A STEP TOWARDS INTERACTIVE DISPLAYS OF DIGITAL ELEVATION MODELS

Thomas K. Poiker
Lori A. Griswold
Simon Fraser University
Canada

BIOGRAPHICAL SKETCH

Thomas K. Poiker (formerly Peucker) studied at several West German Universities, receiving his PhD (Geography, 1966) from the University of Heidelberg. He spent several post-doctoral periods at Harvard University and the Universities of Washington and Maryland. He is presently Professor of Geography and Computing Science at Simon Fraser University where he has taught since 1967. He is also Affiliate Professor at the University of Washington, Editor-in-Chief of Geo-Processing (Elsevier, Amsterdam), Principal of GeoProcessing Associates, Inc. and Senior Associate with Tomlinson Associates.

Lori Ann Griswold received her B.A. in Geography (emphasis cartography) from the State University of New York at Buffalo and is currently M.A. candidate (emphasis Computer Mapping) at Simon Fraser University.

ABSTRACT

Many full-fledged Digital Elevation Models (DEMs) are in existence and the number of programs that display digital surfaces are counted in the hundreds. Surprisingly, however, most of them do nothing more than contour the surfaces or display them by parallel profiles with the hidden portions removed. Few of them allow to add other information (road, houses, etc) and none is known to allow adding or deleting information by pointing at the 3-D display. The major bottleneck for the inclusion of higher sophistication, especially interactiveness, into the display of surfaces is a conceptional one: the transformation from one coordinate system to another is considered one-directional and visibility is a quantity computed during the conversion and lost at the same time. The solution is to make visibility a property of the spatial unit (point, square, triangle). The paper describes the conceptual aspects of this idea and then applies them to the display of three-dimensional surfaces. A few new and potentially powerful ways of computing visibility are demonstrated.

INTRODUCTION

For over two decades, researchers in different disciplines have worked in the area of Digital Elevation Models (DEMs). Many programs have been developed for the contouring of surfaces (Gold, 1984), three-dimensional display of

different types (Peucker et al, 1975), slope analysis, highway planning, land-use analysis (Woodham, 1985), etc. Extensive research has been done on the types of data structures that work best for operations on surfaces (Peucker, 1978) and much care has been exercised to make the displays as pleasing as possible (Wolters, 1969).

Considering the extent of research and development in the field of DEMs, it is surprising how unsophisticated the results still are. The level of sophistication that we mean here is related to the graphic complexity of the results, not to their graphic quality.

This allegation has to be explained more. Most of the work in the field of Digital Elevation Models has been for very well defined results of relatively simple structure. In most cases, the computer method is of a variation that looks primitive in comparison to the manual equivalent.

A good example is that of the perspective display of surfaces, or block-diagrams as they are called in the field of manual production. All programs but a few use vertical profiles with the removal of hidden lines. A handful display contours and even less allow the combination of these two graphic primitives or support the display of random features on the surface. If we compare this to the sophisticated and artistically magnificent block-diagrams whose production has almost vanished by now, the computer products can only be called pitiful.

COMPUTER GRAPHICS

Whereas in the early stages of the computer's development, computer cartography pioneered several of the concepts and techniques in computer graphics, it has fallen behind in the pace of development in recent years compared with it's sister discipline. There are several fields in graphics that the mapping community has largely left untouched over the last years.

A good example for this is Computer-Aided Design (CAD). Whereas cartography has participated intensively in the development of the early stages of CAD systems, it has stayed with it's simpler aspects and left the more complex developments to computer graphics.

Like three-dimensional modelling. In computer mapping, the treatment of three-dimensional surfaces has been so far almost entirely for the purpose of display, very little for analysis even though the demand always seemed to be there and many people had promised different capabilities for a long time.

An example may clarify what we mean: Once a three-dimensional display of a surface (a block diagram) has been produced, one might want to perform one or more of the following functions:

(a) Shade all those areas that have steep slopes.
(b) Draw another data set (e.g. roads) on the surface, only showing the visible portions, of course.

(c) Point at the screen (through a tablet, etc) and request the position (i.e. the data-base coordinates) of the point that is indicated.
(d) Draw on the surface (again through a tablet or the like) additional graphic information.
(e) Slowly rotate the surface without recomputing the visibility of every element.

DIGITAL ELEVATION MODELS

With most DEMs that are presently available, the visibility of the elements of a surface is computed during the production of the display and the information is lost once the display is completed. The most frequent method computes one profile after the other, starting with the closest to the viewer, and testing each subsequent profile against the horizon which is the sum of the highest segments of all previous profiles. As the system passes a spatial unit of the surface (a square or a triangle), all information that has come into the picture at that point has to be ready and is displayed right then.

With this method, none of the above functions can be performed. For the first two, the system has to remember the visibility for later use, for the next two the system has to be able to inverse the projection function to find all the points on the surface that have the given coordinates on the screen, and then find the one point that is visible. And the last function will work if we can compute the visibility of an element for more than one viewpoint at a time.

In our continuing research in the area of Digital Elevation Models, we have developed a concept that allows the implementation of all these functions and have developed an algorithm for the visibility of surfaces that supports these functions.

THE CONCEPT

The concept is very simple: Instead of preparing the complete data base for the display of each visible element, make the visibility a property of the database and deal with it as with all other variables in the base. In other words, the computation of the visibility is separated from the production of the display.

Even though the separation of visibility and display into two processes is fairly obvious, it leads to a chain of other considerations. If these two functions can be separated , why not separate the functions into even more processes?

As will be shown later, part of the computation of the visibility of a unit is the computation of the "normal vector", i.e. the vector that stands perpendicular to the surface at the unit. The visibility is then determined by computing the angle between the view axis and the normal vector. If that angle is less than 90°, the unit is visible. If the

normal vector is stored as part of the data base, half of the visibility test for any view axis is done.

In the following it will be shown how these ideas were implemented in our DEM for the modelling of surfaces. First, we describe a new algorithm for the computation of visibility on a surface and then we discuss its application for the different functions described above.

THE TIN SYSTEM

The framework for developing the visibility is the Triangular Irregular Network (TIN) (Peucker, et al, 1977). The TIN system is based on irregularly distributed points that are first triangulated into a series of connecting facets. The triangulation algorithm allows the generalisation of the surface by selecting points based on Cheybyshev's approximation criterion.

The facets are always triangles which are defined by their three vertices and three edges. In the main data structure, the vertices are stored with their geometric coordinates and pointers which represent the labels of those points which are at the endpoints of the connecting edges. The sequence of the edges around a point is always clockwise. Triangles can therefore be identified by finding one vertex, determining one of its neighbors (pointers) and the next pointer in the pointer list. A secondary data structure is based on the triangles and contains pointers to the three vertices and the three neighboring triangles.

The DEM was initially implemented on an IBM mainframe and contained, besides the basic structures and the triangulation algorithm, programs for contouring, block-diagrams, visibility charts, relief shading, inclined contours, slope map, and random drawings on the surface.

In 1978, a third generation of TIN was initiated on microcomputers (the first generation was a simple system developed by D. Cochrane (1974) as his Master's Thesis). This new system is termed Micro-TIN (uTIN) and operates under UCSD Pascal (Poiker, 1982).

VISIBILITY

As has been said before, the computation of the visibility takes place in stages. The major steps involve the following computations:

(1) "Exposure".
(2) "View vector".
(3) "Potential visibility".
(4) "Potential horizons"
(5) "Display space".
(6) "Visible horizons".
(7) "Visible triangles".

Several of these levels supply data that can be entered in the original data base. Here is a description of the different stages.

1. The "exposure" is the vector that stands perpendicular to a triangle, with the origin in the center of the triangle. This line is also called the normal vector. In vector notation it is the cross-product of the co-ordinates of the triangle's three vertices. Stored with the other data for each triangle, it serves the computation of the visibility from any observation point as well as for relief shading, slope analysis, etc.

2. In the case where parallel projection, relief shading, sun intensity, etc., are computed, the "view vector" is given for all triangles as one value. If the observation point is a finite distance from the surface, it has to be computed individually for every triangle. The view vector is computed in vector form.

3. When the view vector and the normal vector are given for a triangle, the "potential visibility" is positive if the angle between the two is less than 90 degrees. That means that the triangle will be visible unless some other triangles lie in front. The cosine of the angle between two vectors is their dot-product. If the cosine is less than one, the triangle is visible.

4. If two triangles are neighbors and one of them is visible and the other is not, the separating edge is part of a "horizon" or a valley. We call them "potential horizons" since all final horizons will come from this set of edges. Valleys can be eliminated by testing the valley-edge against both third vertices or kept for the next step where they would fall out automatically.

5. Up to this point, all computations have been undertaken in what is called "data space", i.e. using the co-ordinates as they are given in the original data base. Now the coordinates have to be converted from "data space" to "display space", i.e. into the (two-dimensional) coordinates of the final display. But remember, the data set that has to be converted is largely reduced compared to the original data set: only approximately half of all tiangles are potentially visible, all others can be eliminated at this stage. It should be noted here that each triangle has to have a unique label which has to be carried over into the display file. The reason for this extra parameter will become clear later.

6. When an artist draws a landscape with mountains, he/she usually starts with the horizons and then fills in the rest (David H. Douglas has incorporated this function into his block-diagram program and finds the results very appealing. Douglas 1971). To do this in our system, we first have to eliminate those horizons that lie behind and below others. This implies a trivial intersection of all horizons and the deletion of those segments that are hidden by other segments. Of course, all valleys would be eliminated at this stage. We are left with all "visible horizons".

7. Next, all triangles are tested against these horizons. In doing this, the triangles will fall into three groups:
   (a) Triangular facets that are totally visible.
   (b) Triangular facets that are partially visible.
   (c) Triangular facets that are totally non-visible.

If the visibility of a triangle is being stored in the original data structure, each triangle is labelled appropriately but at the same time a display file, i.e. a list of all the visible triangles, is established. The triangles that are partially visible are split into smaller triangles and the visible sub-triangles are added to a special list in the original data base and entered as visible triangles into the display file. This means that the original data base is not enlarged each time a display is computed.

This last process can be done in a more elegant way: Each string of edges that form one continuous horizon throws a shadow that is also a continuous string of lines and the two together create a closed polygon. This method seems to be faster, at least for the determination of shadow areas and the so-called radar maps. However, care has to be taken with shadow areas that have visible islands, i.e., mountains that become visible behind other mountains.

USE OF THE VISIBILITY

After the computation of the visibility of a surface, we are left with two data files: One is the augmented original file, augmented by the visibility information and two extra files, one giving a list of all fully visible triangle and the other containing those triangles that are broken out of partially visible triangles. The other file is the display file which contains the visible triangles in display coordinates. Both files are so structured that for any element (triangle, vertex, edge) in either file its complementary element in the other file can be accessed without long comparisons. The strong relationship between the two files is essential for the success of any further use of these files.

If we now return to the five functions that we want to perform with our DEM, we can summarize them into three different groups.
   (a) Convert from the original data base to the display file.
   (b) Convert from the display file to the original data base.
   (c) Change the external condition slightly.

An interesting aspect of this approach is that most operations can be performed with much less overhead than one might initially expect. For example, there is no need to develop any transformation parameters for the conversion from the display file to the original data file and the slight change of the observation point can be performed with many of the data from the previous view.

ADDITIONS TO THE DISPLAY

Once the basic display file has been developed, any additional computation becomes a local process. If we want to draw a curve on the displayed surface, we do this using the following steps: Given the curve in the original coordinate system, every point along the curve is determined with respect to the triangles of the DEM. In other words, its "local coordinates" are computed. These local coordinates contain the number of the triangle in which the point is located and the x,y coordinates of the point, standardized to the local coordinate system of the triangle. These points are then transferred into the display file and converted from the local coordinate system of the same triangle (this time in display coordinates) into the coordinates of the display system. This might sound more complicated than it is, in most cases the computation is two-dimensional and can use many shortcuts.

Any set of data in the original file can be converted into the display file. Smoothing usually takes place in the display file. Equally, one can draw on a digitizer or tablet (or with a mouse) and the line will be displayed on the display as if it was drawn directly on the surface. Hachuring is also done first in the original data base and the display of the parallel lines takes advantages of the similarity between adjacent lines, a characteristic that is usually called graphic coherence.

MANUAL INTERACTION

When we point at the screen with a stylus, a mouse of a tablet, or digitize a hardcopy of a three-dimensional display, we actually interact with the display file and the program finds for every point that we identify the corresponding triangle in which the point is located.

From there onwards, the process is basically the same as the conversion from the original data base. Since the triangles are known in both data sets, the conversion is again a local problem. Asking for the coordinates of a location that the user points at is therefore a direct process.

Drawing on the screen is a little more complex. Here, the display coordinates have to be converted to data base coordinates and then back to display coordinates. Since the system has to remember everything that the user does with the surface, a direct drawing on the screen (as is done on several graphic systems with a bit-mapped display) is not possible since the results cannot be changed afterwards.

ANIMATION

With this approach the production of several views of a surface from observation points that are changed only slightly (the typical flight over a surface), several time-saving shortcuts can be employed since the change from one image to the next will be minor. Some of these shortcuts are:

(a) Recompute the coordinates of the visible points only every second frame.
(b) Recompute actual visibility only after the observation point has passed through a few degrees (without recomputing the horizons).
(c) Ignore partially visible triangles, treat them as if they were completely visible and draw the horizon as a thicker line.
(d) Recompute the horizons only after the observation point has passed through a few degrees, more than in the previous points.
(e) Maintain the normal vector for each triangle.

CONCLUSION

At the beginning of this paper, a fairly strong statement was made about the state of development of Digital Elevation Models. It is obvious that the presented work is only a very small part of an evolution of DEM studies that eventually will lead to some very powerful tools for surface handling. We believe, however, that we are marching in the right direction, even if very slowly.

REFERENCES

Cochrane, D. 1974, A System for Triangular Representation of Surfaces. Master's Thesis, Simon Fraser University.

Douglas, D.H. 1971, personal communication.

Gold, Christopher M. 1984, Common-Sense Automated Contouring--Some Generalizations. Cartographica, vol. 21, 2+3, pp. 121-129.

Peucker, T.K. 1978, Data Structures for Digital Terrain Models: Discussion and Comparison. Harvard Papers on Geographic Information Systems, Dutton, G., ed., vol. 5, Cambridge, Mass. 1978.

Peucker, T.K., R.J. Fowler, J.J. Little and D.M. Mark 1978, The Triangulated Irregular Network (TIN). Proceed., Digital Terrain Model Symposium, ASP, St. Louis, May 1978.

Peucker, T.K., M. Tichenor and W.-D. Rase (1975), The Automation of Relief Representation. In: J.C. Davis and M. McCullagh (eds.), Display and Analysis of Spatial Data, New York, pp. 187-197.

Poiker, T.K. (ed.) 1982, The uTIN Programs: User's Guide and Programmers Guide, Simon Fraser University.

Walters, R.F. 1969, Contouring by Machine, a user guide. Amer. Assoc. of Petrol. Geologists, Bull., vol. 53. pp. 2324-2340.

Woodham R.J. and T.K. Lee 1985, Photometric Method for Radiometric Correction of Multi-Spectral Scanner Data Canadian Journal for Remote Sensing, to appear.