The Feudal Priority Algorithm on Hidden-Surface Removal

Han-Ming Chen

Wen-Teng Wang

Department of Mechanical Engineering

National Taiwan University

ABSTRACT

Development of a real-time shaded rendering approach for a frequently changing viewpoint or view vector is very important in the simulation of 3-D objects in Computer-Aided Design. A new approach is proposed in this paper to meet this demand in a very efficient manner.

A pre-processing phase, in which a feudal priority tree is established for all polygons of an object, and a post-processing phase, in which a rendering priority list is searched for from the feudal priority tree for a new viewpoint or view vector, are included in our approach. The most time-consuming work is finished in the pre-processing phase which only has to be executed once for an object, and the relatively simple task is left to the post-processing phase, which is repeated when the viewpoint or view vector is changed.

For the pre-processing phase, a static version and a dynamic version are proposed in this paper. The one-way priority relations of all polygons are computed in the former part of the dynamic pre-processing in a more efficient way than that in the static pre-processing, but the latter part of the dynamic pre-processing is still based on the static pre-processing.

A new concept of "absolute priority" is introduced to systematically reduce the polygons in which a separating plane is to be searched for so the probability of finding the separating plane is much increased. This is the basis to implement another important concept of "separating before splitting" by which the polygon splittings are much reduced. Hence the efficiency in the pre-processing and the post-processing phases is highly increased.

CR Categories and Subject Descriptors: I.3 [COMPUTER GRAPHICS]: I.3.7 [Three-Dimensional Graphics and Realism] - Hidden line/surface removal, Visible line/surface algorithms.
Additional Key Words and Phrases: The Binary Space-Partitioning Tree Algorithm.

Taipei, Taiwan 107, R.O.C. e-mail: hmchen@ccms.ntu.edu.tw

INTRODUCTION

In 1969, Schumacker first presented some very important notions on the subject of visual simulation [10]. Schumacker [10, 11, 12, 15] observed that within a cluster the face priority is a property of the topology of the cluster and can be calculated independently of the viewpoint if the environment, i.e., objects, can be divided into several adequate clusters. The cluster priority is determined by isolating clusters with separating planes and is dependent on the location of the viewpoint relative to the separating planes.

A subsequent development in this field was the Binary Space-Partitioning (BSP) Tree Algorithm. It was developed by Fuchs, Kedem, and Naylor [5, 7, 6] in 1980. The BSP tree algorithm is based on the work of Schumacker [5, 4]. Its most fundamental notion is to separate the space into two subspaces by a properly selected plane such that no polygon in the subspace on the viewpoint side is obstructed by any polygon in the subspace on the other side. This algorithm pushes much of its work into a pre-processing phase in which a BSP tree is computed and established. Once the BSP tree is established, the post-processing work becomes very simple to each new viewpoint.

Newell, Newell, and Sancha [8] developed an ordering test and a face-splitting routine in their algorithm to find the priority list for 3-D polygons. In their algorithm, it is still necessary to repeat the whole procedure for a new viewpoint, but its basic ideas are very helpful for establishing the "one-way priority" table used in our research [1, 2].

Computer-Aided Design and Manufacturing has become more and more important in modern industry. To render 3-D objects on the computer screen is an important step in CAD/CAM. But the efficiency of this rendering step completely relies upon a good Hidden-Surface Removal algorithm which can handle objects in any shape at a fast speed.

One of the possible disadvantage of the BSP tree algorithm is that the output polygons in the tree would be significantly more than the input polygons so the number of splittings is very large [5]. Another weakness of this algorithm is that the appropriate partitioning hyperplane selection is quite complicated and difficult. Therefore, we have developed a new method which we have named "Feudal Priority Algorithm" which also includes pre-processing and post-processing phases and can compute the rendering priority of polygons in any shape. In the "dynamic pre-processing" of our approach, the number of splittings and the number of output polygons is much fewer than those in the pre-processing of the BSP tree algorithm, and even the number of one-way priority relations to be computed is fewer than that in the pre-processing of the BSP tree algorithm. Hence the dynamic pre-processing has a faster speed than the pre-processing of the

BSP tree algorithm. The efficiency of the post-processing is only affected by the number of the output polygons in the feudal priority tree or the BSP tree so our approach has a much higher speed than the BSP tree algorithm in the post-processing work.

FUNDAMENTAL CONCEPTS

If there exits any face(polygon) of an object and you can reach both sides of this face without penetrating any other face, this object is "open-volume". If you can just reach at most one side of each face of an object without penetrating other faces, this object is "closed-volume". All real world objects are constructed with closed-volume objects which are the most interesting objects in the fields of CAD and CAM. But many theoretical applications in the areas of Graphics, Mathematics, etc. make use of examples which contain open-volume objects. A polygon is a typical example of an open-volume object.

The objects discussed in this paper only consist of polygons, i.e., flat faces, which are either convex or concave. The term polygon is used to denote the union of the boundary and the interior of a plane region which is bounded by several successive line segments [9]. The following concepts are valid for both closed-volume and open-volume objects.

[Definition 1] One-way Priority

The "one-way priority" of a polygon P relative to a polygon Q is represented by the symbol "P -> Q" and is divided into the following four categories by substituting the x, y, and z coordinates of all vertices of the polygon P into the plane equation of the polygon Q:

(1) P is on the front side of Q if at least one vertex of P makes the plane equation of Q greater than 0 and all other vertices of P make the plane equation of Q not less than 0. This category is represented by the symbols "P <| Q" or "Q |> P".
(2) P is on the back side of Q if at least one vertex of P makes the plane equation of Q less than 0 and all other vertices of P make the plane equation of Q not greater than 0. This category is represented by the symbols "P >| Q" or "Q |< P".
(3) P is cut by Q if at least one vertex of P makes the plane equation of Q greater than 0 and at least one vertex of P makes the plane equation of Q less than 0. This category is represented by the symbol "P \- Q".
(4) P and Q are coplanar if all vertices of P make the plane equation of Q equal to 0. This category is represented by the symbol "P -- Q". Coplanar polygons have equal rendering priority.

The one-way priority is the most important basis in our approach.

The "absolute priority" is divided into the "absolute front priority" and the "absolute back priority" which are defined as the following:

[Definition 2] Absolute Front Priority

If no other polygons are on the front side of a polygon P, P has the "absolute front priority" to those polygons which are on the back side of P. Polygons which are coplanar and have the same normal direction with P have the same priority as P. All these polygons with the absolute front priority have the same priority. In Figure 1, polygons 1, 2, 3, 4, and 7 have the same priority in which polygons 2 and 3 are coplanar. The arrows in Figure 1 represent the normal vectors of polygons.

All the polygons with the absolute front priority are put in a bunch Fj and all other remaining polygons are put in a group G. In this paper, the term "bunch" is used to put the polygons which have the same priority, but the polygons in the term "group" may not have the same priority.

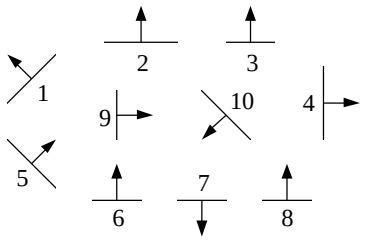

Figure 1. Absolute front priority and absolute back priority.

[Definition 3] Absolute Back Priority

If no other polygons are on the back side of a polygon P, P has the "absolute back priority" to those polygons which are on the front side of P. A polygon has the same priority as P if it is coplanar and has the same normal direction with P. All these polygons with the absolute back priority have the same priority. In Figure 1, polygons 5, 6, and 8 have the same priority in which polygons 6 and 8 are coplanar.

All the polygons with the absolute back priority are put in a bunch Bj and all other remaining polygons are put in a group G. The polygons in Fj and the polygons in Bj have the same priority so they are put in the same level in a "feudal priority tree".

The notion of the separating plane in [10] is extremely helpful in reducing polygon splittings. From the definition of linear separability, a separating plane for a group of polygons is easily found from their one-way priority relations.

[Definition 4] Linear Separability

Two sets of points S1 and S2 in E3(3-dimensional Euclidean space) are said to be linearly separable if there exists a plane P such that S1 and S2 lie on the opposite sides of P [9].

[Definition 5] Separating Planes

Polygons P and Q are said to be separated by a plane S or a plane S is said to be the separating plane of polygons P and Q, if either (1) or (2) is true.

(1) P <| S and Q >| S
(2) Q <| S and P >| S

The "relative priority" of two groups of polygons is determined by the "switch plane". If a separating plane can be found as the switch plane, it is much better than a splitting plane being selected as this switch plane because polygons are divided into two groups by the separating plane without any splitting.

[Definition 6] Relative Priority with A Separating Plane

After all polygons with absolute front priority or absolute back priority have been removed, there is no polygon which has absolute priority. In the remaining polygons, if a polygon S is

found which can be a separating plane to all other polygons, separate all other polygons except the coplanar polygons of S into groups C and D. If the group C is on the front side of S and the group D is on the back side of S, this case can be represented by the symbols "C <| S |< D" or "D >| S |> C". In Figure 2, (polygons 4 and 5) <| polygon 3 |< (polygons 1 and 2).

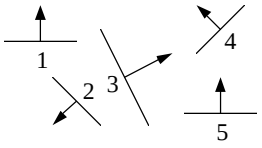

Figure 2. Relative priority with a separating plane.

[Definition 7] Relative Priority with A Splitting Plane

If a separating plane can not be found in the above case, a splitting plane S is selected. From the one-way priority relation of S, the polygons cut by this splitting plane can be easily found and split into smaller polygons. Then all output polygons except S and its coplanar polygons are separated into groups C and D. If C is on the front side of S and D is on the back side of S, this case can be represented by the symbols "C <| S |< D" or "D >| S |> C". In Figure 3, (polygons 1 and 2a) <| polygon 3 |< (polygons 2b, 4, and 5). The polygon 2 is split into polygons 2a and 2b by the splitting plane 3.

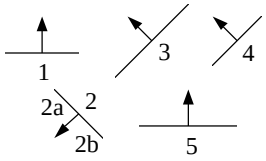

Figure 3. Relative priority with a splitting plane.

PROCEDURE

For each polygon, list all vertices in a sequence which makes the first three vertices be in clockwise direction on the outside surface in order to match the convention of the left-handed coordinate system. This is very important for closed-volume objects because the normal vector of each polygon is set to point outward for doing the back-face culling work. Then compute the plane equation of each polygon with the coordinates of the first three vertices of the polygon and store the coefficients for the plane equation.

The procedure developed in our approach for rendering 3-D objects involves a preprocessing phase and a postprocessing phase which are described next, starting with closed-volume objects. In the preprocessing phase, both a static version and a dynamic version are proposed in this paper.

Now, a "feudal priority tree" is to be established step by step and a simple closed-volume object shown in Figure 4 is used to demonstrate the whole procedure.

The Static Preprocessing Phase

1. Establishing One-way Priority Tables

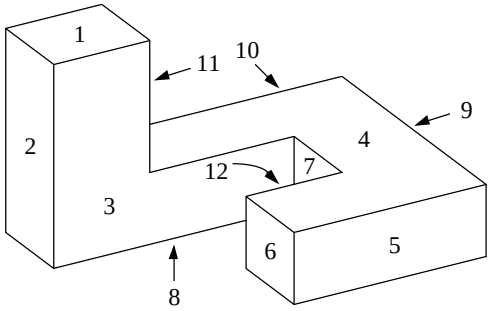

Figure 4. A closed-volume object.

For the i-th polygon of a closed-volume object, substitute the coordinates of all vertices of every other polygon sequentially into the plane equation of the i-th polygon, and then compute and decide the one-way priority relation of every other polygon to the i-th polygon. According to Definition 1, the one-way priority relation of all other polygons to the i-th polygon are divided into four categories: (1) front side, (2) back side, (3) cutting, and (4) coplanar. List every other polygon under its appropriate category in the i-th row in a one-way priority table.

In the i-th row, if there are polygons under the category "coplanar", it is not necessary to compute the one-way priority relation for these coplanar polygons. Just copy the i-th row into the rows of all these coplanar polygons, and then only exchange the polygons under categories "front side" and "back side" for those coplanar polygons with the reverse normal direction with the i-th polygon.

Table 1. A part of the one-way priority table for the object in Figure 4.

Q	P -> Q			
	P <| Q	P >| Q	P \- Q	P -- Q
1		2,3,4,5,6,7,8, 9,10,11,12		
2		1,3,4,5,6,7,8, 9,10,11,12		
3	5,6,7,12	1,2,10,11	4,8,9	
4	1,11	5,6,7,8,9,12	2,3,10	

2. Adding Absolute Priority Polygons to The Feudal Priority Tree and Deleting Them from One-way Priority Tables

(1) In the i-th row, if no polygon is under the categories "back side" and "cutting", the i-th polygon is an absolute back priority polygon and can be added into the bunch Bj on the right side of the current connecting node as shown in Figure 5. Then the i-th row is deleted from the one-way priority table.

(2) In the i-th row, if no polygon is under the categories "front side" and "cutting", the i-th polygon is an absolute front priority polygon and can be added into the bunch Fj on the left side of the current connecting node. Then the i-th row is deleted from the one-way priority table.

(3) After all absolute priority polygons in the above steps (1) and (2) have been found, delete these polygons from all the rows remaining in the one-way priority table.

(4) Repeat steps (1) through (3) again. If there are polygons found in step (1), add them into the bunch Bj+1 linked on the right side of the last bunch Bj. If there are polygons found in step (2), add them into the bunch Fj+1 linked on the left side of the last bunch Fj. If there are no polygons found in steps (1) and (2), stop here and execute the next procedure.

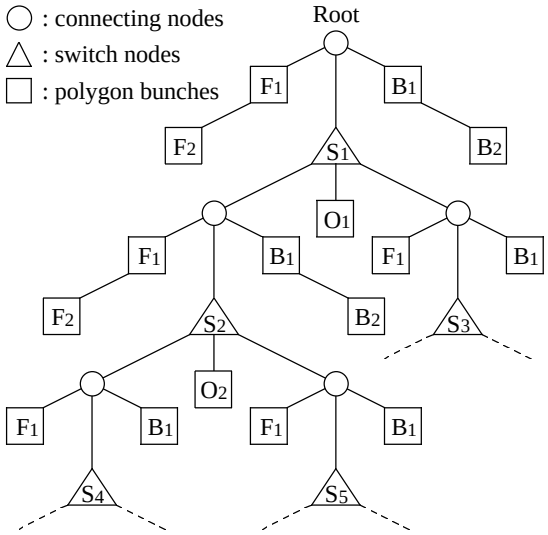

Figure 5. A feudal priority tree.

The polygons in the bunches Fj+1 and Bj+1 and all the polygons below Fj+1 and Bj+1 in the feudal priority tree are surrounded by the polygons in the bunches Fj and Bj such that this relation is not changed while the viewpoint or the view vector is moved. Hence this relation is called the "absolute priority".

After removing absolute front priority polygons 1, 2, 5, 8, 9, and 10, the one-way priority table of the object in Figure 4 becomes Table 2 in which no absolute priority polygon can be found.

Table 2. The one-way priority table for the object in Figure 4 after all absolute priority polygons are removed.

Q	P -> Q			
	P <| Q	P >| Q	P \- Q	P -- Q
3	6,7,12	11	4	
4	11	6,7,12	3	
6	11	7,12	3,4	
7	3,6,11,12		4	
11	4,6,7,12		3	
12	3,7,11	6	4	

3. Separating The Remaining Polygons with A Separating Plane

In the i-th row of the current one-way priority table, if no polygon is under the category "cutting", the polygon Qi can be a separating plane to all other polygons except those polygons coplanar with Qi. If more than one separating plane can be found, the separating plane with the most balanced polygons on the front and back sides is selected. Setting this separating plane Sk as a switch node in Figure 5, those polygons on the front side of Sk are put in a group Gf as the left branch, those polygons on the back side of Sk are put in a group Gb as the right branch, and those polygons coplanar with Sk are put in a bunch Ok under the switch node Sk. Revise the current one-way priority table into two smaller ones for Gf and Gb separately. Because the rendering order of Gf and Gb is determined by the angle between the normal vector of Sk and the view vector it is called "relative priority".

Then go back to do procedure 2 for Gf and Gb individually. If there is no polygon by which all other polygons can be separated into two groups, do the next procedure to select a splitting plane.

4. Splitting The Remaining Polygons with A Splitting Plane

In the current one-way priority table, a polygon Qi with (1) the least polygons under the category "cutting" and (2) the most balanced polygons under the categories "front side" and "back side" is selected as the splitting plane Sk. Those polygons under the category "cutting" are split by Sk into smaller polygons. Those split smaller polygons on the front side of Sk and the polygons under the category "front side" are put in a group Gf as the left branch of a switch node Sk in Figure 5. The other split smaller polygons on the back side of Sk and the polygons under the category "back side" are put in a group Gb as the right branch of the switch node Sk. Those polygons coplanar with Sk are put in a bunch Ok under the node Sk. Two smaller one-way priority tables are modified from the current one based on the groups Gf and Gb. The rendering order of Gf and Gb is also determined by the switch plane Sk as in the above procedure.

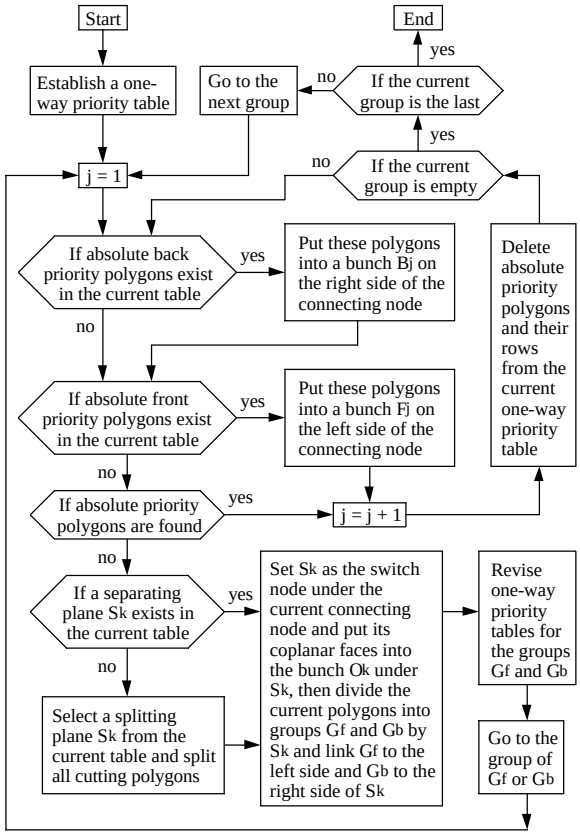

Figure 6. The procedure of the static preprocessing.

Return to do procedure 2 for Gf and Gb individually. In procedure 2, after all absolute priority polygons were removed from Gf and Gb, the separating planes or splitting planes chosen from the remaining polygons in Gf and Gb are set as the switch nodes Sk+1 and Sk+2 separately in the feudal priority tree in Figure 5.

The whole procedure for the static preprocessing phase of our approach is illustrated as shown in Figure 6.

In the "static preprocessing phase", all absolute priority polygons and all separating planes are searched for after their corresponding one-way priority tables have been completely built. But the "dynamic preprocessing phase" is developed to find the absolute priority polygons and the separating planes after each row of the one-way priority tables has been just set up.

In Table 2, there is no separating plane that can be found. Polygons 3, 4, and 12 are good candidates as the splitting plane. Polygon 3 is selected to be the splitting plane and split polygon 4 into polygons 13 and 14 as in Figure 7. Polygons 6, 7, 12, and 14 are in the group Gf and polygons 11 and 13 are in the group Gb. The two one-way priority tables of these two groups are shown in Table 3. In the group Gf of Table 3, two absolute front priority polygons 6 and 14 can be removed and put in a bunch F1 under the left connecting node of the switch node 3, and then its one-way priority table becomes Table 4. Absolute back priority polygons 7 and 12 in Table 4 are put in a bunch B2. Absolute back priority polygons 11 and 13 in the group Gb are put in a bunch B1 under the right connecting node of the switch node 3. The feudal priority tree for the object in Figure 7 is shown in Figure 8.

The Dynamic Preprocessing Phase

1. Establishing One-way Priority Tables and Searching Absolute Priority Polygons or Separating Planes

As the one-way priority relation of the polygon Qi has been computed, go through the following criteria step by step and then execute the appropriate procedure:

(1) If no polygon is under the categories "back side" and "cutting", do procedures 2(1) and 2(3).
(2) If no polygon is under the categories "front side" and "cutting", do procedures 2(2) and 2(3).
(3) If no polygon is under the category "cutting", do procedure 3.
(4) If Qi is not the last polygon in the current group, go to the next polygon Qi in this group. If the one-way priority relation of Qi has not been computed, compute it. Then do procedure 1 for Qi.
(5) If the current group is not the last group, go to the first polygon Qi of the next group. If the one-way priority relation of Qi has not been computed, compute it. Then do procedure 1 for Qi.
(6) Start to do the static preprocessing for all the one-way priority tables in the feudal priority tree.

While doing the static preprocessing in the latter part of the dynamic preprocessing phase, do procedure 2 of the static preprocessing directly because the one-way priority relations of all the polygons have been built. In order to prevent the building of the feudal priority tree from being affected by the order of the input polygons in the data file, the input polygon is randomly selected in procedure 1 of the dynamic preprocessing phase. In contrast with the dynamic preprocessing, the order of the input polygons in the data file in the static preprocessing is not important because the feudal priority tree is built after the whole one-way priority table has been built.

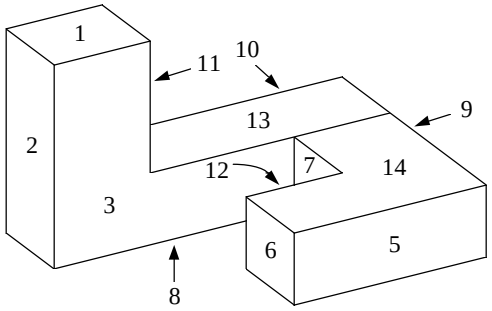

Figure 7. The closed-volume object in Figure 4 after splitting.

Table 3. Two one-way priority tables split from Table 2 by polygon 3.

Q	P -> Q		
	P <| Q	P >| Q	P \- Q
6		7,12,14	
7	6,12		14
12	7	6	14
14		6,7,12	

Q	P -> Q
	P <| Q
11	13
13	11

Table 4. The one-way priority table for the left one in Table 3 after absolute front priority polygons 6 and 14 are removed.

Q	P -> Q		
	P <| Q	P >| Q	P \- Q
7	12		
12	7		

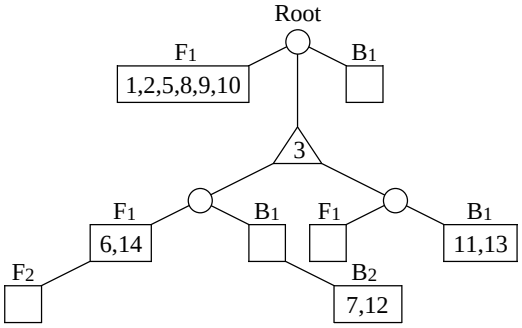

Figure 8. The feudal priority tree built by the static preprocessing for the object in Figure 7.

2. Adding Absolute Priority Polygons to The Feudal Priority Tree and Deleting Them from One-way Priority Tables

(1) Qi which is an absolute back priority polygon is added into the bunch Bj+1 connected to the last bunch Bj on the right side of the current connecting node in Figure 5. The coplanar polygons of Qi having the same normal direction with Qi are also put into Bj+1. The other coplanar polygons of Qi having the reverse normal direction with Qi are put into the bunch Fj+1 connected to the last bunch Fj on the left side of the current connecting node.

(2) Qi which is an absolute front priority polygon is added into the bunch Fj+1 connected to the last bunch Fj on the left side of the current connecting node in Figure 5. The coplanar polygons of Qi having the same normal direction with Qi are also put into Fj+1. The other coplanar polygons of Qi having the reverse normal direction with Qi are put into the bunch Bj+1 connected to the last bunch Bj on the right side of the current connecting node.

(3) The row of Qi is deleted from the one-way priority table. Qi and its coplanar polygons are removed from the previous rows of the one-way priority table. Then search the previous rows of the current table with procedure 1 to find if absolute priority polygons or separating planes exist.

The following polygons in this group are not necessary in computing the one-way priority relations to Qi and its coplanar polygons so the one-way priority relations to be computed can be reduced.

A random order 2, 8, 7, 11, 1, 5, 6, 3, 10, 4, 12, 9 is used to select the input polygons of the object in Figure 4. After the one-way priority row of polygon 2 has been computed, this row is removed for polygon 2 as an absolute front priority polygon. Then the absolute front priority polygon 8 is also removed. The polygons 7 and 11 are not necessary to compute the one-way priority relations to the polygons 2 and 8 in Table 5. The polygons 2 and 8 also do not appear in the row of polygon 1 which is found to be an absolute front priority polygon and will be removed from Table 5.

Table 5. Computing the one-way priority relation of polygon 1 of the object in Figure 4 after absolute priority polygons 2 and 8 are removed.

Q	P -> Q			
	P <| Q	P >| Q	P \- Q	P -- Q
7	3,6,11,12	9	4,5,10	
11	4,5,6,7,9,12		3,10	
1		3,4,5,6,7,9, 10,11,12		

3. Separating The Remaining Polygons with A Separating Plane

The polygon Qi can be a separating plane Sk to all other polygons in the current group except its coplanar polygons. Set Sk as the switch node under the current connecting node in Figure 5 and put its coplanar polygons into the bunch Ok under Sk, and then the polygons on the front side of Sk are put in a group Gf as the left branch and the polygons on the back side of Sk are put in a group Gb as the right branch of the switch node Sk. Revise the current one-way priority table into two one-way priority tables for Gf and Gb. Then search the previous rows of the group Gf or Gb with procedure 1 to find if absolute priority polygons or separating planes exist. While searching the previous rows in a table, only those Qi in which there is no polygon under the category "cutting" can be an absolute priority polygon or a separating plane.

The following polygons in one group are not necessary in computing the one-way priority relations to the polygons in other groups so the one-way priority relations to be computed can be much reduced. The whole procedure for the dynamic preprocessing phase of our approach is illustrated as shown in Figure 9.

The feudal priority tree built by the dynamic preprocessing for the object in Figure 4 with the input polygons being selected in the order 2, 8, 7, 11, 1, 5, 6, 3, 10, 4, 12, 9 is shown in Figure 10. The part below the switch node of the polygon 3 is built by the static preprocessing.

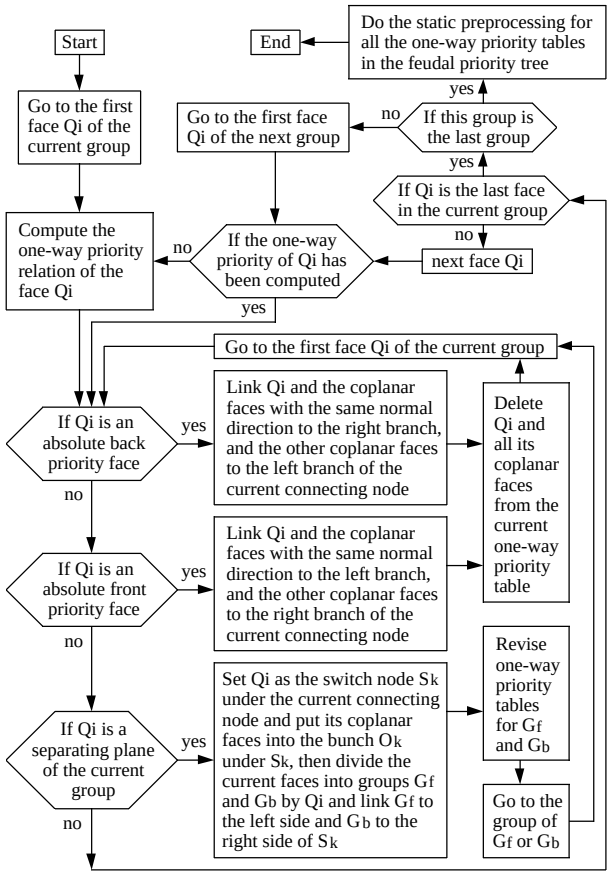

Figure 9. The procedure of the dynamic preprocessing.

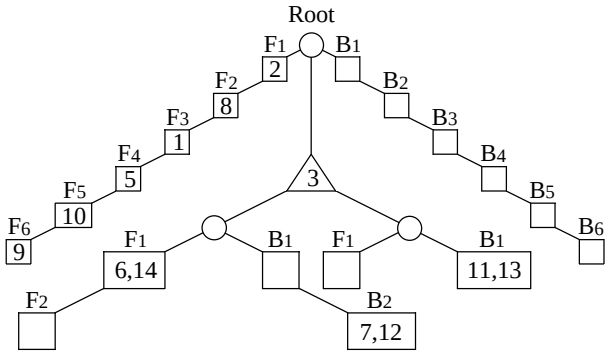

Figure 10. A feudal priority tree built by the dynamic preprocessing for the object in Figure 7.

The Postprocessing Phase

First, the "forward face" and the "backward face" are defined as the following:

(1) A forward face is a polygon whose normal vector forming an angle with the view vector is not greater than 90^o.
(2) A backward face is a polygon whose normal vector forming an angle with the view vector is greater than 90^o.

Instead of the viewpoint, the view vector is used in this paper to compute and decide if one face is forward or backward. The operations to compute the dot product of the view vector with the normal vector of one face involves 3 multiplications and 2 additions, but the operations to substitute the viewpoint into the plane equation of one face includes 3 multiplications and 3 additions. The forward and backward directions are also defined consistently with the direction of the view vector.

For a new view vector, all forward faces in the bunch Fj in Figure 5 are put into the sub-bunch Fjf and all backward faces are put into the sub-bunch Fjb as shown in Figure 11. Similarly, each bunch Bj is divided into the sub-bunches Bjf and Bjb for forward faces and backward faces respectively. For closed-volume objects, all the forward faces are fully obstructed by the backward faces and are invisible so all the faces in the sub-bunches, Fjf and Bjf, can be discarded before drawing. This procedure is known as "back-face culling".

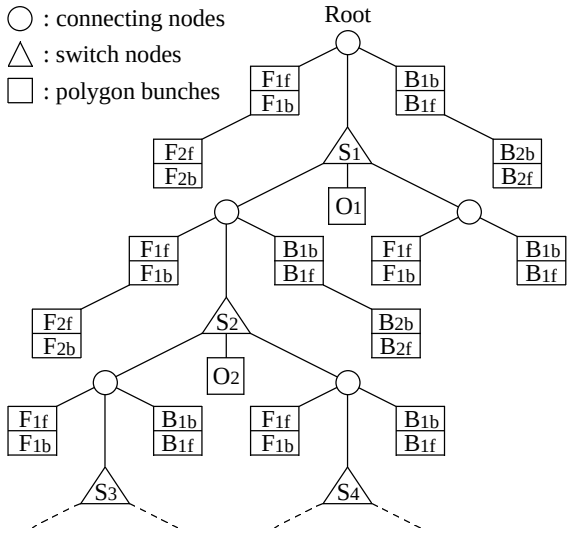

Figure 11. A feudal priority tree for the postprocessing.

Because the faces in Fj and Bj have the same priority, the feudal priority tree is searched level by level. The searching procedure starts from the root which is the first connecting node in the feudal priority tree:

(1) From the head of this connecting node, put the faces of Bjb into the next class element of the rendering priority linked list, and then put the faces of B(j+1)b into the next class element to the previous one, until the tail of this connecting node is reached. (a) Under this connecting node, if a switch node Sk exists and Sk is a forward face, do the connecting node which is on the front side of Sk by procedure (1). (b) Under this connecting node, if a switch node Sk exists and Sk is a backward face, do the connecting node which is on the back side of Sk by procedure (1). (c) If no switch node is under this connecting node, call procedure (2).
(2) From the tail of this connecting node, put the faces of Fjb into the next class element of the priority list, and then put the faces of F(j-1)b into the next class element to the previous one, until the head of this connecting node is reached. (a) If a switch node Sk is above this connecting node, and the other connecting node under Sk has not been done with procedure (1), put the face Sk and its coplanar faces in Ok into the next class, and then go to the other connecting node under Sk and do procedure (1). (b) If a switch node Sk is above this

connecting node, and the other connecting node under Sk has been done with procedure (1), go to the upper connecting node and do procedure (2). (c) If no switch node is above this connecting node, the postprocessing has been finished.

The faces(or polygons) within a "class" element in the rendering priority linked list have the same rendering priority. Therefore, no matter what the drawing order for these faces is, the same picture is obtained . The whole procedure for the postprocessing phase of our approach is illustrated as shown in Figure 12.

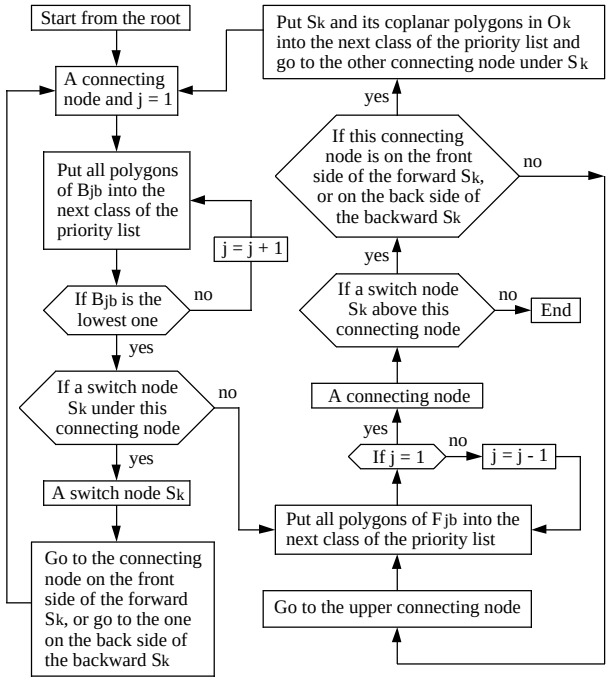

Figure 12. The procedure of the postprocessing for closed-volume objects.

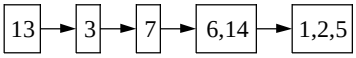

Figure 13. The priority list searched from the feudal priority tree in Figure 8.

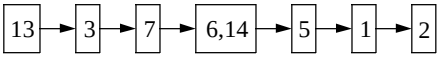

Figure 14. The priority list searched from the feudal priority tree in Figure 10.

For the object in Figure 4, the priority lists searched from the feudal priority trees in the Figures 8 and 10 are shown in Figures 13 and 14 respectively. The priority list obtained from the static preprocessing is always shorter than or at most equal to that obtained from the dynamic preprocessing.

The Procedure for Open-Volume Objects

The procedure of the static preprocessing or the dynamic preprocessing is the same for both closed-volume objects and open-volume objects because the feudal priority tree is constructed for both forward faces and backward faces. Within the postprocessing phase, the forward faces of closed-volume objects are invisible to the viewer so the faces of Fjf and Bjf are discarded in Figure 12. But all the faces of open-volume objects might be visible to the viewer, forward faces can not be culled so the faces of Fjf must be put together with the faces of Bjb into the same class element in the priority list. The class of the faces for Fjb also includes the faces of Bjf.

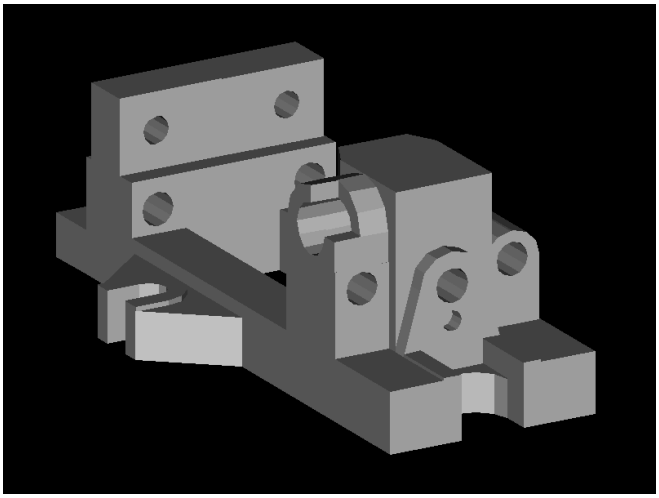

Figure 15. A gray shading display of the base of a machinist‘s vise which is a closed-volume object.

Figure 16. A color display of a house which is a closed-volume object.

IMPLEMENTATION

The static preprocessing and the dynamic preprocessing respectively with the postprocessing have been successfully implemented as two programs in the C language and tested on personal computers to do real-time rendering for both closed-volume and open-volume 3-D objects.

Two examples are displayed in Figures 15 and 16 which are the orthographic projection.

The data of polygons and running time in Tables 6 and 7 are the average values on executing 20 times the corresponding programs for the closed-volume objects in the Figures 15 and 16 respectively. A polygon is assumed to be split into just two smaller polygons so the difference between output polygons and input polygons is the number of splittings. The numbers in the row of "One-way priority relations" are the times of the execution to substitute all the vertices of a polygon into the plane equation of another polygon.

Table 6. Comparing our approach with the BSP tree algorithm for the example in Figure 15.

304 input polygons	FP alg. with static pre-processing	FP alg. with dynamic pre-processing	BSP tree algorithm
Output polygons	324	324	579
Splittings	20	20	275
One-way priority relations	63162	14234	28636
Connecting nodes	71	106	
Switch nodes	35	70	441
Time of pre-processing (ms)	444	113	125
Time of post-processing (ms)	0.603	0.767	7.55

Table 7. Comparing our approach with the BSP tree algorithm for the example in Figure 16.

736 input polygons	FP alg. with static pre-processing	FP alg. with dynamic pre-processing	BSP tree algorithm
Output polygons	741	741	1290
Splittings	5	5	554
One-way priority relations	141942	45744	84596
Connecting nodes	111	221	
Switch nodes	55	135	1013
Time of pre-processing (ms)	1430	406	356
Time of post-processing (ms)	1.23	1.37	17.8

All the nodes in a BSP tree are switch nodes because the searching order depends on the dot product of the view vector and the normal vector of the faces in the nodes.

EVALUATION AND DISCUSSION

The BSP tree algorithm is an extremely efficient method for calculating the visibility relationships among a static group of 3-D polygons as seen from a frequently moving viewpoint [4]. The most time-consuming tasks in the computer for our approach and the BSP tree algorithm are (1) computing the one-way priority relation between each pair of polygons, and (2) splitting polygons. Hence our approach is proposed to compare with the BSP tree algorithm on these two tasks in the preprocessing phase and the postprocessing phase.

The Preprocessing Phase

1. Computing One-way Priority Relations

Usually, establishing a BSP tree with fewer nodes can save time in preprocessing calculation. While establishing this tree at each node, in order to select a polygon as the splitting plane which intersects the fewest polygons n(n-1) one-way priority relations between any two polygons must be computed, but this procedure can not always lead to a BSP tree with the fewest nodes. Therefore, it is necessary to compute one-way priority relations much more than N(N-1) to build a BSP tree, where N is the number of input polygons and n is the number of polygons in a sub-space in the BSP tree. If the splitting plane is selected randomly, it may significantly increase splittings and output polygons. An alternative method is to select five candidate polygons randomly in each sub-space, and then one of the five candidates with the fewest splittings is chosen as the splitting plane [6, 14]. In this paper, the data for the BSP tree algorithm in the Tables 6 and 7 are based on this alternative method. Consequently, the procedure for establishing an optimal BSP tree is uncertain, difficult, and complicated.

In the static preprocessing of our approach, the one-way priority table is established on N(N-1) one-way priority relations. After computing the one-way priority relation of the polygon Qi in the dynamic preprocessing, if Qi is an absolute priority polygon, it is removed before the one-way priority relation of the next input polygon is computed so the following polygons are not necessary in computing one-way priority relations to these removed polygons. As the current polygons are divided into two groups with a separating plane Qi, the polygons in one group are not necessary to compute the one-way priority relations to the polygons in the other group. Therefore, the one-way priority relations to be computed in the dynamic preprocessing are much fewer than that in the static preprocessing and even fewer than that in the preprocessing of the BSP tree algorithm.

2. Splitting Polygons

In this paper, a polygon is assumed to be split by a splitting plane into two smaller polygons. In fact, a concave polygon may be split into more than two simply-connected polygons [13, 3]. To separate simply-connected polygons from the split polygons is complicated and time-consuming work. Hence more polygons to be split will definitely lower the efficiency in the preprocessing and will produce much more output polygons to decrease the efficiency in the postprocessing.

The dynamic preprocessing has much fewer polygon splittings and fewer one-way priority relations to be computed than those in the preprocessing of the BSP tree algorithm. Hence the dynamic preprocessing is faster than the preprocessing of the BSP tree algorithm. If the separation for simply-connected polygons is also involved, the dynamic preprocessing is definitely the best one among these three preprocessing methods. Because the static preprocessing has to compute the fixed N(N-1) one-way priority relations, it is slower than the dynamic preprocessing.

The Postprocessing Phase

In our approach and the BSP tree algorithm, the preprocessing phase which does the most time-consuming work is just run one time for an object to establish a feudal priority tree or a BSP tree respectively. For any new view vector or viewpoint, only the postprocessing phase, which is much faster than the preprocessing phase, is executed again.

In either the feudal priority tree or the BSP tree for both closed-volume and open-volume objects, almost each output polygon has to be decided if it is forward or backward in the postprocessing so the inner product of its normal vector with the view vector must be computed. This is the most time-consuming work in the postprocessing so more output polygons will slow down the efficiency of the postprocessing. The output polygons for an object in a feudal priority tree built by either the static preprocessing or the dynamic preprocessing is much fewer than that in a BSP tree so the postprocessing efficiency of the feudal priority algorithm is greatly better than that of the BSP tree algorithm.

CONCLUSIONS

Our approach introduces a new concept of "absolute priority" to remove several outward polygons by which the other polygons are surrounded before searching for a separating plane in the current polygon group so the possibility of finding the separating plane is highly enhanced. Hence the splittings are greatly reduced. This is the important concept of "separating before splitting" because if there are more splittings, then it is necessary to spend much more time in preprocessing and this produces much more output polygons which greatly decrease the efficiency both in the preprocessing and the postprocessing [1, 2]. Therefore, if separating is possible, to separate polygons is always better than to split polygons. If there are several planes with the least cutting polygons, select the one with the most balanced polygons on its front side and back side as the splitting plane.

The approach proposed in this paper has been compared with the well known BSP tree algorithm in both the preprocessing and the postprocessing:

(1) In the dynamic preprocessing, an absolute priority polygon is removed at once as it is found and the current polygons are immediately separated into two groups while a separating plane is searched for; therefore, the one-way priority relations to be computed are much fewer than that in the static preprocessing and even fewer than that in the preprocessing of the BSP tree algorithm. The efficiency of the dynamic preprocessing is better than that of the preprocessing of the BSP tree algorithm. If all split polygons are separated into simply-connected polygons, the dynamic preprocessing is much faster than the preprocessing of the BSP tree algorithm.

(2) Polygon splittings either in the dynamic preprocessing or in the static preprocessing are much fewer than that in the preprocessing of the BSP tree algorithm. Consequently, the output polygons in the feudal priority tree is much fewer than that in the BSP tree so the efficiency of the postprocessing of our approach is much better than that of the BSP tree algorithm.

The dynamic preprocessing is very efficient in both the preprocessing and the postprocessing. One drawback of the dynamic preprocessing is that its feudal priority tree is not as brief and uncomplicated as that built by the static preprocessing so the rendering priority list obtained by the dynamic preprocessing is longer than that obtained by the static preprocessing.

A topological property in our approach is that the polygons in the two bunches with the same level linked to a connecting node have the equal priority while searching the priority list. Among the polygons in the same class of a priority list, no one can be obstructed by the other polygons; hence these polygons have the same rendering priority that is another topological property in our approach.

Our approach keeps most output polygons without splitting so this maintains their completeness. But most input polygons are split by the BSP tree algorithm into smaller polygons. The information of the input polygons in our approach is kept more complete than that in the BSP tree algorithm so this is a great advantage for our approach in the CAM application.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the National Science Council of The Republic of China under the grant NSC82-0115-E-002-396, and specially thank Jenn-Tsuen Wu for his help in creating the data file of the example house.

REFERENCES

[1] CHEN, HAN-MING. *A Real-Time Rendering System for 3-D Objects in Computer-Aided Design and Manufacturing*, Ph.D. Dissertation, University of California at Berkeley, May 1991.

[2] CHEN, HAN-MING, and ADIGA, S. An Ingenious Algorithm for Hidden-Surface Removal. *The Proceedings of The Second International Conference on CAD & CG*. September 1991, Hangzhou, China, pp.159-164.

[3] CHEN, HAN-MING, and JIANG, L. S. Partitioning A Concave Polygon into Simply-Connected Polygons. *The Proceedings of The Twenty-third Midwestern Mechanics Conference*, October 1993, Lincoln, U.S.A., pp.75-77.

[4] FOLEY, J. D., VAN DAM, A., FEINER, S. K., and HUGHES, J. F. *Computer Graphics: Principles and Practice*, second ed. Addison-Wesley, Reading, MA, 1990.

[5] FUCHS, H., KEDEM, Z. M., and NAYLOR, B. F. On Visible Surface Generation by A Priori Tree Structures. *Computer Graphics*, Vol.14(3), July 1980, pp.124-133.

[6] FUCHS, H., ABRAM, G. D., and GRANT, E. D. Near Real-Time Shaded Display of Rigid Objects. *Computer Graphics*, Vol.17(3), July 1983, pp.65-72.

[7] NAYLOR, B. F. *A Priori Based Techniques for Determining Visibility Priority for 3-D Scenes*, Ph.D. Dissertation, University of Texas at Dallas, May 1981.

[8] NEWELL, M. E., NEWELL, R. G., and SANCHA, T. L. A Solution to the Hidden Surface Problem. *The Proceedings of the ACM National Conference*, 1972, pp.443-450.

[9] PREPARATA, F. P., and SHAMOS, M. I. *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[10] SCHUMACKER, R. A., BRAND, B., GILLILAND, M. G., and SHARP, W. H. Study for Applying Computer-Generated Images to Visual Simulation. *Technical Report AFHRL-TR-69-14*, NTIS AD700375fR, U.S. Air Force Human Resources Lab., Air Force Systems Command, Brooks AFB, TX, September 1969.

[11] SUTHERLAND, I. E., SPROULL, R. F., and SCHUMACKER R. A. Sorting and the Hidden-Surface Problem. *The Proceedings of the National Computer Conference*, 1973, pp.685-693.

[12] SUTHERLAND, I. E., SPROULL, R. F., and SCHUMACKER, R. A. A Characterization of Ten Hidden-Surface Algorithms. *ACM Computing Surveys*, Vol.6(1), March 1974, pp.1-55.

[13] SUTHERLAND, I. E., and HODGMAN, G. W. Reentrant Polygon Clipping. *Communications of the ACM*, Vol.17(1), January 1974, pp.32-42.

[14] THIBAULT, W. C., and NAYLOR, B. F. Set Operations on Polyhedra Using Binary Space Partitioning Trees. *Computer Graphics*, Vol.21(4), July 1987, pp.153-162.
[15] YAO, F. F. On the Priority Approach to Hidden-Surface Algorithms. *The Proceedings of the IEEE Symposium on the Foundations of Computer Science*, 21st Annual, 1980, pp.301-307.