Fuchs79b

from Computer Graphics, Vol. 13, No. 2 (August 1979)
SIGGRAPH '79 Conference Proceedings

PREDETERMINING VISIBILITY PRIORITY IN 3-D SCENES

(Preliminary Report)

Henry Fuchs
Computer Science Department
University of North Carolina
Chapel Hill, NC

Zvi M. Kedem
Bruce Naylor
Mathematical Sciences
University of Texas at Dallas
Richardson, TX

Abstract

The principal calculation performed by all visible surface algorithms is the determination of the visible polygon at each pixel in the image. Of the many possible speedups and efficiencies found for this problem, only one published algorithm (developed almost a decade ago by a group at General Electric) took advantage of an observation that many visibility calculations could be performed without knowledge of the eventual viewing position and orientation -- once for all possible images. The method is based on a "potential obscuration" relation between polygons in the simulated environment. Unfortunately, the method worked only for certain objects; unmanagable objects had to be manually (and expertly!) subdivided into managable pieces.

Described in this paper is a solution to this problem which allows substantial a priori visibility determination for all possible objects without any manual intervention. The method also identifies the (hopefully, few) visibility calculations which remain to be performed after the viewing position is specified. Also discussed is the development of still stronger solutions which could further reduce the number of these visibility calculations remaining at image generation time.

The reduction in overall processing and memory requirements enabled by this approach may be quite significant, especially for those applications (e.g., 3-D simulation, animation, interactive design) in which numerous visible surface images are generated from a relatively stable data base.

Introduction

The basic ideas in Shumacker et al (1969) are succintly described in Sutherland, Sproull, and Schumacker (1974), from which we quote here. (Note that polygons are referred to as "faces", and objects or parts of objects are referred to as "clusters".)

> The notion that face priority within a cluster can be computed independent of the viewpoint is extremely important. Consider the top view of an object, as shown in figure 1. If, for any viewpoint, we eliminate the back faces (relative to that viewpoint), the numbers assigned to each face in the figure are the priority numbers. A cluster is a collection of faces that can be assigned a fixed set of priority numbers which after back edges are removed, provide correct priority from any viewpoint.
>
> The computation of face priority requires computing whether face A can, from any viewpoint, hide face B. If so, face A has priority over face B. These computations are performed for all faces of a cluster, and a priority graph is constructed. If there are any circuits in the graph (e.g., face A has priority over face B, and face B has priority over face A), the faces cannot be assigned priorities that will produce a correct image. In this case, the cluster will have to be split manually into smaller clusters.

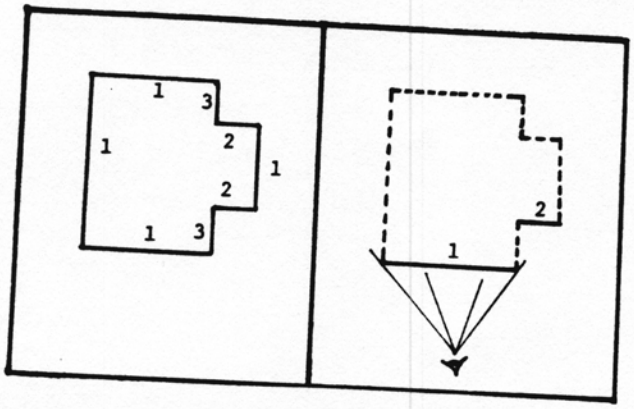


Figure 1: Face priority. a) Top view of an object with face priority numbers assigned (the lowest number corresponds to the highest priority). b) The same object with a specific viewpoint located. The dashed lines show back faces. Face 1 takes priority over face 2. [From Sutherland, Sproull, and Schumacker (1974)].

Although it is at first encouraging that viewpoint-independent visibility priority can be determined for many objects, it is rather discouraging that objects for which this cannot be done are so easy to find (e.g., figure 2). This lack of "dependability" of the approach, we believe, has kept it from widespread use. We present in this paper a solution to this problem which allows, without manual intervention, the generation of priority structures which exhibit minimum variance under viewpoint modification.

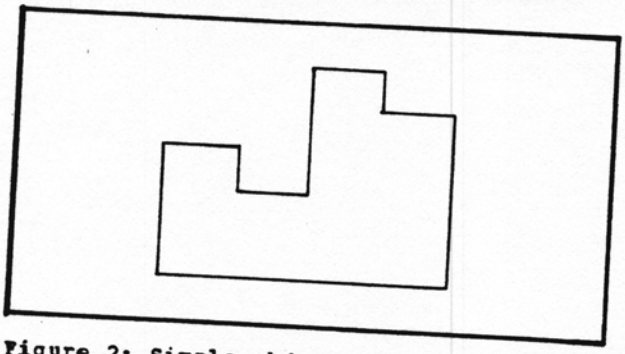


Figure 2: Simple object with no viewpoint-independent priority.

Problem Description

Let then $R = \{r_1, r_2, \ldots, r_n\}$ be the set of (convex) polygons to be considered. This set may be first considered as describing a single object, or possibly later the entire scene of objects. Schumacker et al attempted construction of a function $f: R \rightarrow Z^+, Z^+ = \{1,2\ldots\}$, having the property that if both r_i and r_j are visible "from outside the object" and if r_i and r_j are both "facing" toward the viewer, then $f(r_i) \leq f(r_j)$ if and only if r_j cannot obstruct r_i. Further, for any point P outside the object (see figure 3) if there is a ray from P which intersects both r_i and r_j and if r_i and r_j "face" P, then r_i obscures r_j (along this ray) if and only if $f(r_i) < f(r_j)$.

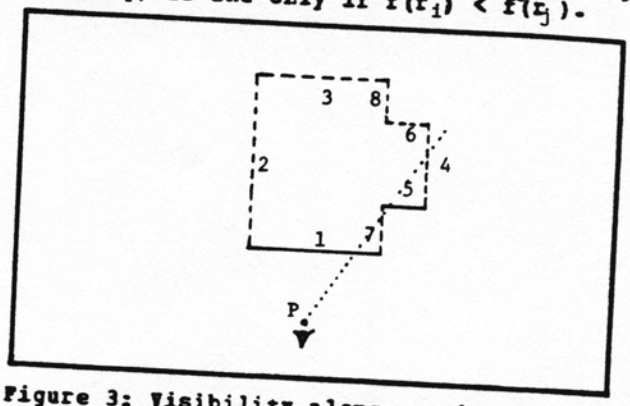


Figure 3: Visibility along a viewing ray (with an associated $f(r_i)$ beside each polygon r_i.)

We now formally describe our interpretation of Schumacker's method as partially based on Sutherland, Sproull and Schumacker (1974) and on personal communication with him (Schumacker). As above, we define a relation < on the set R: $r_i < r_j$ if and only if there exists a ray from some viewing position P which pierces the "front sides" of both r_i and r_j and whose point of intersection with r_i is closer to P than its point of intersection with r_j. (See the Appendix for a more formal discussion of how this relation is calculated.) This relation is modeled by a (directed) graph G which we will refer to as a "priority graph". If this graph is acyclic then one can define a function $f: R \rightarrow Z^+$ satisfying the "visibility" conditions described above. We differ slightly from Schumacker's approach in that we shall require f to be one-to-one. This variation does not restrict the generality of the method, but will be useful in the subsequent development. We borrow a term used in the context of a partial order and will refer to such a function as a consistent enumeration of (R,<) (see, e.g., Preparata and Yeh (1973)). (This is also sometimes called a topological sorting; see, e.g., Knuth (1973).)

We give two examples of objects (for simplicity in 2-space), one object whose associated graph is acyclic and has a consistent enumeration (figure 4) and one whose graph contains a cycle and thus does not have a consistent enumeration (figure 5).

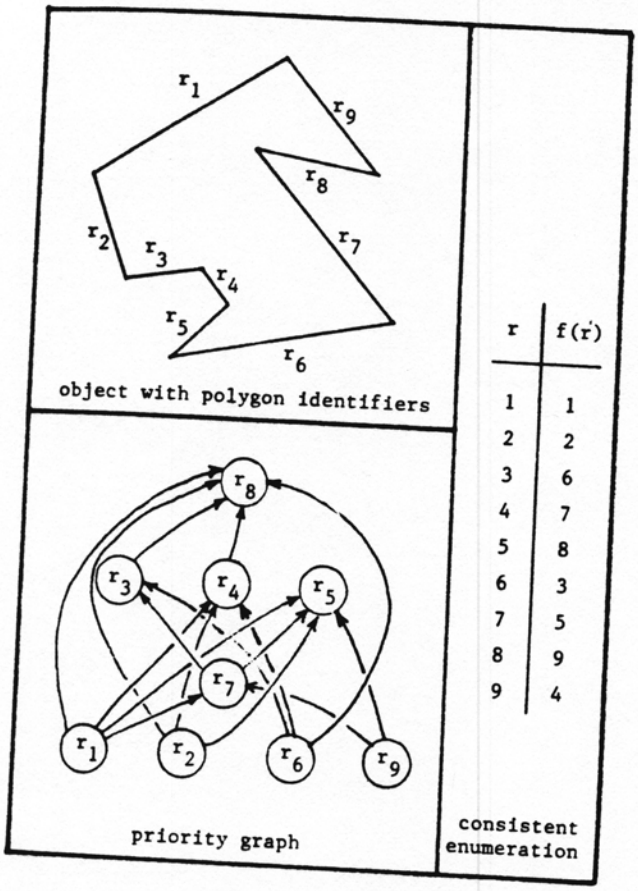


Figure 4: An object which admits consistent enumeration.

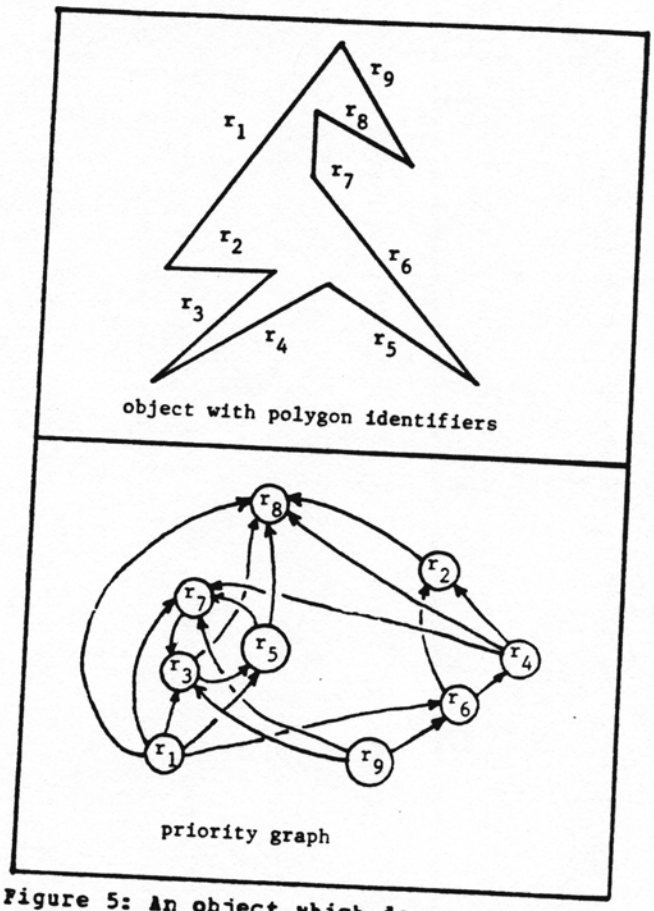


Figure 5: An object which does not admit consistent enumeration.

If the graph was not acyclic Schumacker manually attempted to split the object into several pieces, with each of which an acyclic graph could be associated. The difficulty with this approach was that no general method was found to split up complicated objects, and thus many of the more interesting cases could not be handled without manual intervention.

Proposed Solution

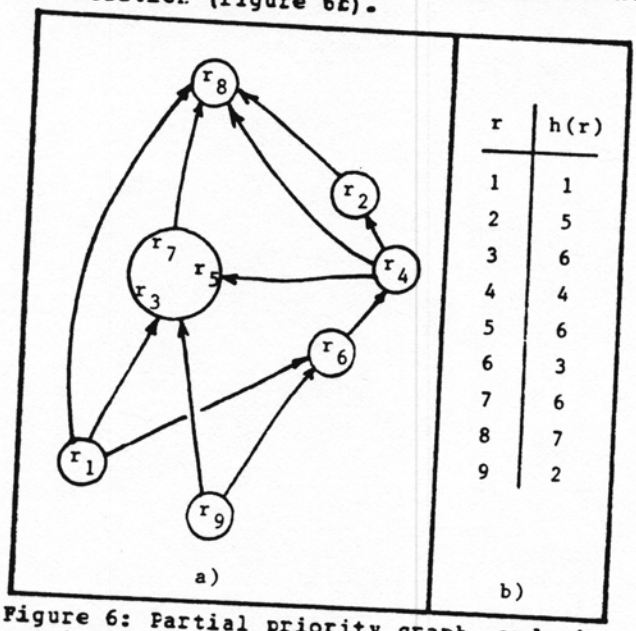
We propose a two-stage approach for solving this problem:

1. Extraction of as much viewpoint - independent visibility information as possible from an object's visibility priority graph. (This involves analysis of the graph and isolation of "difficult" subgraphs.)
2. Further analysis of these remaining difficult subgraphs with the intent of either
   a. assigning a more complex visibility code than a partial ordering, or
   b. automatically splitting some of the object polygons to simplify these difficult subgraphs.

The first stage involves isolating the "difficult", cyclic subgraphs in our priority graph. To do this we partition the priority graph into strongly connected components. (A subset of vertices of a graph spans a strongly connected component if and only if it is a maximal subset of vertices such that any vertex in the subset is reachable from any other vertex in this subset. See, e.g. Berge (1973).)

Let $U_1, U_2, \ldots, U_m$ be the (pairwise disjoint) subsets of vertices spanning the strongly connected components of the priority graph G. (Each U_i is a set of polygons.) We define a new "partial priority" graph G' whose vertices are $U_1, \ldots, U_m$, and there is an arc from U_i to U_j if and only if there are two vertices in G, $r_a \in U_i$ and $r_b \in U_j$, such that there is an arc in G from r_a to r_b. There are well-known efficient methods for calculating the graph G' from the original graph G. (See, e.g., Aho, Hopcroft & Ullman

(1974).) As G' is acyclic there exists a consistent enumeration f: $\{U_1, U_2, \ldots, U_m\}$ -> Z^+. This consistent enumeration of U_i's can be extended to a <u>partial enumeration</u> of R by defining a function h:R --> Z^+ where $h(r) = f(U_i)$ for $r \in U_i$. We present G', for the previous unworkable case (figure 5) if figure 6a. This graph G', as any "partial priority" graph, is always acyclic and thus admits consistent enumeration (figure 6b).



r	h(r)
1	1
2	5
3	6
4	4
5	6
6	3
7	6
8	7
9	2

Figure 6: Partial priority graph and its consistent enumeration (compare with figure 5).

Thus we have assigned some enumeration h:R --> Z^+ whose meaning is as follows: For any view, if r_a and r_b are both "front facing" and are pierced by a ray from the viewing position, then if $h(r_a) < h(r_b)$ then r_a is visible and may obscure r_b along this ray. (Note that the meaning of $h(r_a) = h(r_b)$ for $r_a \neq r_b$ is different here from the one in Schumacker's approach. In our case $h(r_a) = h(r_b)$ means that it is not possible to decide relative visibility of r_a and r_b using h only, and some additional calculations will be necessary.)

This completes the first stage of our solution. We note that even if the analysis process is stopped here that substantial reduction in the number of visibility calculations is likely.

To illustrate this, let us consider the "innermost" level of a visible surface algorithm -- that of visibility calculation for one pixel. There may be several polygons potentially visible. Let then $S = \{s_1, \ldots, s_q\} \subseteq R$ be the set of polygons potentially visible at this pixel. (A polygon s_i is "potentially visible" if and only if it covers the pixel but is not necessarily the nearest one to the viewing position.) Let $\hat{h} = \min\{h(s_i) \mid i=1,\ldots,q\}$ and let $\hat{S} = \{s_i \mid h(s_i) = \hat{h}\}$. To find the polygon visible at this pixel it is sufficient to consider <u>only</u> the elements of $\hat{S}$. If $|\hat{S}| << |S| = q$ then the visible polygon can be determined efficiently. (In particular, if $|\hat{S}| = 1$ then there is a single polygon with highest priority and no further visibility processing is needed.)

Discussion and Extensions

We are currently implementing the above-described solution and are studying the feasibility of still more powerful techniques. Clearly the actual implementation will answer some significant questions -- most importantly the percentage of pixels in images for which $|\hat{S}| = 1$, the percentage for which $|\hat{S}| << |S|$, and the distribution of these pixels throughout the image and the distribution among many "similar" images.

Several additional enhancements may reduce still further the numbers of calculations needed to generate each image. Let us consider, for instance, the situation in which one polygon A obscures only a part of another polygon B (figure 7a). B could be split along the plane of A into two polygons (say, B_1 and B_2) such that A can <u>not</u> obscure B_2 (figure 7b). Conversely, if the plane of B splits A (figure 7c) then A can be split into two pieces such that only one piece potentially obscures B (figure 7d). Thus in some cases strongly connected components can be entirely broken up by this method (figure 8a and 8b). Still more efficiencies may be attained by generalizing the original goal of a pricri visibility calculations.

Until now we have discussed only techniques whose results are valid for all possible viewing positions P and all possible viewing rays r. A more generous view of visibility priority requirements may be useful as some derived information may only be valid within some limited scope.

Some possible scopes of visibility priority information:

1. All P's and all r's: This is the most useful information, one to which the above discussion has so far been limited.
2. Some P's and all r's: This may be information whose validity is limited to certain regions in

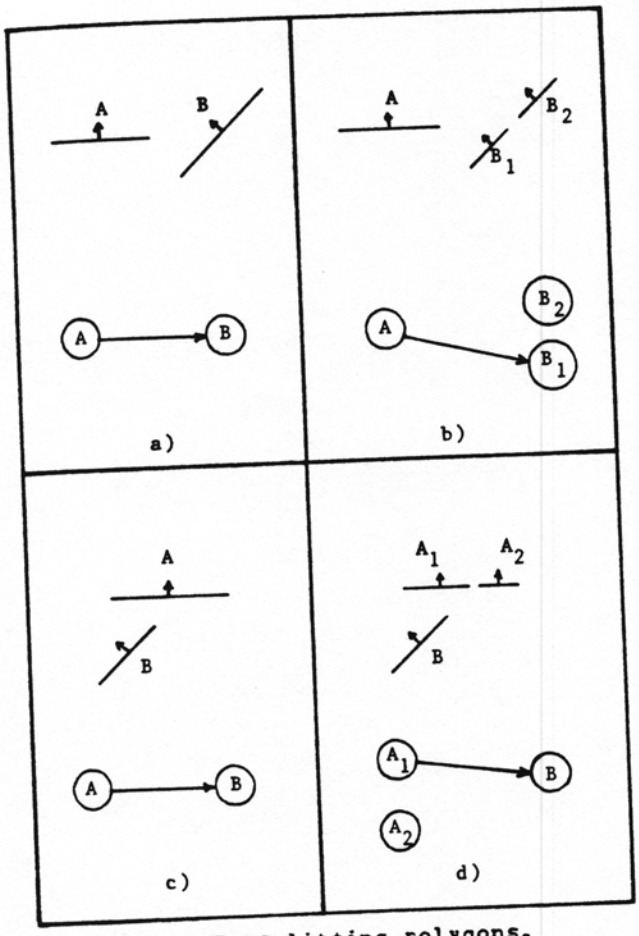


Figure 7: Splitting polygons.

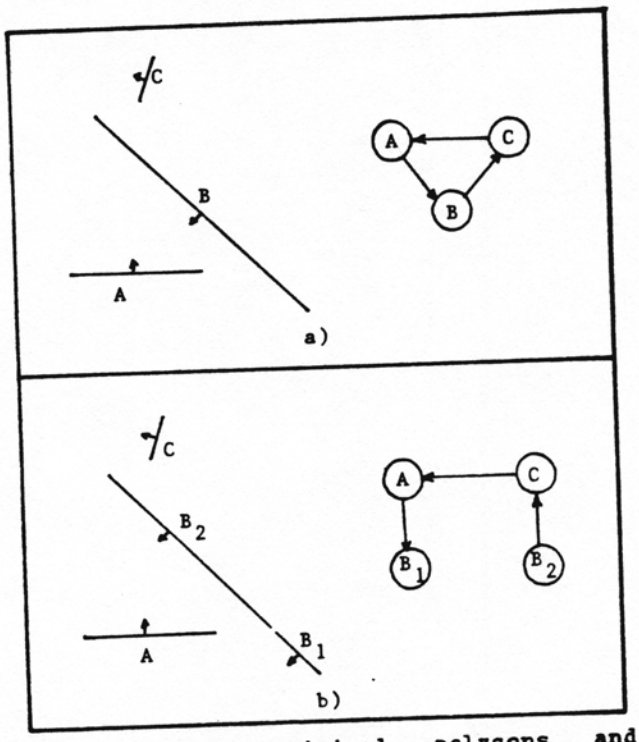


Figure 8: a) Original polygons and strongly connected component graph. b) Split polygons and resulting (acyclic) priority graph.

the 3-space environment; we may wish to utilize such information by subdividing the environment space and making modifications in the priority ordering whenever P crosses from one subspace to another.

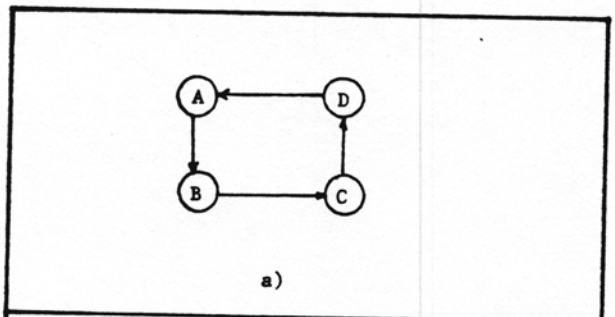
3. One particular P and all r's: This may correspond to information valid for all the images generated from one particular viewing position. When the viewing position is moved, such information would have to be modified (by ,perhaps, altering parts of the priority graph.)

4. One particular P and one particular r: This is information valid only for certain pixels within certain image. In this situation, the usefulness of our general "a priori" approach will depend on the specific number of calculations remaining to this stage, as compared with the number of calculations needed by the standard distarce computations used in most visible surface algorithms.

(Other scopes, of course, may also prove useful.)

One approach we are currently developing for this last situation is to assign more sophisticated enumerations to the elements of strongly connected components. Let us consider, for instance, the simple strongly connected components of figure 9a, consisting of a ring of four vertices. The lack of arcs between nodes A and C and between nodes B and D implies that one node (polygon) in each pair can never obscure the other in the pair, from any point in space. (That is, there is no relation between A and C and between B and D.) This implies that at any given pixel in the image, at most one (but not both) of the pair AC and one of the pair BD may be potentially visible. Thus all the pixels which have more than one potentially visible polygons belong to one of four classes: A & B (potentially visible), B & C, C & D, D & A. In all the cases the priority assignment is easy to determine. The various cases are summarized in figure 9b.

Left yet to be resolved are strongly connected components with more complex structure than a simple cycle. For these situations it may be possible to show that for certain (most?) geometric



a)

Potentially Visible at a Pixel: A	B	C	D	Highest Priority (i.e., visible)
Y	Y	N	N	A
N	Y	Y	N	B
N	N	Y	Y	C
Y	N	N	Y	D

b)

Figure 9: a) Strongly connected component. b) The only possible conflicting cases.

configurations there is no point P with viewing ray r which can pierce all polygons of any complete subgraph. Should this prove to be the case, then a calculation determining which polygon(s) are <u>not</u> potentially visible at a pixel (by backfacing or not falling onto the pixel) will be sufficient to determine the priority of the remaining potentially visible ones.

These and other techniques are currently being investigated.

Appendix

Determining the Priority Graph

We assume for the following discussion that each polygon in our set
- a. is oriented, in that it has two faces: a front face and a back face,
- b. does not interpenetrate another polygon,
- c. is convex.

Consider now that the plane in which a polygon lies naturally divides the rest of the three-dimensional space into two half-spaces associated with the two sides of the polygon: the front half-space, and the back half-space. for a polygon r they will be denoted by FHS(r) and BHS(r), respectively.

We now wish to analyze the following situation: let there be two non-intersecting oriented polygons r_1 and r_2 in the space. Under which circumstances can we say that from some viewing position r_1 (partially) obstructs r_2? We shall denote such "potential" obstruction of r_2 by writing $r_1 < r_2$.

It is easy to see that r_1 obstructs r_2 from a viewing position P along the ray α if and only if the following conditions are satisfied:
1. P lies in $FHS(r_1) \cap FHS(r_2)$, and
2. A point travelling from P along α
   - a. travels for some distance in a region of the space which is in both $FHS(r_1)$ and $FHS(r_2)$ until it "pierces" r_1 then
   - b. travels in a region which is in $BHS(r_1)$ and in $FHS(r_2)$ until it pierces r_2, then
   - c. travels in a region which is in both $BHS(r_1)$ and $BHS(r_2)$.

This informal discussion leads to the following characterization:

<u>Lemma</u>: $r_1 < r_2$ if and only if
1. $r_1 \cap FHS(r_2) \neq \emptyset$, and
2. $r_2 \cap BHS(r_1) \neq \emptyset$.

(Here r_1 and r_2 are treated as planar regions in the three-dimensional space.)

<u>Proof</u>: Is not presented here. It is a straightforward formalization of the previous discussion. □

It is worthwhile to examine the situation deeper. The two planes on which the polygons r_1 and r_2 lie divide the rest of the space into four "cones". (We are excluding here, for simplicity, the case where the two planes are parallel.) These four cones are simply:

$FHS(r_1) \cap FHS(r_2)$,
$FHS(r_1) \cap BHS(r_2)$,
$BHS(r_1) \cap FHS(r_2)$,
$BHS(r_1) \cap BHS(r_2)$.

We can now state:
$r_1 < r_2$ <=> both r_1 and r_2 have non-empty intersections with the boundary of $BHS(r_1) \cap FHS(r_2)$.

For a polygon r we shall denote by CONVEX(r) the smallest subset of the vertices of r that spans the (convex) hull of r. We can now state a computationally

attractive condition for (potential) obstruction:

Theorem: $r_1 < r_2$ if and only if there exist $P_1 \in CONVEX(r_1)$ and $P_2 \in CONVEX(r_2)$ such that $P_1 \in FHS(r_2)$ and $P_2 \in BHS(r_1)$.

Proof: From the above it follows that $r_1 < r_2$ if and only if there exist $\tilde{P}_1$ inside or on the boundary of r_1, and $\tilde{P}_2$ inside or on the boundary of r_2 such that $\tilde{P}_1 \in FHS(r_2)$ and $\tilde{P}_2 \in BHS(r_1)$. Let now $\tilde{P}_1 \in FHS(r_2)$. Intersect $\tilde{P}_1$ with a line in the plane of r_1; this line must cross the boundary of the enclosing convex hull defined by $CONVEX(r_1)$ in exactly two places; call these points Q_1 and Q_2. Since $\tilde{P}_1$ lies between Q_1 and Q_2, one easily sees that either $Q_1 \in FHS(r_2)$ or $Q_2 \in FHS(r_2)$. Without loss of generality, assume $Q_1 \in FHS(r_2)$. Q_1 lies between two points in $CONVEX(r_2)$; call these P_i and P_j. Similarly since $Q_1 \in FHS(r_2)$ either $P_i \in FHS(r_2)$ or $P_j \in FHS(r_2)$. Thus there exists some $P_k \in CONVEX(r_1)$ such that $P_k \in FHS(r_2)$. The treatment of r_1 is analagous. □

We can thus easily check whether $r_1 < r_2$ using $O(CONVEX(r_1) + CONVEX(r_2))$ operations. In the case where the polygons are defined in terms of a list of vertices one has an $O(\#(r_1) + \#(r_2))$ algorithm, where $\#(r)$ denotes the number of points in the vertex-list defining r. (Simply examine all the vertices of a polygon r without limiting consideration to CONVEX(r) only.)

With the additional observation that the distance between the vertices of r_1 (respectively r_2) and the plane defined by r_2 (respectively r_1) is bimodal, an $O(\log(\#(r_1) + \#(r_2))$ algorithm can be designed; it is probably not worthwhile to do so as $\#(r_1) + \#(r_2)$ is rather small. (Simply examine all the vertices of a polygon r without limiting consideration to CONVEX(r) only.)

References

Aho, A.V., Hopcroft, J.E. & Ullman, J.D. (1974) The Design and Analysis of Computer Algorithms, Addison-Wesley

Berge, C. (1973) Graphs and Hypergraphs, North Holland

Knuth, D.E. (1973) The Art of Computer Programming: Volume 1/Fundamental Algorithms, (Second Edition), Addison-Wesley

Preparata, F.P. & Yeh, R.T. (1973) Introduction to Discrete Structures, Addison-Wesley

Schumacker, R.A., Brand, R., Gilliland, M. & Sharp, W. (1969) " Study for Applying Computer-Generated Images to Visual Simulation", AFHRI-TR-69-14, U.S. Air Force Human Resources Laboratory

Sutherland, I.E., Sproull, R.F. & Schumacker, R.A. (1974) "A Characterization of Ten Hidden-Surface Algorithms", ACM Computing Surveys, 6 (1):1-55

-- end --