A HIDDEN LINE ALGORITHM FOR HALFTONE PICTURE REPRESENTATION

John E. Warnock

Utah University

Prepared for:

Advanced Research Projects Agency

20 May 1968

Technical Report 4-5

John E. Warnock

A HIDDEN LINE ALGORITHM FOR HALFTONE PICTURE REPRESENTATION

May 20, 1968

COMPUTER SCIENCE

Information Processing System

University of Utah

Salt Lake City, Utah

Advanced Research Projects Agency • Department of Defense • ARPA order 829

Program code number 6D30

The purpose of this paper is to consider the problem of producing two dimensional picture representations of objects described in 3 space. In discussing the main problem of the removal of hidden surfaces in a picture representation, the motivating philosophy of a particular approach is closely examined, and a description of a possible implementation of this approach is described. The advantages of the scheme are also outlined in the paper.

A HIDDEN LINE ALGORITHM FOR HALFTONE PICTURE REPRESENTATION

by John Warnock, Research Mathematician
Computer Science, University of Utah

In exploring applications in computer graphics, one finds quickly that the representation of three dimensional objects in picture form is both a desirable and necessary capability. Applications dealing with any form of spatial design or with visual environment simulation need the ability to represent objects and their relationships to each other in three dimensional space. This capability is enhanced if the variety and complexity of pictures is great and if the pictures can be presented with a wide range of intensities or in color if possible. Other features that are desirable are the abilities to have more than one light source and, if possible, to consider secondary illumination on the objects viewed.

If these capabilities are present then the designer or user of a graphics system has open to him a large latitude of visual representational capability in a wide range of application areas. Because this is so, the algorithms to solve the problems associated with three dimensional representation have had a great deal of effort focused upon them. However, the advances have been slow and difficult.

The main obstacle to accomplishing these tasks is the enormity of the computation required to decide which parts of the objects viewed are seen and which are hidden from view. This paper describes an approach to this problem that is desirable because it is simple, because the computation time is not severe, and because the additional problem of shading and multiple light sources seems soluble.

To motivate the description of the algorithm, let me describe what I believe is a procedure (founded or not) that the human eye and brain uses when examining a picture of a set of objects. Suppose the picture examined is a desk with a coffee cup and pencil on it. The eye quickly determines that large areas on the top of the desk are open and therefore simple from an information content point of view. The eye will not dwell on open areas such as the top of the desk but instead will search out the complexity in the picture and will dwell on that complexity until the positional information about the objects is understood. From there the eye will search out other complex areas and process those. In doing this the eye seems to be dwelling only on a portion of the picture long enough to assimilate the detail and information content of that portion.

If a portion of the picture has no salient features, the time spent by the eye and brain is small. If a portion of the picture is very complex, then the eye may examine subportions of that picture in order to understand the simpler subareas. In short, if the picture is simple, the processing that the eye and brain go through to understand the picture is minor. If there are complex features in the picture, then further processing is required but only on those complex features.

What I am going to describe is an algorithm that copies the intent of the above description. In trying to emulate the above process, the algorithm will look at smaller and smaller portions of a picture until the portions are completely understood in terms of the decision processes used in the algorithm.

The division of the picture into portions is carried out systematically by the algorithm so that the resultant "understood" picture is a collection of simple portions of the picture that may be combined to form the entire picture (see Fig. 1 as an example of this subdivision).

The operation that divides the picture proceeds by dividing the total view plane into four subsquares. The lower left square is then examined to see if it is simple enough to process. If not, it is divided into 4 subsquares and the question is asked again. The process continues until the subsquare considered is either simple enough to handle or the resolution limit of the picture is attained. The algorithm basically asks the question: "Is the portion of the picture I am looking at simple enough to compute?" If the answer is "yes," output is produced. If the answer is "no," a subdivision of the square occurs and the question is repeated. This physical division of the view plane into successively smaller pieces can really be thought of as taking a difficult problem (a complex picture), and dividing it into easier subproblems (simpler pictures) until either the subproblem is solved by default or until it is solved by some decision procedure. The decision process to determine when a square may be output can range in complexity and power. It can be simple to the extent that if anything not blank is in the subsquare, a subdivision must be made. (This is the situation in our first implementation.) If this is the case, then the resultant output is a set of resolution size squares that lie along the visible boundaries of the objects being viewed. On the other hand, the decision process may be complex and handle squares with visible line segments. In this case, the number of output squares can be reduced but the computation per square goes up.

In looking at pictures from the subdivision point of view it turns out that the problem of hidden lines goes away. This is so because as the portion of the view plane examined gets smaller and smaller, the objects being viewed in that portion become well ordered in terms of their distance from the observer. In other words, the objects in a given picture may or may not be ordered in terms of their distance from the eye, and yet at any single point through the

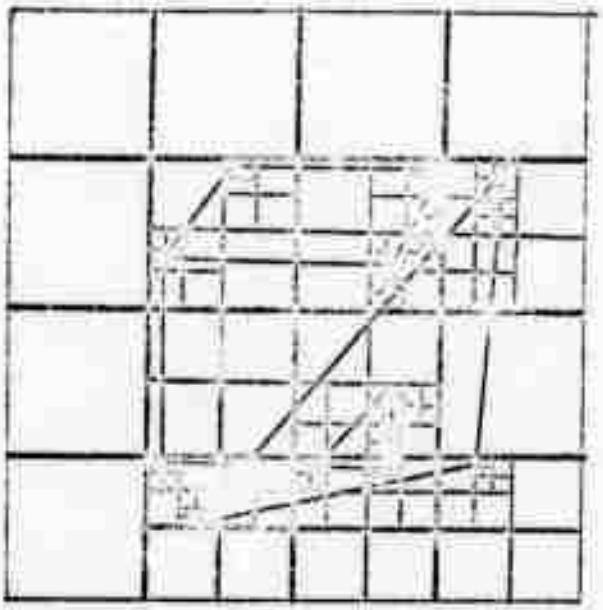

FIG 1

view plane the objects are well ordered. The successive subdivision process allows the local order at some portion of the view plane of the objects to be detected for largest possible areas.

It probably should be noted that at each successive refinement of the portions of the view plane information is gathered so that if further refinement occurs the information collected in the parent squares can be used to the advantage of the algorithm in subsequent refinements. For instance, if an object is found to be outside a given portion of the picture, then it will be outside in refinements of that portion and therefore need not be referenced. This means that as refinement proceeds on a square, the number of objects which are referenced generally becomes smaller and smaller. It can also be noted that since this is the case it is possible to produce highly complex pictures with vast numbers of objects with the advantageous use of secondary storage since generally it is true that after a few refinements of a given picture the list length of the objects relevant to the portion being examined should fit into memory. Consequently the number of secondary storage accesses should not be too severe.

It should also be noted that this entire process can really be thought of as a large number of logically independent activities since disjoint squares have disjoint computational paths. The common element to these independent activities is a common data base, and even the data referenced, as computation proceeds, tends to split apart if the squares separate from one another. This implies that if parallel processing becomes readily available in the future then the algorithm can use this capability to a great advantage.

All of the above considerations make this approach of analyzing objects in pictures very attractive. In the preceeding paragraphs I have attempted to outline the motivation and reasoning on which the hidden surface algorithm

depends. This has been useful since many of the particular implementation techniques used are quite independent of the philosophy and motivation of the algorithm. Because of this, much detail about the nature of the input and about the specific strategies used in the algorithm have been left out of the above discussion. What I will now attempt to do is to solidify the structure of our implementation of the algorithm and give particular programming techniques that are used.

The Input Data

For the purposes of this implementation (even though it appears generalization is possible) the input data is any set of planar polygons described as sets of sequences of points determining closed paths in 3-space. The ordering of these points acts as a criterion for determining the interior and exterior of the polygons. For instance, in fugure 2 the eight ordered outside points determine the exterior of the polygon whereas the 4 interior oppositely ordered points determine a window in the polygon.

This scheme allows the user to describe a wide variety of shapes and objects. Curved surfaces can be constructed by using small planar patches to approximate these surfaces. Other objects are constructed by putting a number of polygonal faces together.

Although the polygons may be positioned anywhere in 3-space the view plane for this implementation is assumed to be the x-y plane such that $0 \leq x, y \leq 512$ and the view point at the point $(256,256,-\infty)$. This means that objects to be viewed must be appropriately positioned and transformed into the first properties and appear with proper perspective and lighting. Once the data has been described and transformed in this manner, the algorithm is ready to accept it.

Data Preprocessing

In the body of the program there are a series of calculations pertinent to a given polygon that are needed over and over. Because of this, some memory is used to store these values in the interest of speed. The most important of these values are the coefficients of the plane of the individual polygons, where the plane is given by $z=Ax+By+C$ (polygons perpendicular to the view plane are eliminated). With these coefficients it is possible to determine a number of things. First the distance from a point on the view plane to the plane of a given polygon can be computed with 3 adds and 2 multiplies. Second, it is possible to determine when 3 planes pass through a common line with these coefficients. Last, the angle of incidence between the light source when it is at the eye and the plane of a polygon can be determined from the coefficients. All these facts justify the additional storage required to retain these numbers.

Other types of information that are computed in the preprocessing stage are min{x}, max{x}, min{y}, max{y}, min{z}, max{z}, for the collection of points that comprise a polygon. These numbers make it easier to determine the relationship between a given subdivision square and a polygon.

Internal Data-Structure

The points of the polygons in this implementation of the algorithm are stored as one point per 36 bit word or 12 bits per coordinate. The order of the points is assumed by their order in memory and the existence of a closed subpath of a polygon is detected by repeating the first point of the path at the end. For example, the polygon in figure 2 would be stored:

pt 1, pt 2, pt 3, pt 4, pt 5, pt 6, pt 7, pt 8, pt 1, pt 9, pt 10, pt 11, pt 12, pt 9.

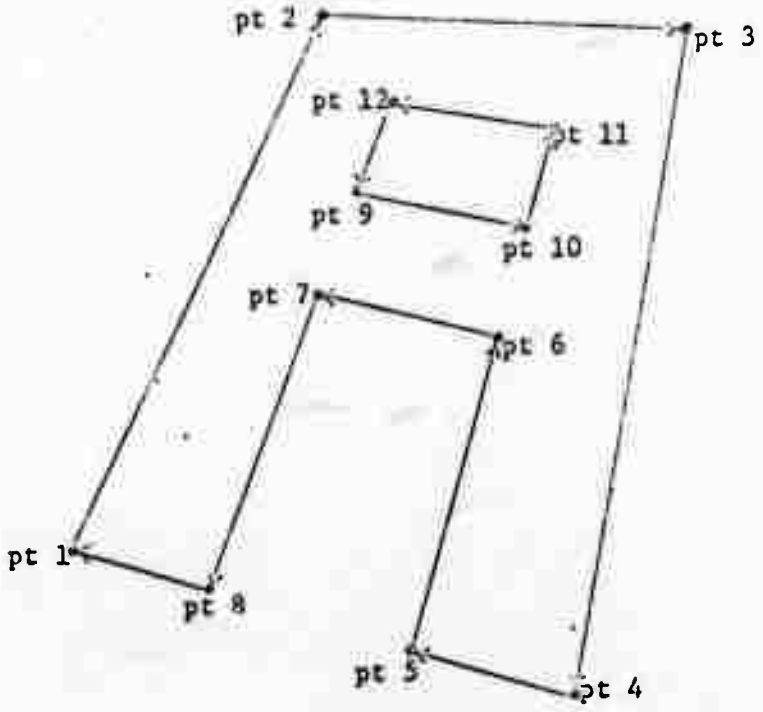

FIG. 2

Two other arrays are stored in memory that are pertinent to the polygons. One array contains the preprocessed data associated with the polygons. The other is an array of pointers that gives the polygon table a list structure. To do this one of the pointers indicates where a given polygon begins in the polygon table. Another pointer indicates the pointer to the next polygon in the list structure. The drawing in figure 3 indicates this memory layout.

The Picture Subdivider and Analyzer

With the above data structure the algorithm can proceed to analyze the picture. To do this the subdivider portion of the program divides the picture into four squares and examines the relationship of the lower left square with each of the polygons. In this implementation only three relationships between squares and polygons are considered. Either the square contains some edge or vertex of the polygon, the square is wholly outside the polygon, or the square is wholly inside the polygon. For purposes of discussion we will refer to these relationships as "involved", "out", and "enclosing" respectively.

As the relationships of the square to the polygons are determined, the list of polygons is arranged so that all "enclosing" polygons are first, "involved" polygons next and "out" polygons last. This is done so that all subsequent refinements of the lower-left square may use this ordering information easily. At this point the depths from the view plane to the planes of "involved" and "enclosing" polygons are calculated at the four corners of the square. These depths provide sufficient information to detect if an "enclosing" polygon covers all other polygons pertinent to that square. If one does or if the lists of "enclosing" and "involved" polygons are empty then the refinement process terminates and the next square examined is the lower right square. In any other case, some edge may possibly be seen and so the lower left square

is subdivided and the lower left square of that subdivision is processed. As the polygons are examined with respect to this square, then the previous "enclosing" list can get longer and the "out" list can get longer. Therefore more useful information is gathered about the smaller square and its relation to the polygons. At this point the depth computations are repeated and the decision question asked again. This process of subdividing, collecting information, and going through the decision process is repeated until the resolution size of the picture is attained, (in this implementation this is 9 subdivisions) or until the decision procedure decides that there are no visible edges present in the square. When no visible lines occur in a square, then the algorithm proceeds directly to the next square of the same size at that level, or if there are none, to the next larger square at the next level. If the resolution limit of the picture is reached, then the coordinates of the lower left corner of the square are stored. Along with these coordinates is sent information about what the square can "see." If the upper left corner is not contained in any polygon, then nothing is sent with the coordinates, otherwise the pointer to the polygon nearest to the observer that contains the upper left corner is stored.

The Shader

The output of the above described process is a set of small squares that lie along the visible boundaries of the objects in the picture. Each square has a pointer to the polygon that is visible just to the left of the square. These squares are sorted in descending order by the x-coordinates of the lower left corner of the square within the y-coordinate. The squares in this order can be used to produce a raster scan output that can be sent to a scope. The scan is built right to left within top to bottom. The scan sends zero intensity to the scope until the coordinates of the raster equal the

coordinates of the first *square* encountered in the list. From here, shading is determined by the polygon to which the square points until the raster matches the next square, or until the end of the current line is reached. On each new line the intensity is reset to zero. The search continues changing intensities as squares are encountered in the list until the end of the picture is reached, and the entire picture has been produced.

This paper has included just one of the many implementations that are possible with the motivating philosophy behind the algorithm. The implementation given is by no means the best from a timing point of view, but it is fairly simply to carry out and has given us promising results and statistics to use for implementations that are yet to be done.

It is our real hope that the approach is flexible and general enough to provide adequate picture representation capabilities to satisfy a large segment of future graphics applications.

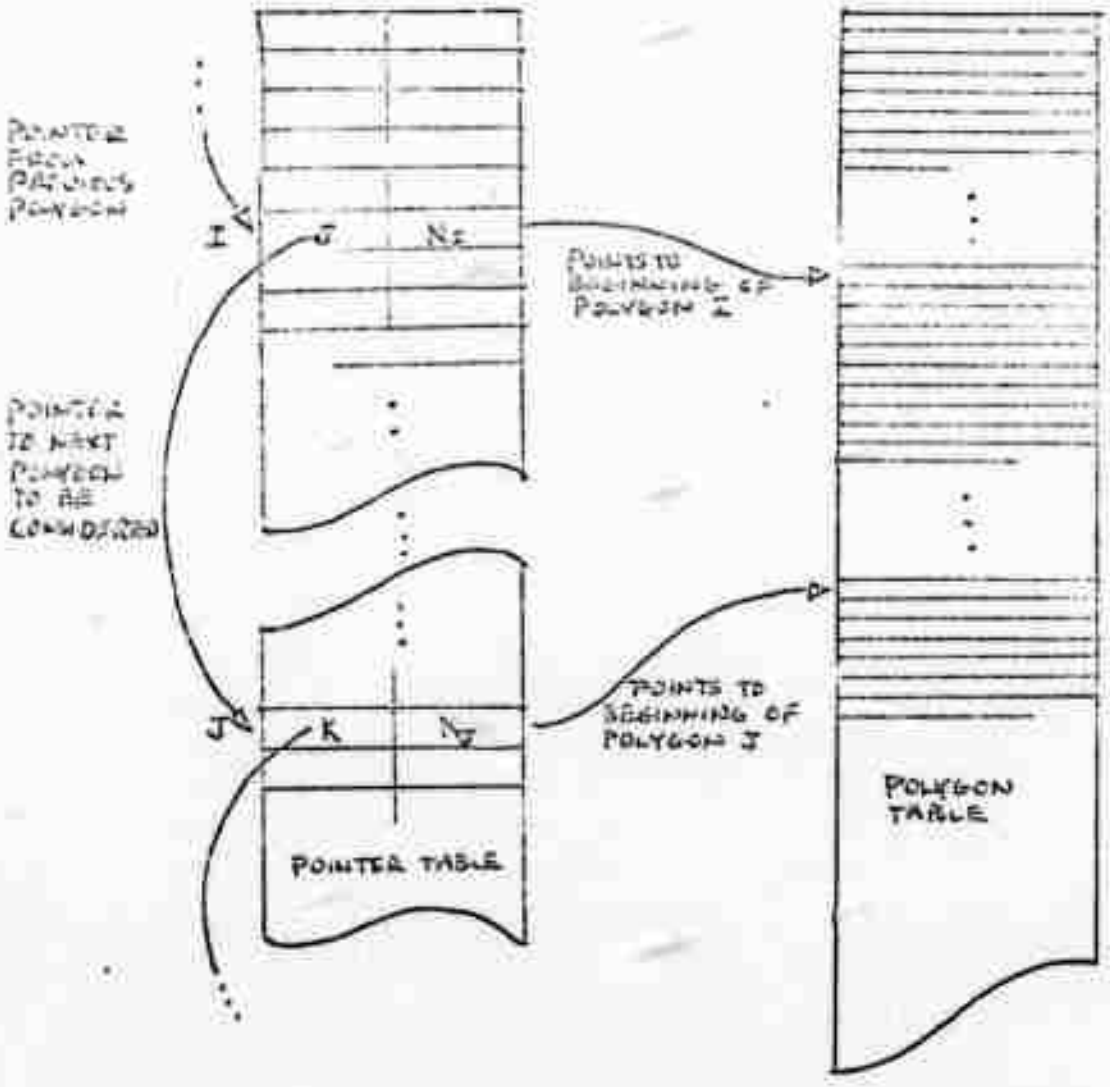

FIG. 3