Vol. 3, No. 1: 1–14

Clustered Backface Culling

Andreas Johannsen and Michael B. Carter
Engineering Animation, Inc.

Abstract. This paper presents a simple, practical, and effective backface-culling technique for clusters of polygons, as well as a method for generating efficient clusters from a set of triangle strips. The cluster-backface test is directly derived from the traditional single-polygon test, and has about the same complexity. Memory requirements are 40 bytes per test. Cluster-backface tests may be arranged hierarchically, and frontface tests added for symmetry. Experiments show graphics performance improvements of up to 50 percent in terms of the number of effective polygons rendered per second.

1. Introduction

Shaded rendering of polygonal models at interactive frame rates has been a core research area of computer graphics since its inception. Contemporary workstations and personal computers with hardware z-buffer graphics accelerators are capable of rendering several million triangles a second. However, advances in graphics accelerators have constantly been outpaced by the demands of applications; models with tens or hundreds of millions of polygons are routinely encountered, for example, in the design and evaluation of mechanical systems such as cars and airplanes. Approaches to the large-model rendering problem concentrate on identifying a minimal subset of polygons that represents the given model, and rendering only this subset. One long-used technique is *backface culling*, whereby only the polygons facing in the direction of the viewpoint are rendered.

The rendering pipeline of common graphics accelerators consists, roughly speaking, of two stages: a transform stage that transforms three-dimensional

1086-7651/98 $0.50 per page

vertex and normal data by a 4×4 matrix to view coordinates, and a rasterizer stage that produces shaded pixels and places them into a frame buffer. Also, polygonal data typically reside in RAM and have to be transferred to the graphics accelerator by some means, e.g. the system bus.

With respect to the transfer/transform/rasterizer pipeline, traditional hardware-based backface culling operates after the transform stage and before the rasterizer stage. Pipeline throughput is limited by its slowest stage; if this happens to be the rasterizer, then backface culling may improve overall graphics performance by up to a factor of two, as half of all polygons face away from the viewer on average. However, if the pipeline bottleneck is presented by the transfer or transform stage, as is not uncommon for contemporary graphics accelerators, then backface culling may have no effect on performance, or even be detrimental. In these cases, it is desirable to perform backface culling on the host CPU before sending data to the graphics accelerator. The traditional single-polygon backface-culling test, however, takes time linear in the number of polygons, and proves impractical for this purpose (even though the test may be accelerated [Zhang, Koff 97]).

In order to perform backface culling in sub-linear time, it is necessary to have a single test decide upon the orientation of a whole cluster of polygons. Such a test has to be conservative and identify clusters that are guaranteed to contain only back-facing polygons; any cluster for which this fact cannot be established, or that contains at least one front-facing polygon, must be rendered. Thus, culling effectiveness for clusters of polygons is smaller than for single polygons, and polygons that are actually back-facing may erroneously be sent to the graphics accelerator (but will consequently be culled by the accelerator).

For simple models, clusters of back-facing polygons may be identified by hand [Foley et al. 90]. Consider, for example, a cube centered at the origin, with faces perpendicular to the coordinate axes. If the eye point of a perspective view is located in the positive octant, then the three faces with negative normal coordinates are all back-facing. This relationship can easily be established for each of the eight octants, and the result stored in a lookup table. At rendering time, this table is indexed with the octant in which the eye point is located.

Manually-constructed tables are clearly impractical for all but very simple and/or symmetric polyhedra. Aspect graphs [Gigus et al. 91] subdivide space into cells that correspond to eye-point equivalence classes with respect to face visibility. However, for a polyhedron with N faces, there are worst case $O(N^9)$ cells, which makes such a general indexing technique impractical.

For convex polyhedra, Tanimoto suggests an algorithm that incrementally updates the object-space outline separating back-facing and front-facing polygons, using frame to frame coherency, [Tanimoto 77]. For general polyhedra, Kumar proposes a cluster-culling test based on the explicit construction of an object-space cell that contains all eye points for which the cluster is backfacing, [Kumar et al. 96]. Such a construction is awkward, and the ensuing space subdivision consumes considerable time and memory.

This paper presents a simple, practical, and effective backface-culling technique for clusters of polygons. The cluster-backface test is directly derived from the traditional single-polygon test, and has about the same complexity; its operation is reminiscent of the simple cube example given above. Memory requirements are 40 bytes per test, assuming single-precision floating point data. Cluster-backface tests may be arranged hierarchically, and frontface tests added for symmetry (consuming only an additional four bytes per test). Experiments show graphics performance improvements of up to 50 percent in terms of the number of effective polygons rendered per second.

2. Cluster-Backface Test

Let the eye point of a perspective view be given as $\mathbf{e} = [e^x, e^y, e^z]^T$ in world coordinates. Any plane polygon with normal $\mathbf{n}$ and one vertex point $\mathbf{p}$, also in world coordinates, faces away from $\mathbf{e}$ iff $\mathbf{n} \cdot (\mathbf{e} - \mathbf{p}) \leq 0$. This is the traditional backface-culling test. Now, consider a cluster of N polygons, with normals $\mathbf{n}_i$ and points $\mathbf{p}_i$; all polygons face away from $\mathbf{e}$ iff

$$\mathbf{n}_i \cdot (\mathbf{e} - \mathbf{p}_i) \leq O, \quad \text{for } 1 \leq i \leq N. \tag{1}$$

2.1. Derivation

The goal is to derive one succinct test that uses precomputed quantities depending on the $\mathbf{n}_i$ and $\mathbf{p}_i$, but not $\mathbf{e}$. The following simple inequalities are needed for the derivation of the cluster-backface culling test:

$$\max_i(a_i - b_i) \leq \max_i(a_i) - \min_i(b_i) \tag{2}$$

$$\max_i(a_i + b_i) \leq \max_i(a_i) + \max_i(b_i) \tag{3}$$

$$\max_i(a_i, k) = \begin{cases} \max_i(a_i)k & \text{for } k \geq 0 \quad (4a) \\ \min_i(a_i)k & \text{for } k < 0 \quad (4b) \end{cases}$$

As a first step, the N inequalities of (1) are equivalent to the single inequality

$$\max_i[\mathbf{n}_i \cdot (\mathbf{e} - \mathbf{p}_i)] \leq 0. \tag{5}$$

The lefthand side of (5) still contains $\mathbf{e}$, but an upper bound may be found as follows. Distributing the dot product and applying (2),

$$\max_i[\mathbf{n}_i \cdot (\mathbf{e} - \mathbf{p}_i)] \leq \max_i(\mathbf{n}_i \cdot \mathbf{e}) - \min_i(\mathbf{n}_i \cdot \mathbf{p}_i), \tag{6}$$

which is a step in the right direction, as the minimum plane offset $d \equiv \min_i(\mathbf{n}_i \cdot \mathbf{p}_i)$ may be precomputed. Substituting d and expanding the dot product, we can rewrite (6) as

$$\max_i[\mathbf{n}_i \cdot (\mathbf{e} - \mathbf{p}_i)] \leq \max_i(n_i^x e^x + n_i^y e^y + n_i^z e^z) - d$$

and using (3), we get

$$\max_i[\mathbf{n}_i \cdot (\mathbf{e} - \mathbf{p}_i)] \leq \max_i(n_i^x e^x) + \max_i(n_i^y e^y) + \max_i(n_i^z e^z) - d.$$

If $\mathbf{e}$ lies in the positive octant, with all three coordinates e^x, e^y, e^z positive, then (4a) applies to all $\max_i()$ terms:

$$\max_i[\mathbf{n}_i \cdot (\mathbf{e} - \mathbf{p}_i)] \leq \max_i(n_i^x)e^x + \max_i(n_i^y)e^y + \max_i(n_i^z)e^z - d. \tag{7}$$

For any negative coordinate of $\mathbf{e}$, the corresponding $\max_i()$ term changes to a $\min_i()$ term according to (4b). This may be expressed in vector notation by defining component-wise for $c \in \{x, y, z\}$,

$$\text{Comp}_{\mathbf{e}}(\mathbf{a}, \mathbf{b})^c \equiv \begin{cases} a^c & \text{for } e^c \leq 0. \\ b^c & \text{for } e^c > 0. \end{cases}$$

Then, expression (7) can be written as

$$\max_i[\mathbf{n}_i \cdot (\mathbf{e} - \mathbf{p}_i)] \leq \text{Comp}_{\mathbf{e}}(\mathrm{N}, \mathrm{M}) \cdot \mathbf{e} - d,$$

with

$$\mathrm{N} \equiv [\min_i(n_i^x), \min_i(n_i^y), \min_i(n_i^z)]^T$$
$$\mathrm{M} \equiv [\max_i(n_i^x), \max_i(n_i^y), \max_i(n_i^z)]^T$$

Considering that the righthand side of the inequality (7) is an upper bound for $\max_i[\mathbf{n}_i \cdot (\mathbf{e} - \mathbf{p}_i)]$, a backface-culling test for clusters is

$$\text{Comp}_{\mathbf{e}}(\mathrm{N}, \mathrm{M}) \cdot \mathbf{e} - d \leq 0 \tag{8}$$

with pre-computed quantities $\mathrm{N}, \mathrm{M}, d$ as defined above. These quantities, two 3-vectors and a scalar, consume 28 bytes of storage space in single precision.

2.2. *Interpretation and Examples*

For any particular eye point $\mathbf{e}$, the lefthand side of (8) merely presents a plane equation with normal $\text{Comp}_{\mathbf{e}}(\mathrm{N}, \mathrm{M})$ and offset d. Therefore, the test may be

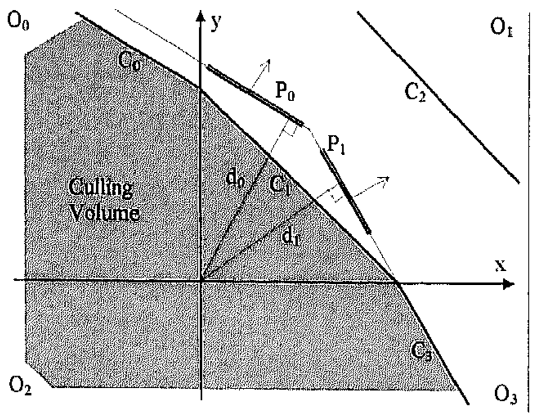

Figure 1. Cluster-backface test geometry (two-dimensional).

interpreted as defining eight culling planes, one associated with each octant within which an eye point may lie.

All culling planes C_i meet at coordinate axes, and therefore along octant boundary planes. Consider, for example, the four octants with a positive x-coordinate. The four corresponding culling planes have the same normal vector x-coordinate M_x, so they all intersect the x-axis at $x_0 = \frac{d}{M_x}$. Thus, the culling planes define a *culling volume*: if the eye point lies anywhere in the culling volume, all polygons in the cluster are backfacing and may be culled.

As a two-dimensional example, consider two polygons P_0 and P_1 as in Figure 1, rotated 30° and 60° versus the x-axis, respectively. Polygon normals are

$$\mathbf{n}_0 = \begin{bmatrix} 0.5 \\ \frac{\sqrt{3}}{2} \end{bmatrix} \quad \text{and} \quad \mathbf{n}_1 = \begin{bmatrix} \frac{\sqrt{3}}{2} \\ 0.5 \end{bmatrix}$$

yielding

$$\mathbf{N} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \quad \text{and} \quad \mathbf{M} = \begin{bmatrix} \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix}$$

The minimum plane offset is the one for P_1, $d_1 = \mathbf{n}_1 \cdot \mathbf{p}_1$, which results in culling planes C_i for octants O_i, as indicated in Figure 1.

Note that C_2 lies outside of O_2, so the cluster $\{\mathrm{P}_0, \mathrm{P}_1\}$ will be culled for all eye points within O_2. Generally, the culling planes C_2 and C_3 must share a y-axis intersection. Likewise, culling planes C_2 and C_0 must share an x-axis intersection. In the example of Figure 1, both intersections are positive, so C_2 lies outside of O_2, and all of O_2 lies within the culling volume.

As an important special case, consider N coplanar polygons. All plane normals and offsets are the same,

$$\mathbf{n}_i = \mathbf{n}, \quad \mathbf{n}_i \cdot \mathbf{p}_i = t, \quad \text{for } 1 \le i \le N,$$

consequently, the precomputed data is $\mathbf{N} = \mathbf{M} = \mathbf{n}$, $d = t$, and all culling planes C_i coincide with the polygon plane. Therefore, the cluster-backface test is sharp for coplanar polygons.

Another special case occurs for arbitrary clusters, when the eye point is located at the coordinate system origin. Substituting $\mathbf{e} = \mathbf{0}$ into (8), the cluster-backface test is $-\min_i(\mathbf{n}_i \cdot \mathbf{p}_i) \leq 0$, which is exactly the case when there are only positive polygon plane offsets. Therefore, the cluster-backface test is also sharp for $\mathbf{e} = \mathbf{0}$.

2.3. Hierarchy

Two polygon clusters' precomputed data can be merged in the following way:

$$\begin{aligned}\mathbf{N}_* &= [\min(\mathbf{N}_1^x, \mathbf{N}_2^x), \min(\mathbf{N}_1^y, \mathbf{N}_2^y), \min(\mathbf{N}_1^z, \mathbf{N}_2^z)]\\ \mathbf{M}_* &= [\min(\mathbf{M}_1^x, \mathbf{M}_2^x), \min(\mathbf{M}_1^y, \mathbf{M}_2^y), \min(\mathbf{M}_1^z, \mathbf{M}_2^z)]\\ d_* &= \min(d_1, d_2)\end{aligned}$$

If backface-culling tests are arranged hierarchically, it makes sense to include frontface culling tests in order to create symmetric culling behavior. Hierarchy traversal may be pruned at inner nodes for frontfacing as well as backfacing clusters. The derivation of a frontface test proceeds as in section 2.1, only with all polygon normals reversed, and yields $\mathrm{Comp}_\mathbf{e}(-\mathbf{M}, -\mathbf{N}) \cdot \mathbf{e} + \max_i(\mathbf{n}_i \cdot \mathbf{p}_i) \leq 0$. Note that the only data that has to be stored in addition to that of a backface test the scalar is $\max_i(\mathbf{n}_i \cdot \mathbf{p}_i)$.

2.4. Local Test Coordinates

Examining equation (8), d becomes arbitrarily negative as the $\mathbf{p}_i$ move away from the world-coordinate origin in a direction opposite that of the normals $\mathbf{n}_i$. In this way, the cluster-backface test may never be satisfied for reasonable eye points. It will be more successful if the geometry being culled lies closer to the origin.

In order to combat the effect of arbitrarily-positioned $\mathbf{p}_i$, a local coordinate system may be chosen for each cluster-backface test, tailored to the polygons in the cluster. This involves transforming polygon data to the local system during preprocessing, in order to compute $\mathbf{N}, \mathbf{M}, d$. At rendering time, the eye point needs to be transformed into the same local coordinates before performing the cluster-culling test.

To minimize eye-point transformation overhead and memory requirements, only origin translations are considered here, i.e., the local test-coordinate system is world-coordinate axis-aligned, but has origin $\mathbf{b}$. Since it is difficult to determine the optimal value for $\mathbf{b}$, we try a variety of values including the eight corners of the cluster's bounding box and the cluster's centroid. A method for evaluating the appropriateness of a particular value of $\mathbf{b}$ is discussed below in Section 3, in terms of near-field culling probabilities.

A point $\mathbf{x}$ in world coordinates is transformed into test coordinates by $\mathbf{x}' = \mathbf{x} - \mathbf{b}$. Of the precomputed data, only d is affected, $d = \min_i[\mathbf{n}_i \cdot (\mathbf{p}_i - \mathbf{b})]$, and the eye point to be used in (8) is $\mathbf{e}' = \mathbf{e} - \mathbf{b}$. The cluster-backface test variables are now $\mathbf{N}, \mathbf{M}, d$, and $\mathbf{b}$, consuming 40 bytes in single precision.

Two polygon clusters' precomputed data can be merged as above, however a new test-coordinate origin $\mathbf{b}_*$ needs to be chosen, and d_* recomputed as $d_* = \min(\min_i[\mathbf{n}_{1,i} \cdot (\mathbf{p}_{1,i} - \mathbf{b}_*), \min_i[\mathbf{n}_{2,i} \cdot (\mathbf{p}_{2,i} - \mathbf{b}_*)]$.

Notice that re-expanding with bias vector $\mathbf{b}_*$ forces us to iterate over all the polygons explicitly to compute the merged $\mathbf{d}_*$. There is no way to shift the expansion to a new center vector $\mathbf{b}_*$ without incurring this additional work. Choosing a test-coordinate system in this fashion may be considered analogous to shifting the center of expansion in a Taylor series. There are ways of shifting the center of analytic expansions, such as the multi-pole expansion [Greengard, Rokhlin 87], but the losses incurred by the max and min operators preclude us from doing the same here.

3. Cluster Generation

Given a backface-culling test for clusters of polygons, we need an algorithm to form these clusters efficiently from the original geometry. In order to drive any algorithm, there has to be a measure of cluster appropriateness. In the present context, it makes sense to speak in terms of the average probability that a cluster will be culled.

Strictly speaking, the eye point may be viewing a cluster of polygons from anywhere in three-dimensional space. For many applications, however, eye points will tend to be located close to the geometry—within a few times the size of the cluster's bounding box. An axis-aligned box of size $R \times R \times R$ around the cluster defines a finite volume within which to calculate the cluster's culling probability. Here, we assume that R will be about 10 times the size of the model's bounding-box diagonal. Within this box, the cluster's culling probability is computed as the ratio of the volume enclosed by the culling volume (Section 2.2) to the box volume. We call this probability the *near-field culling probability*, because eye points are restricted to being relatively near the cluster.

The geometric size of the culling volume is the sum of eight sub-volumes, one for each octant in which a culling plane is defined. In each octant, the culling volume is the volume inside a cube of edge length $R/2$, and below the culling plane.

Near-field culling probabilities are heavily affected by different choices of the local test-coordinate system. As described in Section 2.4, it makes sense to place the test-coordinate system's origin in the vicinity of the cluster polygons. A few reasonable origin candidates may be examined, and the one with maximum near-field probability accepted, for example, the eight corners of the cluster's bounding box, and the polygon points' centroid.

For maximum rendering performance, models are typically not specified in individual polygons, but in tristrips. Some tristrips may be quite long—long enough, in fact, to have a culling probability of zero. In order to address backface culling effectively, long tristrips must be broken into shorter ones, thereby increasing the culling probability of each piece. The fragments must not, however, become so short that the benefits of tristripping are totally lost.

Consider two clusters A and B. Is it faster to render them separately, performing two cluster-backface tests, or to render them as a single merged cluster, C, performing just one test? This is the driving condition used in the cluster-forming algorithm.

Let T be the time it takes to perform one cluster-backface test, P_X the near-field culling probability of a cluster X, and C_X the time to render cluster X. The average time to test and render cluster X is $T + (1 - P_X)C_X$. For two clusters A and B, a merged cluster C makes sense only if the following condition is satisfied:

$$T + (1 - P_C)C_C < 2T + (1 - P_A)C_A + (1 - P_B)C_B.$$

Assuming the cost of rendering a single merged cluster equals the sum of the costs of its constituent clusters,

$$\begin{aligned}(1 - P_C)(C_A + C_B) < T + (1 - P_A)C_A + (1 - P_B)C_B \Leftrightarrow \\ P_C > \frac{T + P_A C_A + P_B C_B}{C_A + C_B}.\end{aligned} \tag{9}$$

The rendering cost C_X can be determined by examining the tristrips contained in the cluster. Assuming a transfer/transform bound rendering pipeline, C_X may be approximated by the rendering cost R of one vertex times the number of vertices in the cluster. Assuming N and M vertices in clusters A and B, respectively, substituting into (9) yields

$$P_C > \frac{\frac{T}{R} + P_A N + P_B M}{N + M} \tag{10}$$

There is only one unknown quantity in (10): T/R, the ratio of the time to perform one cluster-backface test to the time to render one vertex. For a given machine configuration and vertex complexity, this ratio should be a measurable constant and valid no matter what geometry is being rendered. By vertex complexity, we mean to point out the difference between models containing normals, textures, and/or colors. In each case, the time to render a vertex will be different, but T/R for that particular type of vertex will be constant.

Relationship (10) may be used to drive the breaking of tristrips: individual triangles are added to the resultant tristrip until its culling probability becomes too low.

Algorithm 1 presents a pseudo-code version of the cluster-forming algorithm. The algorithm's input is an unsorted list of tristrips. Its output is a set of clusters to be used for rendering.

Algorithm 1. Cluster-forming.

```
Given a list of tristrips P
Choose a "seed tristrip" S with highest culling probability
Start a new cluster C with S
Do
     Get the next tristrip T from P whose normals MN
        point most in the direction of the normals of S
     If the culling probability of the merged cluster
        of C|T satisfies (10)
          Append T to C
          Remove T from P
     Else
          Break do
     End If
While P is not empty
Output the final cluster C
Recur on remaining primitives in P
```

4. Experimental Results

This section presents data from experiments investigating the effect of clustered backface culling on graphics performance. Three different models are investigated, (Figures 2, 5, 7, and Table 1). "Average" values in the tables below were computed from 100 randomly-selected viewpoints lying on a sphere about the model in question. The sphere radius was chosen so that the model fills the screen. All data were collected on a Hewlett-Packard C200 workstation with an FX/6 graphics accelerator. The presence of an extremely fast graphics accelerator represents the most challenging situation in which to evaluate a backface-culling algorithm.

Model	Number of Polygons	Number of Tristrips	Average % Triangles Backfacing
Spherical Dome	3726	438	55.7
Fishing Reel	3074	970	51.3
Locking Ring	2604	496	53.6

Table 1. Test models used.

Each of the three models was tested with six values for the T/R ratio of cluster-backface test time over vertex-rendering time. This ratio controls the degree of clustering achieved by the algorithm described in Section 3. Lower values of T/R tend to result in smaller and more numerous clusters. Higher values of T/R tend to result in larger and less numerous clusters. The ratio is varied in the experiments to illustrate that for a given hardware platform and vertex complexity, the best performance of the backface-culling algorithm occurs at the same value of T/R regardless of the model being rendered.

Table 2 presents detailed performance and behavioral data for the spherical dome. This is simply a sphere with a flat cut just below the equator. It is highly tristripped, and has a regular topology (Figure 2). The uneven decrease in the number of clusters is due to the large average number of triangles per tristrip.

The fifth column in Table 2, Average Effectiveness, lists the average percentage of geometrically-backfacing triangles culled by cluster-backface tests. Two primary factors conspire to reduce the algorithm's effectiveness below 100%. One is the approximate nature of the cluster-backface test upon multiple, non-coplanar tristrips. Another is the effect of clustering, making it necessary to render a whole cluster, even if only one triangle is frontfacing.

The final column in Table 2, Average Speedup, lists the increase in rendering speed seen from the original tristripped model to the backface-culled model. Notice that this column contains a peak value of 31% at intermediate values of T/R. Looking ahead at the results for the other models tested (Tables 3 and 4), we see in each case that there is a well-defined peak around $T/R = 1.0$.

T/R Ratio	Number of Clusters	Average Clusters Backfacing	Average Triangles Backfacing	Average % Effectiveness	Average % Speedup
0.5	129	50.0%	30.9%	55.5%	16%
1.0	87	48.7%	30.1%	54.1%	14%
1.5	145	47.4%	46.8%	84.0%	31%
2.0	86	46.4%	45.1%	80.9%	31%
2.5	68	46.3%	43.7%	78.5%	26%
3.0	49	44.1%	41.9%	75.3%	18%

Table 2. Results on spherical dome.

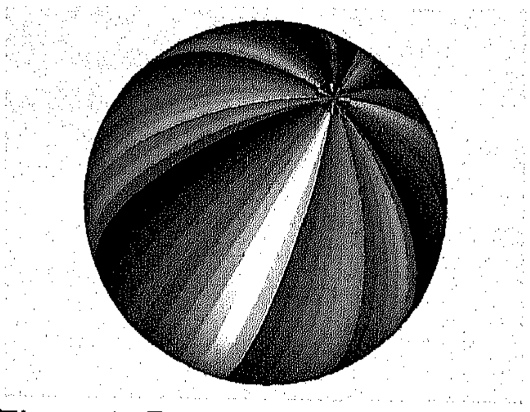

Figure 2. Dome with colored tristrips.

For lower values of T/R, the clustering algorithm forms more clusters. Each of these clusters represents a backface test that must be performed while rendering the model. As more cluster-backface tests are made, less time is spent actually rendering. Moreover, graphics accelerators perform better when multiple primitives can be rendered at once, and more small clusters mean more overhead in the rendering loop.

For larger values of T/R, the clustering algorithm forms fewer clusters, on more widely-varying normals, reducing the near-field culling probabilities. This leads to fewer successful cluster-backface tests, and more actually backfacing triangles sent to the graphics accelerator.

For the dome model, both effectiveness and speedup are at their peak values for $T/R = 1.5$. Because this model is highly tristripped from the outset, it represents a very disadvantageous situation for the clustering algorithm. Even so, it realizes a 31% increase in rendering speed. Figure 3 shows the spherical dome processed with $T/R = 1.5$, and each cluster colored differently.

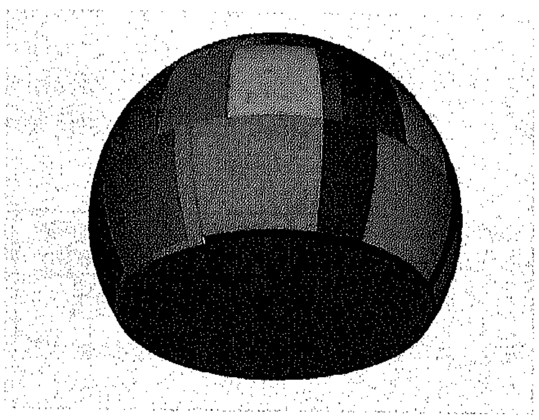

Figure 3. Dome with colored clusters.

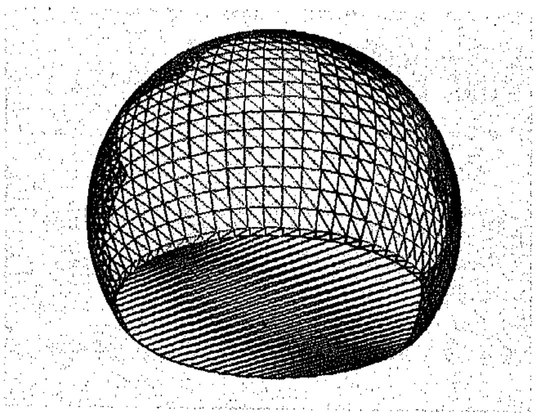

Figure 4. Dome with colored lines.

T/R Ratio	Number of Clusters	Average Clusters Backfacing	Average Triangles Backfacing	Average Effectiveness	Average Speedup
0.5	368	45.4	39.7	77.4	45
1.0	248	41.5	41.3	80.5	50
1.5	164	38.6	38.6	75.3	41
2.0	121	36.6	36.7	71.5	37
2.5	98	35.9	35.4	69.0	29
3.0	81	33.9	33.3	64.8	29

Table 3. Results on fishing reel.

Figure 4 shows the dome in line-rendering mode, to verify that backfacing clusters are indeed not being rendered. Note that some clusters containing backfacing polygons near the silhouette are rendered.

Table 3 shows data for the fishing reel. This model is more irregular in construction, with shorter average tristrips than the dome. Consequently, the clustering algorithm has more choices for forming clusters. The result is a smoothly decreasing number of clusters as T/R increases. For the fishing reel, peak effectiveness and speedup come at $T/R = 1.0$, with substantial values of 80% and 50%, respectively. Figures 5 and 6 show the fishing reel with colored clusters and lines respectively.

Table 4 shows data for the locking ring—a roughly annular shape with several cuts along its perimeter. The data are strikingly similar in character to those found in Table 3. Again, peak effectiveness and speedup comes at $T/R = 1.0$. Figures 7 and 8 show the locking ring with colored clusters and lines respectively.

For the two mechanical parts, peak effectiveness and speedups coincide for the same value of $T/R = 1.0$. This trend holds up well for other models we have examined. The dome model has very long tristrips that must be broken down in order to form clusters, and so behaves differently than the two less-regular models.

Effectiveness is completely independent of machine hardware—it only depends upon the value of T/R and the model geometry. Speedup, on the other hand, is strongly dependent upon the host CPU performance, the graphics-accelerator performance, and effectiveness. The fact that effectiveness and

T/R Ratio	Number of Clusters	Average Clusters Backfacing	Average Triangles Backfacing	Average Effectiveness	Average Speedup
0.5	156	47.7	26.0	48.5	35
1.0	211	42.4	41.5	77.5	47
1.5	157	40.6	40.6	75.8	43
2.0	115	38.5	38.7	72.2	36
2.5	93	36.6	35.8	66.8	31
3.0	76	34.4	34.5	64.4	29

Table 4. Results on locking ring.

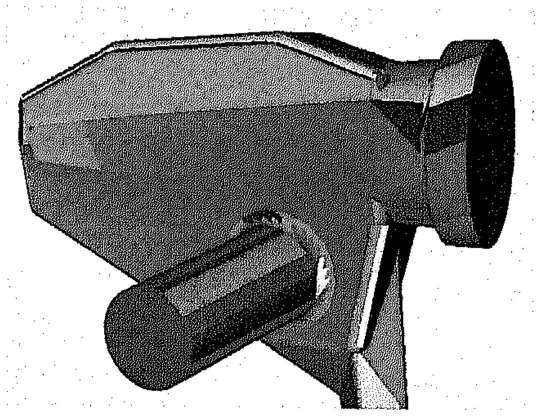

Figure 5. Fishing reel with colored clusters.

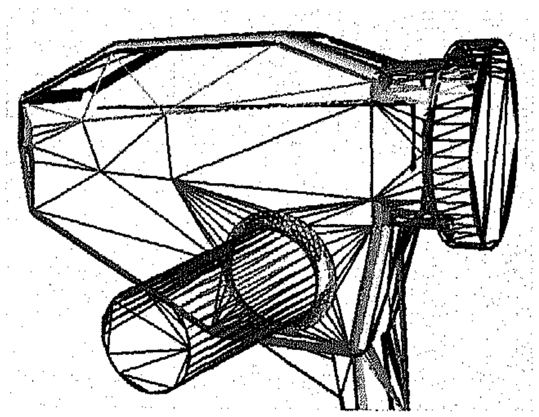

Figure 6. Fishing reel rendered with lines.

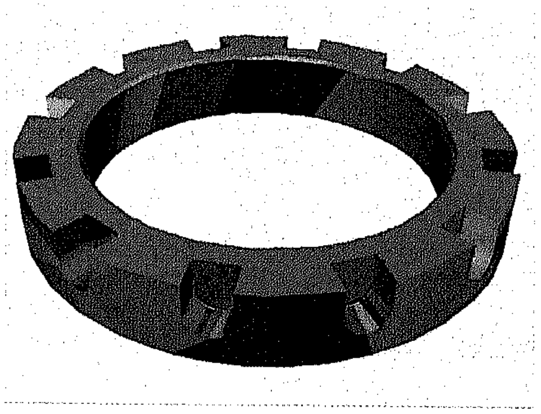

Figure 7. Locking ring with colored clusters.

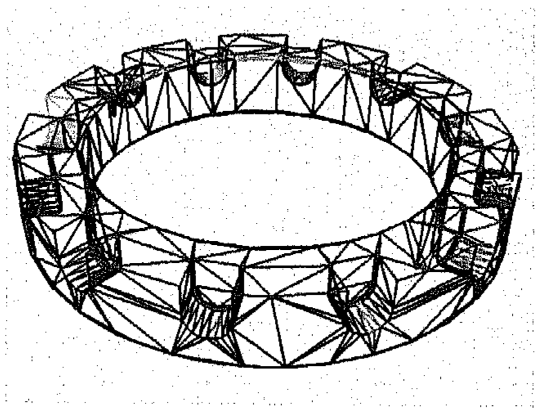

Figure 8. Locking ring rendered with lines.

speedup are in phase with one another implies that the clustering algorithm's output is well-suited to the hardware upon which it has been run.

5. Conclusion

This paper has demonstrated a backface-culling algorithm that achieves aggregate speedups of up to 50% over the tristripped models, with a minimal cost in additional storage. The scheme can cull entire clusters of polygons on the host CPU, bypassing the need to send large amounts of geometric data to the graphics accelerator.

References

[Foley et al. 90] J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics—Principles and Practice.* Second edition, Reading, MA: Addison Wesley, 1990.

[Gigus et al. 91] Z. Gigus, J. Canny, and R. Seidel. "Efficiently Computing and Representing Aspect Graphs of Polyhedral Objects." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:542-551, (1991).

[Greengard, Rokhlin 87] L. Greengard and V. Rokhlin. "A Fast Algorithm for Particle Simulations." *Journal of Computational Physics*, 73: 325-349 (1987).

[Kumar et al. 96] S. Kumar, D. Manocha, B. Garrett, and M. Lin. "Hierarchical Back-Face Computation." *Eurographics Workshop on Rendering* (1996).

[Tanimoto 77] S. Tanimoto. "A Graph-theoretic Real-Time Visible Surface Editing Technique." *Computer Graphics* (Proc. SIGGRAPH 77), 223-228 (1977).

[Zhang, Koff 97] H. Zhang and K. Hoff III. "Fast Backface Culling Using Normal Masks." In *Proc. Symposium on Interactive 3D Graphics*, 103-106 (1997).

Web Information:

http://www.acm.org/jgt/papers/JohannsenCarter98
http://www.eai.com

Andreas Johannsen, Engineering Animation, Inc., 2321 North Loop Drive, Ames, IA 50010-8618 (andreas@eai.com)

Michael B. Carter, Engineering Animation, Inc., 2323 North Loop Drive, Ames, IA 50010-8618 (carter@eai.com)

Received August 19, 1998; accepted in revised form November 17, 1998.