◊ I.7

Detecting Intersection of a Rectangular Solid and a Convex Polyhedron

Ned Greene
Apple Computer and University of California at Santa Cruz
One Infinite Loop, MS 301-3J
Cupertino, CA 95014
greene@apple.com

This Gem presents a fast method for determining whether a rectangular solid intersects a convex polyhedron. Our algorithm is based on the observation that a rectangular solid R intersects a convex polyhedron P if and only if (a) the projections of R and P intersect in all three axis-aligned orthographic views (i.e., front, side, and top views), and (b) R does not lie entirely "outside" the plane of any face of P. After finding the equations of certain lines and planes associated with P, we can determine whether these criteria are satisfied by simply evaluating inequalities. The method is well suited to culling a geometric model organized in an octree to a viewing frustum. For this application, determining whether a bounding box intersects a viewing frustum requires evaluating between one and thirty inequalities. The method can also be applied to detecting the intersection of a rectangular solid and a 3D planar polygon.

◊ Introduction ◊

Beginning with some definitions, we will refer to a convex polyhedron simply as a polyhedron and to an axis-aligned rectangular solid simply as a box. The term "bounding box" will specifically refer to a "tight," axis-aligned rectangle in 2D or rectangular solid in 3D. Axis-aligned orthographic views, the familiar front, side, and top views of engineering drawings, will be referred to simply as orthographic views.

Various computer graphics techniques associate bounding boxes in the shape of rectangular solids with clusters of geometric primitives to enhance efficiency, the idea being that performing a single operation on a bounding box often eliminates the need to process primitives inside the box individually. When culling a geometric model to a viewing frustum, for example, if a bounding box lies outside the frustum, the primitives it contains also lie outside and need not be considered individually. This observation underlies simple, fast procedures for culling models represented in spatial hierarchies.

IBM ISBN 0-12-336155-9
Macintosh ISBN 0-12-336156-7

For example, if geometric primitives are organized in an octree of bounding boxes, parts of the model which lie entirely outside a viewing frustum can be efficiently culled by testing for box-frustum intersection during recursive subdivision of the octree (Garlick et al. 1990).

Since box-polyhedron intersection tests may be performed hundreds or thousands of times in the course of producing a single image of a scene, devising an efficient intersection algorithm is worthwhile. One intuitive approach to the problem combines a test for intersecting bounding boxes, tests to see if a vertex of one solid lies inside the other, and tests to see if an edge of one solid intersects a face of the other. While this method is very straightforward, the expense of finding geometric intersections makes it quite costly. Alternatively, we could employ algorithms for detecting intersection of convex polyhedra that are described in the computational geometry literature, for example, the method of Chazelle and Dobkin (Chazelle and Dobkin 1987). However, such methods are typically complicated, designed with asymptotic performance in mind, and poorly suited to simple problems like deciding whether a box intersects a viewing frustum.

Our approach is both simple and efficient. It does not require finding geometric intersections, but rather, after finding the equations of certain lines and planes associated with a polyhedron, box-polyhedron intersection can be determined by simply comparing bounding boxes and evaluating inequalities. The algorithm performs most efficiently when comparing numerous bounding boxes to the same polyhedron, but even when performing a single intersection test, efficiency compares favorably to other methods.

We precede our analysis of box-polyhedron intersection with a discussion of primitive operations that our algorithm performs — determining whether a box intersects a plane, whether a rectangle intersects a line, and whether a rectangle intersects a polygon.

◇ Box-Plane and Rectangle-Line Intersection ◇

One of the fundamental operations performed by our box-polyhedron intersection algorithm is determining whether a box intersects a plane, and if not, which side of the plane the box lies on. The problem is illustrated in Figure 1, an orthographic projection in which the viewpoint has been chosen so that plane P is perpendicular to the screen. Vector N is normal to P, pointing into P's positive half-space.[1] Box B1 lies in P's negative half-space, box B2 lies in P's positive half-space, and box B3 intersects P. These are the three cases that we wish to distinguish in general.

The first step in distinguishing these cases is to establish which of a box's vertices lies farthest in the "positive direction" (the direction of normal N), call this the *p-vertex*,

[1]Recall that a plane defined by equation Ax+By+Cz+D=0 has normal (A,B,C) and divides space into a *positive half-space* satisfying the equation Ax+By+Cz+D>0 and a *negative half-space* satisfying the equation Ax+By+Cz+D<0.

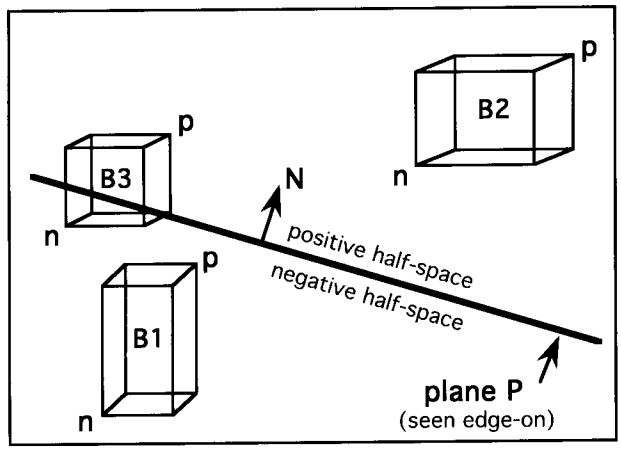

Figure 1.

and which of the box's vertices lies farthest in the "negative direction," call this the *n-vertex*. P- and n-vertices are easily identified, since the p-vertex corresponds to the octant of the plane's normal, and the n-vertex lies at the opposite corner. When an edge or face of the box is parallel to the plane, the octant rule does not specify a unique vertex, but in these cases it doesn't matter which of the obvious candidates is selected. In Figure 1 p-vertices are labeled **p** and n-vertices are labeled **n**. Note that the p- and n-vertices associated with a particular plane have the same relative positions on all axis-aligned bounding boxes, so they need to be identified only once.

Once p- and n-vertices have been identified, distinguishing the three cases of box-plane intersection is very simple. Box B lies entirely in plane P's negative half-space if and only if its p-vertex lies in P's negative half-space, B lies entirely in P's positive half-space if and only if its n-vertex lies in P's positive half-space, and if neither of these relationships holds, B intersects P (Haines and Wallace 1991). It follows that we can determine whether a box lies in a particular half-space by evaluating one plane equation, and that the three cases of box-plane intersection can be distinguished by evaluating one or two plane equations as follows:

```
Given an axis-aligned rectangular solid R with n-vertex (xn,yn,zn) and
p-vertex (xp,yp,zp), and plane P with equation Ax+By+Cz+D=0,

if (A*xp + B*yp + C*zp + D < 0) { R lies entirely in P's negative half-space }
else if (A*xn + B*yn + C*zn + D > 0) { R lies entirely in P's positive half-space }
else{ R intersects P }
```

The two-dimensional problem of determining whether an axis-aligned rectangle intersects a line, lies entirely in the line's negative half-plane, or lies entirely in the line's

positive half-plane is entirely analogous, requiring the evaluation of one or two line equations.

```
Given an axis-aligned rectangle R with n-vertex (xn,yn) and p-vertex (xp,yp),
and line L with equation Ax+By+C=0,

if (A*xp + B*yp + C < 0) { R lies entirely in L's negative half-plane }
else if (A*xn + B*yn + C > 0) { R lies entirely in L's positive half-plane }
else{ R intersects L }
```

◇ Rectangle-Polygon Intersection ◇

We now consider another subproblem, determining whether an axis-aligned rectangle R and a convex polygon P, both lying in the same plane, intersect. It can easily be shown that R intersects P if and only if (a) R intersects P's bounding box and (b) R does not lie entirely "outside"[2] any of the infinite lines defined by P's edges, which we will refer to as *edge-lines*. The problem is illustrated in Figure 2 where P's bounding box B is drawn in dashed lines and P's edge-lines are drawn in dotted lines (e.g., E). Applying the intersection criteria to rectangles R1, R2, and R3 of Figure 2, R1 is found not to intersect P because it does not intersect P's bounding box, R2 is found not to intersect P because it lies outside of edge-line E, and R3 is found to intersect P because it satisfies both intersection criteria.

The rectangle-line intersection method described in the preceding section is an efficient way to evaluate criterion (b). Using this method we can determine whether a rectangle lies outside an edge-line by substituting the coordinates of the rectangle's n-vertex into the edge-line's equation. For example, to show that R2 lies outside edge-line E we would substitute the coordinates of R2's lower left vertex (the n-vertex for E) into the line equation for E.

The rectangle-polygon intersection problem is germane because our algorithm looks for box-polyhedron intersection in orthographic views,[3] and in these views a box always projects to an axis-aligned rectangle and the silhouette of a convex polyhedron is always a convex polygon. These conditions apply, for example, in Figure 3 where the panels labeled "front," "side," and "top" are orthographic views of a box and a polyhedron in the shape of a viewing frustum. Once the polygonal silhouette of the polyhedron has been identified in an orthographic view,[4] determining whether box-polyhedron intersection occurs in that view reduces to the rectangle-polygon intersection problem described above. Incidentally, Figure 3 illustrates that intersection of a box and a polyhedron in

[2]By "outside" we mean on the side opposite polygon P.

[3]Remember that *axis-aligned* is implied in this term.

[4]Finding a convex polyhedron's silhouette edges is particularly straightforward in an orthographic view. In a view down an axis, call it the u-axis, if the u components of the outward-pointing normals of two adjacent faces have opposite signs, their common edge is a silhouette edge.

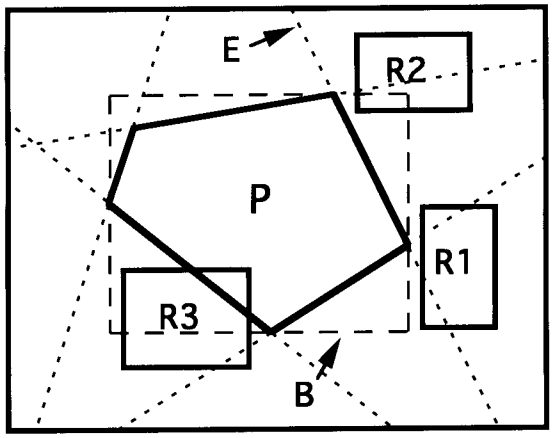

Figure 2. Testing for intersection between a polygon P and various rectangles R_i.

all three orthographic views does not guarantee intersection in 3D, as is apparent in the upper right panel.

It follows from the rules for rectangle-polygon intersection that the projections of a box B and a polyhedron P intersect in all three orthographic views if and only if (1) B intersects P's bounding box and (2) the projection of B does not lie outside any of the edge-lines of P's silhouette in any of the three orthographic views. Our box-polyhedron intersection algorithm uses this formulation to establish intersection in orthographic views, as elaborated in the following section.

◇ Box-Polyhedron Intersection ◇

Our box-polyhedron intersection algorithm is based on the observation that for any two convex polyhedra A and B which do not intersect, there exists a *separating plane* lying between them that is (1) parallel to a face of A, (2) parallel to a face of B, or (3) parallel to an edge of A and an edge of B.[5]

Applying this observation to a rectangular solid R and a convex polyhedron P, we will say that a separating plane which is parallel to a face of R is of *type 1*, a separating plane which is parallel to a face of P is of *type 2*, and a separating plane which is parallel to an edge of R and an edge of P is of *type 3*. If a separating plane of type 1, 2, or 3 exists, R and P do not intersect; otherwise they do.

[5]My thanks to an anonymous reviewer for pointing this out and suggesting the ensuing line of exposition. As a starting point for a proof, consider two non-interpenetrating convex polyhedra whose surfaces are in contact, e.g., a vertex touches a face, an edge touches an edge, an edge touches a face, etc. For any possible configuration, a plane of type 1, 2, or 3 separates the polyhedra except for the point, line segment, or polygon of contact.

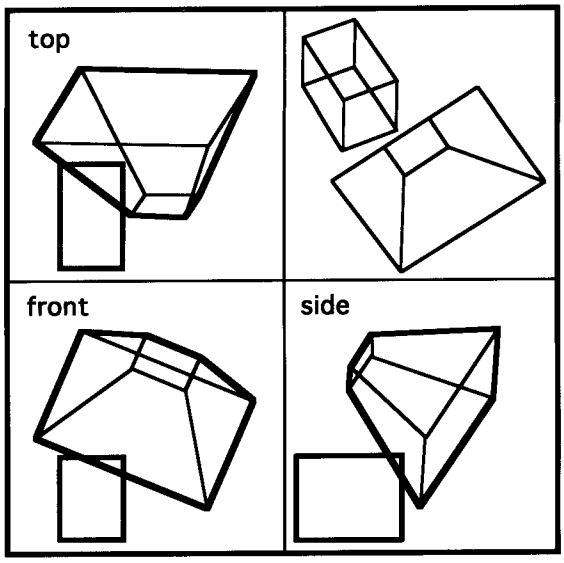

Figure 3. Four views of a rectangular solid and a polyhedron in the shape of a viewing frustum. The front, side, and top views are axis-aligned orthographic views.

We can easily see whether a type 1 plane exists by determining whether P's bounding box intersects R. If so, a type 1 plane does not exist; if not, a type 1 plane does exist, demonstrating that R and P do not intersect, and we are done.

We can see whether a type 2 plane exists by comparing each face-plane of P with the corresponding n-vertex of R using the method presented in the section on box-plane intersection. If any of these n-vertices is outside of its respective face-plane, there exists a type 2 separating plane demonstrating that R and P do not intersect and we are done.

The remaining problem is to determine whether a separating plane of type 3 exists. Since a type 3 plane is parallel to an edge of R, it projects to a "separating line" lying between the projections of R and P in one of the orthographic views. It follows that a type 3 plane exists if and only if the projections of R and P do not intersect in at least one of the three orthographic views. Combining this observation with intersection criterion (2) from the preceding section, the existence of a type 3 plane can be decided by determining whether the projection of R lies outside an edge-line of P in at least one orthographic view.[6] If so, a type 3 plane exists, establishing that R and P do not intersect; if not, a type 3 plane does not exist and we conclude that R and P intersect because no type 1, 2, or 3 separating plane exists.

[6]Note that we have already established that R intersects P's bounding box, satisfying intersection criterion (1) from the preceding section.

Incidentally, the conditions for box-polyhedron intersection can be stated very concisely as follows: Box R intersects polyhedron P if and only if (a) the projections of R and P intersect in all three orthographic views, and (b) R does not lie entirely outside the plane of any face of P. In this formulation the requirement that R intersects P's bounding box is subsumed by condition (a). The analogous intersection conditions for a box R and a 3D planar polygon P are as follows: R and P intersect if and only if the projections of R and P intersect in all three orthographic views and R intersects the plane of P.

◇ Summary of the Algorithm ◇

Let's reiterate the algorithm as it is applied in practice with efficient execution in mind. Assuming for the moment that polyhedron P will be compared to numerous boxes, we begin with the following preliminary steps: 1) for each face of P we find the plane equation and identify which box corner is the n-vertex; 2) for each silhouette edge of P in the three orthographic views we find the line equation and identify which box corner is the n-vertex;[7] and 3) we find P's bounding box. If P will be compared to only a small number of boxes, lazy evaluation of this information may be more efficient.

Now box-polyhedron intersection testing proceeds as follows. If box R does not intersect P's bounding box, we conclude that R does not intersect P and we are done. Next we consider P's face-planes one by one, seeing if R lies entirely outside any of these planes. If so, we conclude that R does not intersect P and we are done. Finally, we consider P's edge-lines[8] one by one, seeing if R's projection lies entirely outside any of these lines in the appropriate orthographic view. If so, we conclude that R does not intersect P and we are done. Otherwise, it has been established that intersection does occur.

To estimate the computational expense of the algorithm we examine the cost of the primitive operations. We can determine whether two 3D bounding boxes intersect by evaluating between one and six simple inequalities of the form "is a < b?" Determining whether a box lies entirely outside a face-plane requires evaluating one plane equation. Determining whether a box's projection lies entirely outside an edge-line requires evaluating one line equation. So when comparing a box to a polyhedron with F faces and E silhouette edges in orthographic views, each box-polyhedron intersection test requires evaluating between one and six simple inequalities, evaluating between zero and F plane equations, and evaluating between zero and E line equations. Summing up, between 1 and 6 + E + F inequalities must be evaluated to show that B and P do not intersect, and all 6 + E + F inequalities must be evaluated to show that B and

[7]Pick either of the two vertices which coincide in the orthographic view.

[8]I.e., the lines defined by P's silhouette edges in the three orthographic views.

P do intersect. For a viewing frustum, E is at most 18 and F is 6, so the maximum number of inequalities which must be evaluated to decide box-frustum intersection is 30. Our cost estimate should also amortize the cost of finding line and plane equations and identifying n-vertices over the number of intersection tests performed.

There are many variations on this basic algorithm. To avoid evaluating all 6 + E + F inequalities whenever intersection occurs, a test can be added to see if the polyhedron lies entirely inside the box or vice versa. When culling geometry to a viewing frustum it may be useful to know which of the frustum's face-planes a box intersects, because they correspond to clipping planes. These refinements are included in the procedure outlined in the following section which we use to cull an octree to a viewing frustum.

◇ Pseudocode ◇

Given a collection of axis-aligned rectangular solids and a convex polyhedron P, the following procedure classifies each rectangular solid R as entirely outside, entirely inside, or partially inside P, and in the latter case reports which face-planes of P are intersected by R. The procedure can be streamlined for applications which only need to detect intersection.

```
/* Routine for Detecting Box-Polyhedron Intersection */

/* Preliminary Step */
Determine P's bounding box.
Find the plane equation of each face of P (these are face-planes).
Determine which vertex of an axis-aligned rectangular solid is the "n-vertex"
    and which is the "p-vertex" for each face-plane of P.
Determine the silhouette edges of P's projection in the three orthographic
    views, and find the line equation of each of these edges (these are edge-lines).
Determine which vertex of an axis-aligned rectangular solid is the "n-vertex"
    for each edge-line of P.

For each rectangular solid R {
    /* Bounding Box Tests */
    if R does not intersect P's bounding box, R is entirely outside P, done with R
    if P's bounding box is entirely inside R, R is partially inside P and R
        intersects all face-planes, done with R

    /* Face-Plane Tests */
    for each face-plane F {
        if R is entirely outside F, R is entirely outside P, done with R
        if R is entirely inside F, set a flag indicating that R does not intersect F
        else R intersects F, set a flag indicating that R intersects F
    }
    if R was entirely inside all face-planes, R is entirely inside P, done with R
```

```
    /* Edge-Line Tests */
    for each edge-line E
        if the projection of R is entirely outside E (in the appropriate
            orthographic projection), R is entirely outside P, done with R

    R is partially inside P and it is known which face-planes R intersects
}
```

◇ Bibliography ◇

(Chazelle and Dobkin 1987) B. Chazelle and D. Dobkin. Intersection of convex objects in two and three dimensions. *Journal of the ACM*, 34(1):1–27, Jan. 1987.

(Garlick et al. 1990) B. Garlick, D. Baum, and J. Winget. Interactive viewing of large geometric data bases using multiprocessor graphics workstations. Siggraph '90, Course Notes, Vol. 28 (Parallel Algorithms and Architectures for 3D Image Generation), pp. 239–245.

(Haines and Wallace 1991) E. Haines and J. Wallace. Shaft culling for efficient ray-traced radiosity. SIGGRAPH '91, Course Notes, Vol. 12 (Frontiers in Rendering), pp. 2.1–2.28.