Open access • Journal Article • DOI:10.1145/965141.563896

Hidden surface removal using polygon area sorting — Source link

WeilerKevin, AthertonPeter

Published on: 20 Jul 1977 - Computer Graphics (ACMPUB27New York, NY, USA)

Topics: Hidden line removal, Polygon and Hidden surface determination

HIDDEN SURFACE REMOVAL USING POLYGON AREA SORTING

by

Kevin Weiler and Peter Atherton

Program of Computer Graphics

Cornell University

Ithaca, New York 14853

ABSTRACT

A polygon hidden surface and hidden line removal algorithm is presented. The algorithm recursively subdivides the image into polygon shaped windows until the depth order within the window is found. Accuracy of the input data is preserved.

The approach is based on a two-dimensional polygon clipper which is sufficiently general to clip a concave polygon with holes to the borders of a concave polygon with holes.

A major advantage of the algorithm is that the polygon form of the output is the same as the polygon form of the input. This allows entering previously calculated images to the system for further processing. Shadow casting may then be performed by first producing a hidden surface removed view from the vantage point of the light source and then resubmitting these tagged polygons for hidden surface removal from the position of the observer. Planar surface detail also becomes easy to represent without increasing the complexity of the hidden surface problem. Translucency is also possible.

Calculation times are primarily related to the visible complexity of the final image, but can range from a linear to an exponential relationship with the number of input polygons depending on the particular environment portrayed. To avoid excessive computation time, the implementation uses a screen area subdivision preprocessor to create several windows, each containing a specified number of polygons. The hidden surface algorithm is applied to each of these windows separately. This technique avoids the difficulties of subdividing by screen area down to the screen resolution level while maintaining the advantages of the polygon area sort method.

COMPUTING REVIEWS CLASSIFICATION: 3.2, 4.9, 4.40, 4.41

KEYWORDS: Hidden Surface Removal, Hidden Line Removal, Polygon Clipping, Polygon Area Sorting, Shadowing, Graphics

INTRODUCTION

A new method for the computation of visible surfaces for environments composed of polygons is presented. The output of the method is in the form of polygons, making it useful in a variety of situations including the usual display applications. The primary components of the method consist of a generalized algorithm for polygon clipping and a hidden surface removal algorithm. The polygon clipper is sufficiently general to clip a concave polygon with holes to the area of a concave polygon with holes. The hidden surface algorithm operates on polygon input which has already been transformed and clipped to the final viewing space. The X and Y axis of this viewing space are thus parallel to the display surface and the Z axis is parallel to the line of sight.

Many visible surface algorithms have been developed, each with unique characteristics and capabilities. A survey presented by Sutherland et al [11] provides a method of categorization as well as a statistical comparison of many of the polygon based algorithms.

Their classification divides the algorithms into three types: object space, image space and list-priority algorithms. The "object space" methods perform the hidden surface computations for potentially visible polygons within the environment to an arbitrary precision usually limited only by machine precision. The "image space" methods are performed to less resolution and determine what is visible within a prescribed area on the output display, usually a raster dot. The "list-priority" algorithms work partially in each of these two domains.

Siggraph '77, July 20-22 San Jose, California

The performance of "object space" methods (Roberts [8], Appel [1], Galimberti [4]) and "list-priority" methods (Newell, Newell and Sancha [7], Schumacher [9]) is dependent on the complexity of the environment. Since all of these algorithms make comparisons between items (objects, polygons, edges), the number of sorting steps required can rise exponentially with the number of input items. However, the computational time is independent of the resolution or size of the image.

In contrast, the "image-space" algorithms make polygon to screen area comparisons (Warnock [12], Watkins [13], Bouknight [3], and depth map or Z-buffer algorithms). Therefore, the number of sorting steps is possibly linear with the number of input polygons, but can vary exponentially with the resolution required.

The algorithm presented qualifies as an object space algorithm since all of its calculations are performed to arbitrary precision. The sorting methods used include a preliminary partial depth sort, an x-y polygon sort, and a conclusive depth sort which may involve recursive subdivision of the original area.

The algorithm probably most closely resembles the Warnock (image space) algorithm in its method of operation. The major difference between them is that the Warnock algorithm performs the x-y sort by screen area, while the new algorithm does the x-y sort by polygon area. Polygon area coherence across the surface of the polygon is preserved as much as possible thereby reducing the number of lateral sorts required. Both algorithms use the techniques of recursive subdivision when necessary. The Warnock algorithm will subdivide until a solution or a preset resolution level has been reached. The new algorithm continues to subdivide only until the proper depth order has been established.

Computation times of the new algorithm vary with both environmental complexity and the visible complexity of the image, and partially depend on the validity of the initial depth sort. For an environment consisting of a number of polygons entirely obscured by a single forward polygon, with a correct initial depth sort, the number of lateral comparisons required is linearly related to the number of input polygons. For environments where every polygon may be partially obscured, but is visible as one piece, the number of lateral comparisons is related to one half the square of the number of input polygons. If few polygons are entirely obscured and many are split into several visible pieces, the relationship to the number of input polygons can be worse than n^2. In practice, the relationship is usually better. Additions to the algorithm as described later may be used to limit the effects of exponential growth rates.

The characteristics of the algorithm allow several options not always available in existing approaches. Of particular importance is the capability for generating perspective images with shadows. Translucency and surface detailing are also possible.

Figure 1. Dice with shadows and surface detail.

Because the output of the algorithm is in the form of polygons as opposed to a raster format, and because the output data does not overlay itself on the image plane, the algorithm effectively solves for hidden lines as well as hidden surfaces. Additional line visibility information can also be stored to enhance CRT displays by eliminating double brightness lines.

HIDDEN SURFACE ALGORITHM

In general, the algorithm selects a polygon shaped area in the x-y plane from the vantage point of the observer and solves the hidden surface problem in that area completely before going on to any other area. This area may itself be subdivided recursively if there is an error in the initial depth sort. Output from the algorithm never overlaps on the x-y plane since each visible area has had all polygons behind it removed. The algorithm proceeds from front to back across the transformed object space, producing portions of the final image along the way and temporarily reversing direction only when an initial depth sort error is detected.

The hidden surface algorithm involves four steps:

a) a preliminary rough depth sort,

b) an x-y polygon area sort to the area of the currently most forward polygon,

c) a depth sort by removal of polygons behind the current forward polygon, and

d) a conclusive depth sort by recursive subdivision when necessary.

The initial sorting step attempts to place the list of input polygons into a rough depth priority order, from those closest to the observer to those furthest away. Any reasonable criterion for a sorting key, such as ordering on the nearest Z value of each polygon, is acceptable. This step is not mandatory but greatly increases the efficiency of the algorithm in later stages. The initial depth sorting operation is only performed once at the beginning of processing and is not repeated.

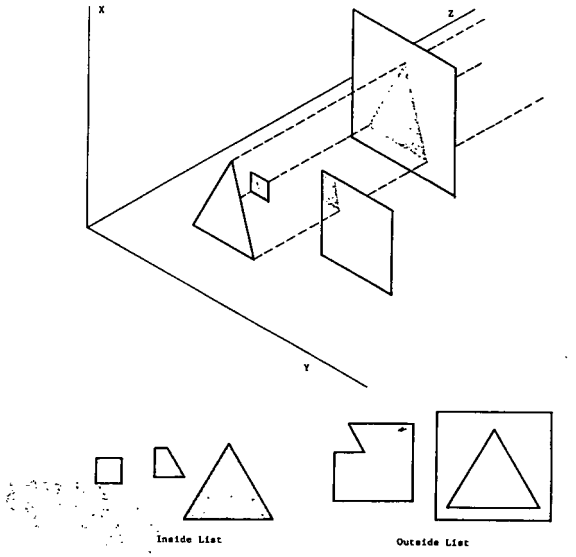


Figure 2. Example of an X-Y clip of a polygon environment by the triangle (leftmost). Two output lists are generated. One list consists of polygons inside the clipping volume created by the X-Y clip to the triangle. The other list consists of polygons outside the clipping volume. Note that non-convex polygons and polygons with holes can be generated by the clipping process.

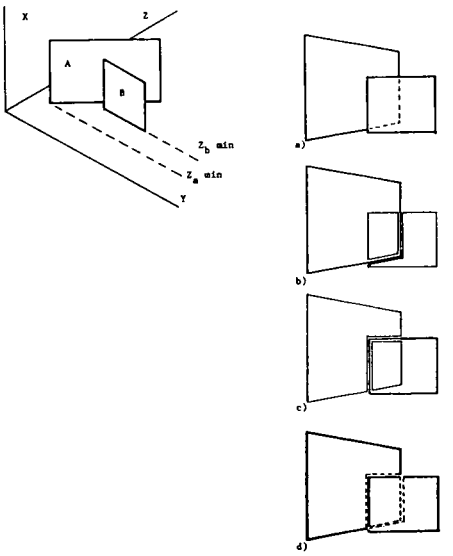


Figure 3. Recursive subdivision caused by initial depth sort error. Polygon A is in front of polygon B by Z-minimum sort but B obscures A.
a) View of scene from observer position.
b) Clip by polygon A. Inside piece of B in front - recursion is performed.
c) Clip of inside list by polygon B, piece of A behind B is removed.
d) Final display.

The first polygon on the sorted input list is then used to clip the remainder of the list into new lists of polygons inside and outside of the clip polygon (Figure 2). In essence, this x-y clipping subdivision is equivalent to lateral area sorting.

The process now examines the inside list and removes any polygons located behind the current clip polygon since they are hidden from view.

Next, if any remaining polygons on the inside list are located in front of the clip polygon, an error in the initial depth sort has been discovered and the algorithm recursively subdivides the region of interest by repeating the clipping process with the offending polygon as the clip polygon and the current inside list as the input list (Figure 3). Finally, when the recursive subdivision has been completed, the inside list is displayed and the algorithm repeats the entire process using the outside list as input. The process is continued until the outside list is exhausted.

It is important to note that the clip polygon actually used as the clipping template is a copy of the original polygon rather than several pieces of its remainder. While keeping a copy of the original increases the storage requirements, the number of clipping edges and the number of clips to perform can be minimized. In this way computation time can be substantially reduced.

After obscured polygons have been removed and before recursive subdivision, a check must be made for the case of cyclic overlap where a single polygon lies both in front of and behind a polygon (Figure 4a). A stack is kept of polygon names which have been used as clipping polygons for this screen area, but have not finished processing because of recursive subdivision. If the algorithm is ready to make a recursive subdivision because a polygon is in front of the current clip polygon, a check is made to see if the name of that polygon is on the stack. If it is, a case of cyclic overlap exists and no additional recursion is necessary since all material behind that polygon has already been removed. This cyclic overlap condition occurs as a result of clipping to the original copy of polygons instead of their remainders. The reduction in the number of clips required outweighs the disadvantage of the simple check required for cyclic overlap. Note that another case of cyclic overlap involving several polygons is implicitly handled by the algorithm (Figure 4b).

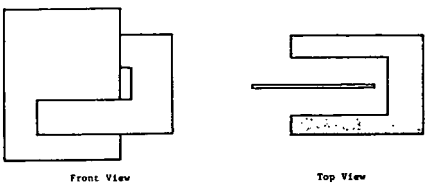


Figure 4a. Cyclic overlap with single polygon.

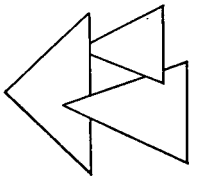


Figure 4b. Cyclic overlap with several polygons.

POLYGON CLIPPING ALGORITHM

Two dimensional polygon clipping is central to the hidden surface removal approach presented.

If only convex polygons were allowed in a scene, clipping a scene by the convex areas of the polygons could quickly yield non-convex areas and holes (Figure 2). Thus even for a restricted environment, a polygon clipper capable of handling concave polygons with holes is necessary.

A clipping algorithm capable of clipping concave polygons with holes to the inside portion of a convex area has been described by Sutherland and Hodgman [10]. The algorithm has the merit of simplicity and is particularly useful for screen subdivision and viewbox clipping. A modified version of this algorithm would clip polygons to a plane and create output polygons on each side of the clipping plane. This version could be used to clip to the borders of a convex polygon yielding intact inside and outside polygons if the planar clip was applied against each edge of the convex clipping polygon. The entire exterior space would then be clipped by infinite planes; however, the effects of each border clip would not be localized and many new exterior polygons would be created. Since this is undesirable in a situation where computational complexity may increase greatly with the number of polygons, another method of clipping has been developed which minimizes the number of polygons created during the clipping process (Figure 5) [14].

The polygon clipper presented is a generalized x-y polygon clipper which is capable of clipping a concave polygon with holes to the borders of concave polygons with holes. Clipping is performed to the borders of the clip polygon. The polygon which is clipped is the subject polygon.

Any new borders created by the clipping of the subject polygon to the area of the clip polygon need only be identical to portions of the borders of the clip polygon. Using this concept, no new edges not already present in either the clip or subject polygon need be introduced and the number of output polygons from the process can be minimized. While the clip is a two-dimensional clip, depth information can be preserved in all output for use by hidden surface calculations.

The creation of new polygons due to intersections of the boundaries of the clip and subject polygons is performed by partial transversals of both boundaries. If the outside borders of the subject polygon are followed in a clockwise direction, the borders of the newly clipped output polygons can be found by making a right turn at each place the two polygons intersect. The process continues until the starting point is arrived at again (Figure 6). The inner or hole borders of the subject polygon must be followed in a counter-clockwise manner in order to use this right turn rule. Note that the borders of the clip polygon used to complete the new polygons must be traversed twice, once in each direction.

Additional techniques are necessary to deal with cases where the borders of the clip polygon do not intersect with borders of the subject polygon but lie completely inside the area of the subject polygon.

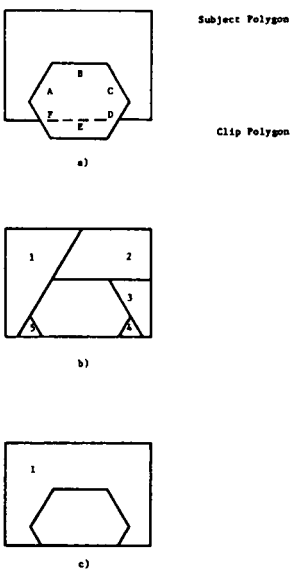


Figure 5. Results of clipping a subject polygon (square) by a clip polygon (hexagon) using different clipping algorithms
a) Subject polygon and clip polygon
b) Result of clipping with infinite planes starting clockwise with edge A
c) Result of general polygon clipper

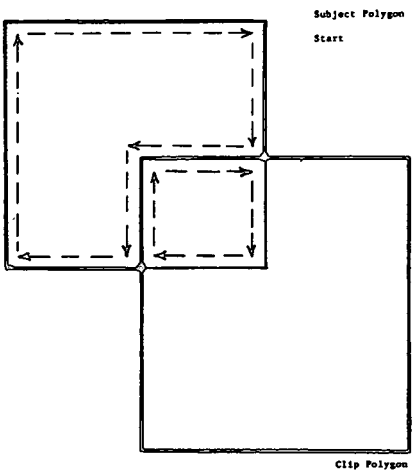


Figure 6. Clipped output polygons caused by intersecting contours can be traced out by following the borders of the subject polygon in a clockwise manner and making a right turn at every intersection of the contours. Note that the borders of the clip polygon are traversed twice in order to form the two output polygons.

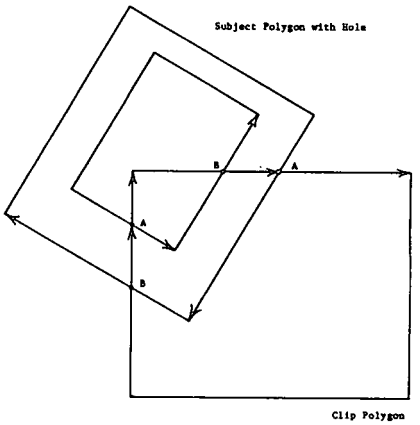


Figure 7. The two types of intersections are based on the directionality of the contours. Note that outside contours are clockwise and holes are counterclockwise.
a) Clip polygon border passes to the left (outside) of the subject polygon border.
b) Clip polygon border passes to the right (inside) of the subject polygon border. Note that types alternate along any contour.

A more detailed description of the clipper follows:

A polygon is defined as an area enclosed by a series of edges consisting of straight lines. These edges may touch upon one another at single non-contiguous points. There are no intrinsic limits as to the number of edges or holes a polygon may have. Contours are the edges or boundaries of a polygon. The term main contour refers to the exterior boundaries of a polygon, while hole contour refers to the interior boundaries of a polygon.

The algorithm represents a polygon as a circular list of vertices, one list for the main contour and one list for each of the holes. The vertices of the main contour are linked in clockwise order and the holes in counter-clockwise order. Using this order, as one follows along the chain of vertices of a polygon, the outside is always to the left while the interior of the polygon is always to the right (Figure 7).

The clip polygon remains the same after the clipping process as before. The subject polygon may be fragmented by the clipping process. The results of the clipping process are two lists of polygons, one of polygons inside the clip polygon area and one of polygons outside the clip polygon (Figure 2). The clipping process is as follows:

1. The borders of the two polygons are compared for intersections. At each intersection a new false vertex is added into the contour chain of each of the two polygons. The new vertices are tagged to indicate they are intersection vertices. A link is established between each pair of new vertices, permitting travel between two polygons wherever they intersect on the x-y plane. If care is taken in placement of intersections where the subject and clip polygon contours are identical in the x-y plane, no degenerate polygons will be produced by the clipping process.

2. Contours which have no intersections are now processed. Each contour of the subject polygon which has no intersections is placed on one of two holding lists. One holding list is for contours inside of the clip polygon; the other is for contours outside of it. Clip polygon contours outside of the subject polygon are ignored. Clip polygon contours inside of the subject polygon in effect cut a hole in the subject polygon, producing two new contours. In this case two copies of the clip polygon contour are made (one in reverse order); one copy is placed on each of the holding lists.

3. Two lists of intersection vertices found on all of the subject polygon contours are formed. The first list contains only those intersections where the clip polygon border passes to the outside of the subject polygon (from the point of view of the subject polygon this occurs whenever the clip polygon passes to the left) (Figure 7). The second list contains those intersections where the clip polygon border passes to the inside (to the right). These two types of intersections will be found to alternate along any given contour and the number of intersections will always be even. This means only one determination of intersection type is necessary per contour.

4. The actual clipping is now performed (Figure 8):

a) An intersection vertex is removed from the first intersection list to be used as a starting point. If the list is exhausted, the clipping is complete; Go to step 5.

b) Follow along the subject polygon vertex chain until the next intersection is reached.

c) Jump to the clip polygon.

d) Copy the chain of polygon vertices until the next intersection vertex is reached.

e) Jump back to the subject polygon.

f) Repeat steps "b" to "e" until the starting point has been reached. At this point the contour of a new inside polygon has just been closed.

The outside polygons can be created by a second pass if the contour vertex chain is double-linked (bi-directional). This is accomplished by starting at intersection vertices from the second intersection vertex list and following the reverse links during traversal of the clip polygon. Otherwise the contours of the outside polygons must be closed during the first pass. This can be done by making a second copy of the vertex chain during the clip polygon traversal (step 4d) in reverse order and attaching its "loose" ends to the unused intersection points at its begin and end locations. A second pass is still needed to find out where these outside contours are. All polygons created are placed on the proper holding lists.

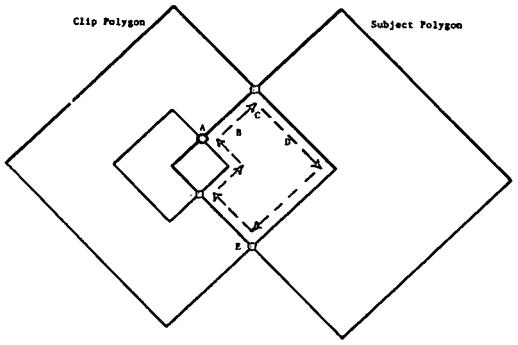


Figure 8. Example of creation of an inside polygon during the clipping process.
a) Intersection vertex starting point of subject polygon
b) Follow subject polygon until next vertex intersection
c) Jump to clip polygon
d) Copy chain of clip polygon vertices
e) Jump back to the subject polygon

5. All holes on the holding lists are attached to the proper main contours. There are several methods of determining which polygons are hole contours. The conceptually simplest method is to test directionality of the contours since main contours will always be clockwise and hole contours will always be counterclockwise. A more efficient method is based on using the highest of a precedence of types of intersections located on a contour. The types used in this precedence are related to the cause of the intersection, such as a main contour intersecting a hole contour, a hole contour intersecting a hole contour, etc. The clipping process is now complete.

EXTENSIONS

Several extensions to the hidden surface algorithm allow greater versatility and efficiency. Of those described below, surface detail, shadowing, and screen subdivision have been implemented.

Surface Detail

Polygons that describe information such as color differences or designs within the boundaries of a planar polygon are referred to here as surface detail. Since they do not affect the boundaries of the polygon to which they belong, they cannot affect hidden surface calculation and should not be included in it. Instead, whenever a polygon is output from the hidden surface computations, the surface detail belonging to the original source of that polygon is clipped to the area of the output polygon. Any of the surface detail within the bounds of the output polygon is then output at this time. This technique can greatly simplify the hidden surface problem for those situations in which surface detail might have otherwise been specified as regular polygons involved in the hidden surface removal process.

Shadowing

The polygon area sort approach lends itself to the generation of shadows because the output of the algorithm is in the form of polygons which are suitable for further processing.

Shadow creation is then reduced to the problem of producing a hidden surface removed view of a scene from the position of the light source. Visible polygons from this point of view are transformed back to the original space and are treated as surface detail of a lighter shade on their source polygons. After this initial shadowing, a normal hidden surface removed view can be taken from any viewpoint to create a correctly shadowed scene. Multiple light sources may be represented using the same process.

Since full machine precision of the output is possible, this particular technique shows promise of being useful not only for display purposes, but also for engineering applications such as energy analyses related to solar heat gain [2].

Translucent Polygons

Translucent polygons can be represented with a slight modification of the depth culling portion of the algorithm. When a translucent polygon becomes the clip polygon, polygons which are behind it should not be removed, but instead tagged and identified as being obscured by that particular translucent polygon. When a polygon is obscured by several translucent polygons, the effect can be accumulated. Since display output is not made for a given area until after the hidden surface removal process for that area is complete, images can be correctly rendered. Partial shadows and shades cast by translucent planes would be handled by the shadowing process in the same manner as normal shadows.

Screen Subdivision

Reducing the exponential rate of the number of sorting steps required for visible surface computation is highly desirable. Some mechanism for dealing with large numbers of polygons or polygons with large numbers of edges which exceed the capacity of main storage should also be provided. The benefits of the polygon area sort approach should be maintained.

By taking the approach of a Warnock style screen area subdivision, the image can be divided into areas each containing a specified maximum number of polygons. Each of these areas can then be processed by the hidden surface removal system separately. This method keeps the number of polygons within storage capacities, while effectively reducing the number of lateral sorting steps to an almost linear relationship with the number of polygons. This is accomplished by reducing the range of the exponential growth factors of the hidden surface removal to the maximum number of polygons allowed within each screen subdivision. The overall number of

lateral sorting steps for hidden surface removal is then almost linear to the number of polygons. The screen subdivision process itself follows an n log n growth rate.

Note that it is possible that the polygons can also be subdivided along the Z axis if their depth exceeds the specified limit. An example would be a large number of screen-sized polygons parallel to the display. Two methods can be used to deal with this case.

The first solution, valuable only for hidden surface removal, uses a technique similar to the frame buffer overlay technique of Newell, et al [7]. The image is subdivided along the Z axis into several "boxes" of space containing a specific number of polygons. The boxes are ordered from back to front and each box is separately solved. The results are output to a frame buffer in order, with the results from each succeeding box overlaying previous results. This technique, while sufficient for most display purposes and quicker than the one presented in the next paragraph, loses some of the advantages of the polygon area sort algorithm and cannot produce fully shadowed images or hidden line removed images.

A more general solution is to divide the scene by Z subdivisions into boxes as before. The farthest box is then solved, and the solution of this last box is treated as surface detail on the plane of a distant polygon parallel to the screen and with the same x-y limits as the box (such surface detail, while associated with the "backdrop" plane for hidden surface removal, can still maintain all depth information for three dimensional output) (Figure 9). This newly created polygon is then added to the next to last box and that box is solved. The process repeats until all of the boxes have been solved. The techniques of surface detail and of consolidation (described later) are used here to reduce the numbers of polygons involved in the hidden surface problem. This solution has the same effect as the first method visually, but is different in that no external overlay techniques are used in order that the solution be entirely expressed in terms of polygons. This difference in the two solutions illustrates one of the primary differences between the Newell, et al, approach and the hidden surface algorithm presented here.

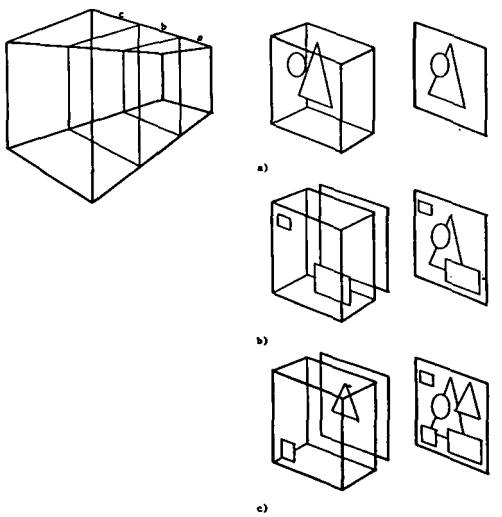


Figure 9. Z subdivision example. The space is divided into 3 "boxes" by Z subdivision to reduce the number of polygons to be solved in the visible surface problem at any one time. The last box (a) is solved and the solution is expressed as a single "backdrop" plane with surface detail. This new plane is solved together with the next box (b) to produce a new backdrop plane, and so on, until the entire space has been solved.

Consolidation

While the hidden surface algorithm takes advantage of polygon area coherence, even greater gains can be achieved by taking advantage of object coherence where several related polygons obscure objects behind them [11]. An example is the case where solid objects are represented as a series of polygons.

Consolidation can be accomplished by creating a new silhouette polygon exactly encompassing all the polygons of the group. Individual component polygons can then be represented as surface detail of the new polygon (Figure 10). This technique is particularly valuable for convex polyhedra, where it is known that the component polygons do not overlap each other after removal of the backward facing polygons.

The advantage of consolidation is that one polygon replaces several polygons in the hidden surface computations, thus reducing the number of polygon clips and depth tests required. Furthermore, since the number of sorting steps required is related to the number of polygons, any method of reducing the number of polygons

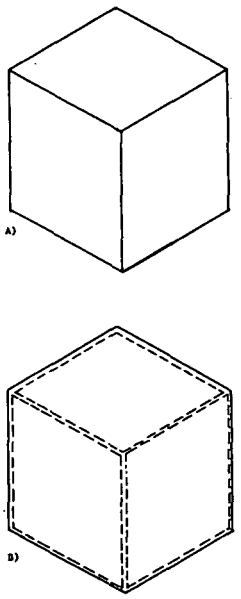


Figure 10. Consolidation of 3 polygons representing a cube (a) into a single polygon with 3 surface detail polygons (shown dotted) (b)

offers the greatest potential in reducing overal computation time.

IMPLEMENTATION

The hidden surface and hidden line removal system described has been implemented at Cornell's Laboratory for Computer Graphics [5]. The program was written in FORTRAN IV and runs on a PDP 11/50 with a floating point processor under the RSX-11M operating system. The available display equipment includes both static and dynamic vector displays, as well as a video frame buffer and color monitor. Some sample photographs of several environments are shown in Figures 1 and 11.

The system was designed as a flexible subroutine package for use in a variety of applications programs. Complete matrix transformation, viewbox clipping, and backplane removal facilities are provided. Once the input data has been defined in an input data file it may be repeatedly transformed for an unlimited number of views as specified by a matrix file. A filming capability for the generation of long sequences of images is also provided.

The system is organized as four separate tasks including the user's task, a data preparation task, the hidden surface removal task, and a monitor task. This system reduces the complexities of user interface requirements and increases the flexibility of the runtime configuration in terms of sequential or concurrent execution and of core usage. All communication between tasks is limited to file access and system messages. System dependent functions are contained only in top level control routines in each task.

The user task is not required to be aware of the details of the configuration or that any file system exists; all interaction takes place through interface routines provided by the hidden surface removal system. Sufficient information is provided in all output files so that any file may be displayed on any vector or raster output device without prior knowledge of the contents of the output file.

CONCLUSION

A hidden surface and hidden line removal algorithm using polygon area sorting has been presented. A generalized polygon clipper, capable of clipping concave polygons with holes to concave polygons with holes, is incorporated allowing polygon format to be maintained for both the input and output. Calculation times are primarily related to the visible complexity of the final image.

Inherent characteristics of the polygon area sorting algorithm give rise to both positive and negative features. Disadvantages are the relative complexity of the clipping and the need to render polygons separately as contrasted to generating the output on a scan line basis. Advantages include flexible polygon representations which can provide for the creation of complex environments and arbitrary output precision. Perhaps the primary advantage is the similarity between the output and input forms, enabling shadow generation and surface details to be treated in a manner consistent with the entire hidden surface removal process.

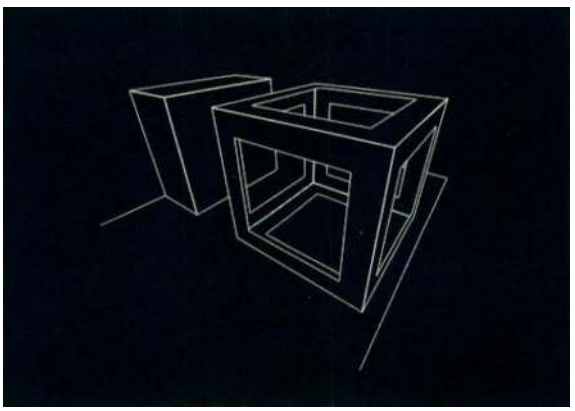
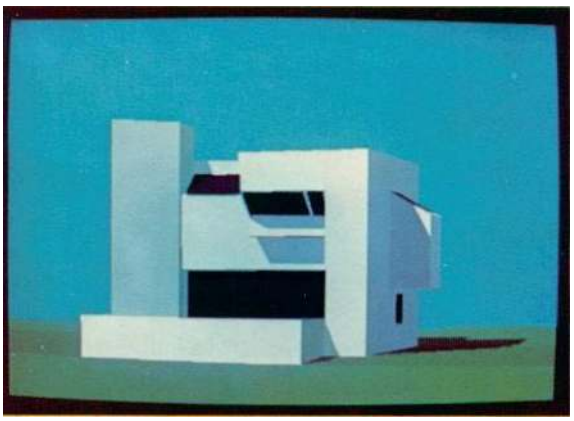
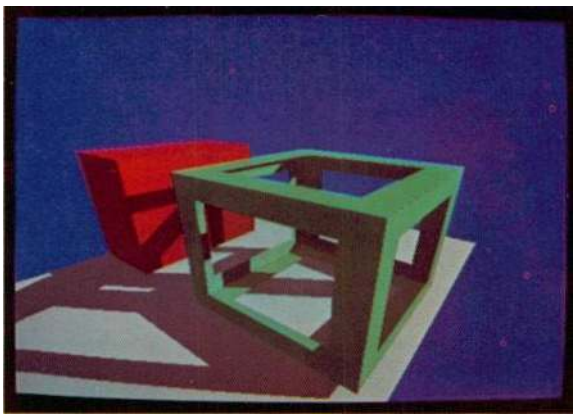


Figure 11. (above) Hidden line image of cubes. (below) Hidden surface image of cubes. (left) House with shadows.





ACKNOWLEDGEMENTS

The research is part of a project sponsored by the National Science Foundation under a grant number DCR74-14694 entitled "Development of Computer Graphics Techniques and Applications" (Dr. Donald P. Greenberg, Principle Investigator). The authors wish to particularly thank Ted Crane and David Bessel for their work in the implementation of the color display system.

REFERENCES

1. Appel, A., "The Notion of Quantitative invisibility and the Machine Rendering of Solids", Proceedings ACM National Conference (1967), pp. 387-393.

2. Atherton, Peter R., "Polygon Shadow Generation", M.S. Thesis, Cornell University, Ithaca, N.Y. (1977), (forthcoming).

3. Bouknight, W.J., "A Procedure for Generation of Three Dimensional Half-toned Computer Graphics Representations", Comm. ACM, 13, 9 (Sept. 1970) pp. 527-536.

4. Galimberti, R., and Montanari, U., "An Algorithm for Hidden-Line Elimination", Comm. ACM, 12, 4, (April 1969), pp. 206-211.

5. Greenberg, Donald P., "An Interdisciplinary Laboratory for Graphics Research and Applications", Proceedings of the Fourth Annual Conference on Computer Graphics, Interactive Techniques and Image Processing — SIGGRAPH, 1977.

6. Myers, A.J., "An Efficient Visible Surface Program", CGRG, Ohio State U., (July 1975).

7. Newell, M.E., Newell, R.G. and Sancha, T.L., "A Solution to the Hidden Surface Problem", Proceedings ACM National Conference, (1972), pp. 443-450.

8. Roberts, L.G., "Machine Perception of Three-Dimensional Solids", MIT Lincoln Laboratory, TR 315, (May 1963).

9. Schumacher, R.A., Brand, B., Gilliand, M. and Sharp, W., "Study for Applying Computer Generated Images to Visual Simulation", AFHRL-TR-69-14, U.S. Air Force Human Resources Laboratory, (Sept. 1969).

10. Sutherland, I.E., and Hodgman, G.W., "Re-entrant Polygon Clipping", Communications of the ACM, Vol. 17, No. 1, (Jan. 1974), pp. 32-42.

11. Sutherland, I.E., Sproull, R.F., and Schumacker, R.A., "A Characterization of Ten Hidden Surface Algorithms", ACM Computing Surveys, Vol. 6, No. 1, (Mar. 1974), pp. 1-55.

12. Warnock, J.E., "A Hidden Surface Algorithm for Computer Generated Halftone Pictures", Dept. Comp. Sci., U. of Utah, (1969).

13. Watkins, G.S., "A Real-Time Visible Surface Algorithm", Comp. Sci, Dept., U. of Utah, UTECH-CSC-70-101, (June 1975).

14. Weiler, Kevin J., "Hidden Surface Removal Using Polygon Area Sorting", M.S. Thesis, Cornell University, Ithaca, N.Y. (1977), (forthcoming).