1993

Back-Face Culling Applied to Collision Detection of Polyhedra

George Vaněček

Report Number:
93-020

BACK-FACE CULLING APPLIED TO COLLISION DETECTION OF POLYHEDRA

George Vanecek, Jr.

CSD-TR-93-020
March 1993

Back-face Culling applied to Collision Detection of Polyhedra

George Vaněček Jr.

Department of Computer Science
Purdue University
West Lafayette, IN 47907, USA
(317) 497-7088, **fax:** (317) 494-0739

Back-face culling is a preprocessing technique used in computer graphics to speed up the rendering of polyhedra. In this paper we show how this technique can be modified to reduce unnecessary checking of boundary elements in collision detection for a physical-based simulations and animation systems. At each time step, we determine a priori which faces cannot be part of the contact between two polyhedra and thus can be culled. In the computer graphics technique, the normal vector of a polygon is compared with the view direction. Here, the normal is compared to one or possibly several relative velocity vectors, and the face is culled when its motion is in the opposite direction of the normal vector.

We also give an algorithm that takes linear time in terms of the number of faces, and on the average eliminates half of the polygons. Due to its low computational overhead, when it is used as a front end to a collision detection system, a noticeable improvement in performance can be achieved.

Keywords: culling, collision detection, animation, simulation.

1 Introduction

We show in this paper that when two moving objects can potentially collide, at any instance of time roughly half of their total surface area need never be checked for collision. We apply a well known computer-graphics technique called back-face culling [5] in which the technique checks the orientation of polygons in relation to a given view and culls polygons which can not be seen. A polygon is culled when its normal vector points in the same general direction as the view vector. On the average this technique eliminates half the number of polygons from being rendered. Presently, the technique is supported by computer graphics hardware in conjunction with Z-buffering to render the visible polygons of opaque polyhedra.

There is a direct relationship between this problem and the problem of detecting possible contact between two moving objects. At any instance of time, some polygons of a moving object are facing in the general direction of motion and some are facing backwards. When considering pairs of objects, the polygons of one object that face backwards cannot collide with the other object and these polygons need not be checked for contact.

In physical-based simulations, we do not deal simply with the bounding polygons but with full Brep descriptions consisting of faces, edges and vertices. This technique can be used to preprocess the faces, edges and vertices at each time step prior any method used for collision checking. For example, Bouma and Vaněček have shown [2] that a full set-theoretic intersection is required to analyze contact in a physical-based simulation based on dynamics. The culling technique can speed up the performance of the intersection operation by an average factor of two since roughly half of each object is always facing in the opposite direction of motion. When complex objects consisting of thousands of faces are simulated, this factor of two can provide a noticeable improvement in performance of the collision detection algorithm.

To appreciate its merits in reducing the number of faces considered in collision checking, we can compare it with the commonly used bounding-volume techniques. In general, a complex object can be approximated by a bounding volume (such as a sphere, or its convex hull) leading to the following observation: if a bounding volume is not penetrated by the other object (or its bounding volume) then neither is the object within. When objects are far apart, bounding volumes can quickly determine the separation of objects [5].

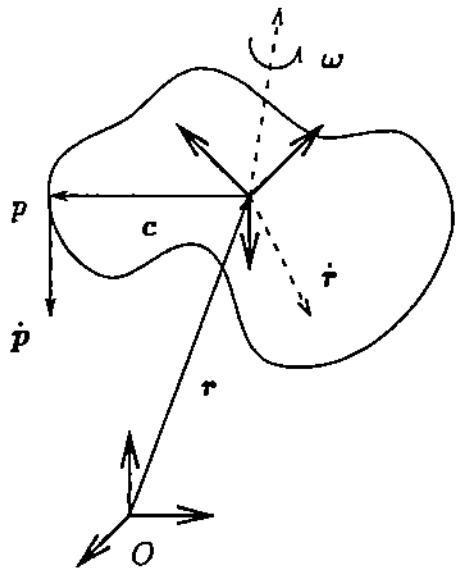

Figure 1: *Instantaneous velocity $\dot{p}$ at point p.*

However, when the objects are in close proximity, most (if not all) of the bounding-volume techniques fail to detect the separation of two nonpenetrating objects. Even hierarchical bounding-volume techniques such as [12] fail to reduce the complexity in close proximity of objects since the bounding volumes describe the entire objects and not the elements composing the objects (such as the faces). Unlike the bounding volumes, the presented culling technique does not depend on the geometry or pose of both objects being tested. It only depends on the relative-velocity vectors and the surface normals. Thus the same results are obtained whether the objects are far apart or in total contact.

We continue this paper by working out the details of relative velocity in Section 2 and show that it possesses a property that the relative-velocity vectors can be linearly interpolated between any two points. This property is then applied in Section 3 to formulate the technique of culling faces for two moving objects. The culling algorithm and its complexity is given next. In the last section, we summarize the results and conjecture some future results.

2 Preliminaries

Consider a polyhedron at some time t positioned and oriented in the global frame of reference with its local center indicated by the vector $\boldsymbol{r}$. The *velocity* of the center is given

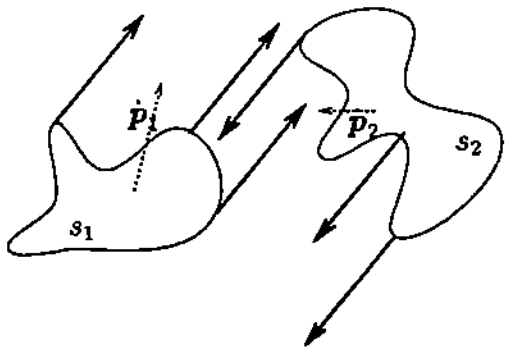

Figure 2: Relative Velocities (dark vectors) of two objects. Here $\omega_1 = \omega_2 = \emptyset$.

as the time derivative $\dot{r}$ and the *angular velocity* about the center is given as ω. Using r, $\dot{r}$, and ω, any point $p = r + c$ has an *instantaneous velocity* (as shown in Figure 1) given by the equation

$$\dot{p} = \dot{r} + \omega \times c. \tag{1}$$

From this equation we can compute the relative velocity of the point in relation to another object and show that the relative-velocity vector-space is linear. This property leads directly to the culling technique.

2.1 Relative Velocities

Given two polyhedra s_1 and s_2 in the global frame of reference at time t, the *relative velocity* at point p of object s_i as seen by an observer fixed on s_j is

$$\dot{p}_{ij} = \dot{p}_i - \dot{p}_j, \tag{2}$$

where the instantaneous velocities $\dot{p}_i$ and $\dot{p}_j$ are defined by Eq. (1). As a simple example, look at the relative velocities, shown as dark vectors, for the two objects in Figure 2. The objects have linear velocities (shown as dotted vectors) but no angular velocities.

Expanding Equation (2), we get the relative velocity in terms of the vectors c_i and c_j as

$$\begin{aligned} \dot{p}_{ij} &= \dot{r}_i + \omega_i \times c_i - (\dot{r}_j + \omega_j \times c_j) \\ &= \dot{r}_{ij} + \omega_i \times c_i - \omega_j \times c_j, \end{aligned} \tag{3}$$

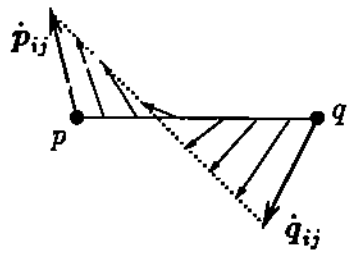

Figure 3: *Linear interpolation of relative velocity vectors.*

with $\dot{r}_{ij} = \dot{r}_i - \dot{r}_j$. To get the equation in terms of point p, we note that $c_i = p - r_i$ and $c_j = p - r_j$, and obtain .

$$\begin{aligned}
\dot{p}_{ij} &= \dot{r}_{ij} + \omega_i \times (p - r_i) - \omega_j \times (p - r_j) \\
&= \dot{r}_{ij} + \omega_i \times p - \omega_i \times r_i - \omega_j \times p + \omega_j \times r_j \\
&= (\dot{r}_{ij} - \omega_i \times r_i + \omega_j \times r_j) - p \times \omega_i + p \times \omega_j \\
&= a_{ij} + p \times (\omega_j - \omega_i) \\
&= a_{ij} + p \times \omega_{ji} \qquad (4)
\end{aligned}$$

where $a_{ij} = \dot{r}_{ij} - \omega_i \times r_i + \omega_j \times r_j$, and $\omega_{ji} = \omega_j - \omega_i$ are constants for a give time t.

2.2 Linear Property of Relative Velocities

When objects are rotating, points need not have the same relative velocity. This raises the following question: given any two distinct points p and q for which we know the relative velocities, can we state anything about the relative velocities of points on the line segment between p and q? We can show that the relative velocity of any point $tp + (1-t)q$, for $t \in \mathbf{R}$, is a linear combination of the relative velocity vectors $\dot{p}_{ij}$ and $\dot{q}_{ij}$ (see Figure 3). For readability, we now let $v_{ij}(p) \equiv \dot{p}_{ij}$. To show this property, we specifically want to demonstrate that

$$v_{ij}(tp + (1-t)q) = tv_{ij}(p) + (1-t)v_{ij}(q), \qquad (5)$$

for $0 \leq t \leq 1$. If this equality is correct, the two sides should cancel, and so the proof follows from Eq. (4):

$$\begin{aligned}
&v_{ij}(tp + (1-t)q) - tv_{ij}(p) - (1-t)v_{ij}(q) \\
&= (a_{ij} + (tp + (1-t)q) \times \omega_{ji}) - t(a_{ij} + p \times \omega_{ji}) - (1-t)(a_{ij} + q \times \omega_{ji})
\end{aligned}$$

$$
\begin{aligned}
&= a_{ij} + tp \times \omega_{ji} + q \times \omega_{ji} - tq \times \omega_{ji} - ta_{ij} - tp \times \omega_{ji} - (1-t)(a_{ij} + q \times \omega_{ji}) \\
&= (1-t)a_{ij} + (1-t)q \times \omega_{ji} - (1-t)(a_{ij} + q \times \omega_{ji}) \\
&= \emptyset
\end{aligned}
$$

This linear combination property for relative velocities generalizes to n points. Given n distinct points $p_1, \ldots, p_n$ in $\mathbf{E}^3$, let p be any convex combination of these points given by $\alpha_1 p_1 + \ldots + \alpha_n p_n$, where $1 = \sum_{i=1}^{n} \alpha_i$. It follows from Eq. (5) that

$$
v_{ij}(p) = \sum_{k=1}^{n} \alpha_k v_{ij}(p_k). \tag{6}
$$

To show this property visually, refer to the two torii in Figure 4. The torii are shown as wire frames and the relative velocities at each vertex are shown as line segments.

3 The Culling Technique

When we think of an object (say, s_i) moving in the presence of another object (say s_j), we think of s_i as having a front and a rear in terms of its relative velocity. We do not expect a collision to occur in the rear of s_i. To illustrate this, refer again to the torii as seen in Figure 5. The faces that face in the direction of motion are shaded, and the faces that face towards the rear are left as wire frames.

We can state this more formally. Given a face f of s_i in the global frame of reference, let $p \in f$. The angle θ between the normal vector n_f of face f and $\dot{p}_{ij}$ describes whether p is moving towards the outside directly above f or not. If follows that if θ is less than $\pi/2$,

$$
\dot{p}_{ij} \cdot n_f \geq 0,
$$

indicating that in the local neighborhood of p, p is moving towards the empty space above f and therefore p can possibly collide with some part of s_j within this local neighborhood (refer to Figure 6). Accordingly,

$$
\forall p \in f \; (\dot{p}_{ij} \cdot n_f < 0) \tag{7}
$$

implies that the entire face is moving away from any portion of s_j that may lie directly above it. Therefore f cannot collide and can be culled. Furthermore, note that $\dot{p}_{ij} = -\dot{p}_{ji}$. This means that if p is on face f_i of s_i and on f_j of s_j, and $\dot{p}_{ij} \cdot n_{f_i} < 0$ it must be the

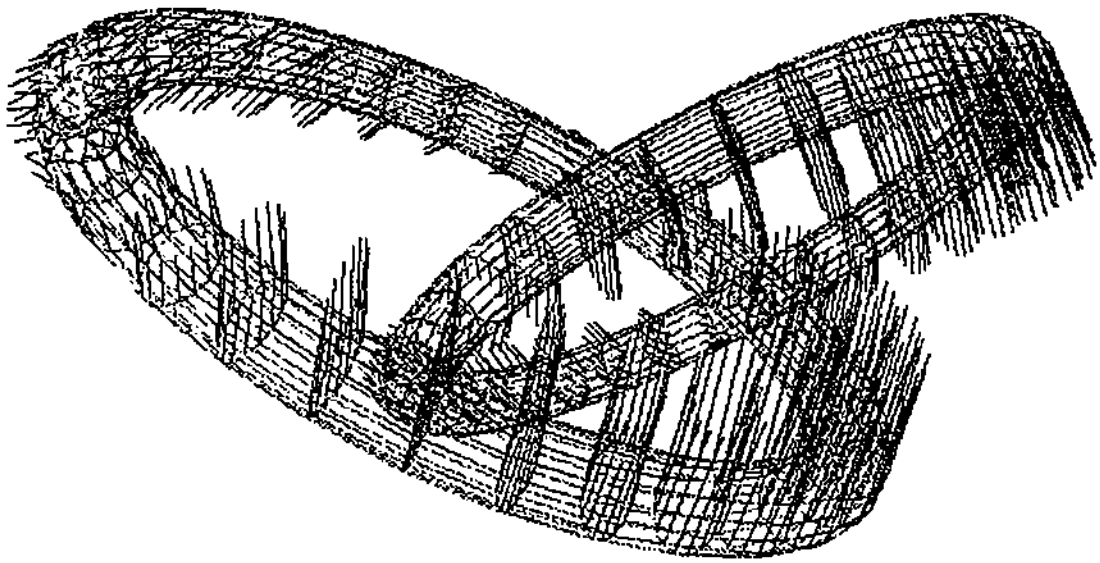

Figure 4: *Two torii with lines indicating the relative velocities.*

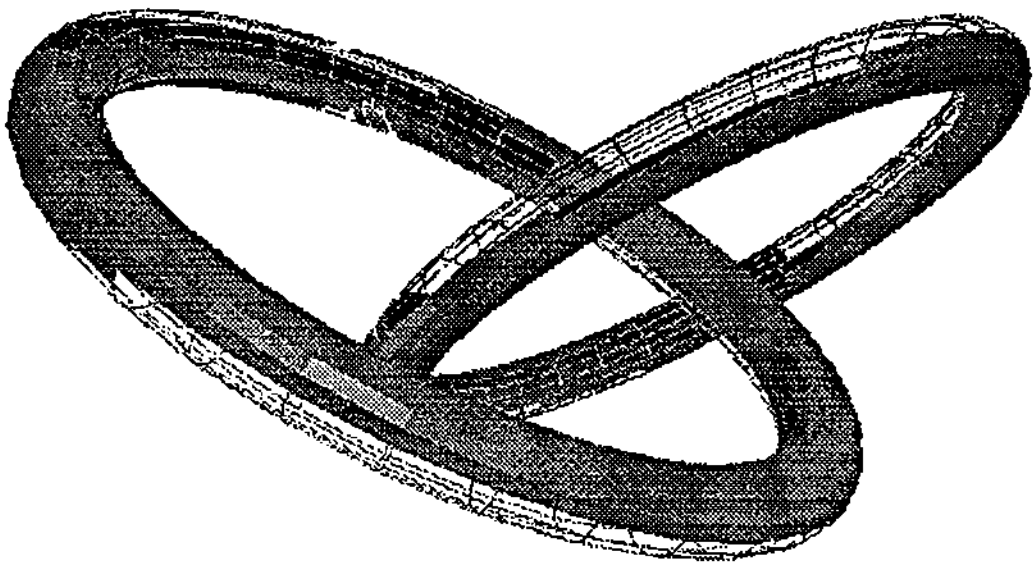

Figure 5: *Two torii with faces facing in the direction of motion shown filled.*

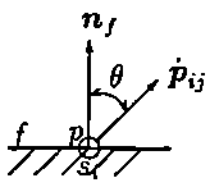

Figure 6: Point p, fixed on s_i, is moving away from Face f when $\theta < \pi/2$.

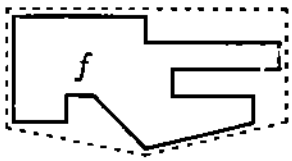

Figure 7: Bounding polygon $\Pi(f)$ approximating complex face f.

case that $\dot{\boldsymbol{p}}_{ji} \cdot \boldsymbol{n}_{f_j} < 0$, since $\boldsymbol{n}_{f_i} = -\boldsymbol{n}_{f_j}$. Thus both f_i and f_j are culled. This indicates that although contact may exist at time t, it cannot exists at time $t' > t$.

The above predicate cannot be implemented efficiently since it requires a check of the relative velocity direction at every point on the face. However, since the relative velocity vector-space is linear, we can make the following simplification. Let $\Pi(f)$ be a convex polygon of n points enclosing face f. Using the points of the polygon, $p_1, \ldots, p_n$ and applying Eq. (6), we can make the following reduction:

$$\forall k = 1, \ldots, n \ (\boldsymbol{v}_{ij}(p_k) \cdot \boldsymbol{n}_f < 0) \ \Rightarrow \ \forall p \in f \ (\boldsymbol{v}_{ij}(p) \cdot \boldsymbol{n}_f < 0).$$

Thus we can guarantee Property (7) by checking only a small finite set of points. Note that this is only a sufficient condition. Clearly the other direction does not always hold. This means that sometimes we may fail this simple test even though the face satisfies the property. How accurate the test is depends on how tightly the convex polygon approximates the face. If we let $\Pi(f)$ be the exact convex hull of f, we get both the necessary and sufficient condition. However, in terms of efficiency, this is not always appropriate. Consider, as an example a circular face with 256 vertices. Checking all 256 points is too inefficient. We can use instead a tightly approximating convex polygon consisting of only, lets say, eight points(refer to Figure 7). This idea was introduced for objects in [8] in trying to improve the efficiency of ray tracing.

There are two special cases of this property worth mentioning. When both objects are moving without rotating, Equation (7) becomes $\dot{\boldsymbol{r}}_{ij} \cdot \boldsymbol{n}_f < 0$, and when object s_j is stationary, this is $\dot{\boldsymbol{r}}_i \cdot \boldsymbol{n}_f < 0$. In both cases, only a single vector comparison is needed for each face.

4 Implementation

This technique has been implemented in C++ within Proxima [9], a system based on the Brep-index data structure [10]. The Brep-index supports contact analysis [2] for the Newton dynamics simulation system [4, 6, 7]. The culling technique serves as its front-end.

In the system, the objects are given in their local frames of reference and mapped to the global frame in each time step by the transformation

$$p = p^{\mathrm{L}}\mathrm{R}_i + \boldsymbol{r}_i.$$

Here, point p^{L} is the corresponding point of p in the local frame of reference, and the mapping uses a 3-by-3 rotational matrix, R_i, and the translation vector $\boldsymbol{r}_i$. Since the boundary points and face normals are given in the local frames, the culling algorithm computes the relative velocities in the local frames of reference of object s_i instead of the global frames as outlined in the previous sections. To do this, we rewrite Equation (4) which is a function of the global p as a function of p^{L} in the local frame. Thus,

$$\dot{\boldsymbol{p}}^{\mathrm{L}}_{ij} = \boldsymbol{a}^{\mathrm{L}}_{ij} + p^{\mathrm{L}} \times \boldsymbol{\omega}^{\mathrm{L}}_{ji}, \tag{8}$$

where $\boldsymbol{a}^{\mathrm{L}}_{ij} = \boldsymbol{a}_{ij}\mathrm{R}_i^{-1}$, and $\boldsymbol{\omega}^{\mathrm{L}}_{ji} = \boldsymbol{\omega}_{ji}\mathrm{R}_i^{-1}$.

The algorithm to cull faces and return the set of unculled faces can now be stated. It takes as arguments the set of faces F_i of s_i in its local frame of reference, the rotation matrix R_i, and the motion parameters for both s_i and s_j. With each face f we associate the bounding convex polygon, $\Pi(f) = [p_1, \ldots]$.

function cull(F_i, R_i, $\dot{\boldsymbol{r}}_i$, $\boldsymbol{\omega}_i$, $\dot{\boldsymbol{r}}_j$, $\boldsymbol{\omega}_j$) : **set**
begin
  $S \leftarrow \{\}$
  $\boldsymbol{a}^L_{ij} \leftarrow (\dot{\boldsymbol{r}}_{ij} - \boldsymbol{\omega}_i \times \boldsymbol{r}_i + \boldsymbol{\omega}_j \times \boldsymbol{r}_j)\mathrm{R}_i^{-1}$
  $\boldsymbol{\omega}^L_{ji} \leftarrow (\boldsymbol{\omega}_j - \boldsymbol{\omega}_i)\mathrm{R}_i^{-1}$
  for each face $f \in F_i$ **do**
    if $\exists k = 1, \ldots, |\Pi(f)| \ \ni \ (\boldsymbol{a}^L_{ij} + p_k \times \boldsymbol{\omega}^L_{ji}) \cdot \boldsymbol{n}_f \geq -\epsilon$ **then**
      $S \leftarrow S \cup \{f\}$
  return S *Return Unculled Faces*
end

In the algorithm, we use a small tolerance $\epsilon > 0$ to test for zero.

Without the bounding polygons for each face, the time complexity of Function cull is $O(2e + f)$ where e is the number of edges and f is the number of faces. Each face is visited once and culled if its relative-velocity vectors at the vertices all point in the opposite direction of the face normal. Since for a given face the number of edges is the same as the number of vertices, each edge in the object is visited twice, once for each adjacent face. Since, however, we only check the bounding polygon points rather than all the vertices of the face, we can bound the polygon to have at most eight points, and obtain a time complexity of $O(f)$.

5 Summary

This paper presented a linear time algorithm to eliminate on the average half of all the faces of two moving objects. The technique can supplement any collision detection algorithm that checks the pairwise intersection of faces of the two objects.

This technique could be easily supported in hardware on top of existing graphics-hardware pipelines and with a z-buffer feedback could possibly provide a hardware based collision detection. We are currently investigating this possibility.

We are also investigating the combined use of the culling technique and bounding-volumes. For example, adding bounding spheres to the objects and checking the support planes of faces against the spheres results in higher than 50 percent culling. It is hoped that simple a priori checks will cull most faces leaving a small subset of candidate faces which would need to be checked for interpenetration by other means.

Acknowlegements

The author wishes to thank Bill Bouma for his input on dynamics and for reviewing this paper.

References

[1] W. Bouma and G. Vaněček, Jr. "Collision Detection and Analysis in a Physical Based Simulation," *Proceedings of the Eurographics Workshop on Animation and Simulation*, 191–203, September 1991.

[2] W. Bouma and G. Vaněček, Jr. "Contact Analysis in a Physical Based Simulation," *ACM Symposium on Solid Modeling and Applications*, Montreal Canada, May, 1993.

[3] J. F. Cremer, "An Architecture for General Purpose Physical System Simulation–Integrating Geometry, Dynamics, and Control." Ph.D. Thesis, TR 89–987, Department of Computer Science, Cornell University, April 1989.

[4] J. F. Cremer and A. J. Stewart. "The Architecture of *Newton*, a General-Purpose Dynamics Simulator", *IEEE International Conference on Robotics and Automation*, pages 1806–1811, 1989.

[5] J. D. Foley, A. van Dam, S. K. Feiner and J. F. Hughes. *Computer Graphics, Principles and Practice*, Second Edition, Addison-Wesley, 1990.

[6] C. M. Hoffmann and J. E. Hopcroft. "Simulation of Physical Systems from Geometric Models", *IEEE Journal of Robotics and Automation*, RA–3(3):194–206, June 1987.

[7] C. M. Hoffmann and J. E. Hopcroft. "Model Generation and Modification for Dynamic Systems from Geometric Data," *CAD Based Programming for Sensory Robots*, F50:481–492, 1988.

[8] D. S. Kay and J. T. Kajiya, "Ray Tracing Complex Scenes," *Computer Graphics*, 20(4):269–278, (Proc. SIGGRAPH '86).

[9] G. Sun and P. Van Vleet and G. Vaněček, Jr. "Proxima, a Polyhedral Distance and Classification Support in C++", Department of Computer Science, Purdue University, Technical Report, CER–92–41, November 1992.

[10] G. Vaněček, Jr., "Brep-Index: A Multi-dimensional Space Partitioning Tree," *First ACM/SIGGRAPH Symposium on Solid Modeling Foundations and CAD/CAD Applications*, Austin Texas, 35–44, June 1991.

[11] G. Vaněček, Jr., "A Data Structure for Analyzing Collisions of Moving Objects," *IEEE Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences*, Kailua-Kona Hawaii, Vol I:671–680, January 1991.

[12] H. Weghorst, G. Hooper, and D. P. Greenberg. "Improved Computational Methods for Ray Tracing," *ACM Trans. Graphics*, 3(1):52–69, 1984.