Polygon Databases

Is a Digital Terrain Elevation Database (DTED) really a Polygon Database?

Jed Margolin

Short answer: If a Digital Terrain Elevation Database (DTED) is used to produce polygons, then it is a polygon database.

<u>1.</u> A Polygon Database can be stored in several ways.

<u>A.</u> Each Polygon is defined by its points (X,Y,Z). "Z" is the elevation.

This is not an efficient way to store polygons because points may be shared by other polygons. The duplication of points not only uses more storage space, it wastes processing time during runtime because the duplicated points must also be transformed.

It is also an inefficient use of storage in view of how the points in a DTED are stored.

<u>B.</u> The DTED stores points in an array representing rows and columns of elevations. Only the elevations need to be stored in the array because the position of the point (X,Y) is derived from its position in the array. It is derived from its position in the array by applying the appropriate scaling and offset.

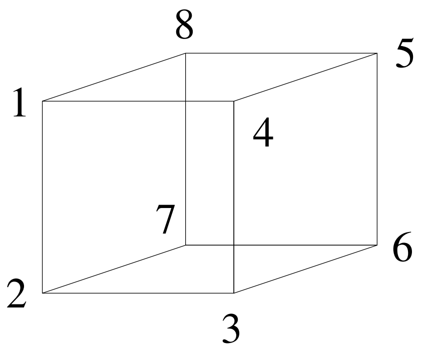
Note that the actual storage medium doesn't know it is storing an array. The storage medium is only storing a series of bytes. It is the program that reads the data that knows how long a row is so it can divide it into an array of rows and columns .

Thus, while Storage Method 1 (above) stores X,Y,Z for each point, the DTED only needs to store Z.

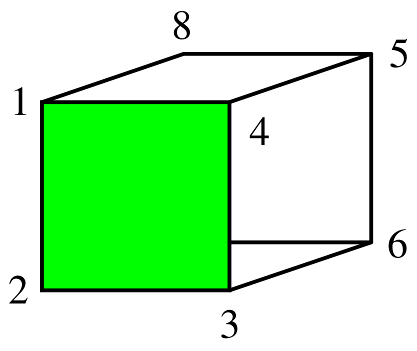
The next part is to have a list that describes how the points are connected.

Let's use a cube as an example.

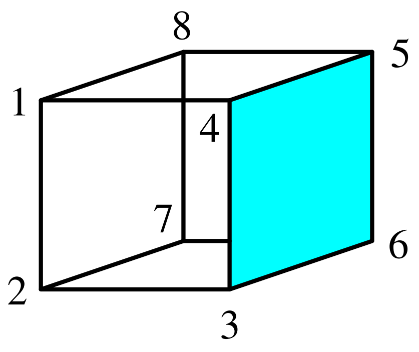
A cube has 8 points and 6 faces.



The front face uses points 1, 2, 3, and 4.



The next face uses points 3,4,5,6.



Points 3 and 4 are used in both faces (polygons). The other four faces are done in a similar fashion.

Indeed, each point in a cube is used by three faces.

Storing each point in each polygon would use three times as many points as is necessary and, since each point would need to be transformed during runtime, would require three times as much processing.

At Atari, our polygon database for each object consisted of a ***Point List*** and a ***Face List***.

The Pont List contained the points making up the polygons used in an object and the Face List described which points were used in each polygon. The Face List also described how they were connected since polygons were always described in a clockwise manner.

As a result, the polygon database for a cube contained only 8 points to be stored and transformed and used by the 6 faces (polygons).

Storing the polygons in Method 1 above would require storing and transforming 24 points (Four for each of the six faces.)

A Digital Terrain Elevation Database is a Point List.

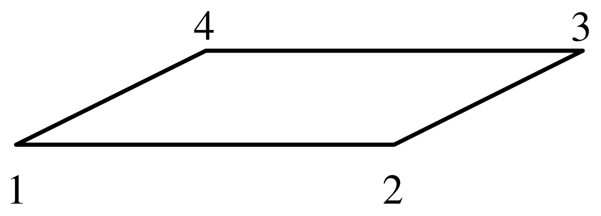
The manner in which the polygons are connected is generally very simple and can be either a simple Face List or an algorithm.

Together, the Digital Terrain Elevation Database and its Face List (or Algorithm) constitute a Polygon Database.

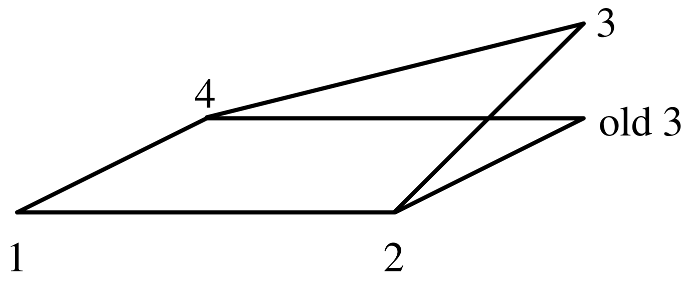
The proof is to look at the display. If there are polygons on the display, then the Digital Terrain Elevation Database is a Polygon Database.

A side issue when using a Digital Terrain Elevation Database is that using the four intersections of two rows and two columns is not sufficient because a polygon made of four points may not be coplanar (in the same plane).

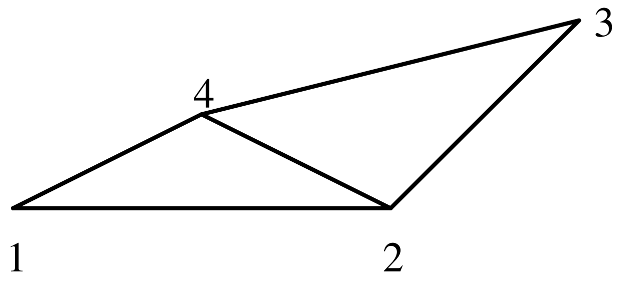
In this example the points are coplanar.



In the following example, they are not.

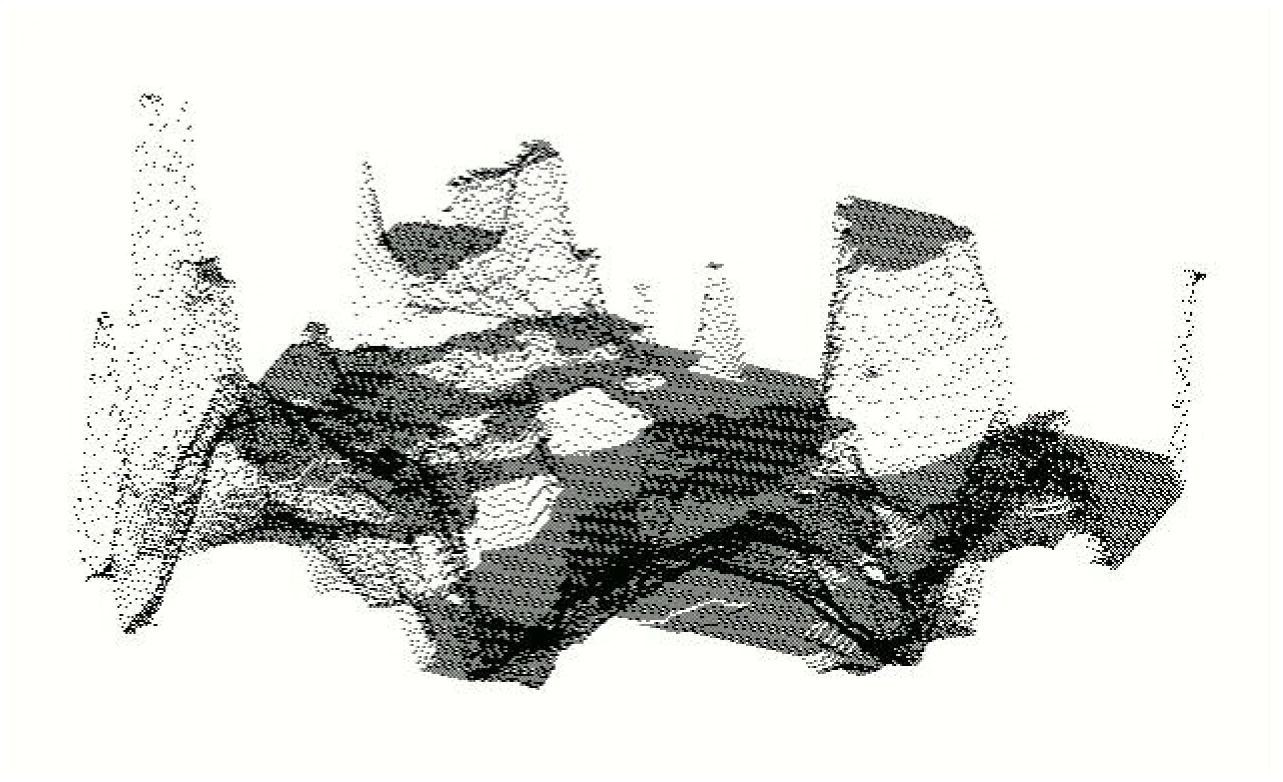


The standard technique is to divide the square formed by rows and columns into two triangles because, not only are three points guaranteed to be in the same plane, the three points define the plane.

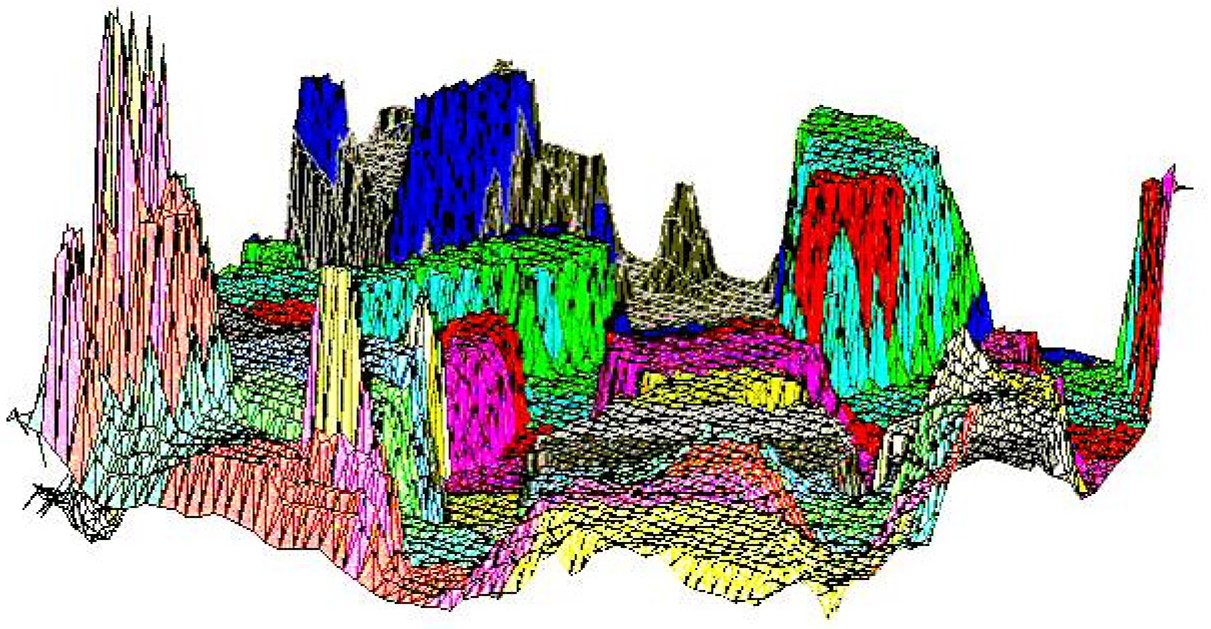


Although the Face List is admittedly simple, it is a Face List which, with the Digital Terrain Elevation Database, forms a Polygon Database.

The alternative to displaying the Digital Terrain Elevation Database as polygons is to display only the points in the database. If you display only points there is no way to remove "hidden points" because there are no surfaces to test them against. (Things can only be hidden behind surfaces.) The result is a jumble which looks like this (the only useful features are the highest peaks):



This same scene rendered in polygons. (The polygons are crude because I had only a few colors to work with and there is no clipping; only polygon sorting):



2. At the time my invention described in ‘073 was made, the prior art showed:

a. 2D moving maps;
b. 2D moving maps with 2D rotation that was faked by reading out the memory in particular sequences;
c. 2D terrain with apparent 3D effects produced by, again, “faking it.”

The 3D math I had developed and which was used in a number of Atari’s coin-operated games (3D simulator games like Battlezone, Star Wars, Hard Drivin’, Stun Runner, Race Drivin’, and Steel Talons) was mathematically accurate and efficient and could be run on cost-effective hardware.

These algorithms are taught in the ‘073 patent.

Jed Margolin
Reno, NV
November 15, 2007
Revised 7/29/2008 (added point and polygon pictures)