MKGRID reads the point file produced by RDDTED, and creates a list file. The following sample run of MKGRID will read the file *w115n37.pts*, build a terrain model from the data, and then write the terrain model to the file *w115n37.lst*.

```
% mkgrid
enter point file name (filename.pts): w115n37.pts
origin of .pts file is:
    latitude = 37.000000
   longitude = -115.000000

enter origin of output file in degrees (114W = -114)
          latitude    (-90.0 .. 90.0): 37.0
          longitude [-180.0 .. 180.0]: -115.0

enter output file name (filename.lst): w115n37.lst

enter a line of text for the file header (end with $):
grid spacing 500 meters  37N,115W to 37.25N,114.75W
$

end of mkgrid
```

A.7 Link Point Rings

The program LINKRING reads a terrain list file and creates a ring list of edges and polygons associated with each point.

```
% linkring
enter name of list file (filename.lst): w115n37.lst
linking pt rings
making polys ccw and computing normals
end of linkring
```

A.8 Make Hierarchical Model

The program MKTREE reads a terrain list file and makes a hierarchical terrain database (HTDB). The HTDB consists of two files: a file containing the search tree, and a file containing the polygon data.

```
% mktree
enter list filename (no extensions): w115n37

file w115n37.lst opened
origin of list file is:
latitude is =  37.000000
longitude is = -115.000000

enter new origin
latitude in degrees north   (-90.0 .. 90.0) : 37.5
longitude in degrees east [-180.0 .. 180.0] : -114.5

build all or select limits (a,s)?: s
enter North Latitude   (DD.dddd) : 35.5
      South Latitude   (DD.dddd) : 38.5
       West Longitude (DDD.dddd) : -117.0
       East Longitude (DDD.dddd) : -114.0

enter tree filename (no extensions): redflag
file redflag.tre opened
file redflag.bkt opened

enter number of levels in tree: 4
end of mktree
```

A.9 Add Lower Level-Of-Detail to Hierarchical Model

The program ADDTREE reads a terrain list file and makes a hierarchical terrain database (HTDB). The HTDB consists of two files: a file containing the search tree, and a file containing the polygon data.

```
% addtree
enter list filename (no extensions): w115n37a

file w115n37a.lst opened
origin of list file is:
latitude  = 37.000000
longitude = -115.000000

enter tree filename (no extensions): redflag
file redflag.tre opened
file redflag.bkt opened

origin of tree file is:
latitude  = 37.500000
longitude = -114.500000

add all or select limits (a,s)?: s
enter North Latitude   (DD.dddd) : 35.5
      South Latitude   (DD.dddd) : 38.5
       West Longitude (DDD.dddd) : -117.0
       East Longitude (DDD.dddd) : -114.0

enter level of new data: 3
end of addtree
```

A.10 Make Polygon File

MKVUFILE reads a terrain list file and builds a polygon file that may be viewed with the VIEW program.

```
% mkvufile

enter list filename (filename.lst): w115n37.lst

 originLat = 37.0
 originLong = -115.0

view all or select limits (a,s)?: s
enter North Latitude   (DD.dddd): 37.6
      South Latitude   (DD.dddd): 37.4
       West Longitude (DDD.dddd): -114.6
       East Longitude (DDD.dddd): -114.4

enter viewit filename (filename.new): w115n37.new

enter origin of output file in degrees
          latitude    (-90.0 .. 90.0): 37.5
          longitude [-180.0 .. 180.0]: -114.5

enter units to convert to (m ft none): ft

end of mkvufile
```

MKVUTREE reads a hierarchical terrain model file and builds a polygon file that may be viewed with the VIEW program.

```
% mkvutree
enter the filename (no extensions): redflag
file redflag.tre opened
file redflag.bkt opened

 originLat = 37.0
 originLong = -115.0

view all or select limits (a,s)?: s
enter North Latitude   (DD.dddd): 37.6
      South Latitude   (DD.dddd): 37.4
       West Longitude (DDD.dddd): -114.6
       East Longitude (DDD.dddd): -114.4

enter viewit filename (filename.new): w115n37.new

enter origin of output file in degrees
          latitude    (-90.0 .. 90.0): 37.5
          longitude [-180.0 .. 180.0]: -114.5

enter units to convert to (m ft none): ft

end of mkvutree
```

Appendix B. *Algorithms*

This appendix outlines several of the algorithms that were used to build the terrain model from elevation points.

B.1 Determine Point Position

Before a point can be connected to the terrain patch, its position relative to the terrain patch must be determined. A point can be on an edge (either an exterior edge, or an interior edge), inside one of the triangles in the terrain patch, or exterior to the terrain patch (see Fig 18). A points' position is determined by finding the closest edge. If the distance to the closest edge is zero, then the point is on the edge (see Fig 18, points a and b). If the distance is not zero, and the closest edge is an interior edge, then the point is inside one of the triangles adjacent to the closest edge (see Fig 18, point c). If the closest edge is an exterior edge, and the closest point on that edge is an end point, then the point is exterior to the terrain patch (see Fig 18, point d). Otherwise, the point might be exterior to the terrain patch, or inside the polygon adjacent to the closest edge (see Fig 18, points e and f).

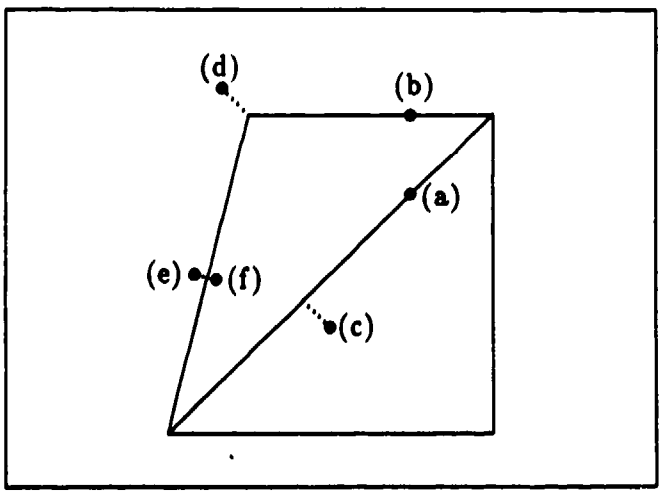
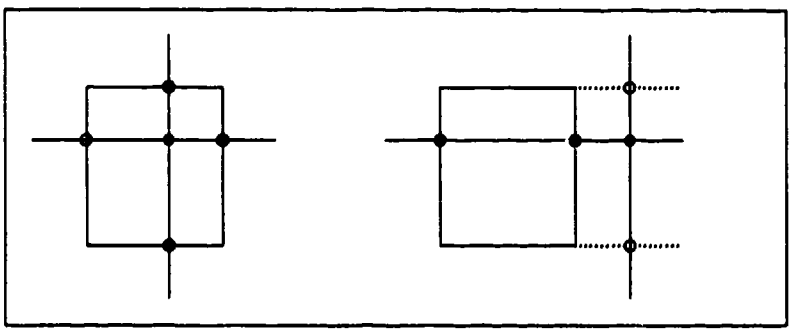


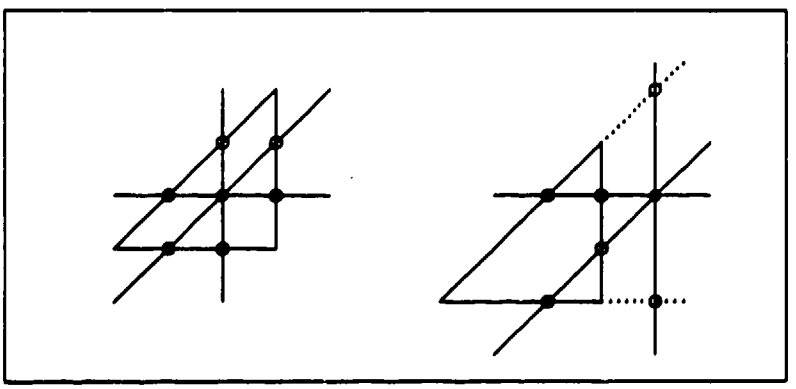
Figure 18. Determine Point Position

B.2 Inside/Outside Polygon Algorithm

The algorithm used to determine whether a point is inside a triangle, or outside a triangle, is similar to the algorithm commonly used to determine if a point is inside or outside a rectangular area. A point is inside a rectangular area if the lines drawn through the point parallel to each side intercept the sides of the rectangle. If any of the interceptions are not on the sides of the rectangle, then the point is outside of the rectangle (see Fig 19 (b)). To determine whether a point is inside a triangle, lines are drawn through the point parallel to each side of the triangle. If all of these lines intercept the sides of the triangle, then the point is inside the triangle. If any of the interceptions are outside of the triangle, then the point is outside of the triangle (see Fig 19 (a)).



(a) Rectangles

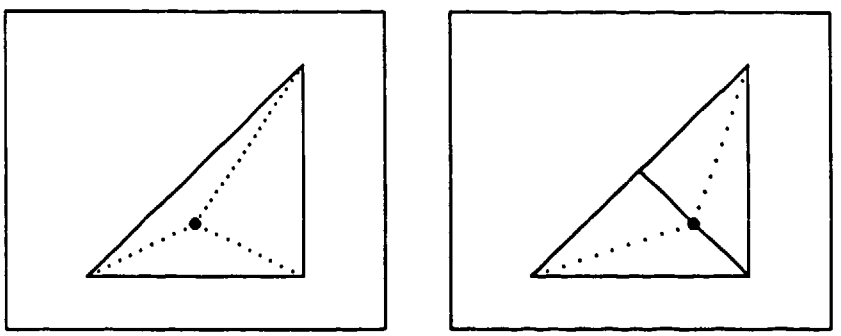


(b) Triangles

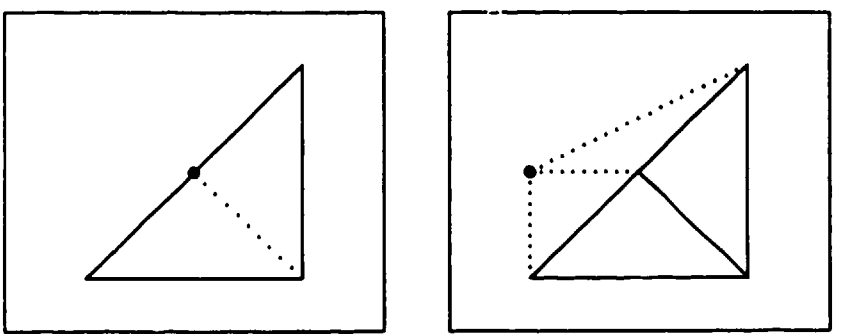
Figure 19. Inside/Outside Polygon Algorithm

B.3 Connecting Points

Once a points' position relative to the terrain patch has been determined, it can be connected to the terrain patch. If it is inside one of the triangles (see Fig 20a), then it is connected to the three vertices of the containing triangle, adding three new edges, and dividing the containing triangle into three triangles. If it is on one of the interior edges (see Fig 20b), it is connected to the third point of each of the two adjacent triangles, adding two new edges, dividing the containing edge into two edges, and dividing each of the adjacent triangles into two triangles. If the new point is on one of the exterior edges (see Fig 20c), then it is connected to the third point of the adjacent triangle, adding one new edge, dividing the containing edge into two edges, and dividing the adjacent triangle into two triangles. If the new point is exterior to the terrain patch (see Fig 20d), then it is connected to an exterior point if the new edge does not cross any other edge.



(a) Pt Inside Triangle (b) On Interior Edge



(c) On Exterior Edge (d) Outside Patch

Figure 20. Connecting Points

B.4 Adjusting Polygons

During the process of building a terrain patch, a point is connected to every other point if the new edge formed does not cross any other edge. This will guarantee that the exterior edge of the terrain patch is convex (a requirement for the algorithm that connects the patches). It will also form long, thin triangles (see Fig 21). An edge formed by connecting two points that are close to each other is more likely to reflect the actual contours of the terrain than an edge formed by connecting two points that are far apart. After the terrain patch is built, each pair of adjacent triangles is examined to determine whether the edge that they share should be adjusted (see Fig 22).

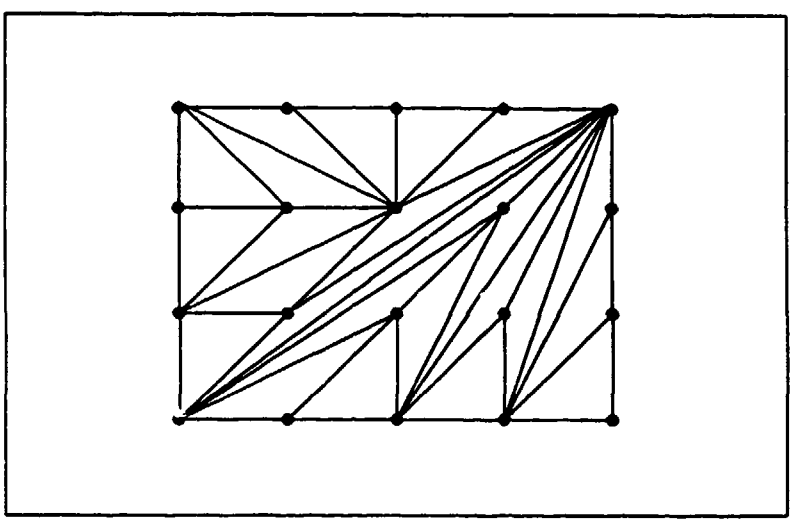
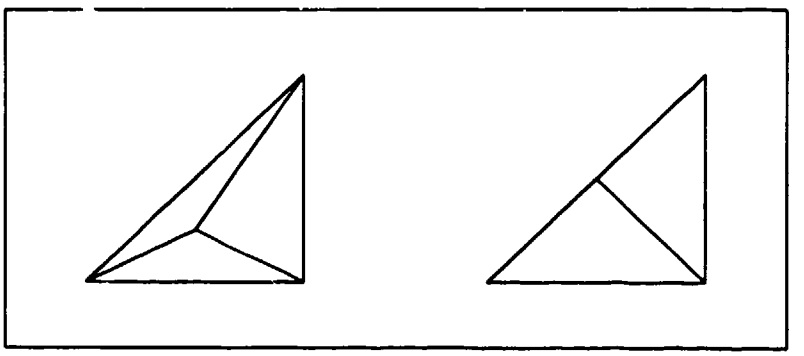
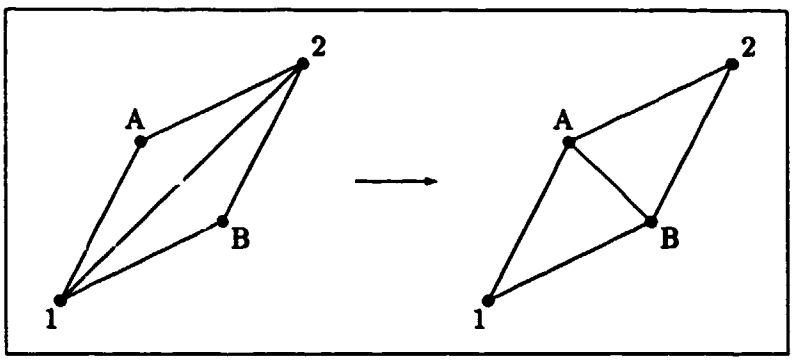


Figure 21. Long, Thin Triangles



(a) Non-Adjustable Triangles



(b) Adjustable Triangles

Figure 22. Adjusting Triangles

Appendix C. *DMA Data*

DMA digital data is distribution limited. This means that its distribution is limited to U.S. Government agencies and their contractors. In order to obtain DMA DTED or DFAD data, send a request to:

AFIA/INTB
Bolling AFB
Washington, D.C. 20332

The letter should contain the reason for requesting the data, the area of the data, and whether 1600 bpi, or 800 bpi magnetic tapes are required.

Requests from other than U.S. Government agencies and their contractors should be sent to:

Director
DMA Aerospace Center
3200 South Second Street
St. Louis, Missouri 63118 – 3399
ATTN: PR

Appendix D. *NOAA Data*

The National Oceanic and Atmospheric Administration (NOAA), produces several digital data files that may be purchased by the public. In order to obtain aeronautical data (such as NAVAID Data which is produced in cooperation with the Federal Aviation Administration and includes all aeronautical navigation aids within the continental United States, Alaska, Hawaii, Puerto Rico, and the U. S. Virgin Islands, plus bordering facilities in Canada, Mexico, and the Atlantic and Pacific areas), send a request to:

NOAA Distribution Branch
N/CG33
National Ocean Service
6501 Lafayette Avenue
Riverdale, MD 20737

In order to obtain nautical data (such as aids to navigation, digital shoreline data, or other chart data), send a request to:

Chief, Mapping and Charting Branch
C&GS
N/CG22
6001 Executive Blvd.
Rockville, MD 20852

Bibliography

1. Baumgart, Bruce G. "A Polyhedron Representation For Computer Vision," *AFIPS Conference Proceedings*. 589-596. Anaheim, California: AFIPS Press, 1975.
2. Baumgart, Bruce G. *Geometric Modeling For Computer Vision*. MS Thesis STAN-CS-74-463. Computer Science Department, Stanford University, Stanford, CA, October 1974 (AD-A002 261).
3. Clark, James H. "Hierarchical Geometric Models for Visible Surface Algorithms," *Communications of the ACM*: 547-554 (October 1976).
4. Costenbader, J. L. "CIG Data Bases in an Instance: Bits and Pieces", (AD-P004 318), *The Image III Conference Proceedings Held at Phoenix, Arizona on 30 May - 1 June 1984*, (Proceedings Paper), 151-163, Cameron Station, Alexandria, Virginia: Defense Technical Information Center, (AD-A148 636).
5. Kanko, Mark A. *Geometric Modeling of Flight Information For Graphical Cockpit Display*. MS thesis, AFIT/GCE/ENG/87D-6. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1987 (AD-A190 484).
6. MacDougal, Paul D. *Generation and Management of Object Description Hierarchies For The Simplification of Image Generation*. Report to the National Science Foundation, Division of Mathematical and Computer Sciences, Grant Number DCR-8304185, Ohio State University, Columbus, OH, August 1984.
7. Mirante, Anthony and Nicholas Weingarten. "The Radial Sweep Algorithm for Constructing Triangulated Irregular Networks," *IEEE Computer Graphics & Applications*: 11-21 (May 1982).
8. U.S. Department of Commerce, National Oceanic and Atmospheric Administration, National Ocean Survey. *Cook Inlet, Fire Island To Goose Creek*. Chart 16664, 17th ed. Washington, D.C.: April 21, 1979.
9. Oliver, Michael R. and others. *Interactive, Networked, Moving Platform Simulators*. Naval Postgraduate School, Monterey, California, February 1988.
10. Ooi, Beng Chin and others. "Spatial kd-Tree: An Indexing Mechanism for Spatial Database," *Proceedings of the 11th Annual International Computer Software & Applications Conference*. 433-438. Tokyo, Japan: IEEE Press, 1987.
11. Rebo, Robert K. *A Helmet-Mounted Virtual Environment Display System*. MS Thesis, AFIT/GCS/ENG/88D-17. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December, 1988.
12. Ross, Harry D. *Analysis and Preparation of a Digital Terrain Data Base for Flight Simulator Use*. MS thesis, AFIT/GEO/MA/81D-1. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, November 1981 (AD-A115 547).
13. Samet, Hanan. "The Quadtree and Related Hierarchical Data Structures," *Computing Surveys, 16*: 187-260 (June 1984).

Vita

Lieutenant LeeAnne Roberts was born on November 11, 1950 in Chico, California. She graduated from Hiram Johnson Sr. High School in Sacramento, California in 1968 and entered California State University at Sacramento, California. She graduated from CSUS in 1972 with a Bachelor of Arts in Physics and Applied Mathematics and entered the University of California at Riverside, California. She graduated from UCR in 1974 with a Master of Arts in Physics. Lieutenant Roberts entered the National Oceanic and Atmospheric Administration Commissioned Corps in April of 1977. She served aboard the NOAA Ship *Fairweather* from 1977 through 1979. She was assigned to the Marine Data Systems Project, Rockville, MD, from 1979 through 1984. She then served aboard the NOAA Ship *Whiting* as 4th Officer from 1984 through 1985. She was assigned to the National Meteorological Center for 18 months before entering the School of Engineering, Air Force Institute of Technology in June 1987.

Permanent address: c/o NOAA Commissioned
Personnel
National Oceanic &
Atmospheric Administration
Rockville, MD 20852

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/88D-18	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AAMRL 8b. OFFICE SYMBOL (If applicable) HEA	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) See Block 19	
12. PERSONAL AUTHOR(S) LeeAnne Roberts, M.A., B.A., LT, NOAA	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM _____ TO _____ 14. DATE OF REPORT (Year, Month, Day) 1988 December 15. PAGE COUNT 52	
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
08	02		Terrain Models, Digital Maps, Topographic Maps, Computer graphics, Hierarchical Data Structures

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: A SOFTWARE SYSTEM TO CREATE A HIERARCHICAL, MULTIPLE LEVEL OF DETAIL TERRAIN MODEL

Thesis Advisor: Phil Amburn, Major, USAF
Professor of Computer Systems

Abstract: See reverse

Approved for public release IAW AFR 190-1
10 Jan. 1989

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Phil Amburn, Major, USAF	22b. TELEPHONE (Include Area Code) (513) 255-3576 22c. OFFICE SYMBOL AFIT/ENG

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Block 19. Abstract

The purpose of this effort was to design and implement a software system that will generate a hierarchical database composed of more than one level of detail of a terrain model and cultural objects from digital elevation and cultural data files.

The software system designed and implemented in this effort was written in the C language and implemented on a Digital Equipment GPX Workstation and on a Silicon Graphics Iris 3130 Workstation.

Although the primary source of elevation data used in this effort was the Defense Mapping Agencies' (DMA) Digital Terrain Elevation Data (DTED), a method was developed to build a terrain model from non-gridded elevation data. In addition to allowing the use of non-DMA data, this method also makes it possible to build a terrain model from a non-gridded subset of a DMA DTED data file.

Two database formats were designed and implemented in this effort. The first database (which was used to generate the terrain model) contains lists of the vertices, exterior edges, interior edges, and polygons (and their interrelations) that form the terrain model. The second database contains polygons (that form the terrain model) and cultural features (such as radio towers) at one or more levels of detail, and a search tree (so that data in a particular area may be retrieved quickly).

The list database may be used to generate contours, or non-real-time images of terrain. The hierarchical database may be used in a simulator environment.