to produce an image of a highly glazed patterned teapot, as in Figure 10.

Resource Requirements

The images shown in this paper were all generated on a PDP-11/45 computer having a 256K-byte random access frame buffer which was used as the depth buffer. The main routines were written in Fortran and the critical parts were written in assembly language. The computation time of the extended subdivision algorithm is roughly proportional to the area covered by visible objects. Images of nontextured objects of the type used in this paper take about 25 minutes. The addition of texture or reflection increases this time by about 10 percent. All images have a resolution of 512×512 picture elements.

Conclusions

By refining and extending Catmull's subdivision algorithm, images can be generated having a far higher degree of naturalness than was previously possible. These generalizations result in improved techniques for generating patterns and texture, and in the new capability for simulating reflections.

References

1. Bui-Tuong Phong. Illumination for computer generated images. *Comm. ACM 18*, 6 (June 1975), 311–317.
2. Catmull, E.A. Computer display of curved surfaces. Proc. Conf. on Comptr. Graphics, Pattern Recognition, and Data Structure, May 1975, pp. 11–17 (IEEE Cat. No. 75CH0981-1C).
3. Crow, F.C. The aliasing problem in computer-synthesized shaded images. Tech. Rep. UTEC-CSC-76-015, Dep. Comptr. Sci., U. of Utah, Salt Lake City, Utah, March 1976.
4. Forrest, A.R. On Coons and other methods for the representation of curved surfaces. *Computer Graphics and Image Processing 1* (1972), 341.
5. Gouraud, H. Computer display of curved surfaces. Tech. Rep. UTEC-CSC-71-113, Dep. Comptr. Sci., U. of Utah, Salt Lake City, Utah, June 1971.
6. Newell, M.E., Newell, R.G., and Sancha, T.L. A solution to the hidden surface problem. Proc. ACM 1972 Ann. Conf., Boston, pp. 443–450.
7. Oppenheim, A.V., and Schafer, R.W. *Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, N.J., 1975, pp. 26–34.
8. Sutherland, I.E., Sproull, R.F., and Schumaker, R.A. A characterization of ten hidden-surface algorithms. *Computing Surveys 6*, 1 (March 1974), 1–55.
9. Warnock, J.E. A hidden-line algorithm for halftone picture representation. Rep. TR 4-15, Dep. Comptr. Sci., U. of Utah, Salt Lake City, Utah, 1969.
10. Watkins, G.S. A real-time visible surface algorithm. Tech. Rep. UTEC-CSC-70-101, Dep. Comptr. Sci., U. of Utah, Salt Lake City, Utah, June 1970.

Graphics and Image Processing

Hierarchical Geometric Models for Visible Surface Algorithms

James H. Clark
University of California at Santa Cruz

The geometric structure inherent in the definition of the shapes of three-dimensional objects and environments is used not just to define their relative motion and placement, but also to assist in solving many other problems of systems for producing pictures by computer. By using an extension of traditional structure information, or a geometric hierarchy, five significant improvements to current techniques are possible. First, the range of complexity of an environment is greatly increased while the visible complexity of any given scene is kept within a fixed upper limit. Second, a meaningful way is provided to vary the amount of detail presented in a scene. Third, "clipping" becomes a very fast logarithmic search for the resolvable parts of the environment within the field of view. Fourth, frame to frame coherence and clipping define a graphical "working set," or fraction of the total structure that should be present in primary store for immediate access by the visible surface algorithm. Finally, the geometric structure suggests a recursive descent, visible surface algorithm in which the computation time potentially grows linearly with the visible complexity of the scene.

Key Words and Phrases: visible surface algorithms, hidden surface algorithms, hierarchical data structures, geometric models

CR Categories: 5.31, 8.2

A version of this paper was presented at SIGGRAPH 76: The Third Annual Conference on Computer Graphics, Interactive Techniques, and Image Processing, The Wharton School, University of Pennsylvania, July 14–16, 1976.

Author's address: Information Sciences, University of California, Santa Cruz, CA 95064.

1. Introduction

1.1 Background

Early research in computer graphics was concerned with the organization and presentation of graphical information in the form of real-time line drawings on a CRT. Many of the concepts of structuring graphical information were developed by Sutherland in Sketchpad [19], and the line-drawing graphical displays that resulted from his early research remain the most widely used today. With the development of integrated circuit technology, research interests shifted to producing very realistic, shaded, color pictures of the visible parts of complex three-dimensional objects. Because of the desire to utilize television technology, the algorithms for producing these pictures generated output for a raster CRT. The pioneering works in this area were by Schumacker et al. [18] and Wylie et al. [23].

Computer produced pictures now provide one of the most direct and useful ways of communicating with the computer. The ability to produce shaded pictures that illustrate mathematical functions and physical properties of mathematical models is of incontestable value in both research and education. With the development of computer controlled simulators, a real-time computer displayed environment is now used to train pilots of aircraft [11, 16], spacecraft [9] and ocean vessels [2]. Other significant uses of computer pictures include computer aided design [4], modeling of chemical structures [22], and computer animation [7, 12]. With this increased value of computer generated pictures, comes an increasing need to devise efficient algorithms that improve the realism and enhance the descriptive power of these pictures.

1.2 Motivation for New Research

The underlying motivation for new research on computer produced pictures is to either enhance the realism of the pictures or improve the performance of the algorithms that generate them. Most recent research has addressed a combination of these issues.

There are three basic approaches to improving picture quality. The first is to devise clever ways to add information value to a scene without significantly increasing the total amount of information in the database for the scene, for example, without increasing the number of polygons used in representing the objects. Approaches of this type usually make subtle changes to the visible surface and shading algorithms that result in greatly improved pictures. Examples are the improvements to shading algorithms devised by H. Gouraud [10] and Bui-Tuong Phong [15].

The second approach is to employ more refined mathematical models for the objects being rendered and to devise algorithms that can find the visible surfaces using these models. The goal of these methods is to model smooth surfaces with surface patches, such as Coons patches [5] or B-splines [4, 17], rather than with clusters of polygons, and still not increase the size of the database. Catmull's [3] ingenious algorithm is an example of this approach. The benefit of these methods is that an arbitrarily refined description of the model is present, thus allowing much better renditions of contours and shading. The disadvantage is that because of nonlinear mathematics, the algorithms are less efficient than polygon-based algorithms.

The third approach is to increase the information in the database and employ more structured methods for handling the increased information. The motivation for this approach is that the information value of a scene grows in proportion to the amount of information in the database for the scene. Newell's [13] algorithm is an example of this approach.

The structured approach appears to be the most promising of these approaches since it potentially improves both picture quality and algorithm performance. However, there are several problems associated with this approach. First, increased complexity of a scene, or increased information in the database, has less value as the resolution limits of the display are approached. It makes no sense to use 500 polygons in describing an object if it covers only 20 raster units of the display. How do we select only that portion of the data base that has meaning in the context of the resolution of the viewing device? Second, how do we accommodate the increased storage requirements of this additional information? We might, for example, wish to model a human body to the extent that a closeup view of the eye shows the patterns of the iris, yet such a fine description of the entire body will indeed require large amounts of store. Third, how much information must be presented to convey the information content of the scene? In other words, we would like to present the minimal information needed to convey the meaning of what is being viewed. For example, when we view the human body mentioned above from a very large distance, we might need to present only "specks" for the eyes, or perhaps just a "block" for the head, totally eliminating the eyes from consideration. The amount of information "needed" can be the subject of psychological debate, but it is clear that even coarse decisions will yield more manageable scenes than attempting to use all of the available information.

These issues have not previously been addressed in a unified way. The research described here represents an attempt to solve these and related problems.

2. Summary of Existing Algorithms

Visible surface algorithms may be categorized according to whether they employ polygons, parametric surface patches, or procedures as the underlying method of modeling the surfaces they render. The most thoroughly studied types of algorithms use polygons. However, because of the shortcomings of representing smooth surfaces with faceted clusters of polygons, some research interest has recently been devoted to parametric surface algorithms, which allow higher degrees of continuity than just positional continuity. The algorithms for these different modeling methods will be discussed separately.

2.1 Polygon-Based Algorithms

A highly informative survey of existing polygon-based visible surface algorithms has been written by Sutherland et al. [20]. As they point out, a convenient way to classify these algorithms is according to the order in which they sort the image space polygons that are potentially visible in a scene. The basic difference between the major algorithms is in whether they sort in depth (from the viewpoint) before the vertical-horizontal sort, or vice versa.

Depth-first sort. The most significant algorithms to use this sorting order are due to Schumacker et al. [18] and Newell et al. [14]. Schumacker utilizes this order along with a polygon clustering concept to achieve a coherence from one frame to the next, while Newell utilizes it to render translucent images. By first computing a priority ordering of polygons according to their image space distance from the screen, they are able to establish which polygon *segments* on a given scan line have visibility priority.

Newell uses this information to write those segments with a lesser priority into a scan-line buffer before writing in those with a greater priority. Thus greater priority segments which are from translucent polygons only modify the intensity values in the buffer rather than completely overwriting them. While there is clearly a considerable overhead in writing into the buffer segments that might eventually be obscured, some beautiful pictures have resulted from this work.

Schumacker's goal is to produce real-time picture sequences. Rather than writing the polygon segment information for a scan-line into a buffer according to its priority, a set of priority-ordered hardware registers are simultaneously loaded with the priority-ordered segment information. Then as the scan line is displayed, the register information is counted down and a combinational-logic network selects the appropriate highest priority register according to its lateral displacement on the screen. This approach requires a separate set of registers for each polygon segment that intersects the scan line. Nonetheless, it represents the first real-time solution to the visible surface problem [9].

There are two very significant features to Schumacker's work. First, he makes use of a priori knowledge of the database to compute fixed priorities for clusters of polygons. If the polygons in a group of polygons are not subject to changes in relative placement, they form a *cluster* and may be assigned fixed priorities which work no matter from where the cluster is viewed. Thus part of the priority ordering is fixed with the environment and need not be recomputed each frame. Second, he shows that if the environment is restricted so that the clusters are linearly separable, an intercluster priority can be established that does not change unless the viewpoint crosses one of the separating planes; hence, the priority ordering remains fixed from one frame to the next unless one of the planes is crossed.

This work by Schumacker and coworkers represents the only visible surface algorithm to make use of both structured information (clustering) and frame to frame coherence (relatively constant intercluster priority). These very important concepts will be discussed in more detail later.

Depth-last sort. The algorithms that use this sorting order have been devised by Watkins [21], Bouknight [1], and Wylie et al. [23]. They are referred to as scan-line algorithms and differ only in their use of various image-space coherence properties. All three first perform a vertical bucket (radix) sort of polygon edges according to their uppermost vertices. Then for each scan line, the various polygon segments on that scan line are sorted according to their horizontal displacements. The depth sort is deferred until last under the assumption that the initial two sorts will decrease the number of depth comparisons needed to determine final visibility.

Of the three approaches, Watkins' is the most economical because of its uses of scan-line coherence and a logarithmic depth search. The assumption of scan-line coherence is that in going from one scan line to the next, the number of changed polygon segments is small; hence the horizontal sort may be optimized to take advantage of this. Watkins' is the only other algorithm besides Schumacker's that has been implemented in hardware.

2.2 Parametric Surface Algorithms

Modeling smooth surfaces with collections of polygons leads to problems both in shading the surface and in rendering the contour edges. While there have been a number of very clever improvements to the quality of such pictures without significantly increasing the amount of information used, notably those of Gouraud [10], Phong [15], and Crow [6], the most direct approach is to employ a more refined model, such as parametric surface patches. Such patches can be used to define the surface using no more, and usually even less, information than is required with polygons. Yet they can join together with tangent or even higher continuity, thus eliminating the above problems. The difficulty with this method is that the mathematics is no longer linear; to explicitly solve for such things as the curve of intersection of two bi-cubic patches or of a patch and a clipping plane are very difficult problems.

Catmull [3] solves such problems, but not explicitly. Rather, he does so by employing the discrete character of the image space, a recursive algorithm, and what he calls a Z-buffer. For each patch in the environment his algorithm asks: does the patch extend over more than a single raster unit? If the answer is yes, the patch is subdivided (by a very fast algorithm for bi-cubic patches) into four patches and the same question is *recursively* asked of these patches. When the answer finally is no, an attempt is made to write the intensity and depth coordinates for the resulting "patch" into a buffer for the raster unit, or *pixel*, in question. The attempt fails if the pixel buffer already has in it a depth coordinate nearer to the observer (with some minor modifications to allow for translucent patches).

A very significant feature of Catmull's algorithm is that, despite the more complex mathematics, it will actually work faster than polygon-based algorithms if the object being rendered occupies a very small area of the screen. Because of the recursive structure of the algorithm, it will "structure" the surface no more finely than the resolution of the display dictates, whereas current polygon-based algorithms keep the same structural description, i.e. the same number of polygons, no matter how much of the screen area is occupied. This notion of structuring will be extended to include polygon-based algorithms in the next section.

2.3 Procedurally Modeled Objects

Newell [13] has recently employed procedural modeling to solve the visible surface problem for complex scenes. According to this approach, objects are modeled using procedures which "know how" to render themselves in terms of their own primitives, which might include activations of other object procedures; such knowledge includes rendering only their visible parts. This is a very general way to represent objects.

Although the underlying philosophy of this approach is very general, in the actual implementation Newell user polygons as the basic primitives for the objects. The object procedures are activated according to a priority ordering so that more distant objects are activated first. Each procedure renders the object it represents by activating the Watkins process, the results of which are written into a frame buffer. The net result is therefore a "hybrid" Watkins/Newell priority algorithm.

The significant point about this algorithm is not the procedural modeling but that it represents another example of structuring to simplify the total soiting problem, namely that the geometric primitives of one object need be compared with those of another only when the objects overlap.

3. Hierarchical Approach

It was indicated in the previous section that, aside from uses of image-space coherence to reduce the amount of sorting required, the most fruitful gains in visible surface algorithm research have resulted from structuring the environments being rendered. However, the structures employed take a diverse variety of forms, from Catmull's implicit structuring of surface patches to Newell's procedural objects. What is needed is a single, unified, structural approach that embodies all of the ideas from these algorithms. Before presenting one such approach it is instructive to consider two ways in which structure has been utilized to prepare objects for visible surface processing.

3.1 Existing Uses of Structure

Defining relative placement. The benefits of a position or motion structure have been realized for some time. Sutherland used such concepts in two dimensions in Sketchpad, and a number of graphics hardware companies incorporate transformation hardware in their display devices to accommodate structural descriptions. Most of the visible surface algorithms presented used a position or motion structure to describe positions and orientations of objects relative to each other. However, all but the few mentioned in Section 2 disregard the structure at the visible surface algorithm level. That is, all polygons of the objects are transformed into a common screen coordinate system in which the visible surface algorithm works.

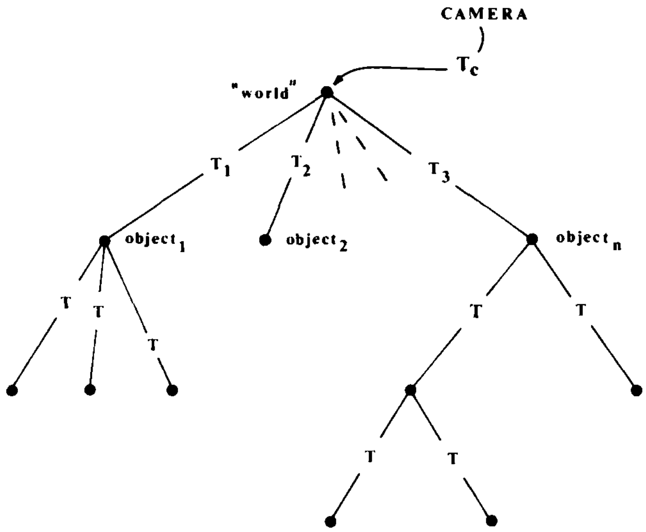
An example of such a structure is shown in Figure 1. Each node in the hierarchy represents a set of geometric primitives (e.g. polygons) defining the node and the arc leading to the node represents a transformation defining the orientation and placement of the node relative to its "parent." Because each node has its own unique transformation defining it, it may represent one of many "instances" of the same primitive description, or data set. This is a very convenient and general way to define and place objects.

Decreasing clipping time. When simulating a camera in a computer-generated environment, some parts of the environment must be "clipped" to the field of view of the simulated camera. This can be done either by transforming all of the geometric primitives of each object into the camera, or screen, coordinate system and clipping each of them separately or by first clipping some bounding volume of the object to see if it intersects the boundaries of the field of view. If it does not, then the parts of the object lie either totally within or totally outside of the field of view and thus need not be separately clipped. This utilization of the above mentioned position hierarchy is implicitly assumed, although this author does not know if the authors of the various algorithms actually made such use of it.

3.2 New Uses of Structure

Varying environment detail. By choosing to represent an object with a certain amount of detail, one fixes the minimum distance from which the object may be displayed with a realistic rendering. For example, a dodecahedron looks like a sphere from a sufficiently large distance and thus can be used to model it so long

Fig. 1. The traditional motion structure used to position objects relative to the "world" and subobjects relative to objects. Each arc in the graph represents a transformation.



as it is viewed from that or a greater distance. However, if it must ever be viewed more closely, it will look like a dodecahedron. One solution to this is simply to define it with the most detail that will ever be necessary. However, then it might have far more detail than is needed to represent it at large distances, and in a complex environment with many such objects, there would be too many polygons (or other geometric primitives) for the visible surface algorithms to efficiently handle.

As mentioned in Section 2, the solution to this problem has been to define objects relatively coarsely and employ clever algorithms that smooth appropriate contours or improve shading to make the object look more realistic at close observation. The difficulty with these approaches is that at best the range of viewing depth is only slightly improved, and the problem of too much detail at large distances usually remains. Although these approaches have yielded results of unquestionable value, it seems evident that multiple levels of description must be used to adequately represent complex environments.[1]

How does one represent these multiple levels of description? A solution is to define "objects" in a hierarchy like that of Figure 2. The entire environment is itself an "object" and is represented as a rooted tree. ("Object" is a generic term for the things represented by nodes of the tree. This generic term will be used for the remainder of this paper.) There are two types of arcs in the tree, those that represent transformations as before and those that represent pointers to more detailed structure (the identity transformation). Each nonterminal node represents a "sufficient" description of the "object" if it covers no more than some small area of the display; the arcs leading from the node point to more detailed "objects" which collectively define a more detailed version of the original object if its description is insufficient because it covers a larger area of the screen. The terminal nodes of the tree represent either polygons or surface patches (or other primitives) according to whether they are primitive elements of a faceted or a smooth object.

As an example of such a description, consider a model of the human body. When viewed at a very large distance, for example when the body covers only 3 or 4 display raster units, it is sufficient to model the body with a single rectangular polyhedron with appropriate color. Therefore the uppermost node, or "object," for this body represents this simple description. If the body is viewed from a closer distance—for example, if its topmost node's description covers 16 raster units—then this topmost description is no longer sufficient, and the next level of more refined description is needed. At this next level the body is now perhaps described as a collection of rectangular polyhedra appropriately attached to each other, for example using one polyhedron for each of the arms and legs, the head and the torso. Then so long as each of these "objects" covers only a few raster units of the display, their description is "sufficient." When the viewing distance decreases such that any of them covers a critical maximum area of the display, its more detailed subobjects are used to replace its description. This process is carried out to whatever maximum level of detail will be needed. For example, a terminal level of description of the fingertip might be several surface patches (which could be implicitly structured even more finely using Catmull's algorithm).

The body described is just one "object" of an environment, or larger hierarchy. There might be many such bodies, or other objects. The significant point, however, is that in a complex environment, the amount of information presented about the various objects in the environment varies according to the fraction of the field of view occupied by these objects.

It is worth noting again that Catmull's algorithm, described in the previous section, implicitly built such a structure. His algorithm used this structure in such a way that, despite the more complex mathematics of surface patches, it outperforms polygon-based algorithms if the surface occupies a small area of the screen. Thus it seems that such a structure should lead to improvements in polygon-based algorithms as well.

Clipping: a truncated logarithmic search. The choice of this structural representation poses another problem. How does one select only that portion of a potentially very large hierarchy that is meaningful in the context of the viewpoint and the resolution of the viewing device? In other words, clipping in a broader sense must mean selecting not only that part of the environment within the field of view (the usual meaning) but also just the *resolvable* part. This implies finding the visible nodes of

[1] Actually, Evans and Sutherland made use of a three-level description of the New York skyline in its Maritime simulation, but in an ad hoc way [2].

the tree, as shown in Figure 2. The contour shown in the figure represents a possible set of objects that are within the field of view and are both not too large and not too small for the screen areas they occupy.

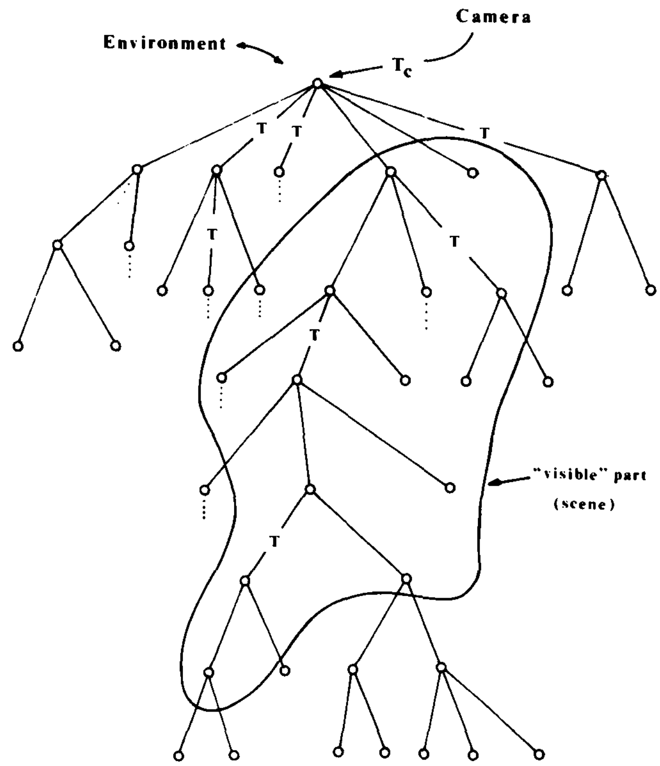
In order to efficiently perform this clipping operation some minimal description of object sizes must be available. For example, a bounding rectangular box or a bounding sphere would be sufficient information to test whether an object is totally within or totally outside of the field of view. The minimum necessary information is the center and radius of a bounding sphere.

This general structure therefore suggests a very fast clipping algorithm which recursively descends the tree, transforming (if necessary) this minimal information into perspective viewing coordinates and testing both the area occupied by the bounding sphere and its intersection with the boundaries of the field of view. The criterion for descending a level is the area test, while the criterion for inclusion/rejection is the field of view boundary test. Only after either the area test terminates the descent or the terminal level of representation is reached is it necessary to actually transform and possibly clip the polygons or surface patches represented by the node. Clipping therefore resembles a logarithmic search that is truncated by the area (resolvability) test.

This relatively simple mechanism for varying the detail in a scene suggests several other interesting possibilities. Since the center of attention of a scene is often its geometric center, one might effectively render the scene with a center-weighting of detail. In other words, the maximum area an object is allowed to cover before splitting it into its subobjects becomes larger towards the periphery of the field of view. This is somewhat analogous to the center-weighted metering systems of some cameras. Likewise, since moving objects are less resolved by both the human eye (because of saccadic suppression) and a camera (because of blurring), one can render them with an amount of detail that varies inversely with their speeds. Indeed, an entire scene might be rendered with less detail if the camera is moving. Thus "clipping" can be extended to include these concepts as well.

Graphical working set. Since the problems addressed by this model are those associated with producing pictures and picture sequences of very complex environments, the excessive storage needed for the geometric description of these environments must somehow be accommodated. Denning's "working set" model for program behavior provides a useful analogy [8]. According to this model, a computer program that makes excessive demands on immediate-access store is structured or segmented, and its storage demands are managed in such a way that only those segments most recently in use are actually kept in immediate-access store. The remaining potentially large number of segments are kept on a slower, secondary store, such as a disk. The "working set" is that set of segments available for immediate access, and is usually defined by a time average of past program reference patterns. Reference to an unavailable segment causes that segment to become part of the working set, and segments not accessed after some period of time are deleted from the working set.

Fig. 2. A very deep hierarchy that structures the environment much more than the traditional motion structure. Arcs in this graph represent either transformations or pointers to more refined definitions of the node. The visible part contour represents a possible result of clipping.



This working set model coupled with the broader sense of clipping mentioned above suggests a suitable way to accomplish a particular type of frame coherence. The working set in this context is that set of objects in the hierarchy that are "near" to the field of view, inside it, or "near" to the resolution of the image space. Only if an object is a member of this set is its description kept in immediate-access store. The set membership will change slowly since the differences between one scene and the next are usually small. Those cases in which the differences are large due to fast camera (or object) motion are easily accommodated by rendering the scene (or object) with less detail, as mentioned above. Moreover, the minimal description of node size needed for clipping suffices as the graphical analog of the segment table used in the computer program context. That is, this minimal clipping description must always be available in immediate-access store to facilitate determining the working set. This working set model therefore seems particularly well suited to the graphics context.

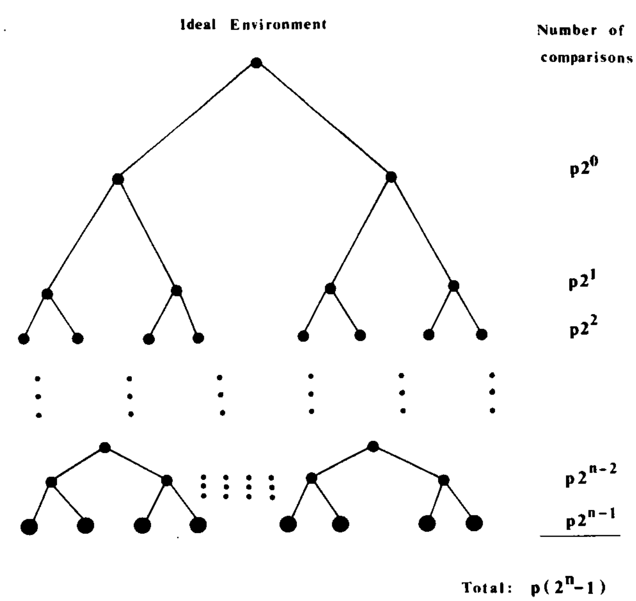
Improving existing algorithms. There are two ways in which a geometric hierarchy should lead to improvements in existing algorithms. The first is by reducing the number of comparisons needed to sort objects and the second is by eliminating from potential consideration an entire portion of the environment because an object obscures it.

Since sorting is the central problem of visible surface algorithms, the performance of these algorithms improves with improved sorting methods. Indeed, many of the fast visible surface algorithms that have been discussed have resulted from clever utilization of image-space coherences, such as scan-line coherence, to improve sorting speeds. In the present hierarchical framework, the geometric proximity of the subobjects of an object provides an object-space coherence that can also be utilized to decrease sorting time.

For example, consider an ideal case of a binary tree as shown in Figure 3. Each node of the tree has associated with it a bounding volume, but since this ideal tree is the result of clipping, only the terminal nodes actually represent geometric primitives, e.g. polygons or patches. Assuming that there are n levels in the tree, not counting the root node, there are $m = 2^n$ terminal nodes.

If the structure is ignored, then the fastest possible sort of these terminal nodes is accomplished with proportional to $m \log_2 m$ comparisons using a quicksort. However, if the structure is utilized and if the bounding volumes of siblings do not overlap, which is admittedly an optimum arrangement, then the number of required operations is $p2^0$ for the first level, $p2^1$ for the second level, $p2^2$ for the third, etc., where p is a proportionality factor. Summing the number of operations performed at all levels yields $p\sum_{i=0}^{n-1} 2^i = p(2^n - 1)$, or roughly pm. In other words, by using the structure, in the optimum situation of no overlap, the sorting time grows linearly rather than as $m \log_2 m$.

Fig. 3. An ideal binary hierarchy in which none of the terminal nodes overlap. The first $p2^0$ comparison sorts all objects into two classes, the second $p2^1$ comparisons sort them into 4 classes, etc. Summing all comparisons from all levels yields $p(2^n - 1)$ comparisons.



Of course, this analysis holds only for a binary hierarchy in which none of the siblings' bounding volumes overlap, which is an idealized situation. A binary hierarchy might not be appropriate, and any complex environment will no doubt have some overlap, although presumably not a very large amount. However, the point here is that sorting methods which utilize the geometric structure can yield a considerable performance improvement over those which do not, even under less than ideal conditions.

The other improvement provided by a deeply structured geometric hierarchy is that of eliminating a potentially large part of the structure from consideration because an object obscures it. Such an improvement requires defining for each object (in the generic sense) both a simple occluded volume, Δ, such that if Δ is obscured then the entire object is obscured, and a simple occluding volume, δ, such that if δ obscures something then that thing is sure to be obscured by the object. Clearly, Δ exists for all objects, whereas δ might not exist for some objects, such as an open-ended cylinder or a transparent object. Δ can be just the bounding sphere used in clipping, but δ is in general additional information that must be kept for each object.

Recursive descent, visible surface algorithm. The above considerations suggest a totally new recursive-descent visible surface algorithm in which at each level all objects are sorted according to their bounding volumes. If any of the bounding volumes overlap both laterally and vertically then the occlusion test potentially allows one (or more) of the objects, and hence all of its descendents, to be totally eliminated from consideration.

Using the ordering thus obtained, the same sorting and occlusion tests are recursively applied to each of the descendants of these objects; in those cases where two or more objects' bounding volumes overlap in all three dimensions, indicating potential intersections, the descendents of these objects are treated as if they have the same parent nodes at the next level of recursion. Of course, recursion terminates when a terminal node is reached, and the net result of descending the tree is a very rapid sort of the primitives represented by these terminal nodes. Under ideal conditions, the computation time of this algorithm grows linearly with the *visible* complexity of the scene.

Since both this algorithm and the clipping algorithm described above recursively descend a tree structure, it seems natural to combine them. Doing so not only potentially eliminates area tests on occluded objects but also potentially decreases the size of the working set. If all processing is performed by a single processor, such as a general purpose computer, then the algorithms are probably most conveniently integrated into a single algorithm. However, if multiple processors are available, whether special purpose hardware or general purpose computers, then the algorithms might be left separate or combined according to whether parallelism is achieved by pipelining or otherwise.

Building structured databases. Obtaining a good graphical database is a very time consuming and difficult part of computer picture research. Databases obtained by careful measurement of real objects, by "building" objects from collections of simple mathematical objects, or by sculpturing surfaces in three dimensions [4] are at least as valuable as the visible surface algorithms that render them.

At first glance it appears that the structural framework multiplies the dimensions of this problem since multiple descriptions of the same object must be defined. However, in the case of carefully measured real objects, the multiple descriptions can be produced by judicious "bottom-up" pruning of existing definitions of the objects in their most detailed form. Therefore use can be made of all objects that have already been defined.

Those existing objects modeled with surface patches also present no problem. The coarser, high-level descriptions of these objects can be obtained by replacing the patches themselves with polygons and proceeding with the "bottom-up" pruning mentioned above to obtain even coarser descriptions. The finer, low-level descriptions of the objects can be obtained by "top-down" splitting of the surface patches, as in the Catmull algorithm. This can be done either at display time or beforehand in building the database; the difference is the traditional time/space tradeoff.

4. Conclusions

All of the recent major advances in computer picture research have resulted from either explicitly or implicitly incorporating structure information in the geometric modeling techniques. This research represents an attempt to encompass all of these advances in a more general structural framework as a unified approach to solving a number of the important problems of systems for producing computer pictures.

The proposed hierarchical models potentially solve a number of these problems. They provide a meaningful way to vary the amount of detail in a scene both according to the screen area occupied by the objects in the scene and according to the speed with which an object or the camera is moving. They also extend the total range of definition of the object space and suggest convenient ways to rapidly access objects by utilizing a graphical working set to accomplish frame coherence.

An important aspect of the hierarchical models is that by providing a way to vary detail they can yield an incremental improvement to existing systems for producing computer pictures without modifying their visible surface algorithms. Another incremental improvement is then possible by incorporating the structure in the sorting phases of existing algorithms. A final improvement is suggested by a totally new recursive descent visible surface algorithm in which the computation time potentially grows linearly with the visible complexity of a scene rather than as a worse than linear function of the object-space complexity.

References

1. Bouknight, W.J. A procedure for generation of three-dimensional half-toned computer graphics representations. *Comm. ACM, 13*, 9 (Sept. 1970), 527.
2. Computer Aided Operations and Research Facility, U.S. Maritime Service Simulator (principal contractor Philco-Ford, visible-surface processor by Evans and Sutherland Comptr. Corp.).
3. Catmull, E. A subdivision algorithm for computer display of curved surfaces. Tech. Rep. UTEC-CSc-74-133, U. of Utah, Salt Lake City, Utah, Dec. 1974.
4. Clark, J.H. 3-D design of free-form B-spline surfaces. UTEC-CSc-74-120, Ph.D. Th., U. of Utah, Salt Lake City, Utah, (abridged version Designing surfaces in 3-D. *Comm. ACM 19*, 8 (Aug. 1976), 464–470.)
5. Coons, S.A. Surfaces for computer-aided design of space forms. Project MAC TR-41., M.I.T., Cambridge, Mass., June 1967.
6. Crow, F.C., and Bui-Tuong Phong. Improved Rendition of Polygonal Models of Curved Surfaces. Proc. Second USA-Japan Comptr. Conf., Aug. 1975, p. 475.
7. Csuri, C. Computer animation, Computer Graphics *9*, 1 (1975), 92–101 (Issue of Proc. Second Ann. Conf. Comptr. Graphics and Interactive Techniques).
8. Denning, P.J. The working set model for program behavior. *Comm. ACM, 11*, 5 (May 1968), 323–333.
9. Electonic scene generator expansion system. Final Rep., NASA Contract NAS 9-11065, Defense Electronic Div., General Electric Corp., Syracuse, N.Y., Dec. 1971.
10. Gouraud, H. Computer display of curved surfaces. *IEEE Trans. Computers C-20* (June 1971), 623.
11. Nasa-Ames Short Take-off and Landing Simulator (built by Evans and Sutherland Comptr. Corp.).
12. New York Inst. Tech., Comptr. Animation Dep.
13. Newell, M. The utilization of procedure models in digital image synthesis. Ph.D. Th., Comptr. Sci., U. of Utah, Salt Lake City, Utah, 1975.
14. Newell, M.E., Newell, R.G., and Sancha, T.L. A new solution to the hidden-surface problem. Proc. ACM 1972 Ann. Conf., pp. 443–448.
15. Bui-Tuong Phong. Illumination for computer generated pictures. *Comm. ACM 18*, 6 (June 1975), 311–317.
16. Rediflow Flight Simulation, Ltd., NOVOVIEW Visual Systems (video system provided by E&S Comptr. Corp.).
17. Riesenfeld, R.E. Applications of B-spline approximation to geometric problems of computer aided design. Ph.D. Th., Syracuse U., Syracuse, N.Y., 1972.
18. Schumacker, R.A., Brand, B., Gilliland, M., and Sharp, W. Study for applying computer-generated images to visual simulations. AFHRL-TR-69-74, US Air Force Human Resources Lab., Washington, D.C., Sept. 1969.
19. Sutherland, I.E. Sketchpad: a man-machine graphical communication system. TR 296, M.I.T Lincoln Labs, M.I.T., Cambridge, Mass., Jan. 1963.
20. Sutherland, I.E., Sproull, R.F., and Schumacker, R.A. A characterization of ten hidden-surface algorithms. Computing Surveys, *6*, 1 (March 1974), 1–55.
21. Watkins, G.S. A real-time visible-surface algorithm. UTECH-CSc-70-101, Ph.D. Th., Comptr. Sci. Dep., U. of Utah, Salt Lake City, Utah, June, 1970.
22. Wipke, T., et al. *Computer Representation and Manipulation of Chemical Information*. Wylie Interscience, New York, 1974.
23. Wylie, C., Romney, R.S., Evans, D.C., and Erdahl, A. Halftone perspective drawings by computer. Proc. AFIPS 1967 FJCC, Vol. 31, AFIPS Press, Montvale, N.J., pp. 49–58.