```
    // for a point gives the signed distance of the point to the plane
    float dist = Dot(s.c, p.n) - p.d;
    // If sphere center within +/-radius from plane, plane intersects sphere
    return Abs(dist) <= s.r;
}
```

To determine if the sphere lies fully behind (inside the negative halfspace of) plane π, the test changes to:

```
// Determine whether sphere s is fully behind (inside negative halfspace of) plane p
int InsideSpherePlane(Sphere s, Plane p)
{
    float dist = Dot(s.c, p.n) - p.d;
    return dist < -s.r;
}
```

If instead the negative halfspace of the plane is to be considered solid for the test with the sphere, the test changes to:

```
// Determine whether sphere s intersects negative halfspace of plane p
int TestSphereHalfspace(Sphere s, Plane p)
{
    float dist = Dot(s.c, p.n) - p.d;
    return dist <= s.r;
}
```

5.2.3 Testing Box Against Plane

Let a plane P be given by $(\mathbf{n} \cdot X) = d$. Testing if a box B intersects P can also be accomplished with the separating-axis test. Here, only the axis parallel to the plane normal $\mathbf{n}$ need be tested. Because the plane extends indefinitely, there are no edges with which to form edge-edge axis combinations. Axes corresponding to the face normals of the box can be eliminated from testing because the infinite extent of the plane means the plane will never sit fully outside one of the box faces unless it is parallel to the face, a case already handled by the plane normal axis.

Consider first the case of B being an OBB, given by the usual representation of a center C; local coordinate axes $\mathbf{u}_0$, $\mathbf{u}_1$, and $\mathbf{u}_2$; and three scalars e_0, e_1, and e_2 (making the OBB sides $2e_i$ wide for $0 \leq i \leq 2$). Points R in the OBB are given by $R = C \pm a_0\mathbf{u}_0 \pm a_1\mathbf{u}_1 \pm a_2\mathbf{u}_2$, where $|a_i| \leq e_i$. Similarly, the eight vertices V_i, $0 \leq i \leq 7$, of the OBB are given by $V_i = C \pm e_0\mathbf{u}_0 \pm e_1\mathbf{u}_1 \pm e_2\mathbf{u}_2$.

In that any line L parallel to $\mathbf{n}$ serves as a separating axis, a good choice is to have L go through the box center, giving L as $L(t) = C + t\mathbf{n}$. The box center C projects onto L at $t = 0$. Because the OBB is symmetrical about its center, the projection onto L results in a symmetric interval of projection $[C - r\,\mathbf{n}, C + r\,\mathbf{n}]$, centered at C, with a halfwidth (or radius) of r. Testing if B intersects P now amounts to computing the radius r of the projection interval and checking if the distance of the center point of B to P is less than r.

Because points of the OBB farthest away from its center are the vertices of the OBB, the overall maximum projection radius onto any vector $\mathbf{n}$ will be realized by one of the vertices. Thus, it is sufficient to consider only these when computing the radius r. The projection radii r_i of the eight vertices are given by

$$r_i = (V_i - C) \cdot \mathbf{n} = (C \pm e_0\mathbf{u}_0 \pm e_1\mathbf{u}_1 \pm e_2\mathbf{u}_2 - C) \cdot \mathbf{n} = (\pm e_0\mathbf{u}_0 \pm e_1\mathbf{u}_1 \pm e_2\mathbf{u}_2) \cdot \mathbf{n}.$$

Due to the distributive properties of the dot product, this expression can be written as

$$r_i = \pm(e_0\mathbf{u}_0 \cdot \mathbf{n}) \pm (e_1\mathbf{u}_1 \cdot \mathbf{n}) \pm (e_2\mathbf{u}_2 \cdot \mathbf{n}).$$

The maximum positive radius r is obtained when all involved terms are positive, corresponding to only positive steps along $\mathbf{n}$, which is achieved by taking the absolute value of the terms before adding them up:

$$r = |e_0\mathbf{u}_0 \cdot \mathbf{n}| + |e_1\mathbf{u}_1 \cdot \mathbf{n}| + |e_2\mathbf{u}_2 \cdot \mathbf{n}|\,.$$

Because the extents are assumed positive, r can be written as

$$r = e_0\,|\mathbf{u}_0 \cdot \mathbf{n}| + e_1\,|\mathbf{u}_1 \cdot \mathbf{n}| + e_2\,|\mathbf{u}_2 \cdot \mathbf{n}|\,.$$

When the separating-axis vector $\mathbf{n}$ is *not* a unit vector, r instead becomes

$$r = (e_0\,|\mathbf{u}_0 \cdot \mathbf{n}| + e_1\,|\mathbf{u}_1 \cdot \mathbf{n}| + e_2\,|\mathbf{u}_2 \cdot \mathbf{n}|) \big/ \|\mathbf{n}\|\,.$$

The signed distance s of C from P is obtained by evaluating the plane equation for C, giving $s = \mathbf{n} \cdot C - d$. Another way of obtaining s is to compute the distance u of P from C, $s = -u$. Recall that P is $\mathbf{n} \cdot X = d$, where $d = Q \cdot \mathbf{n}$ for some point Q on the plane. As all points on P project to a single point on L, it is sufficient to work with the projection of Q onto L. The distance u can therefore be computed as $u = (Q - C) \cdot \mathbf{n} = Q \cdot \mathbf{n} - C \cdot \mathbf{n} = d - C \cdot \mathbf{n}$, which up to a sign change is equivalent

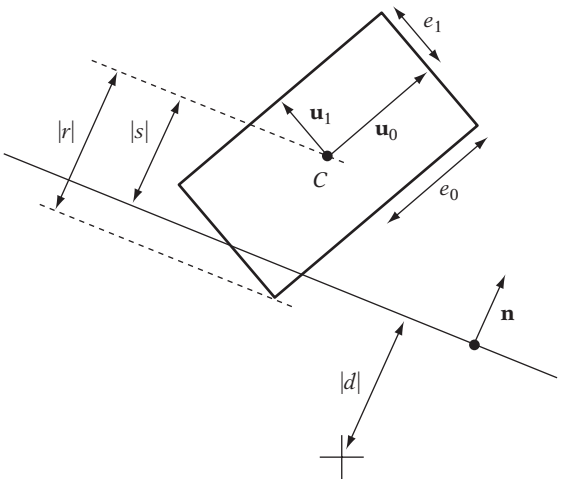

Figure 5.16 Testing intersection of an OBB against a plane.

to the evaluation of the plane equation for the box center. Figure 5.16 illustrates the quantities used in the test.

Because the intersection between the OBB B and the plane P occurs when $-r \leq s \leq r$, or equivalently when $|s| \leq r$, there is now sufficient information to implement the test.

```
// Test if OBB b intersects plane p
int TestOBBPlane(OBB b, Plane p)
{
    // Compute the projection interval radius of b onto L(t) = b.c + t * p.n
    float r = b.e[0]*Abs(Dot(p.n, b.u[0])) +
              b.e[1]*Abs(Dot(p.n, b.u[1])) +
              b.e[2]*Abs(Dot(p.n, b.u[2]));
    // Compute distance of box center from plane
    float s = Dot(p.n, b.c) - p.d;
    // Intersection occurs when distance s falls within [-r,+r] interval
    return Abs(s) <= r;
}
```

It is not necessary for $\mathbf{n}$ to be normalized for the test to work. If $\mathbf{n}$ is nonunit, both r and s will be a factor $\|\mathbf{n}\|$ larger, which does not affect the test.

Other tests can be easily implemented in a similar vein. For example, the OBB falls inside the negative halfspace of the plane if $s \leq -r$. If $r \leq s$, the OBB lies fully in the positive halfspace of the plane. For an OBB given as $B = C + k_0\mathbf{v}_0 + k_1\mathbf{v}_1 + k_2\mathbf{v}_2$,

$0 \leq k_0, k_1, k_2 \leq 1$, the radius r of the OBB is instead obtained by $r = (|\mathbf{v}_0 \cdot \mathbf{n}| + |\mathbf{v}_1 \cdot \mathbf{n}| + |\mathbf{v}_2 \cdot \mathbf{n}|)/2$. For an axis-aligned box B, the local axes $\mathbf{u}_0$, $\mathbf{u}_1$, and $\mathbf{u}_2$ are known in advance, and thus the code can be simplified accordingly.

```
// Test if AABB b intersects plane p
int TestAABBPlane(AABB b, Plane p)
{
    // These two lines not necessary with a (center, extents) AABB representation
    Point c = (b.max + b.min) * 0.5f;  // Compute AABB center
    Point e = b.max - c;  // Compute positive extents

    // Compute the projection interval radius of b onto L(t) = b.c + t * p.n
    float r = e[0]*Abs(p.n[0]) + e[1]*Abs(p.n[1]) + e[2]*Abs(p.n[2]);
    // Compute distance of box center from plane
    float s = Dot(p.n, c) - p.d;
    // Intersection occurs when distance s falls within [-r,+r] interval
    return Abs(s) <= r;
}
```

This test is equivalent to finding an AABB vertex most distant along the plane normal and making sure that vertex and the vertex diagonally opposite lie on opposite sides of the plane.

5.2.4 Testing Cone Against Plane

Let a plane be given by $(\mathbf{n} \cdot X) = d$, where $d = -P \cdot \mathbf{n}$ for a point P on the plane and $\mathbf{n}$ is unit. Let a cone be specified by its tip T, normalized axis direction $\mathbf{d}$, height h, and a bottom radius r (Figure 5.17). The cone is intersecting the negative halfspace of the plane if any point of the cone lies inside the negative halfspace; that is, if there is a point X of the cone for which $(\mathbf{n} \cdot X) < d$. For a cone only two points must be tested for this condition.

- The tip T of the cone
- The point Q on the circular endcap of the cone, farthest in the direction of $-\mathbf{n}$

For the second test, Q must be located. Thanks to the format the cone is given in, Q is easily obtained by stepping from the tip along the direction vector to the circular bottom endcap and down the endcap toward the plane:

$$Q = T + h\,\mathbf{v} + r\,\mathbf{m}.$$