V.1

FAST RAY–CONVEX POLYHEDRON INTERSECTION

Eric Haines
3D /Eye, Inc.
Ithaca, New York

The standard solution to ray-polyhedron intersection is to test the ray against each polygon and find the closest intersection, if any. If the polyhedron is convex, the ray-polyhedron test can be accelerated by considering the *polyhedron* to be the space inside a set of planes. This definition also drastically reduces the memory requirements for such polyhedra, as none of the vertices and their connectivities need to be stored; only the plane equations for the faces are needed. Finally, the ray-polyhedron test outlined here avoids the problems that can occur when the shared edge of two polygons is intersected by a ray, since there no longer are any edges. There is no chance of a ray "slipping through the cracks" by having its intersection point on an edge not considered being inside either polygon.

The algorithm is based on the ideas of Roth (1981) and Kay and Kajiya (1986). The basic idea is that each plane of the polyhedron defines a half-space: All points to one side of this space are considered inside the plane (also considering points on the plane as inside). The logical intersection of the half-spaces of all the convex polyhedron's planes is the volume defined by the polyhedron. Introducing a ray into this definition changes the problem from three dimensions to one. The intersection of each plane by the ray creates a line segment (unbounded at one end) made of a set of points inside the plane's half-space. By taking the logical intersection of all ray-plane line segments, we find the line segment (if any) in which the ray passes through the polyhedron.

The ray is defined by:

$$R_{\text{origin}} = R_o = [x_o\ y_o\ z_o],$$

$$\mathbf{R}_{\mathbf{direction}} = \mathbf{R_d}\ [x_d\ y_d\ z_d],$$

where $x_d^2 + y_d^2 + z_d^2 = 1$ (i.e., normalized).

The set of points on the ray is represented by the function:

$$R(t) = R_0 + \mathbf{R}_\mathbf{d}^* t,$$

where $t > 0$.

In addition, in ray tracing, it is useful to keep track of $t_{\max}$, the maximum valid distance along the ray. In shadow testing, $t_{\max}$ is set to the distance of the light from the ray's origin. For rays in which the closest object hit is desired, $t_{\max}$ is initialized to *infinity (i*.e., some arbitrarily large distance), then updated to the distance of the object currently the closest as testing progresses. An object intersected beyond $t_{\max}$ does not need to be tested further—for example, for shadow testing, such an object is beyond the light.

We initialize the distances t_{near} to *negative infinity* and t_{far} to $t_{\max}$ These will be used to keep track of the logical intersection of the half-spaces with the ray. If t_{near} ever becomes greater than t_{far}, the ray misses the polyhedron and testing is done.

Each plane is defined in terms of [*a, b, c, d*], which defines the plane

$$a*x + b*y + c*z + d = 0.$$

$\mathbf{P_n}$ is the plane's normal,[*a, b, c*].

The distance from the ray's origin to the intersection with the plane *P* is simply:

$$t = -v_n/v_d,$$

where:

$$v_n = \mathbf{P_n} \cdot R_o + d,$$

$$v_d = \mathbf{P_n} \cdot \mathbf{R_d}.$$

If v_d is 0, then the ray is parallel to the plane and no intersection takes place. In such a case, we check if the ray origin is inside the plane's half-space. If v_n is positive, then the ray origin is outside the plane's half-space. In this case, the ray must miss the polyhedron, so testing is done.

Otherwise, the plane is categorized as front-facing or back-facing. If v_d is positive, the plane faces away from the ray, so this plane is a back-face of the polyhedron; else, the plane is a front-face. If a back-face, the plane can affect t_{far}. If the computed t is less than 0, then the polyhedron is

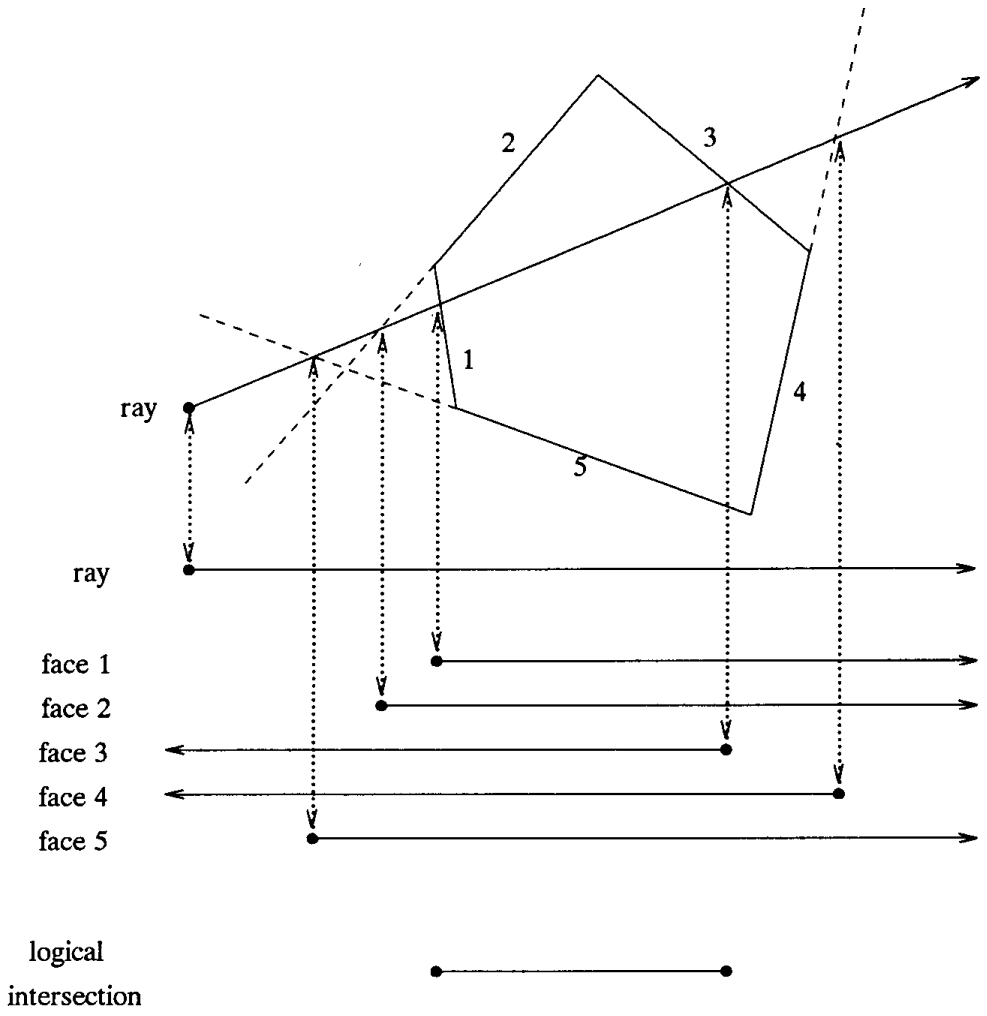

Figure 1.

behind the ray and so is missed. If t is less than t_{far}, t_{far} is updated to t. Similarly, if the plane is a front-face, t_{near} is set to t if t is greater than t_{near}. If t_{near} ever is greater than t_{far}, the ray must miss the polyhedron. Else, the ray hits, with t_{near} being the entry distance (possibly negative) and t_{far} being the distance where the ray exits the convex polyhedron. If t_{near} is negative, the ray originates inside the polyhedron; in this case, check if t_{far} is less than t_{max}; if it is, then t_{far} is the first valid intersection.

An example of this process is shown in Fig. 1. A two-dimensional view of a ray and a set of five planes forming a polyhedron is shown. Below this is the set of segments formed by the ray and the five faces. The ray intersects the plane formed by face 1 and so defines a set of points along the ray inside the half-space to the right of this intersection. The ray intersects the plane of face 2 (shown as a dashed line) at a closer distance, even though the face itself is missed. Face 3 is a back-face and creates a half-space open to the left. Face 4 also is missed, but its plane still defines a half-space. Finally, plane 5 forms a front-face half-space.

The segment in which the line defined by the ray passes through the polyhedron is the logical intersection of the five face segments; that is, where all of these segments overlap. Comparing this segment with the ray's own segment gives which (if either) of the intersection points is the closest along the ray. If there is no logical intersection, the ray misses.