Chapter 3
Basic Primitives

Elementary, my dear Watson.
—Sherlock Holmes

In this chapter we discuss intersection tests for a number of basic primitives. The primitives we consider are spheres, boxes, line segments (rays), triangles, and general polygons. These are the most commonly used primitives in interactive 3D applications. Spheres and boxes are popular bounding-volume types, whereas triangles and polygons are in general the components from which complex models are built. This chapter provides intersection tests for any combination of these primitives.

3.1 Spheres

Spheres are probably the simplest type of primitives for geometric modeling. A sphere can be represented using only four scalars (three for the center point and one for the radius), so they are quite cheap to store. Furthermore, spheres are invariant under rotations, which makes them a good candidate as a bounding volume for rigid bodies. In many applications, spheres are also used as active areas in which events are triggered. For instance, a sliding door that opens when an avatar approaches the door can be simulated by detecting whether the avatar intersects a sphere that represents the active area of the door. An *avatar* is an object in a 3D world that represents the user who is viewing the world from the location of the object. Both the avatar itself as well as the area of the world that is visible/audible by the user are often represented as spheres.

3.1.1 Sphere-Sphere Test

Due to their simplicity, collision detection of spheres is not that hard. Two spheres A and B intersect iff the distance between their centers $\mathbf{c}_A$ and $\mathbf{c}_B$

is at most the sum of their radii ρ_A and ρ_B:

$$A \cap B \neq \emptyset \equiv \|\mathbf{c}_A - \mathbf{c}_B\| \leq \rho_A + \rho_B.$$

We want to avoid the evaluation of square roots as much as possible, since square roots take more time to compute than primitive arithmetic operations such as additions and multiplications. So, for our implementations we rewrite the expression as

$$A \cap B \neq \emptyset \equiv \|\mathbf{c}_A - \mathbf{c}_B\|^2 \leq (\rho_A + \rho_B)^2,$$

which uses only primitive arithmetic operations. The distance between a pair of spheres is the distance between their centers minus the sum of their radii—that is, if the spheres do not intersect, because then the distance is of course zero:

$$d(A,B) = \max(\|\mathbf{c}_A - \mathbf{c}_B\| - (\rho_A + \rho_B), 0).$$

We find a similar expression for the penetration depth, which is zero for nonintersecting objects:

$$p(A,B) = \max(\rho_A + \rho_B - \|\mathbf{c}_A - \mathbf{c}_B\|, 0).$$

The witness points for both a nonzero distance and a nonzero penetration depth are computed in the same way. Let $\mathbf{v} = \mathbf{c}_A - \mathbf{c}_B$, the vector from B's center to A's center. Then, the points

$$\mathbf{p}_A = \mathbf{c}_A - \rho_A \frac{\mathbf{v}}{\|\mathbf{v}\|} \quad \text{and} \quad \mathbf{p}_B = \mathbf{c}_B + \rho_B \frac{\mathbf{v}}{\|\mathbf{v}\|}$$

are the witness points for either the distance or the penetration depth, as can be verified by computing the distance between these points. Note that these expressions are valid only for nonconcentric spheres. For a pair of concentric spheres, $\mathbf{v}$ is the zero vector, and thus the expressions result in division by zero. For concentric spheres, choose an arbitrary nonzero vector $\mathbf{v}$, preferably one of unit length, since this saves normalization, and compute the witness points for the penetration depth using this vector.

3.1.2 Ray-Sphere Test

In Section 2.4, we saw that a four-dimensional space-time collision test on a pair of spheres can be done by performing a ray cast on the CSO of the spheres in local coordinates. A ray is a line segment connecting a source point $\mathbf{s}$ and a target point $\mathbf{t}$. For the four-dimensional intersection test, the points $\mathbf{s}$ and $\mathbf{t}$ are the differences $\mathbf{c}_B - \mathbf{c}_A$ of the centers at $t = 0$

and $t = 1$, respectively. If the ray intersects the CSO, then the *ray cast* returns the smallest $\lambda \in [0,1]$ for which the point $\mathbf{x} = \mathbf{s} + \lambda(\mathbf{t} - \mathbf{s})$ is contained in the CSO. The CSO of a pair of spheres is itself a sphere centered at $\mathbf{c}_A - \mathbf{c}_B$ and with a radius ρ equal to $\rho_A + \rho_B$, the sum of the radii of the spheres. For the ray cast we use the CSO of the spheres in local coordinates (i.e., centered at the origin); thus the CSO is centered at the origin as well. The ray cast is performed in the following way.

Let the direction of the ray be given by the vector $\mathbf{r} = \mathbf{t} - \mathbf{s}$. First, we compute the intersection of the unbounded line $\mathbf{x} = \mathbf{s} + \lambda\mathbf{r}$ and the sphere. The points of intersection of the line and the sphere's boundary are given by

$$\mathbf{x} = \mathbf{s} + \lambda\mathbf{r} \quad \text{and} \quad \|\mathbf{x}\| = \rho.$$

We substitute the first expression in the second and square out the square root,

$$\|\mathbf{s} + \lambda\mathbf{r}\|^2 = \rho^2.$$

This quadratic equation needs some rewriting in order to find the roots:

$$\begin{aligned}
\|\mathbf{s} + \lambda\mathbf{r}\|^2 = \rho^2 &\equiv \|\mathbf{s} + \lambda\mathbf{r}\|^2 - \rho^2 = 0 \\
&\equiv \|\mathbf{s}\|^2 + 2\lambda(\mathbf{s}\cdot\mathbf{r}) + \lambda^2\|\mathbf{r}\|^2 - \rho^2 = 0 \\
&\equiv \|\mathbf{r}\|^2\lambda^2 + 2(\mathbf{s}\cdot\mathbf{r})\lambda + \|\mathbf{s}\|^2 - \rho^2 = 0.
\end{aligned}$$

Thus,

$$\lambda_{1,2} = \frac{-\mathbf{s}\cdot\mathbf{r} \pm \sqrt{(\mathbf{s}\cdot\mathbf{r})^2 - \|\mathbf{r}\|^2(\|\mathbf{s}\|^2 - \rho^2)}}{\|\mathbf{r}\|^2}.$$

A solution exists only if

$$(\mathbf{s}\cdot\mathbf{r})^2 - \|\mathbf{r}\|^2(\|\mathbf{s}\|^2 - \rho^2) \geq 0.$$

This should not be too much of a surprise, since after rewriting this expression as

$$\|\mathbf{s}\|^2 - \left(\frac{\mathbf{s}\cdot\mathbf{r}}{\|\mathbf{r}\|}\right)^2 \leq \rho^2,$$

we see that the left-hand side is the squared distance of the origin to the line, as can be verified in Figure 3.1. The line intersects the sphere iff the distance between the origin and the line is at most the sphere's radius.

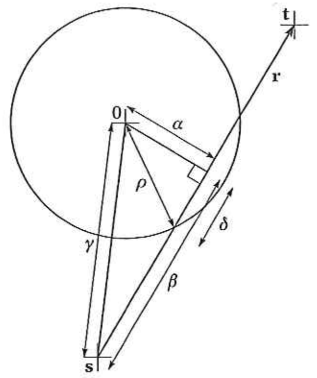

Figure 3.1 The distance α of the origin to the line $\overline{\mathbf{st}}$ is found using the Pythagorean theorem, $\alpha^2 + \beta^2 = \gamma^2$. Here, $\beta = -\mathbf{s} \cdot \mathbf{r}/\|\mathbf{r}\|$ and $\gamma = \|\mathbf{s}\|$. Thus, we find $\alpha^2 = \|\mathbf{s}\|^2 - (\mathbf{s} \cdot \mathbf{r}/\|\mathbf{r}\|)^2$. The line intersects the sphere only if $\alpha^2 \leq \rho^2$, in which case the closest common point of the ray and the sphere lies at a distance of $\beta - \delta$ from $\mathbf{s}$, where $\delta^2 = \rho^2 - \alpha^2$.

The ray itself intersects the sphere iff $[\lambda_1, \lambda_2] \cap [0, 1] \neq \emptyset$. This is the case iff $\lambda_1 \leq 1$ and $\lambda_2 \geq 0$. If $\lambda_1 > 1$, then the ray did not reach far enough to hit the sphere. If $\lambda_2 < 0$, then the ray is pointing away from the sphere. In the case where the ray intersects the sphere, the parameter for the point where the ray enters the sphere is $\lambda_{enter} = \max(\lambda_1, 0)$. The value λ_{enter} can be used as the time of collision of the spheres in the four-dimensional intersection test. The interval of time where the spheres are intersecting is $[\lambda_{enter}, \lambda_{exit}]$, where $\lambda_{exit} = \min(\lambda_2, 1)$. Note that if λ_{exit} does not need to be returned, the value λ_2 is used only for testing whether it is at least zero. In that case, we can leave out the division by $\|\mathbf{r}\|^2$ in the computation of λ_2.

If $0 \leq \lambda_1 \leq 1$, then the point $\mathbf{x} = \mathbf{s} + \lambda_1\mathbf{r}$ is the point where the ray enters the sphere. The normal to the boundary at this point is $\mathbf{x}$, regarded as the vector from the origin to $\mathbf{x}$. If $\lambda_1 < 0 < \lambda_2$, then the point $\mathbf{s}$ is an internal point of the sphere, and thus the ray does not enter the sphere. Algorithm 3.1 summarizes the operations that are performed for a ray cast.

Algorithm 3.1 A ray cast for a ray $\overline{\mathbf{st}}$ and a sphere centered at the origin having radius ρ. The intersection of the ray and the sphere is represented by the interval $[\lambda_{enter}, \lambda_{exit}]$. The point where the ray enters the sphere is returned as $\mathbf{x}$.

```
$\mathbf{r} := \mathbf{t} - \mathbf{s}$;
$\sigma := (\mathbf{s} \cdot \mathbf{r})^2 - \|\mathbf{r}\|^2(\|\mathbf{s}\|^2 - \rho^2)$;
if $\sigma \geq 0$ then
  {{Line $\mathbf{x} = \mathbf{s} + \lambda\mathbf{r}$ intersects the sphere.}}
begin
  $\lambda_1 := (-\mathbf{s} \cdot \mathbf{r} - \sqrt{\sigma})/\|\mathbf{r}\|^2$;
  $\lambda_2 := (-\mathbf{s} \cdot \mathbf{r} + \sqrt{\sigma})/\|\mathbf{r}\|^2$;
  if $\lambda_1 \leq 1$ and $\lambda_2 \geq 0$ then
    {{The ray intersects the sphere, since $[\lambda_1, \lambda_2] \cap [0, 1] \neq \emptyset$.}}
  begin
    $\lambda_{enter} := \max(\lambda_1, 0)$;
    $\lambda_{exit} := \min(\lambda_2, 1)$;
    $\mathbf{x} := \mathbf{s} + \lambda_{enter}\mathbf{r}$;
    return true
  end
end;
return false
```

For a ray of infinite length, we can use almost the same computation. An infinite ray is represented by a source point $\mathbf{s}$ and a unit vector $\mathbf{r}$. The ray is the set of points $\mathbf{x} = \mathbf{s} + \lambda\mathbf{r}$, where $\lambda \geq 0$. Algorithm 3.1 is greatly simplified for infinite rays, since the condition $\lambda_1 \leq 1$ does not have to be checked and all occurrences of $\|\mathbf{r}\|^2$ can be removed.

Let us explore the potential numerical problems in Algorithm 3.1. Suppose that the length of $\mathbf{s}$ is orders of magnitude greater that ρ, $\|\mathbf{s}\| \gg \rho$, and that the ray is pointing at the sphere. Then, $(\mathbf{s} \cdot \mathbf{r})^2 \approx \|\mathbf{r}\|^2\|\mathbf{s}\|^2$, and thus, the relative error in $\tilde{\sigma}$, the computed value of σ, can become quite large. So, the computed value $\tilde{\sigma}$ may be negative when the theoretical value σ is nonnegative or vice versa. Furthermore, suppose that $\tilde{\sigma} \geq 0$ and that the ray is long enough to hit the sphere, $\tilde{\lambda}_1 \leq 1$. Then, the computed value $\tilde{\mathbf{x}}$ for the point where the ray enters the sphere may not lie exactly on the sphere's boundary. We see that the long-range accuracy of the ray cast is limited. Using an infinite ray, $\|\mathbf{r}\| = 1$, for long-range ray casts does not make a dramatic difference, since this will reduce the relative error in $\tilde{\sigma}$ only by a small factor. So, in practical applications of the ray cast, it is best to use relatively short rays, or, for infinite rays, ignore hits with distant spheres.

3.1.3 Line Segment–Sphere Test

For cases where the points of intersection of the ray with the sphere's boundary are of no interest, a simpler test can be used. A line segment intersects a sphere iff the point on the segment closest to the sphere's center is contained by the sphere. A point on the line segment is a point $\mathbf{x} = \mathbf{s} + \lambda\mathbf{r}$, for some $\lambda \in [0, 1]$. We first solve this problem without the constraint $\lambda \in [0, 1]$ and compute the parameter λ for the closest point on the unbounded line through the segment. This is the point $\mathbf{x} = \mathbf{s} + \lambda\mathbf{r}$ for which the vector from the sphere's center to $\mathbf{x}$ is orthogonal to the direction of the line. Thus, for the closest point we have $\mathbf{x} \cdot \mathbf{r} = 0$. We substitute $\mathbf{x} = \mathbf{s} + \lambda\mathbf{r}$ in the latter equation and find

$$(\mathbf{s} + \lambda\mathbf{r}) \cdot \mathbf{r} = 0.$$

This equation gives us the parameter $\lambda = -\mathbf{s} \cdot \mathbf{r}/\|\mathbf{r}\|^2$. If $\lambda \leq 0$, then $\mathbf{s}$ is the closest point of the line segment. If $\lambda \geq 1$, then $\mathbf{t}$ is the closest point. Otherwise, if $0 < \lambda < 1$, then the closest point is an internal point of the line segment. Algorithm 3.2 describes this intersection test.

Algorithm 3.2

An intersection test for a line segment $\overline{\mathbf{st}}$ and a sphere centered at the origin having radius ρ. The point $\mathbf{x}$ is the point of the line segment closest to the sphere's center.

```
$\mathbf{r} := \mathbf{t} - \mathbf{s}$;
$\delta := -\mathbf{s} \cdot \mathbf{r}$;
if $\delta \leq 0$ then $\mathbf{x} := \mathbf{s}$
else if $\delta \geq \|\mathbf{r}\|^2$ then $\mathbf{x} := \mathbf{t}$
else
begin
    $\lambda := \delta / \|\mathbf{r}\|^2$;
    $\{\{0 < \lambda < 1\}\}$
    $\mathbf{x} := \mathbf{s} + \lambda\mathbf{r}$
end;
return $\|\mathbf{x}\|^2 \leq \rho^2$
```

The closest point $\mathbf{x}$ is also the witness point on the line segment for the distance and the penetration depth. The distance is $\max(\|\mathbf{x}\| - \rho, 0)$, and the penetration depth is $\max(\rho - \|\mathbf{x}\|, 0)$. As witness on the sphere we can use the point $\rho\mathbf{x}/\|\mathbf{x}\|$—that is, if $\mathbf{x}$ is not the zero vector. In the degenerate case where the line segment contains the origin, we may use any point $\rho\mathbf{y}/\|\mathbf{y}\|$, for which $\mathbf{y} \perp \mathbf{r}$, as the witness point on the sphere. A good choice is $\mathbf{y} = \mathbf{r} \times \mathbf{e}_i$, where $\mathbf{e}_i$ is the coordinate axis on which $\mathbf{r}$ has the smallest absolute coordinate ($|\mathbf{v} \cdot \mathbf{e}_i|$ is the smallest). In this way, $\mathbf{y}$ can never be zero.

3.2 Axis-Aligned Boxes

Axis-aligned boxes are the most widely used type of bounding volumes because they are easy to compute, cheap to store, and fast to test for

intersection. For axis-aligned boxes represented using min-max representation, testing the intersection is simply done by comparing the extrema:

$$[\mathbf{p}_1, \mathbf{q}_1] \cap [\mathbf{p}_2, \mathbf{q}_2] \neq \emptyset \quad \equiv \quad \mathbf{p}_1 \leq \mathbf{q}_2 \text{ and } \mathbf{p}_2 \leq \mathbf{q}_1.$$

Testing the intersection of a pair of boxes in center-extent representation is not much harder:

$$[\mathbf{c}_1 - \mathbf{h}_1, \mathbf{c}_1 + \mathbf{h}_1] \cap [\mathbf{c}_2 - \mathbf{h}_2, \mathbf{c}_2 + \mathbf{h}_2] \neq \emptyset \quad \equiv \quad |\mathbf{c}_1 - \mathbf{c}_2| \leq \mathbf{h}_1 + \mathbf{h}_2.$$

The center-extent test uses a few arithmetic operations but performs three scalar comparisons and branch instructions fewer than the min-max test. On a platform that has parallel hardware for performing arithmetic operations, the difference in performance is negligible.

3.2.1 Ray-Box Test

For performing a ray cast on an axis-aligned box we enter the realm of line segment clipping. If we merely need to test a line segment for intersection with an axis-aligned box, then the separating-axes test, discussed in Section 3.3, is likely to be faster. There is a considerable amount of literature available on line segments [11, 32, 33, 43, 78]. We borrow techniques from two popular line clippers, Cohen-Sutherland (CS) [43] and Liang-Barsky (LB) [78]. Originally, CS and LB were presented for clipping in 2D; however, both algorithms can be readily generalized to 3D. We will give brief descriptions of the algorithms.

CS uses a classification of points with respect to the six planes supporting the facets of the box. The classification is represented by a 6-bit *outcode*, in which each bit corresponds with a plane. A bit in the outcode is 1 iff the point lies "outside" the corresponding plane—in the positive open halfspace of the plane for plane normals pointing outward. We see that a point that is contained by the box is classified as 000000. If the outcodes of the ray's endpoints contain the same bit (i.e., their bitwise AND is nonzero), then the ray does not hit the box, since the corresponding plane separates the ray from the box. If the bitwise AND of the outcodes of the ray's endpoints is zero, the ray may still miss the box, as illustrated in Figure 3.2 for the 2D case.

We need to compute the parameters of the intersection points of the ray and the planes, and, for this purpose, we use a technique from the LB parametric line clipping algorithm. Let $\mathbf{s}$ be the source point and $\mathbf{t}$ be the target point of the ray. Then, an intersection point can be expressed as $\mathbf{s}+\lambda(\mathbf{t}-\mathbf{s})$, where $\lambda \in [0, 1]$ is its parameter. By classifying the intersection

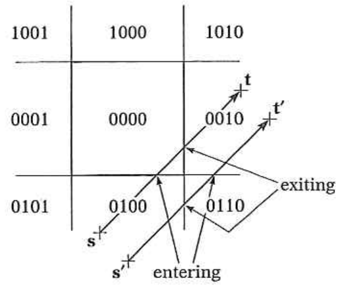

Figure 3.2 A ray cast for axis-aligned boxes using techniques from Cohen-Sutherland and Liang-Barsky line clipping. The ray intersects the box iff the largest parameter of an entering intersection point is at most the smallest parameter of an exiting intersection point.

point as *entering* or *exiting*, we can decide whether the ray hits the box. An intersection point is classified as *entering* if moving from **s** to **t** we go from the positive to the negative halfspace of the corresponding plane, and *exiting* otherwise. It can be seen that a ray intersects the box iff the largest parameter of an entering intersection point is at most the smallest parameter of an exiting intersection point.

We will now explain why CS and LB make such a good team. First of all, under the assumption that the bitwise AND of the outcodes of the endpoints is zero, the 1 bits in either of the two outcodes correspond with the planes for which intersection points need to be computed, since the ray crosses only the planes that correspond with these bits. Moreover, each intersection point corresponding with a 1 bit in the outcode of **s** is entering, and each intersection point corresponding with a 1 bit in **t**'s outcode is exiting. Hence, the ray intersects the box iff the largest parameter corresponding with a 1 bit in **s**'s outcode is at most the smallest parameter corresponding with a 1 bit in **t**'s outcode. We see how neatly LB benefits from the outcodes computed by CS. Considering the long history of line clipping, it is remarkable that a hybrid of CS and LB got published only recently [11].

For a box centered at the origin the parameters are computed in the following way. Let σ_i and τ_i be the ith components of the ray's source **s** and target **t**, respectively, and let η_i be the ith component of the box's extent **h**. Then, the parameters of the intersection points on the lower and upper plane orthogonal to the ith coordinate axis are, respectively,

$$\lambda_- = \frac{-\sigma_i - \eta_i}{\tau_i - \sigma_i} \quad \text{and} \quad \lambda_+ = \frac{-\sigma_i + \eta_i}{\tau_i - \sigma_i}.$$

Since we compute these parameters only for planes for which **s** and **t** lie on different sides, each computed parameter lies in the interval [0, 1], and thus represents an intersection point of the ray and the plane. Pseudocode for the ray cast on boxes is presented in Algorithm 3.3.

Algorithm 3.3 A ray cast for a ray $\overline{\mathbf{st}}$, where $\mathbf{s} = (\sigma_1, \sigma_2, \sigma_3)$ and $\mathbf{t} = (\tau_1, \tau_2, \tau_3)$, and a box centered at the origin having extent (η_1, η_2, η_3). The intersection of the ray and the box is represented by the interval $[\lambda_{enter}, \lambda_{exit}]$. The point where the ray enters the box is returned as **x**.

```
$bits_\mathbf{s} := outcode(\mathbf{s})$;
$bits_\mathbf{t} := outcode(\mathbf{t})$;
if $bits_\mathbf{s}$ & $bits_\mathbf{t} = 0$ then
  {{None of the side planes separate the ray from the box.}}
begin
  $\lambda_{enter} := 0$;
  $\lambda_{exit} := 1$;
  $bit := 1$;
  for $i := 1, 2, 3$ do
  begin
    if $bits_\mathbf{s}$ & $bit \neq 0$ then
      {{Point of intersection is entering.}}
      $\lambda_{enter} := \max(\lambda_{enter}, (-\sigma_i - \eta_i)/(\tau_i - \sigma_i))$;
    else if $bits_\mathbf{t}$ & $bit \neq 0$ then
      {{Point of intersection is exiting.}}
      $\lambda_{exit} := \min(\lambda_{exit}, (-\sigma_i - \eta_i)/(\tau_i - \sigma_i))$;
    $bit := bit \ll 1$;
    if $bits_\mathbf{s}$ & $bit \neq 0$ then
      {{Point of intersection is entering.}}
      $\lambda_{enter} := \max(\lambda_{enter}, (-\sigma_i + \eta_i)/(\tau_i - \sigma_i))$;
    else if $bits_\mathbf{t}$ & $bit \neq 0$ then
      {{Point of intersection is exiting.}}
      $\lambda_{exit} := \min(\lambda_{exit}, (-\sigma_i + \eta_i)/(\tau_i - \sigma_i))$;
    $bit := bit \ll 1$
  end;
  if $\lambda_{enter} \leq \lambda_{exit}$ then
    {{The ray intersects the box, since $[\lambda_{enter}, \lambda_{exit}] \neq \emptyset$.}}
  begin
    $\mathbf{x} := \mathbf{s} + \lambda_{enter}(\mathbf{t} - \mathbf{s})$;
    return true
  end
end;
return false
```

3.2.2 Sphere-Box Test

We conclude this section with a discussion of how a sphere and an axis-aligned box are tested for intersection. This is done by computing the point in the box closest to the sphere's center, and testing whether this point is contained in the sphere. Let the box be centered at the origin with extent $\mathbf{h} = (\eta_1, \eta_2, \eta_3)$, and let the sphere's center be given by $\mathbf{c} = (\gamma_1, \gamma_2, \gamma_3)$. Then, the point in the box closest to $\mathbf{c}$ is the point $\mathbf{x} = (\text{clamp}(\gamma_1, -\eta_1, \eta_1), \text{clamp}(\gamma_2, -\eta_2, \eta_2), \text{clamp}(\gamma_3, -\eta_3, \eta_3))$, where

$$\text{clamp}(\gamma, \alpha, \beta) = \begin{cases} \alpha \text{ if } \gamma < \alpha \\ \beta \text{ if } \gamma > \beta \\ \gamma \text{ otherwise.} \end{cases}$$

The sphere intersects the box iff the point $\mathbf{x}$ is contained in the sphere; that is, the squared distance $\|\mathbf{x} - \mathbf{c}\|^2$ is at most the squared radius ρ^2.

The witness points for a nonzero distance are found in the following way. We take the point $\mathbf{x}$ closest to the sphere's center as the witness point on the boundary of the box. The witness point on the sphere's boundary is the point

$$\mathbf{r} = \mathbf{c} + \rho \frac{\mathbf{v}}{\|\mathbf{v}\|},$$

where $\mathbf{v} = \mathbf{x} - \mathbf{c}$. The distance between the sphere and the box is $\|\mathbf{x} - \mathbf{r}\| = \|\mathbf{v}\| - \rho$—that is, if the sphere and the box do not intersect ($\|\mathbf{v}\| > \rho$), since otherwise the distance is zero.

If the sphere and the box intersect and the center of the sphere is not contained by the box, then the points $\mathbf{r}$ and $\mathbf{x}$ are witness points of a nonzero penetration depth as well. In that case, the penetration depth is $\rho - \|\mathbf{v}\|$. However, if the center is contained by the box, then the points $\mathbf{c}$ and $\mathbf{x}$ coincide. In that case, the witness point for the box is given by the point $\mathbf{y}$ on the boundary of the box closest to the sphere's center. The point $\mathbf{y}$ is found in the following way. First, we select the coordinate axis on which $\mathbf{c}$'s component lies closest to the boundary, as illustrated in Figure 3.3. This is the axis $\mathbf{e}_i$ for which $\delta_i = \eta_i - |\gamma_i|$ has the smallest value. If multiple axes result in a smallest value δ_i, then simply choose one of these axes. Given such an axis $\mathbf{e}_i$, the point on the boundary of the box closest to the sphere's center is

$$\mathbf{y} = \mathbf{c} + \delta_i \,\text{sign}(\gamma_i)\mathbf{e}_i,$$

where

$$\text{sign}(\alpha) = \begin{cases} -1 \text{ if } \alpha < 0 \\ 1 \text{ otherwise.} \end{cases}$$

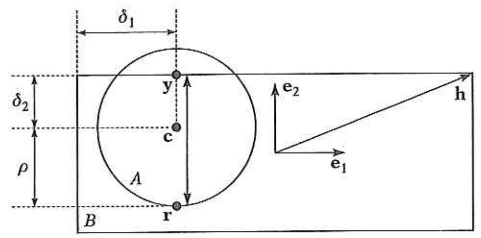

Figure 3.3 Computing the penetration depth of a sphere A and a box B for the case where the sphere's center $\mathbf{c}$ is contained by the box. The point $\mathbf{y}$ is the point on the boundary of B closest to $\mathbf{c}$. The penetration depth vector $\mathbf{y} - \mathbf{r}$ is aligned along $\mathbf{e}_2$, the coordinate axis on which $\mathbf{c}$'s component lies closest to the boundary.

The witness point on the sphere is the point $\mathbf{r} = \mathbf{c} - \rho\,\text{sign}(\gamma_i)\mathbf{e}_i$, and the penetration depth is $\|\mathbf{y} - \mathbf{r}\| = \delta_i + \rho$.

3.3 Separating Axes

For simple polytopes, such as line segments, triangles, and boxes, there exists an easy and fast method for intersection testing. Two objects A and B are disjoint if for some vector $\mathbf{v}$ the projections $\mathbf{v} \cdot A$ and $\mathbf{v} \cdot B$ of the objects onto the vector do not overlap. Such a vector $\mathbf{v}$ is called a *separating axis*. This property is illustrated in Figure 3.4. For nonintersecting convex objects a separating axis always exists. A general proof for this claim is presented in Chapter 4. For now, let's restrict ourselves to separating axes for polytopes.

Theorem 3.1 gives us a straightforward method for finding a separating axis for a pair of polytopes that can successfully be used if the number of

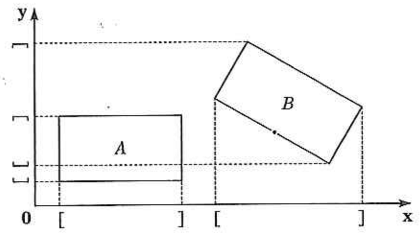

Figure 3.4 The vector $\mathbf{x}$ is a separating axis of A and B, whereas $\mathbf{y}$ is not a separating axis.