3-D GRAPHIC DISPLAY OF DISCRETE SPATIAL DATA BY PRISM MAPS

Wm. Randolph Franklin *
Harry R. Lewis

Center for Research in Computing Technology
Harvard University,
Cambridge, Mass., 02138

CS sections: 8.2, 3.19, 3.23, 3.39, 3.59, 3.79, 3.9, 5.25, 5.31

Key words: prism maps, graphics, 3-D, display, hidden surface, map, perspective, cartography, thematic mapping, shading

1.0 ABSTRACT

An efficient algorithm for displaying 3-D scenes showing a discrete spatially varying surface is described. Given a 2-D map or planar graph composed of polygons where each polygon has a positive real number attribute, a prism is erected on each polygon with height proportional to that attribute. The resulting 3-D scene is plotted with shading and hidden lines removed. Thus the spatial variation of the attribute may be quickly and intuitively grasped by the nontechnical observer. This has applications to areas such as geography if the map is a cartographic map, or to physics if the map diagrams the periodic table. The algorithm takes time O(N*log(N)) where N is the number of edges in the map. Most of the calculations can be done without knowing the prism heights so extra plots with different attributes for the prisms can be produced quickly. This algorithm has been implemented and tested on maps of up to 12000 edges.

2.0 INTRODUCTION

Consider a scene consisting of a base map such as the USA, as shown in Figure 1, with prisms of varying heights on each state. For example, in Figure 2, each state's height is proportional to its number of alcoholics per 100,000. It is very easy to see how the data varies: Nevada is first which we can rationalize easily enough, but Rhode Island is second and Wisconsin is third. This is totally unexpected. We could also obtain these facts from a Census table but that table could not show us the spatial relationships: each of these three states is surrounded by neighbours with far lower rates.

Figure 1: Base map of USA by state.

This example illustrates both what a PRISM-MAP is and why it is so useful. With the information explosion, it is no longer enough to produce data; the data must be in a form that a casual observer can easily and intuitively appreciate or else it is worthless. As computing power becomes cheaper, powerful display techniques like this become more important both because there is more data to display and because the display techniques are less expensive to use.

This paper describes a new, faster algorithm to produce prism maps. Indeed, most of the calculation can be performed on the two dimensional base map, producing an intermediate file that can be combined with different sets of heights to produce new plots.

This algorithm takes time O(N*log(N)) where N is the number of edges in the map. If cost were no object this algorithm would not be necessary, since a three dimensional scene could be generated from

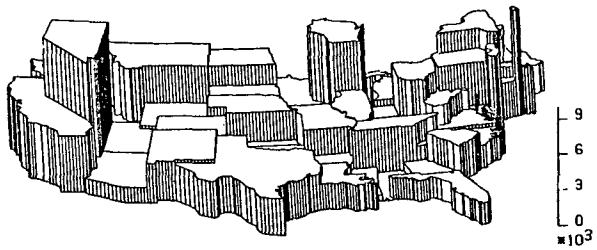

Figure 2: Estimated Alcoholism per 100,000 by state.

* Work partially supported by the Lab for Computer Graphics and Spatial Analysis, Harvard University.

the base map and heights and then fed into a general hidden surface routine. The only previous published solution did just this. Tobler [12], took at least $O(N^2)$ time, and could only process a few hundred edges. This algorithm solves the problem of displaying a discretely varying 3-D surface of a function of two variables. Hidden surface contouring algorithms such as ASPEX [7] handle the continuous case. This algorithm adapts the concept of a horizon line, developed by 1967 for the continuous case by Rens and Tobler [8] in the ASPEX program. Although problems like this are not that well known in the computer science community, there have been attempted solutions for several years by cartographers and geographers. Even though three dimensional plots are more appealing, because of their difficulty, various two dimensional methods have heretofore necessarily been used.

3.0 THE ALGORITHM

3.1 Definition

Prism: A polyhedron that is the extension of a polygon in the XY plane, into the Z direction. The top face is congruent to, parallel to, and straight above the bottom face. The side faces are vertical rectangles. If the 2-D polygon has N sides then the prism has 2N vertices, 3N edges, and N+2 faces. A simple polygon and the prism derived from it are shown in Figure 3.

3.2 Basic Algorithm

The algorithm is basically this:

1. Read the input map and normalize it.
2. Write its edges to a file, one edge per record.
3. Sort the file by minimum Y value of each edge.
4. Process the resulting file with a local processor that:
    1. Reads edges in order, into memory,
    2. Reorders them while in memory so that if edge E_1 hides (defined later) E_2, then E_1 occurs before E_2.
    3. Writes them out.
5. Repeat the following as often as desired, once per plot:
    1. Read the set of prism values or heights into memory.
    2. Read the final sorted file and as each edge is read, draw part of the plot.

The steps will now be explained in more detail.

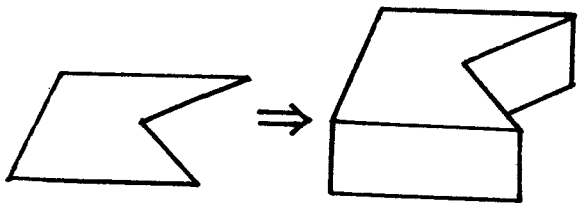

Figure 3: A simple polygon and the prism derived from it.

3.3 Input

The map consists of points, lines and polygons. However the only explicit datatype is a set of straight edges or line segments that form a planar graph. The polygons, marked by unique identification numbers, can be built from the edges. Each edge, E_i, represents a quadruple (A_i, B_i, L_i, R_i). A_i and B_i are the coordinates of the two endpoints. L_i and R_i are the two polygons, on the left and right of E_i (looking from A_i towards B_i). The nonexistent polygon on the exterior is numbered zero.

3.4 Normalization

The map can be observed in perspective in two dimensions from some general point (X,Y) in the plane. It is rotated, scaled and perspectively transformed to make the viewpoint be at (0, -infinity). So now the projection is orthogonal. For the actual 3-D scene, the viewpoint is in 3-space with a line from it to the origin forming a given altitude angle with the horizontal. [11] gives a thorough description of 3-D transformations and projections.

3.5 First Sort

The first sort is by the minimum Y coordinate of each edge. It is very simple and can be done quickly enough by any reasonable external sorting algorithm, such as Knuth [6].

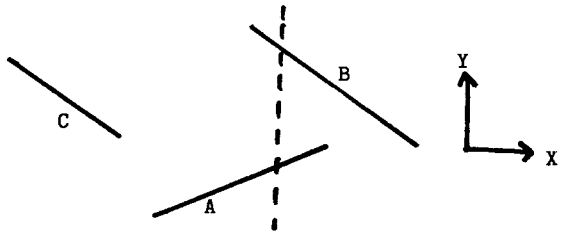

Figure 4: The "hiddenness" relation among map edges.

3.6 The Partial Order

3.6.1 The Partial Order In 2-D -

Definition: Edge A on the input map directly hides edge B iff there exists a vertical line which intersects both A and B with the B intercept being higher, and with no other edges intersecting that line between the A and B intercepts. The vertical line can intersect either edge at an endpoint. This means that at that line, A obscures B as seen from the viewpoint and that no other edge is between them at that point.

Definition: A indirectly hides B iff there is a sequence of N>2 edges C_i with $A=C_1$, $C_N=B$, C_i directly hiding C_{i+1}, and A does not hide B directly.

Definition: A hides B iff A directly hides B or A indirectly hides B.

Notice that "hides" is closed under transitive completion. Thus "hides" is a partial order on the edges of the input map. In Figure 4, A directly hides B but doesn't hide C, directly or indirectly. Note that A can indirectly hide B even though there is no vertical line intersecting both A and B.

Hiding induces a partial order in 2-D because there cannot exist a finite sequence of edges, each hiding the next and the last hiding the first. This is not true in 3-D since three general oblique rectangles, A, B, and C, can be arranged so that A directly hides B, B directly hides C, and C directly hides A.

3.6.2 The Partial Order Extended Into 3-D -

The prisms' tops and sides do not, in general, satisfy the partial order but they can be transformed until they do. Figure 5 shows how a prism top is split into several slices by dropping lines from some of its vertices. After each prism top is split into several smaller polygons, the new set of polygons can be arranged into a partial order. In fact, the 3-D order can be easily formed from the 2-D order. As Figure 5 shows, each top slice is related to one edge of the original polygon. Not all edges, but only the "top edges", induce prism top slices. They are dashed in the figure. Thus the complete 3-D partial order can be formed by sorting the edges in 2-D. Then every edge causes a side polygon of a prism and some edges cause top slices also in the proper place in the order. The top slice polygon caused by any edge in placed immediately before the side polygon from that edge in the 3-D order.

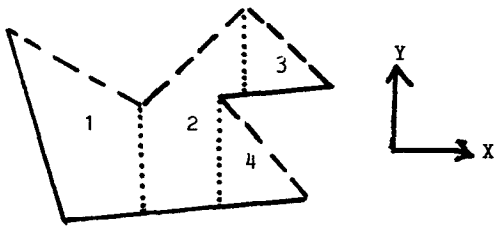

Figure 5: Dividing a prism top into slices by dropping lines down from vertices.

Now given that the prism map's polygons can be ordered as described above, a possible hidden surface algorithm would be to paint them in order onto an initially blank screen, taking care to paint only blank parts of the screen and never to overpaint anything. But first the partial ordering has to be calculated.

3.7 The Final Sort

The algorithm for the final sort is given here briefly. Although it is conceptually simple, it is the most difficult part of the whole prism-map algorithm. The details are given in [5].

1. Run a scan line up the screen, from the bottom to the top.

2. As the scan line rises above the bottom of an edge, E, read it into memory.

3. When E is read into memory, compare it against the edges adjacent to it on the scan line to determine if it directly hides them or them it. If one, say A, hides another, say B, add a link between A and B so each knows about the other.

4. If the scan line rises above the top point of E, then:
   1. If E is has been determined to be hidden by any edges still in memory, then do nothing.
   2. Otherwise write E to the final sorted edge file.

5. If E was written, possibly there are some edges that were remaining in memory only because they

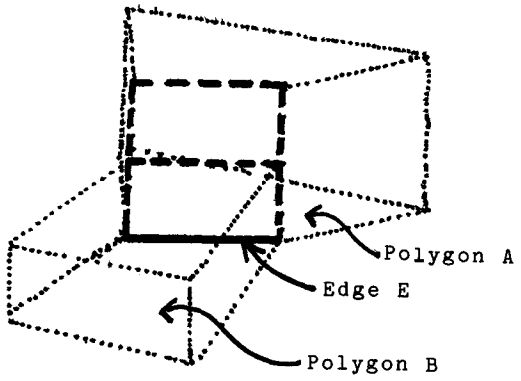

Figure 6: The plot lines derived from one map edge.

were hidden by E and no other edges. If so, write them out, and repeat the process until there are no edges completely below in the scan line remaining in memory unless they are hidden by some edge still in memory.

3.8 Making The Plot

The sorted edge file that was produced in the previous section can now be used with any set of prism heights to produce a plot. The basic algorithm uses a concept of a horizon line that has been used previously to draw hidden surface plots of net representations of bivariate functions. See references [7] and [8].

A horizon line is a function Y=F(X,T) where X and Y address the plotter screen and T is the elapsed time. The line stretches across the plot from left to right and since it is a function never doubles back on itself. At T=0, it lies along the bottom edge of the plot and it increases with T. At any time, it cuts the plots into two regions: The area below has been calculated and plotted while the area above has not been touched yet. As a new part of the plot is calculated, the horizon line is raised above it to include it. Thus this is simply an implementation of the simple algorithm mentioned above.

To produce the plot, the prism heights are read and stored in memory in a hash table indexed by polygon number. Then the sorted edge file is read, and each edge induces part of the plot. Consider for example edge E in Figure 6. It has polygon A on its left and B on its right. E causes four lines to be drawn - the dashed lines in the figure. They are two vertical edges common to the two prisms and a top edge of each prism. The heights at which to draw the lines are known since each edge knows the polygons on each side. If the horizon line should cut across the lines, only the part above the horizon line is drawn. This is the way hidden edges are prevented from being drawn. After the lines are drawn, the horizon line is rased above them. Figure 7 shows Figure 2 halfway through its plot with the current horizon line sketched in.

In Figure 6, the left prism is higher than the right one. If it were lower, then only one top edge would be visible to be drawn. since the higher right prism would hide the left one. Also note that every vertical edge of a prism can be induced by two or more edges of the map. Nevertheless it is drawn only once since after the first time it is drawn, the horizon line is raised high enough that it is not drawn again.

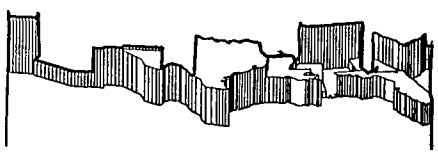

Figure 7: The horizon line when the plot is in progress.

3.9 Shading

The last section described how the plot was drawn and how hidden lines were calculated; this section describes how it is shaded. Two different types of shading are possible: contour lines or vertical shading that assumes an imaginary light source. In either case, extra lines that were not part of the original plot are added to highlight it.

3.9.1 Contour Lines -

These lines run along the sides of the prisms and in the original 3-D scene would be horizontal. If the prisms were cut from thick layers of plywood, the contour lines would be the joins between the plies. They are equidistant and enable the user to count up the side of the prism to determine its height.

Contour lines may produced by not drawing the top edges of the prisms immediately. Instead the top edge is raised gradually from the bottom edge in increments of the contour spacing until its proper value. The complete calculation involving the horizon line is performed for each contour line. The edges are still processed in order: all the contour lines for each edge are drawn before the next edge is read. This seems slow but is necessary to determine which parts of the contour lines are visible. Figure 8 shows per capita public school expenditures by state with the contour lines at multiples of $50. For comparison, Figure 9 shows relative illiteracy, by state. It is shown from the north since the higher southern states would otherwise hide the northern states.

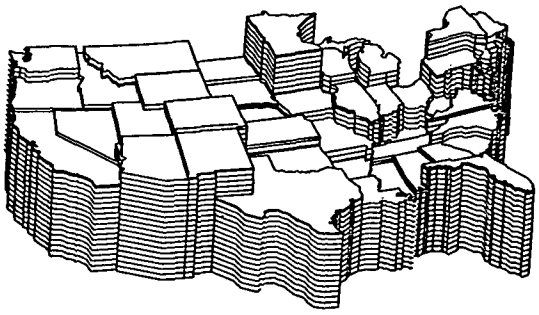

Figure 8: Public school expenditures, dollars per capita; showing contour lines in a prism map.

When the contour lines are drawn, it is usually desirable to draw only some of the vertical edges, those that are in some sense "silhouette edges".

3.9.2 Vertical Shading -

The sides of each prism can be shaded with vertical hatch lines that create a grey scale approximating illumination from a light source. However, the intent here is not to approximate physical reality, (for which see Newell [10]), but to suggest contrasts so as to make the plot easier to understand. As an analogy, it is easier to learn to recognize a person from a skilled caricature than from a photograph since the cartoon emphasizes the features, be they a large nose or whatever, that are not average and plays down the normal ones. Here it is desired to highlight the indentations in the boundaries. To do this, a cosine law raised to a power is used. With a cosine law, the shading on a face is directly proportional to the cosine of the angle between the normal to the face and the direction of an imaginary light source. Raising the cosine to a power increases the amount of very light and very dark areas at the expense of the middle intensity grey areas that would normally cover most of the plot.

In Figure 6, edge E induced two areas to be shaded. They are a vertical side face of prism A and a slice of the top of prism B. The areas in each case are precisely the area between the corresponding top edge and the current horizon line. The top edge of B is drawn first and then the top of B below the edge down to the horizon line. Since the other edges of B that are below E have already been processed, the horizon line cannot be below the bottom of B's top. Thus shading down from the top edge doesn't cause a streak down to the bottom of the plot. After the top edge of B has been drawn, E causes the top corresponding top edge of A to be processed. By now the horizon line is at the top edge of B so that shading down from the top edge of A shades precisely that part of the side face of A which is above B.

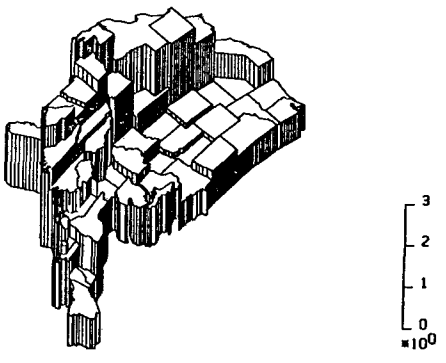

Figure 9: Percent illiteracy, by state, viewed from the northeast.

There are various details to handle if the shading is to be attractive. For instance, the rectangles along an edge are not being shaded in consecutive order so the shading must be designed to blend smoothly from one rectangle to its neighbours. Also a rectangle may be narrower than the shading spacing in which case it randomly gets either zero or one shade lines so that its expected shading density is correct.

4.0 RESOURCES REQUIRED BY THE ALGORITHM

4.1 Time

To analyze the time required by this algorithm, it is necessary to know the distribution of the relevent characteristics of the input scenes as they get bigger. A useful statistic is the average length of the edges as the number of edges, N, increases. This also controls the number of edges that intersect an average scan line which is the number of edges in memory at any time during the final sort. The dependency can be obtained either theoretically from first principles, or experimentally, by measuring actual maps.

For a theoretical analysis, the borders of a map fall into two categories: natural and man-made. Mandlebrot [9], and others have hypothesized that natural coastlines and river beds are fractional dimensional curves that are scale invariant. This is almost impossible to analyze. Manmade boundaries cannot be handled theoretically since they may be straight, or be smoothly curving along parallels of latitude, or be gerrymandered without rhyme or reason.

One useful way to consider a bigger map is as four smaller maps placed side by side and scaled to half the size. This gives the average number of edges crossing a scan line as proportional to SQRT(N). For more analysis, see Chrisman [2].

A heuristic analysis was done by considering national borders from the World Data Bank II [1], generalized [3] to different levels of accuracy containing from 2409 to 14378 edges. Here the average number of edges crossing a scan line was proportional to $N^{0.45}$. Then a map of the USA with the state boundaries (4641 edges) was rotated to six different angles to check the stability of the number of edges crossing a scan line under rotation. The average number was always within one of 15.

Under these assumptions and measured relationships on the input data, the theoretical time required by the algorithm is O(N*log(N)) which is quite satisfactory for a hidden surface algorithm.

4.2 Storage

The whole input file is never in memory at any one time. During the preprocessing stage, three edges need be in memory together. One is being processed and the other two are its neighbours that are needed to set the various bits stored with the edge for shading. The external sorting runs better the more storage it gets, but only needs a small constant amount. The only variable part of the algorithm is the final sort during which all the edges crossing the scan line must be in memory. This is $O(N^{1/2})$ edges or for the 4641 edge USA map, an average of 15 and a maximum of 32. The final plotting requires a constant amount of storage.

5.0 IMPLEMENTATION

This algorithm has been implemented at the Lab for Computer Graphics at Harvard. It is a 6000 line machine independent FLECS program. (FLECS is a Fortran preprocessor adding block structure and in-line routines). It takes 36+7K words on a DEC PDP-10 when compiled with Fortran 10 with no optimization or overlaying. User input is by a pseudo-English interactive command language with about 40 commands. The language was developed by Dougenik [4]. There are many optional parameters to give the user more flexibility that assume reasonable default values if they are not set. PRISM is documented by an extensive users' guide and a program logic manual.

Plotting the USA map with 4641 points takes 130 seconds while each successive plot with a presorted edge file takes 26 seconds. PRISM-MAP has been used on maps of up to 12000 edges, with 36000 edges in the 3-D scene.

6.0 REFERENCES

[1] Anderson, Delmar E. & Angel, James L. & Gorney, Alexander J. World Data Bank II: content, structure & application. An Advanced Study Symposium on Topological Data Structures for Geographic Information Systems, Oct 16-21, 1977, Laboratory for Computer Graphics and Spatial Analysis, Graduate School of Design, Harvard University.

[2] Chrisman, Nick. Concepts of space as a guide to cartographic data structures. An Advanced Study Symposium on Topological Data Structures for Geographic Information Systems, Oct 16-21, 1977, Laboratory for Computer Graphics and Spatial Analysis, Graduate School of Design, Harvard University.

[3] Dougenik, James. LINGUIST: A table driven language model for Odyssey. An Advanced Study Symposium on Topological Data Structures for Geographic Information Systems, Oct 16-21, 1977, Laboratory for Computer Graphics and Spatial Analysis, Graduate School of Design, Harvard University.

[4] Douglas, David & Peucker, Tom. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Canadian Cartographer, Dec 1973.

[5] Franklin, Wm. Randolph. Combinatorics of hidden surface algorithms. Harvard University, Center for Research in Computing Technology, Technical Report, 1978.

[6h] Knuth, D. E. The Art of computer programming, vol. 3, Sorting and Searching, Addison-Wesley, 1973.

[7] The Lab for Computer Graphics and Spatial Analysis, Harvard University. ASPEX users' reference manual, (Jan. 1978).

[8] The Lab for Computer Graphics and Spatial Analysis, Harvard University. SYMVU manual, (1977).

[9] Mandlebrot, Benoit. Fractals: form, change and dimension. WH Freeman and Co., San Francisco, 1977.

[10] Newell, M. E. The progression of realism in computer generated images, ACM 1977 National Conference.

[11] Newman, W. M. & Sproull, R. F. Principles of interactive computer graphics. McGraw-Hill, 1973.

[12] Tobler, Waldo. A Computer Program to Draw Perspective Views of Geographic Data. Cartographic Lab Report, #1, Dept. of Geography, U. of Michigan.

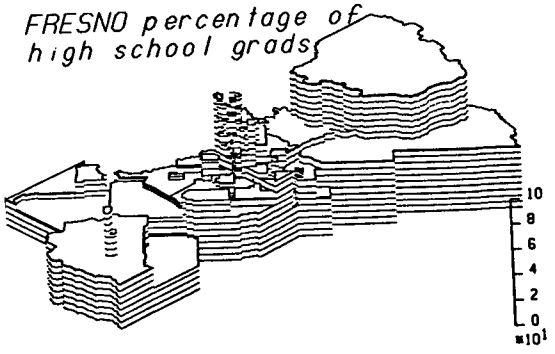

Figure 10: Percentage of high school graduates in Fresno, by census tracts; contours every 5%. No vertical edges at all.