SHADOW ALGORITHMS FOR COMPUTER GRAPHICS

Franklin C. Crow
University of Texas at Austin
Austin, Texas

ABSTRACT

Shadows are advocated for improved comprehension and enhanced realism in computer-synthesized images. A classification of shadow algorithms delineates three approaches: shadow computation during scanout; division of object surfaces into shadowed and unshadowed areas prior to removal of hidden surfaces; and inclusion of shadow volumes in the object data. The classes are related to existing shadow algorithms and implementations within each class are sketched. A brief comparison of the three approaches suggests that the last approach has the most appealing characteristics.

KEY WORDS AND PHRASES: computer graphics, hidden-surface removal, shadows, shading, raster displays

CR CATEGORIES: 8.2

INTRODUCTION

A major deficiency in most computer-synthesized shaded images to date has been the lack of shadows. Although shadows are unneeded when the light source is coincident with the eyepoint, a fact which was used to advantage in many early implementations, many of the more pressing applications for realistic images (eg. spacecraft docking and aircraft landing simulators) require sunlit images. Quite realistic images of scenes which should contain shadows are now made, but the success of these images relies on the assumption of a diffuse light source such as a cloud-masked sun.

There are situations in which shadows can be important. A cast shadow may make an important piece of equipment virtually invisible under actual conditions even though it shows clearly in a simulation without shadows. Applications of computer graphics to architectural siting problems and environmental impact investigations could require the calculation of shadows for evaluating the need for airconditioning or the availability of solar energy. Most importantly, shadows provide valuable positional information; the shadow cast by one object on another can clarify otherwise ambiguous spatial relationships. Moreover, shadows pose an interesting problem; they should receive more attention than they have been getting.

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

Appel [3] and then Bouknight and Kelley [5] have demonstrated solutions to the shadow problem which are discussed in this paper in the context of a classification scheme for shadow algorithms. Three classes of solution are currently identifiable (there may be further undiscovered classes). Appel, Bouknight and Kelley have shown solutions of one class and algorithms suggesting the other two classes have been proposed but not yet implemented.

The first class of algorithm, demonstrated by Appel, Bouknight and Kelley, detects shadow boundaries as the image is produced by a raster-scan. The edges of cast shadows are found by projecting potential shadowing polygon edges onto the surface being scanned. Shadow edges thus formed are then projected onto the image plane. Upon crossing a shadow edge, the color of a scan segment is changed appropriately.

A second class of algorithm involves two passes through a hidden-surface algorithm, or perhaps a single pass through each of two differing algorithms. The first pass distinguishes shadowed and unshadowed surfaces and divides partially shadowed surfaces by determining hidden surfaces from a viewpoint coincident with the light source. The colors of shadowed surfaces are then modified and a second pass operates on the augmented data from the observer's viewpoint.

The third class of shadow algorithm involves calculating the surface enclosing the volume of space swept out by the shadow of an object, its umbra. The umbra surface is then added to the data and treated as an invisible surface which, when pierced, causes a transition into or out of an object shadow.

A more complete explanation of the three classes follows with suggested implementations in each class. These will be preceded by remarks on modeling of the light source and followed by an attempted comparison of the practical difficulties in implementing the three approaches.

MODELING THE LIGHT SOURCE

Light sources are generally modeled as either points or directions. However, an actual light

source has a finite size, a perhaps irregular shape and a definite position in space relative to the objects to be represented. Light sources of finite size cast shadows involving an umbra and penumbra. The umbra is that part of the shadow space which receives no light from the source; the penumbra, that part which receives light from some part of the source but not all of it. Thus there is a dark central area to any such shadow surrounded by a border area in which a smooth change from shadowed to unshadowed takes place. For an irregularly shaped light source the penumbra could be approximated by a linear variation in shade over a strip of fixed width around the periphery of any umbra. Calculating a penumbra can be expected to significantly increase the effort necessary to represent shadows. Therefore, a point light source or one infinitely far removed (specified by a direction only) will be assumed.

Shadow boundaries are determined by projecting the silhouette of one object onto another. The type of projection which may be used is dependent on the position of the light source. The easiest light source for which to calculate shadows is one that is infinitely far removed since shadow boundaries may be found by an orthographic projection. On the other hand, the calculation of shadow boundaries for a light source which has a position in the object space varies in difficulty with the location. If the source lies outside the field of view, shadow boundaries can be calculated by using the same sort of perspective projection used for image display. However, when the light source lies within the field of view, different methods must be used. Since the conventional perspective transformation is accurate only for a limited field of view, either the space must be divided into sectors radiating from the light source, in which the perspective transform can operate, or more complicated three-dimensional geometric methods must be used.

Projective transforms provide convenience and efficiency. However, it is always possible to define shadow boundaries in the object space by using the light source position and the object silhouette to define a surface and then calculating the intersection of that surface with other objects.

CLASS ONE: SHADOW COMPUTATION DURING SCANOUT

Appel [2,3] and then Bouknight and Kelley [5] have shown methods for rendering shadows which calculate shadow boundaries while scanning the image. Appel detects shadow boundaries by extending his notion of quantitative invisibility. Quantitative invisibility is a count of the number of surfaces hiding a vertex (polygonal objects are assumed). Therefore, a line segment is visible only if all points on it have a quantitative invisibility of zero. Changes in quantitative invisibility along a segment are detected by Appel's hidden surface algorithm and only the visible portions are drawn. This method yields a line drawing.

Shadowed surfaces are determined during a scanning procedure which is also used to shade the line drawing. The scan is executed by generating "cutting" planes through the eyepoint which intersect

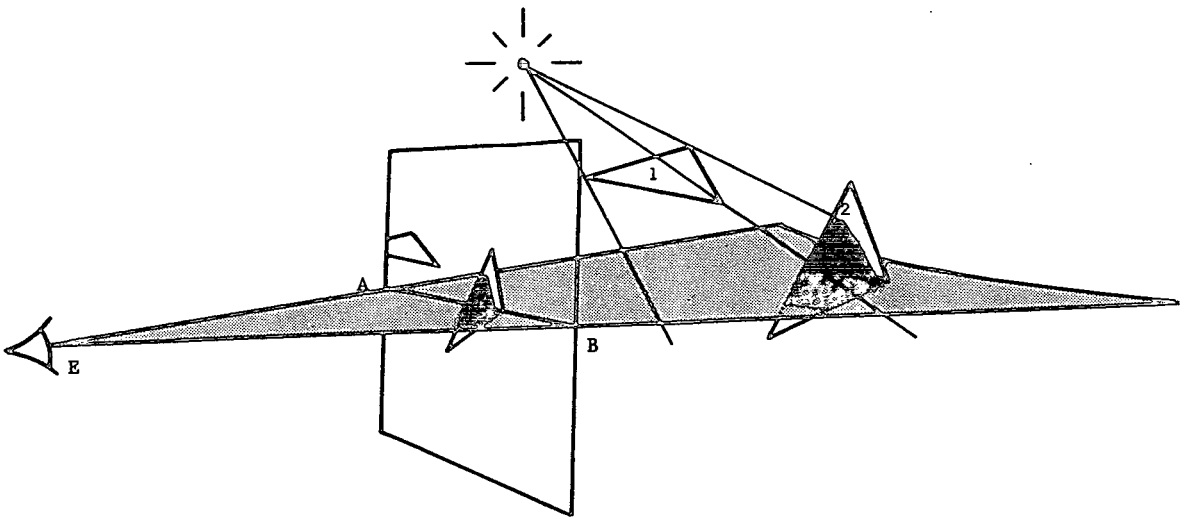

Figure 1: ABE defines a "cutting plane" in Appel's algorithm. Edges of polygon 1 are projected onto polygon 2 to form shadow boundaries which are then projected onto the image plane.

the picture plane in equally-spaced horizontal lines (fig. 1). A set of scan segments is defined by the intersection of visible lines and a cutting plane. The quantitative invisibility with respect to the light source (previously computed for all visible vertices) is then used to determine those segment parts which lie in shadow. Further detail is available in Appel's publications [1,2,3].

Bouknight and Kelley developed a similar method for shadow detection [4,5]. However, they enjoyed an advantage in that their hidden-surface algorithm was already based on a scanning process. A secondary scan was used to detect shadow boundaries calculated by projecting edges upon the surfaces being scanned. The primary scan followed a raster pattern in image space which generated the secondary scan, the corresponding path across the visible surface in object space. Therefore, shadow edges occurred where edges of other polygons projected onto a secondary scan segment.

A procedure for finding all polygons which could cast shadows on a given polygon was used to diminish the edge-projection computation. This routine transformed all polygons to a pseudo-spherical coordinate space with its origin at the light source. Polygons were then tested for overlap and a linked list was formed for each polygon, enabling the other polygons which might cast shadows on it to be easily found (In figure 1, polygon 2 would be linked to polygon 1). An expansion of this overlap test leads to the second class of algorithms as will be seen below.

The general approach exemplified by Bouknight-Kelley can be analyzed as two basic operations: (1) the shadow priority ordering of polygons and (2) the calculation of projected shadow boundaries. It is worth noting that these two operations are independent of the hidden surface algorithm used for display and this could be applied to virtually any polygon-based algorithm.

Many variations on the Bouknight and Kelley algorithm can be developed. For example, the computation for their pseudo-spherical overlap test grows as the square of the number of polygons. It would therefore be advantageous to divide the viewable object space into sectors radiating from the light source position. This would allow all polygons in a sector to be sorted to a shadow priority order without reference to other sectors. Shadow priority determination requires a special sort such as the one used by Newell et al [9]. The behavior of this algorithm (also obeying an N-squared growth law) is discussed by Sutherland et al [12].

Under favorable conditions, sectorization can change the N-squared growth law of the Bouknight-Kelley (or Newell et al) priority scheme to a linear growth law. The growth of the sectored scheme is proportional to S (N/S)**2 where S is the number of sectors and N is the number of polygons (as long as the general distribution of polygons in space remains similar). If N/S is held constant by increasing the number of sectors proportionally with the number of polygons, the priority stage obeys a linear growth law. However, this growth rate is complicated in the limit by the fact that when sectors become so small that a high percentage of polygons overlap sector boundaries, the effective number of polygons increases. This is due to the fact that a polygon overlapping two sectors must be considered in both. However, the potential linear growth rate makes this an attractive approach both here and in the design of sectorable algorithms in general.

The second basic operation, the calculation of shadow boundaries, requires a process akin to clipping. The polygon under consideration must be used as a window against which polygons of a higher priority are clipped. The growth rate of this operation is proportional to the product of the number of edges in shadowed polygons and the number of edges in higher priority polygons, again an N-squared growth rate. However, sectorization can again provide an overall linear growth rate under favorable conditions. (It should be noted that the linked list used here by Bouknight and Kelley is in some sense an optimized sectorization.) Two factors can substantially reduce the constant of proportionality in the growth law: (1) shadow calculation need only be carried through for visible polygons and (2) the calculation may terminate when a polygon is discovered to be completely shadowed.

In closing this section, it should be reemphasized that in all shadow algorithms, a large amount of computation can be saved by considering only the silhouette of a shadowing object instead of each of its polygons individually. This restricts searching to only those edges which cause visible shadow boundaries.

A SECOND CLASS: THE TWO-PASS APPROACH

It would appear that a hidden-surface algorithm could be used to detect which surfaces are hidden from the light source as easily as those which are hidden from the eye. However, to be useful, the algorithm must yield information which can be used to generate an image, as seen from the eyepoint, in a subsequent pass. This restriction limits the class of applicable algorithms.

Sutherland, Sproull and Schumaker proposed that hidden-surface algorithms could be divided into object-space algorithms and image-space algorithms [12]. This distinction turns out to be important since the determination of shadow boundaries must be made in object space so that the resulting information can be merged into the data to be sent to the display algorithm. Thus image-space algorithms which depend on the limited resolution of the display medium to ease the determination of hidden surfaces are inappropriate for this application.

The algorithms characterized by Sutherland et al [12] as operating strictly in object space all suffer from discouraging growth laws (computation increasing with the square of the amount of data). Furthermore, where polygons are considered, they are not treated as entities but broken into individually treated sides. To create shadows, the polygons must be treated as entities so that they may be divided by shadow boundaries and returned

to the data base as smaller polygons to be fed to the display algorithm. Eliminating the object-space algorithms leaves the algorithm shown by Newell, Newell and Sancha [9]. This algorithm provides many useful techniques for splitting polygons and determining overlap but the eventual determination of what part of which surfaces are hidden is done by overwriting in image space.

Sutherland has proposed another algorithm which is more applicable to the problem [11]. Using clipping techniques, a binary sort is executed sending polygons and parts of polygons lying on one side of a line out on one stream and those lying on the other side off on a different stream. Such a process is, of course, exactly what is needed for determining divisions between shadowed and unshadowed parts of surfaces. Furthermore, Sutherland's algorithm holds the promise of a reasonable growth rate. Since the algorithm operates by recursive subdivision of the viewed space via a 2-dimensional binary sort of the data, an N log N growth law may be achievable.

Sutherland also proposed improvements based on considering only "contour" edges in the subdivision process. Contour edges are those edges which separate frontfacing and backfacing polygons at those places where the surface curves behind itself or else edges which lie at the extremes of the surface, for surfaces which don't close on themselves [1]. Thus any area lying within the bounds of a set of contour edges for a single surface can be treated as a unit assuming that the bounded surface is the frontmost surface in the bounded area.

Clark has proposed a general scheme for approaching hidden-surface algorithms which involves recursive descent through a hierarchical data description [7]. The combination of this approach with Sutherland's notion of clipping to contour edges appears to hold promise for an interesting shadow algorithm. If environmental restrictions are imposed so that objects must be broken into linearly separable sub-objects and groups of such objects may also be linearly separated, then an algorithm may be implemented along the following lines.

The first step of this algorithm would use sorting techniques akin to those of Newell, Newell and Sancha to establish a front to back priority ordering of the surfaces under consideration. The hierarchical approach proposed by Clark may be superimposed to first order objects or groups of objects, then to establish an order within such objects or groups. Newell has recently suggested an algorithm for sorting objects to a depth priority which could be applied here [10].

Note that, for purposes of shadow detection, the silhouette of an object may be used to define a "blot", or anti-window, under a perspective projection. Anything lying within the blot and farther from the light source is clearly in shadow. Therefore, the algorithm can proceed by augmenting a collection of blots using the silhouette of each convex sub-object in turn. Moving away from the light source, surfaces hidden by the blot are marked as shadowed. Other surfaces contribute the unshadowed portion of their silhouettes to the collection and are themselves clipped into shadowed and unshadowed portions.

Finding object silhouettes for polygonal objects is eased by a data structure providing links between adjacent polygons. Such a structure is detailed in [6,8]. Since the silhouette is formed solely from the contour edges and all contour edges on a convex surface must lie on the silhouette, the determination of the silhouette consists of finding the closed loop of contour edges.

Strings of contour edges may be formed straightforwardly. First, all polygons must be tagged as frontfacing or backfacing from the light source point of view. Secondly, the neighbor polygons for each frontfacing polygon must be checked; where backfacing adjacent polygons are found, the associated edge must be tagged as a silhouette edge. Lastly, silhouette edges may be linked together by using the adjacent frontfacing polygons to search for additional silhouette edges connected to a known silhouette vertex. For a convex object, a single such string of edges will form the silhouette (fig. 2).

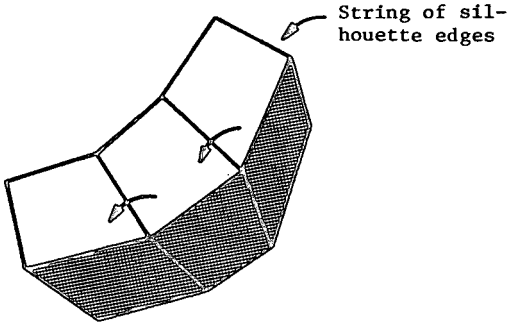

Figure 2: Strings of silhouette edges may be formed by searching for adjacent frontfacing polygons with contour edges.

By using the scheme proposed by Clark, the calculation of some object silhouettes may be avoided. Following the hierarchical division of the data (groups, objects, sub-objects), tests using "bounding boxes" may be used to determine which groups may overlap from the light source point of view. The bounding box is defined by the range of the object vertices over height and width (fig. 3). Any time the bounding box of an object is found to lie totally within the silhouette of an object of higher priority, the first object is in shadow. Similarly, if the bounding box of a convex object (consisting of a single sub-object) fails to overlap any others then the object is clearly not involved in any shadows. In these cases, there is no reason to compute the silhouette.

The shadow algorithm may be driven by the hierarchical organization of the data. Thus groups of objects may be processed in priority order, closest group first. Within each group, objects will be treated in priority order and within each ob-

ject, sub-objects will be treated in priority order. Overlap tests can first determine whether groups may interact. If so, the bounding boxes must be passed to the next lower level of the hierarchy. Overlap tests are then applied to the objects within the group and finally to the sub-objects of each object. If the bounding box of a sub-object overlaps none of the bounding boxes passed down through the hierarchy, it may be ignored. Otherwise its silhouette is computed.

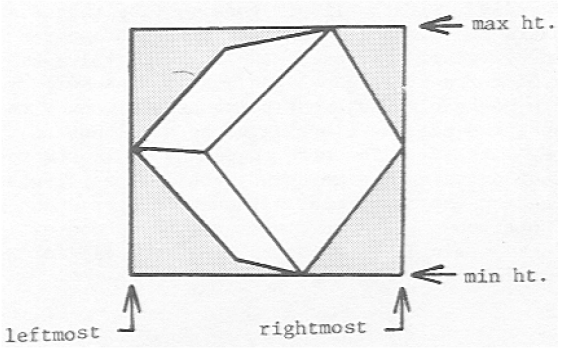

Figure 3: The "bounding box"

Having computed the silhouette of the highest priority sub-object for which it was required, intra-object shadow boundaries may be computed. This can be done by clipping the polygons of lower priority sub-objects to the silhouettes of higher priority sub-objects. If any lower priority sub-object is completely shadowed, it may be tagged as such and ignored in subsequent overlap tests. Partially shadowed sub-objects are clipped into shadowed and unshadowed portions; partial silhouettes are then computed based only on the unshadowed portion. Completely unshadowed sub-objects merely have their silhouettes calculated.

As the algorithm works its way down the priority-ordered list of objects, a minimal set of convex blots is built up, each blot with an associated bounding box. As lower priority objects are treated, they will first be clipped by the higher priority blots then the remaining polygons will be used to compute partial silhouettes to be added to the set of blots. It may be useful to include a provision to absorb a set of blots into a single one in the case where a lower priority sub-object is large enough to provide an enveloping silhouette. However, the overhead in checking for this case may well prove to outweigh the benefits.

Where several groups of objects overlap, an extremely large set of blots is likely to have been built up by the time lower priority groups are treated. To avoid undue growth of computation, the set of blots and untreated data should be sectored so that spatially separate areas may be treated independently. Using the information provided by the bounding boxes, sectorization becomes trivial. It may even be advantageous to resector several times as the collection of blots develops. Also, sectoring based on the bounding boxes of lower priority objects would allow blots which will no longer be needed to be discarded.

Of course this approach depends heavily on a well-conditioned environment (convex sub-objects). It is not clear whether (1) the algorithm could be easily extended to the general case and (2) whether data generation and object modelling techniques could be forced to always deliver such well-conditioned data.

In general, the testing sequences described above will increase in cost with the square of the number of objects involved. However, the division of the data into a hierarchical arrangement and the use of sectoring when the number of blots gets large should minimize the number of necessary tests. Again it must be pointed out that if the light source lies in or near the field of view from the eye position, the space will have to be sectored for shadow determination so that perspective projections may be used. This approach counts heavily on the ease of determining backfacing and frontfacing polygons and on overlap tests both of which are much more easily done after a perspective transform.

Once the shadowed polygons have been determined, any hidden-surface algorithm may be used to generate the eyepoint image from the augmented data. Therefore, an advantage to the two-pass approach is that the process of defining shadows is totally independent of the later process of picture generation; the shadow process may run concurrently in pipeline with the picture-generation process. Note that, given two processors, there is little point in making the shadow detection algorithm more efficient than the display algorithm if they are to be run separately and concurrently. Also, given a static environment and a fixed light source, shadows need be computed only once for a large number of eyepoint positions. In that situation, the efficiency of the shadow algorithm becomes much less important.

THE THIRD CLASS: PROJECTED SHADOW POLYGONS

Shadows may be defined by the projection of edges onto surfaces as in the first and second classes or they may be defined by the volume of space they encompass. The last class of shadow algorithm includes shadow volumes in the hidden-surface computation by adding their surfaces to the data. Assuming a polygonal object, the shadow surface is given by planes defined by contour edges and the light source position. Each such edge defines a polygon whose boundaries are the edge itself, the two lines defined by the light source position and the endpoints of the edge and the bounds of the field of view (fig. 4). The sense of the polygon must be maintained so that the near surface of a shadow volume (frontfacing polygons) may be distinguished from the far surface (backfacing polygons). Thus the polygons facing the light source plus the set of projected shadow polygons for an object define its shadow volume.

Shadow polygons may be treated just like the rest of the data when applied to a scanning hidden-sur-

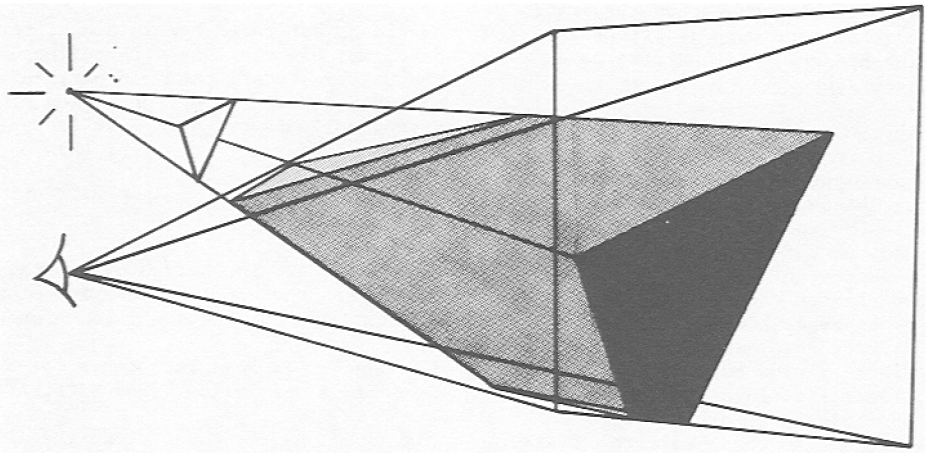

Figure 4: Shadow polygons clipped to the field of view

face algorithms; only the shading for visible surfaces must handled differently. Shadow polygons are themselves invisible, thus they do not count in the determination of visibility. However, the depth order of shadow surfaces and visible surfaces determines shadowing. A frontfacing shadow surface puts anything behind it in shadow while a backfacing shadow surface cancels the effect of a frontfacing one. For example, a post or column might cast a shadow surface consisting of a single polygon pair. Any surface lying between those two shadow polygons would be in shadow while surfaces lying in front of or behind both polygons would be shaded normally.

If the frontmost shadow surface is backfacing, then everything in front of it is in shadow; if the rearmost shadow surface is frontfacing, then everything behind it is in shadow. These cases can occur where the eyepoint is in shadow or a surface casts a shadow over a large part of the field of view. Therefore, surfaces are shadowed whenever they lie in front of a backfacing frontmost shadow polygon or the surface depth count is such that more frontfacing than backfacing shadow polygons have been pierced. Shadow boundaries are formed where a visible surface intersects a shadow polygon.

Modification of a scanning hidden-surface algorithm to handle shadow polygons involves changing only the inner loops where shading must be calculated. Two properties of shadow polygons may be used to simplify computation. First, shadow polygons are invisible. Therefore, scan lines involving only shadow polygons may be ignored. Second, shadow polygons formed by projection of contour edges cannot intersect one another (as long as a single light source is used). Therefore the depth ordering of such polygons is constant.

Using a scanning algorithm of the Bouknight variety (see [4,12] for detailed views of this type of algorithm) shadow polygons may be treated just as other polygons through the y-sort and x-merge procedures. Scanning algorithms generally require maintaining a depth-sorted list of all scan segments which would be pierced by a ray from the eyepoint through the current position on the scanline. Shadow polygons will frequently cause quite lengthy scan segments greatly increasing the average depth complexity over an image. As Sutherland et al [12] pointed out, increased depth complexity may well severely hamper the performance of scanning algorithms.

Shadow polygons, however, need be considered only under certain circumstances during the production of scan segments. The fact that shadow polygons may not intersect allows profitable use of scanline to scanline coherence. The depth ordering of shadow polygons will change only when new polygons are added or old ones deleted as the scan moves down the image. Thus the process of rebuilding and updating the depth-sorted list of shadow polygons can be largely eliminated. The list need only be built where object segments occur. Therefore, a scanline with no object segments can be ignored. Since the depth ordering doesn't change, it will only be necessary to calculate the depth to a shadow surface when it must be compared to the depth of a visible surface.

The priority list of shadow polygons need only be searched when the visible surface in the image changes. Once it is discovered which shadow polygons bound the currently visible surface (in depth) then only those polygons need be checked for possible intersections. Therefore, although there may be considerable depth complexity due to shadows, a depth complexity of two to three shadow surfaces should be all that really affect computing time. However, many of the images made today have an average depth complexity of less than three. Thus a significant increase in the time needed for the scanning process may result from the addition of shadow polygons. However, this effect may prove to be less significant as more highly complex environments are attempted.

A COMPARISON OF THE THREE CLASSES

Comparisons can be made with respect to the additional difficulty involved in representing shadows using each of the above approaches. Three bases

for comparison are used: the additional data storage required; the additional computation required; and the difficulty of the necessary additional software. A scanning hidden-surface algorithm is assumed for this discussion.

At first glance only the second and third classes of algorithm appear to require additional data storage. The two-pass approach requires that surfaces be split along shadow boundaries, or at least that shadow boundaries be included in the data; the shadow polygon approach requires the storage of perhaps numerous shadow polygons. However, neither of these two classes requires the entire scene description to be available for the hidden-surface calculation; backfacing surfaces and data lying outside the field of view may be discarded. On the other hand, the first class of algorithm requires that all object data be available in its original form at all times so that projected shadow boundaries may be calculated during scanout. Therefore, the space left over for use as temporary storage by later stages of the hidden-surface algorithm is severely reduced. It must be concluded that the two-pass approach requires least additional storage, shadow polygons require somewhat more and calculation of shadows during scanout requires by far the most.

Assuming that it will always be more efficient to use only silhouettes for calculating shadow boundaries, the projected shadow polygon approach appears to cause the smallest increase in necessary computation. The definition of shadow polygons is straightforward once silhouette edges are found, and the additional computation in the scanning process is minimized by taking advantage of the special properties of shadow polygons. Furthermore, both other approaches require methods which obey less desirable growth laws. Shadow calculation during scanout requires additional computation to determine which surfaces may cast shadows on each other and then requires the calculation of the lines separating shadowed and unshadowed areas by operations on the object-space data. Bouknight and Kelley reported roughly doubled computation time to include shadows in very simple scenes. The two-pass approach, in its turn, requires an additional solution of the hidden-surface problem. However, since only silhouette edges need to be considered, the first pass should be simplified.

The complexity of the additional software required also appears to be smallest for the projected shadow polygon approach. Algorithms of both the first and second classes require significant new software. However, it could be argued that once a suitable hidden-surface algorithm is available for the two-pass approach, the software for the first pass is just a subset of that needed for the second pass and thus no additional software is needed.

Given a situation in which a scanning hidden-surface algorithm is available, it appears that the shadow polygon approach offers the best solution. However, starting from scratch, there is no clear-cut best choice. Certainly there is much to be learned by implementing an algorithm of any class.

ACKNOWLEDGEMENTS

The ideas expressed herein arose, for the most part, in conversations with colleagues while at the University of Utah. In particular, Ivan Sutherland suggested to me the notion of projected shadow polygons and also provided valuable comments on an earlier draft of this paper.

REFERENCES

[1] Appel, A., The Notion of Quantitative Invisibility and the Machine Rendering of Solids, Proceedings ACM 1967 National Conference.

[2] Appel, A., Some Techniques for Shading Machine Renderings of Solids, 1968 SJCC, AFIPS Vol. 32.

[3] Appel, A., On Calculating the Illusion of Reality, IFIP 1968.

[4] Bouknight, W. J., A Procedure for the Generation of 3-D Half-Toned Computer Graphics Presentations, CACM, Vol. 13, no. 6, Sept. 1970.

[5] Bouknight, W. J. and Kelley, K., An Algorithm for Producing Half-Tone Computer Graphics Presentations with Shadows and Moveable Light Sources, 1970 SJCC, AFIPS Vol. 36.

[6] Bui Tuong Phong and Crow, F. C., Improved Rendition of Polygonal Models of Curved Surfaces, Proc. of the 2nd USA-Japan Computer Conf., 1975.

[7] Clark, J. H., Hierarchical Geometric Models for Visible Surface Algorithms, CACM, Vol. 19 no. 10, Oct. 1976.

[8] Crow, F. C., The Aliasing Problem in Computer-Synthesized Shaded Images, Dept of Computer Science University of Utah, UTEC-CSc-76-015, March 1976. (abridged version to appear in CACM)

[9] Newell, M. G., Newell, R. G. and Sancha, T. L. A Solution to the Hidden-Surface Problem, Proceedings of the 1972 ACM National Conference.

[10] Newell, M. G., The Utilization of Procedural Models in Digital Image Synthesis, Department of Computer Science, University of Utah, UTEC-CSc-76-218, Summer 1975.

[11] Sutherland, I. E., Polygon Sorting by Subdivision: A Solution to the Hidden-Surface Problem, Unpublished, 1973.

[12] Sutherland, I. E., Sproull, R. F. and Schumaker, R. G., A Characterization of Ten Hidden-Surface Algorithms, Computing Surveys, Vol. 6, No. 1, March 1974.