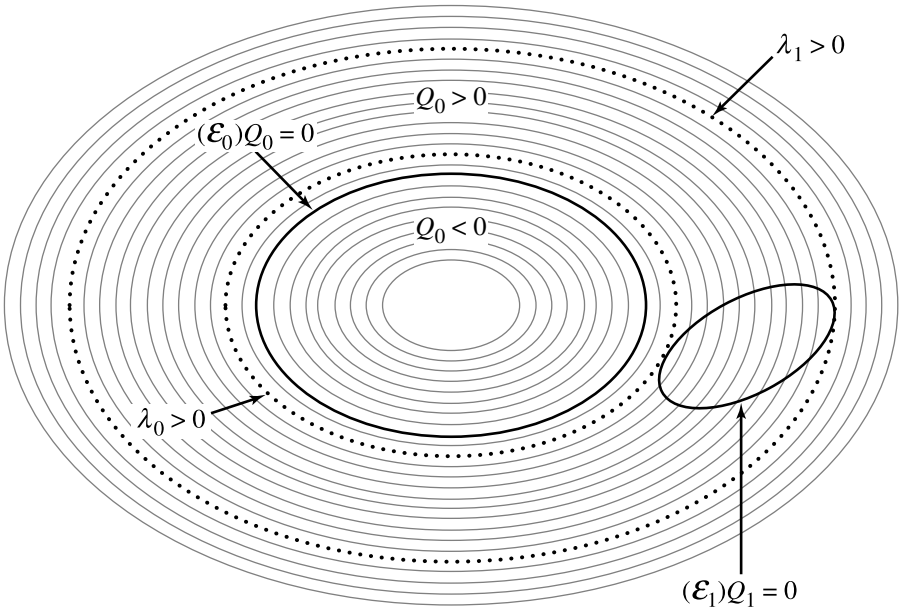

Figure 7.9 $\mathcal{E}_1$ is separated from $\mathcal{E}_0$. The minimum $\mathcal{E}_0$ level curve value λ_0 for $\mathcal{E}_1$ is positive.

$$\mathbf{A}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 6 \end{bmatrix}, \quad B_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad C_0 = -1, \quad \mathbf{A}_1 = \begin{bmatrix} 52 & -36 \\ -36 & 73 \end{bmatrix},$$
$$B_1 = \begin{bmatrix} -32 \\ -74 \end{bmatrix}, \quad C_1 = 28$$

From these are derived

$$\vec{Y}(t) = \begin{bmatrix} 4t(625t + 24) \\ t(2500t + 37) \end{bmatrix}, \quad \delta(t) = 2500t^2 + 385t + 6$$

The polynomial of Equation 7.9 is $p(t) = -156250000t^4 - 48125000t^3 + 1486875t^2 + 94500t + 1008$. The two real-valued roots are $t_0 \doteq -0.331386$ and $t_1 \doteq 0.0589504$. The corresponding $X(t)$ values are $X(t_0) = (x_0, y_0) \doteq (1.5869, 1.71472)$ and $X(t_1) = (x_1, y_1) \doteq (0.383779, 0.290742)$. The axis-aligned ellipse level values at these points are $Q_0(x_0, y_0) = -0.345528$ and $Q_0(x_1, y_1) = 19.1598$. Since $Q_0(x_0, y_0) < 0 < Q_0(x_1, y_1)$, the ellipses intersect. Figure 7.10 shows the two points on $Q_1 = 0$ that have extreme Q_0 values. ■

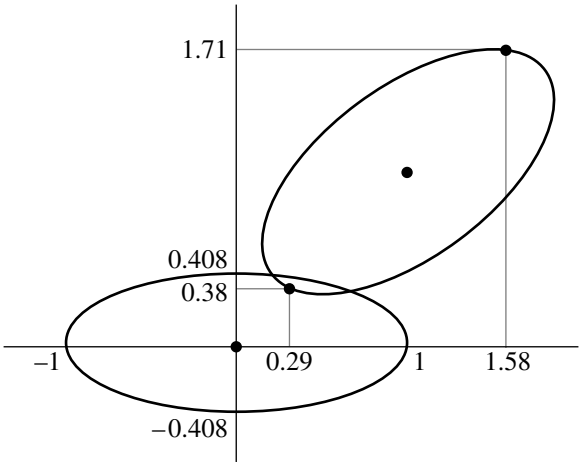

Figure 7.10 Intersection of two ellipses.

7.6 Polynomial Curves

Consider two polynomial curves, $X(s) = \sum_{i=0}^{n} \vec{A}_i s^i$, where $\vec{A}_n \neq \vec{0}$ and $s \in [s_{\min}, s_{\max}]$, and $Y(t) = \sum_{j=0}^{m} \vec{B}_j t^j$, where $\vec{B}_m \neq \vec{0}$ and $t \in [t_{\min}, t_{\max}]$. This section discusses how to compute points of intersection between the curves from both an algebraic and geometric perspective.

7.6.1 Algebraic Method

The straightforward algebraic method is to equate $X(s) = Y(t)$ and solve for the parameters s and t. Observe that the vector equation yields two polynomial equations of degree $\max\{n, m\}$ in the two unknowns s and t. The method of elimination may be used to obtain a single polynomial equation in one variable, $q(s) = 0$. The method of solution is a simple extension to what was shown in the section on intersection finding for lines and polynomial curves, except that the degree of $q(s)$ will be larger than in that case (for the line, $m = 1$; for curves, we generally have $m > 1$).

7.6.2 Polyline Approximation

The root finding of the algebraic method can be computationally expensive. Similar to Section 7.4 for line-curve intersection testing, the time complexity is reduced

by approximating both curves by polylines and finding intersections of the two polylines. The polylines are obtained by subdivision, described in Section A.8. Any intersections between the polylines can be used as approximations to curve-curve intersections if the application is willing to accept that the polylines are suitable approximations to the curves. However, the points of intersection might be used as an attempt to localize the search for actual points of intersection.

7.6.3 Hierarchical Bounding

In Section 7.4 we discussed using coarse-level testing using bounding polygons, bounding boxes, or hierarchies of bounding boxes to allow for an early out when the two underlying objects do not intersect. The same ideas apply to curve-curve intersection testing. If the curves are defined by control points and have the convex hull property, then an early-out algorithm would test first to see if the convex polygons containing the curves intersect. If not, then the curves do not intersect. If so, a finer-level test is used, perhaps directly the algebraic method described earlier.

A hierarchical approach using box trees can also be used. Each curve has a box hierarchy constructed for it. To localize the intersection testing, pairs of boxes, one from each tree, must be compared. This is effectively the 3D-oriented bounding box approach used by Gottschalk, Lin, and Manocha (1996), but in 2D and applied to curve segments instead of polylines. One issue is to perform an amortized analysis to determine at what point the box-box intersection tests become more expensive than the algebraic method for curve-curve intersection tests. At that point the simplicity of box-box intersection tests is outweighed by its excessive cost. A lot of the cost is strongly dependent on how deep the box hierarchies are. Another issue is construction of axis-aligned bounding boxes for curves. This was discussed in Section 7.4.

7.6.4 Rasterization

Finally, a raster approach may be used, even though it is potentially quite expensive to execute. An axis-aligned bounding box $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ is constructed to contain both curves. An $N \times M$ raster is built to represent the box region. The grid points are uniformly chosen as (x_i, y_j) for $0 \le i < N$ and $0 \le j < M$. That is, $x_i = x_{\min} + (x_{\max} - x_{\min})i/(N-1)$ and $y_j = y_{\min} + (y_{\max} - y_{\min})j/(M-1)$. Each curve is drawn into the raster. The step size for the parameter of the curve should be chosen to be small enough so that as the curve is sampled you generate adjacent raster values, potentially with a raster cell drawn multiple times because multiple curve samples fall inside that cell. The overdraw can be minimized by sampling the curve based on arc length rather than the curve parameter. If the raster is initialized with 0, the first curve drawn by or-ing the pixels with 1, and the second curve drawn by

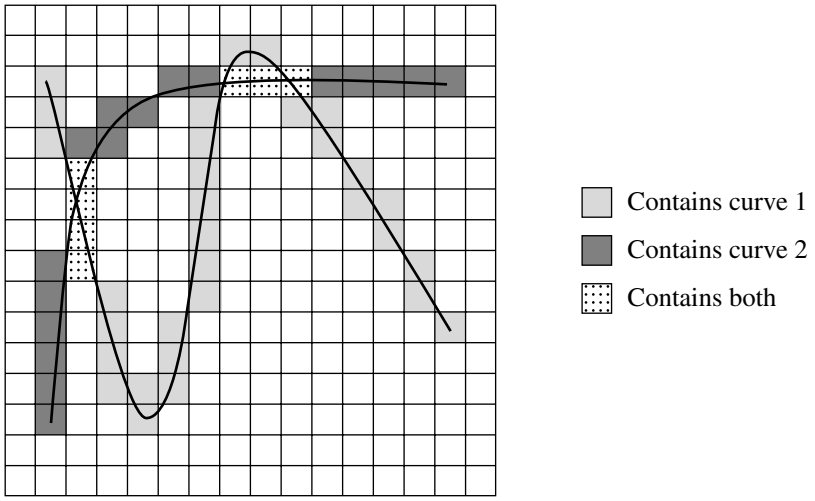

Figure 7.11 Two curves rasterized on a grid that is initially zero. The first curve is rasterized by or-ing the grid with the mask 1 (light gray). The second curve is rasterized by or-ing the grid with the mask 2 (dark gray). Grid cells that contain both curves have a value 3 (dotted).

or-ing the pixels with 2, the pixels that are associated with curve-curve intersections are those with a final value of 3 (see Figure 7.11).

Notice that the leftmost block of pixels (dotted cells) make it uncertain where the curves might intersect, if at all. The problem generally is that two curves can be near each other, yet not intersect, and be rasterized into the same pixels. The solution is to increase the number of cells while reducing the cell size to get a finer-resolution grid. How small a cell size should be to properly detect intersections and not produce false positives is usually information that is not known ahead of time.

Just as in the polyline approach, the application can choose to accept the pixel values as approximations to the actual curve-curve intersections. If more accuracy is desired, the pixels tagged as 3 (and possibly immediate neighbors) can be used as a localization of where the intersections occur. If a contiguous block of pixels is tagged, such as is shown in the left of the grid in Figure 7.11, and if the application chooses to believe the block occurs because of a single intersection of curves, a suitable choice for the approximation is the average of pixel locations. If the application chooses not to accept the pixel values as approximations, then it can store the original curve parameters for samples occurring in a pixel with that pixel. Those parameter values can be used to start a search for intersections using a numerical root finder or a numerical distance calculator.

7.7 The Method of Separating Axes

A set S is *convex* if given any two points P and Q in S, the line segment $(1-t)P+tQ$ for $t \in [0,1]$ is also in S. This section describes the *method of separating axes* in 2D—a method for determining whether or not two stationary convex objects are intersecting. The ideas extend to moving convex objects and are useful for predicting collisions of the objects by computing the first time of contact and for computing the contact set. Two types of geometric queries are considered. The first is a *test-intersection* query that just indicates whether or not an intersection exists for stationary objects or will occur for moving objects. The second is a *find-intersections* query that involves computing the set of intersections for two stationary objects or for two moving objects at the time of first contact. This section describes both types of queries for convex polygons in 2D.

The following notation is used throughout this section. Let C_j for $j = 0, 1$ be the convex polygons with vertices $\{V_i^{(j)}\}_{i=0}^{N_j-1}$ that are counterclockwise ordered. The edges of the polygons are $\vec{e}_i^{\,(j)} = V_{i+1}^{(j)} - V_i^{(j)}$ for $0 \le i < N_j$ and where $V_{N_j}^{(j)} = V_0^{(j)}$. Outward pointing normal vectors to the edges are $\vec{d}_i^{(j)} = \text{Perp}\left(\vec{e}_i^{\,(j)}\right)$, where $\text{Perp}(x, y) = (y, -x)$.

7.7.1 Separation by Projection onto a Line

A test for nonintersection of two convex objects is simply stated: if there exists a line for which the intervals of projection of the two objects onto that line do not intersect, then the objects do not intersect. Such a line is called a *separating line* or, more commonly, a *separating axis* (see Figure 7.12). The translation of a separating line is also a separating line, so it is sufficient to consider lines that contain the origin. Given a line containing the origin and with unit-length direction $\vec{d}$, the projection of a convex set C onto the line is the interval

$$I = [\lambda_{\min}(\vec{d}), \lambda_{\max}(\vec{d})] = [\min\{\vec{d} \cdot \vec{X} : \vec{X} \in C\}, \max\{\vec{d} \cdot \vec{X} : \vec{X} \in C\}]$$

where possibly $\lambda_{\min}(\vec{d}) = -\infty$ or $\lambda_{\max}(\vec{d}) = +\infty$; these cases arise when the convex set is unbounded. Two convex sets C_0 and C_1 are *separated* if there exists a direction $\vec{d}$ such that the projection intervals I_0 and I_1 do not intersect. Specifically they do not intersect when

$$\lambda_{\min}^{(0)}(\vec{d}) > \lambda_{\max}^{(1)}(\vec{d}) \quad \text{or} \quad \lambda_{\max}^{(0)}(\vec{d}) < \lambda_{\min}^{(1)}(\vec{d}) \tag{7.10}$$

The superscript corresponds to the index of the convex set. Although the comparisons are made where $\vec{d}$ is unit length, the comparison results are invariant to changes in length of the vector. This follows from $\lambda_{\min}(t\vec{d}) = t\lambda_{\min}(\vec{d})$ and $\lambda_{\max}(t\vec{d}) = t\lambda_{\max}(\vec{d})$ for $t > 0$. The Boolean value of the pair of comparisons is also invariant

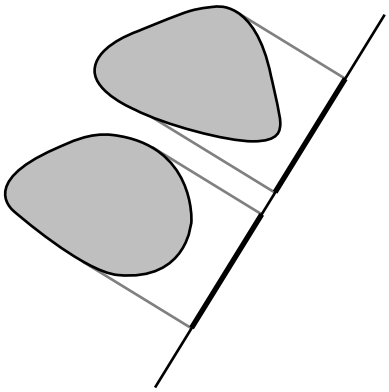

Figure 7.12 Nonintersecting convex objects and a separating line for them.

when $\vec{d}$ is replaced by the opposite direction $-\vec{d}$. This follows from $\lambda_{\min}(-\vec{d}) = -\lambda_{\max}(\vec{d})$ and $\lambda_{\max}(-\vec{d}) = -\lambda_{\min}(\vec{d})$. When $\vec{d}$ is not unit length, the intervals obtained for the separating line tests are not the projections of the object onto the line; rather they are scaled versions of the projection intervals. We make no distinction between the scaled projection and regular projection. We will also use the terminology that the direction vector for a separating line is called a *separating direction*, a direction that is not necessarily unit length.

Please note that in two dimensions, the terminology for separating line or axis is potentially confusing. The separating line separates the *projections* of the objects on that line. The separating line does *not* partition the plane into two regions, each containing an object. In three dimensions, the terminology should not be confusing since a plane would need to be specified to partition space into two regions, each containing an object. No real sense can be made for partitioning space by a line.

7.7.2 Separation of Stationary Convex Polygons

For a pair of convex polygons, only a finite set S of direction vectors needs to be considered for separation tests. That set contains only the normal vectors to the edges of the polygons. Figure 7.13(a) shows two nonintersecting polygons that are separated along a direction determined by the normal to an edge of one polygon. Figure 7.13(b) shows two polygons that intersect; there are no separating directions.

The intuition for why only edge normals must be tested is based on having two convex polygons just touching with no interpenetration. Figure 7.14 shows the three possible configurations: edge-edge contact, vertex-edge contact, and vertex-vertex

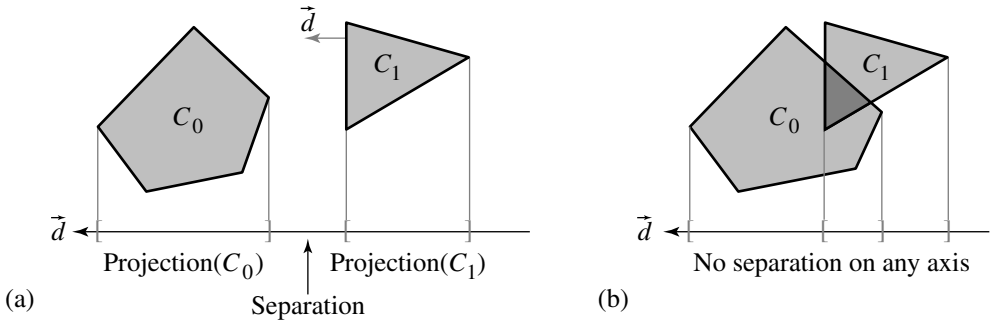

Figure 7.13 (a) Nonintersecting convex polygons. (b) Intersecting convex polygons.

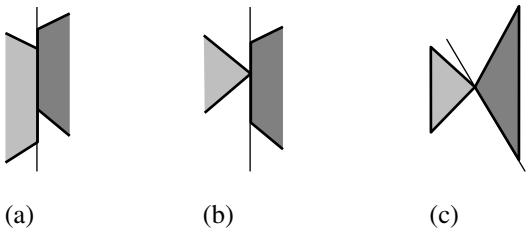

Figure 7.14 (a) Edge-edge contact, (b) vertex-edge contact, and (c) vertex-vertex contact.

contact. The lines between the polygons are perpendicular to the separation lines that would occur for one object translated away from the other by an infinitesimal distance.

The mathematical proof that S contains only edge normals is based on showing that if $\vec{d}$ is a separating direction that is not normal to an edge of either convex polygon, then there must be an edge normal that is also a separating direction. Let $\vec{d} = (\cos\theta, \sin\theta)$ be a separating direction that is not normal to an edge. For the sake of argument, assume that the projection of C_0 onto the separating line is on the left of the projection of C_1. A similar argument directly applies if it were on the right. Since $\vec{d}$ is not an edge normal, only one vertex V_0 of C_0 maps to $\lambda_{\max}^{(0)}$, and only one vertex V_1 of C_1 maps to $\lambda_{\min}^{(1)}$. Let θ_0 be the largest angle smaller than θ so that $\vec{d}_0 = (\cos\theta_0, \sin\theta_0)$ is an edge normal, but $\vec{d}(\phi) = (\cos\phi, \sin\phi)$ is not an edge normal for all $\phi \in (\theta_0, \theta]$. Similarly, let θ_1 be the smallest angle larger than θ so that $\vec{d}_1 =$

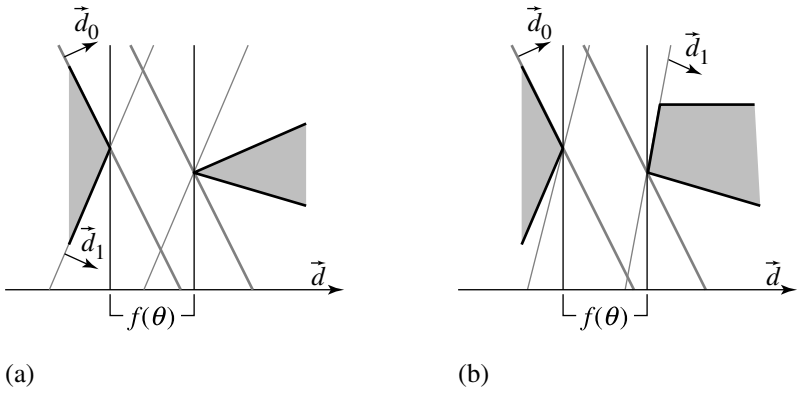

Figure 7.15 The edge normals closest to a non-edge-normal separation direction: (a) from the same triangle and (b) from different triangles.

$(\cos\theta_1, \sin\theta_1)$ is an edge normal, but $\vec{d}(\phi)$ is not an edge normal for all $\phi \in [\theta, \theta_1)$. For all directions $\vec{d}(\phi)$ with $\phi \in (\theta_0, \theta_1)$, V_0 is the unique vertex that maps to $\lambda_{\max}^{(0)}$ and V_1 is the unique vertex that maps to $\lambda_{\min}^{(1)}$. The separation between the intervals is the continuous function $f(\phi) = (\cos\phi, \sin\phi) \cdot (V_1 - V_0) = A\cos(\phi + \psi)$, where A is a constant amplitude and ψ is a constant phase angle. Also, $f(\theta) > 0$ since $\vec{d}$ is a separating direction.

If $f(\theta_0) > 0$, then the edge normal $\vec{d}_0$ is also a separating direction. If $f(\theta_1) > 0$, then the edge normal $\vec{d}_1$ is also a separating direction. Suppose that $f(\theta_0) \le 0$ and $f(\theta_1) \le 0$. Since $f(\theta) > 0$, there must exist two zeros of f on $[\theta_0, \theta_1]$, one smaller than θ and one larger than θ. The zeros of f are separated by π radians. This forces $\theta_1 - \theta_0 \ge \pi$, in which case the angle between the consecutive edge normals $\vec{d}_0$ and $\vec{d}_1$ is at least π radians. This happens only if the angle is exactly π, the two edges sharing V_0 are parallel to $\vec{d}$, and the two edges sharing V_1 are parallel to $\vec{d}$, a contradiction to the angles at those vertices being strictly positive. Therefore, it is impossible that both $f(\theta_0) \le 0$ and $f(\theta_1) \le 0$. In summary, if $f(\theta) > 0$, then either $f(\theta_0) > 0$, in which case $\vec{d}_0$ is a separating edge normal, or $f(\theta_1) > 0$, in which case $\vec{d}_1$ is a separating edge normal.

Figure 7.15 illustrates what $\vec{d}_0$ and $\vec{d}_1$ mean. Figure 7.15(a) shows the case where both nearest edge normals are from the same triangle. Figure 7.15(b) shows the case where the nearest edge normals are from different triangles.

The Direct Implementation

The direct implementation for a separation test for direction $\vec{d}$ just computes the extreme values of the projection and compares them. That is, compute $\lambda_{\min}^{(j)}(\vec{d}) = \min_{0 \le i < N_0}\{\vec{d} \cdot V_i^{(j)}\}$ and $\lambda_{\max}^{(j)}(\vec{d}) = \max_{0 \le i < N_1}\{\vec{d} \cdot V_i^{(j)}\}$ and test the inequalities in Equation 7.10. The pseudocode is listed below.

```
bool TestIntersection(ConvexPolygon C0, ConvexPolygon C1)
{
    // test edge normals of C0 for separation
    for (i0 = 0, i1 = C0.N-1; i0 < C0.N; i1 = i0, i0++) {
        D = Perp(C0.E(i1));  // C0.E(i1) = C0.V(i0) - C0.V(i1)
        ComputeInterval(C0, D, min0, max0);
        ComputeInterval(C1, D, min1, max1);
        if (max1 < min0 || max0 < min1)
            return false;
    }

    // test edge normals of C1 for separation
    for (i0 = 0, i1 = C1.N - 1; i0 < C1.N; i1 = i0, i0++) {
        D = Perp(C1.E(i1));  // C1.E(i1) = C1.V(i0) - C1.V(i1));
        ComputeInterval(C0, D, min0, max0);
        ComputeInterval(C1, D, min1, max1);
        if (max1 < min0 || max0 < min1)
            return false;
    }

    return true;
}

void ComputeInterval(ConvexPolygon C, Point D, float& min, float& max)
{
    min = max = Dot(D, C.V(0));
    for (i = 1; i < C.N; i++) {
        value = Dot(D, C.V(i));
        if (value < min)
            min = value;
        else if (value > max)
            max = value;
    }
}
```

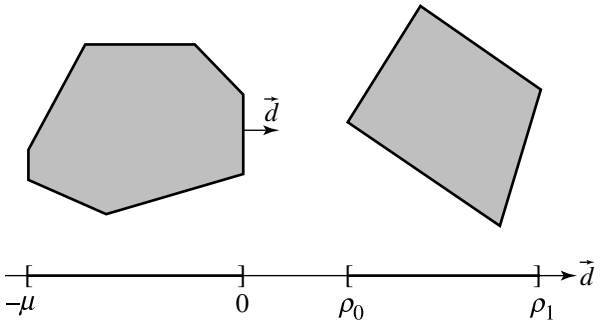

Figure 7.16 Two polygons separated by an edge-normal direction of the first polygon.

Observe that the implementation always processes potential separating lines that contain the origin. When polygons are relatively far from the origin, a variation on the implementation to deal with floating-point errors would involve choosing a potential separating line that contains a polygon vertex, thereby hoping to keep intermediate floating-point values relatively small.

An Alternative Implementation

An alternative algorithm avoids projecting all the vertices for the polygons by only testing for separation using the maximum of the interval for the first polygon and the minimum of the interval for the second polygon. If $\vec{d}$ is an outward pointing normal for the edge $V_{i+1} - V_i$ of the first polygon, then the projection of the first polygon onto the separating line $V_i + t\vec{d}$ is $[-\mu, 0]$, where $\mu > 0$. If the projection of the second polygon onto this line is $[\rho_0, \rho_1]$, then the reduced separation test is $\rho_0 > 0$. Figure 7.16 illustrates two separated polygons using this scheme. The value μ is irrelevant since we only need to compare ρ_0 to 0. Consequently, there is no need to project the vertices of the first polygon to calculate μ. Moreover, the vertices of the second polygon are projected one at a time until either the projected value is negative, in which case $\vec{d}$ is no longer considered for separation, or until all projected values are positive, in which case $\vec{d}$ is a separating direction.

```
bool TestIntersection(ConvexPolygon C0, ConvexPolygon C1)
{
    // Test edges of C0 for separation.  Because of the counterclockwise ordering,
    // the projection interval for C0 is [m,0] where m <= 0.  Only try to determine
    // if C1 is on the 'positive' side of the line.
```