3.3.8 **Algebraic Identities Involving Scalar Triple Products**

Given vectors **u**, **v**, **w**, **x**, **y**, and **z**, the following identities hold for scalar triple products:

$$[\mathbf{u\,v\,w}] = [\mathbf{v\,w\,u}] = [\mathbf{w\,u\,v}] = -[\mathbf{u\,w\,v}] = -[\mathbf{v\,u\,w}] = -[\mathbf{w\,v\,u}]$$

$$[\mathbf{u\,u\,v}] = [\mathbf{v\,u\,v}] = 0$$

$$[\mathbf{u\,v\,w}]^2 = [(\mathbf{u}\times\mathbf{v})\,(\mathbf{v}\times\mathbf{w})\,(\mathbf{w}\times\mathbf{u})]$$

$$\mathbf{u}\,[\mathbf{v\,w\,x}] - \mathbf{v}\,[\mathbf{w\,x\,u}] + \mathbf{w}\,[\mathbf{x\,u\,v}] - \mathbf{x}\,[\mathbf{u\,v\,w}] = \mathbf{0}$$

$$(\mathbf{u}\times\mathbf{v})\times(\mathbf{w}\times\mathbf{x}) = \mathbf{v}\,[\mathbf{u\,w\,x}] - \mathbf{u}\,[\mathbf{v\,w\,x}]$$

$$\left[(\mathbf{u}\times\mathbf{v})\,(\mathbf{w}\times\mathbf{x})\,(\mathbf{y}\times\mathbf{z})\right] = \left[\mathbf{v\,y\,z}\right][\mathbf{u\,w\,x}] - \left[\mathbf{u\,y\,z}\right][\mathbf{v\,w\,x}]$$

$$[(\mathbf{u}+\mathbf{v})\,(\mathbf{v}+\mathbf{w})\,(\mathbf{w}+\mathbf{u})] = 2\,[\mathbf{u\,v\,w}]$$

$$[\mathbf{u\,v\,w}]\left[\mathbf{x\,y\,z}\right] = \begin{vmatrix} \mathbf{u}\cdot\mathbf{x} & \mathbf{u}\cdot\mathbf{y} & \mathbf{u}\cdot\mathbf{z} \\ \mathbf{v}\cdot\mathbf{x} & \mathbf{v}\cdot\mathbf{y} & \mathbf{v}\cdot\mathbf{z} \\ \mathbf{w}\cdot\mathbf{x} & \mathbf{w}\cdot\mathbf{y} & \mathbf{w}\cdot\mathbf{z} \end{vmatrix}$$

$$[(\mathbf{u}-\mathbf{x})\,(\mathbf{v}-\mathbf{x})\,(\mathbf{w}-\mathbf{x})] = [\mathbf{u\,v\,w}] - [\mathbf{u\,v\,x}] - [\mathbf{u\,x\,w}] - [\mathbf{x\,v\,w}]$$

$$= [(\mathbf{u}-\mathbf{x})\,\mathbf{v}\,\mathbf{w}] - [(\mathbf{v}-\mathbf{w})\,\mathbf{x}\,\mathbf{u}]$$

3.4 **Barycentric Coordinates**

A concept useful in several different intersection tests is that of *barycentric coordinates*. Barycentric coordinates parameterize the space that can be formed as a weighted combination of a set of reference points. As a simple example of barycentric coordinates, consider two points, *A* and *B*. A point *P* on the line between them can be expressed as $P = A + t(B - A) = (1 - t)A + tB$ or simply as $P = uA + vB$, where $u + v = 1$. *P* is on the segment *AB* if and only if $0 \le u \le 1$ and $0 \le v \le 1$. Written in the latter way, (u, v) are the barycentric coordinates of *P* with respect to *A* and *B*. The barycentric coordinates of *A* are $(1, 0)$, and for *B* they are $(0, 1)$.

The prefix *bary* comes from Greek, meaning weight, and its use as a prefix is explained by considering *u* and *v* as weights placed at the endpoints *A* and *B* of the segment *AB*, respectively. Then, the point *Q* dividing the segment in the ratio *v*:*u* is the *centroid* or *barycenter*: the center of gravity of the weighted segment and the position at which it must be supported to be balanced.

A typical application of barycentric coordinates is to parameterize triangles (or the planes of the triangles). Consider a triangle *ABC* specified by three noncollinear points *A*, *B*, and *C*. Any point *P* in the plane of the points can then be uniquely expressed as $P = uA + vB + wC$ for some constants *u*, *v*, and *w*, where $u + v + w = 1$. The triplet (u, v, w) corresponds to the barycentric coordinates of the point. For the

triangle *ABC*, the barycentric coordinates of the vertices *A*, *B*, and *C* are $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$, respectively. In general, a point with barycentric coordinates (u, v, w) is inside (or on) the triangle if and only if $0 \leq u, v, w \leq 1$, or alternatively if and only if $0 \leq v \leq 1, 0 \leq w \leq 1$, and $v + w \leq 1$. That barycentric coordinates actually parameterize the plane follows from $P = uA + vB + wC$ really just being a reformulation of $P = A + v(B - A) + w(C - A)$, with v and w arbitrary, as

$$P = A + v(B - A) + w(C - A) = (1 - v - w)A + vB + wC.$$

In the latter formulation, the two independent direction vectors *AB* and *AC* form a coordinate system with origin *A*, allowing any point *P* in the plane to be parameterized in terms of v and w alone. Clearly, barycentric coordinates is a redundant representation in that the third component can be expressed in terms of the first two. It is kept for reasons of symmetry.

To solve for the barycentric coordinates, the expression $P = A + v(B - A) + w(C - A)$ — or equivalently $v(B - A) + w(C - A) = P - A$ — can be written as $v\,\mathbf{v}_0 + w\,\mathbf{v}_1 = \mathbf{v}_2$, where $\mathbf{v}_0 = B - A$, $\mathbf{v}_1 = C - A$, and $\mathbf{v}_2 = P - A$. Now, a 2×2 system of linear equations can be formed by taking the dot product of both sides with both $\mathbf{v}_0$ and $\mathbf{v}_1$:

$$(v\,\mathbf{v}_0 + w\,\mathbf{v}_1) \cdot \mathbf{v}_0 = \mathbf{v}_2 \cdot \mathbf{v}_0, \text{ and}$$

$$(v\,\mathbf{v}_0 + w\,\mathbf{v}_1) \cdot \mathbf{v}_1 = \mathbf{v}_2 \cdot \mathbf{v}_1.$$

Because the dot product is a linear operator, these expressions are equivalent to

$$v\ (\mathbf{v}_0 \cdot \mathbf{v}_0) + w\ (\mathbf{v}_1 \cdot \mathbf{v}_0) = \mathbf{v}_2 \cdot \mathbf{v}_0, \text{ and}$$

$$v\ (\mathbf{v}_0 \cdot \mathbf{v}_1) + w\ (\mathbf{v}_1 \cdot \mathbf{v}_1) = \mathbf{v}_2 \cdot \mathbf{v}_1.$$

This system is easily solved with Cramer's rule. The following code is an implementation computing the barycentric coordinates using this method.

```
// Compute barycentric coordinates (u, v, w) for
// point p with respect to triangle (a, b, c)
void Barycentric(Point a, Point b, Point c, Point p, float &u, float &v, float &w)
{
    Vector v0 = b - a, v1 = c - a, v2 = p - a;
    float d00 = Dot(v0, v0);
    float d01 = Dot(v0, v1);
    float d11 = Dot(v1, v1);
    float d20 = Dot(v2, v0);
```

```
    float d21 = Dot(v2, v1);
    float denom = d00 * d11 - d01 * d01;
    v = (d11 * d20 - d01 * d21) / denom;
    w = (d00 * d21 - d01 * d20) / denom;
    u = 1.0f - v - w;
}
```

If several points are tested against the same triangle, the terms **d00**, **d01**, **d11**, and **denom** only have to be computed once, as they are fixed for a given triangle.

The barycentric coordinates can be computed for a point with respect to a simplex (Section 3.8) of any dimension. For instance, given a tetrahedron specified by the vertices A, B, C, and D, the barycentric coordinates (u, v, w, x) specify a point P in 3D space, $P = uA + vB + wC + xD$ with $u + v + w + x = 1$. If $0 \leq u, v, w, x \leq 1$, then P is inside the tetrahedron.

Given the points specified as $A = (a_x, a_y, a_z)$, $B = (b_x, b_y, b_z)$, $C = (c_x, c_y, c_z)$, $D = (d_x, d_y, d_z)$, and $P = (p_x, p_y, p_z)$, the barycentric coordinates can be solved for by setting up a system of linear equations:

$$
\begin{array}{ccccccccc}
a_x u & + & b_x v & + & c_x w & + & d_x x & = & p_x \\
a_y u & + & b_y v & + & c_y w & + & d_y x & = & p_y \\
a_z u & + & b_z v & + & c_z w & + & d_z x & = & p_z \\
u & + & v & + & w & + & x & = & 1
\end{array}
$$

Alternatively, by subtracting A from both sides of $P = uA + vB + wC + xD$ — giving

$$P - A = v(B - A) + w(C - A) + x(D - A)$$

— it follows that three of the four barycentric coordinate components can be obtained by solving

$$(b_x - a_x)v + (c_x - a_x)w + (d_x - a_x)x = p_x - a_x,$$
$$(b_y - a_y)v + (c_y - a_y)w + (d_y - a_y)x = p_y - a_y, \text{ and}$$
$$(b_z - a_z)v + (c_z - a_z)w + (d_z - a_z)x = p_z - a_z$$

(with the fourth component given by $u = 1 - v - w - x$). Either system is easily solved using Cramer's rule or Gaussian elimination. For example, in the former system the coordinates are given by Cramer's rule as the ratios

$$u = d_{PBCD}/d_{ABCD},$$

$$v = d_{APCD}/d_{ABCD},$$

$$w = d_{ABPD}/d_{ABCD}, \text{ and}$$

$$x = d_{ABCP}/d_{ABCD} = 1 - u - v - w$$

of the following determinants:

$$d_{PBCD} = \begin{vmatrix} p_x & b_x & c_x & d_x \\ p_y & b_y & c_y & d_y \\ p_z & b_z & c_z & d_z \\ 1 & 1 & 1 & 1 \end{vmatrix}, \quad d_{APCD} = \begin{vmatrix} a_x & p_x & c_x & d_x \\ a_y & p_y & c_y & d_y \\ a_z & p_z & c_z & d_z \\ 1 & 1 & 1 & 1 \end{vmatrix}, \quad d_{ABPD} = \begin{vmatrix} a_x & b_x & p_x & d_x \\ a_y & b_y & p_y & d_y \\ a_z & b_z & p_z & d_z \\ 1 & 1 & 1 & 1 \end{vmatrix},$$

$$d_{ABCP} = \begin{vmatrix} a_x & b_x & c_x & p_x \\ a_y & b_y & c_y & p_y \\ a_z & b_z & c_z & p_z \\ 1 & 1 & 1 & 1 \end{vmatrix}, \quad \text{and} \quad d_{ABCD} = \begin{vmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \\ 1 & 1 & 1 & 1 \end{vmatrix}.$$

These determinants correspond to the signed volumes of the tetrahedra *PBCD, APCD, ABPD, ABCP,* and *ABCD* (strictly 1/6 of each signed volume). As shown further ahead, the ratios simplify to being the normalized relative heights of the point over the opposing planes.

Returning to triangles, just as the barycentric coordinates with respect to a tetrahedron can be computed as ratios of volumes the barycentric coordinates with respect to a triangle can be computed as ratios of areas. Specifically, the barycentric coordinates of a given point P can be computed as the ratios of the triangle areas of *PBC, PCA,* and *PAB* with respect to the area of the entire triangle *ABC*. For this reason barycentric coordinates are also called *areal coordinates*. By using signed triangle areas, these expressions are valid for points outside the triangle as well. The barycentric coordinates (u, v, w) are thus given by

```
u = SignedArea(PBC)/SignedArea(ABC),
v = SignedArea(PCA)/SignedArea(ABC), and
w = SignedArea(PAB)/SignedArea(ABC) = 1 - u - v.
```

Because constant factors cancel out, any function proportional to the triangle area can be used in computing these ratios. In particular, the magnitude of the cross product of two triangle edges can be used. The correct sign is maintained by taking
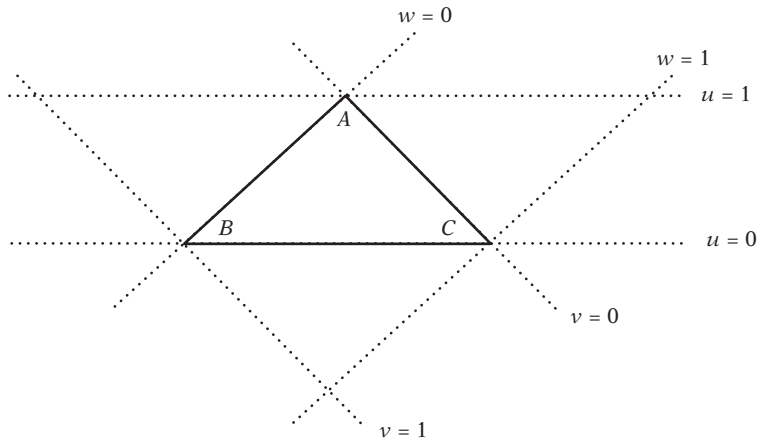
Figure 3.9 Triangle *ABC* with marked "height lines" for $u = 0$, $u = 1$, $v = 0$, $v = 1$, $w = 0$, and $w = 1$.

the dot product of this cross product with the normal of *ABC*. For instance, the signed area for the triangle *PBC* would be computed as

```
SignedArea(PBC) = Dot(Cross(B-P, C-P), Normalize(Cross(B-A, C-A))).
```

Because the area of a triangle can be written as *base · height*/2, and because for each of the previous ratios the triangles involved share the same base, the previous expressions simplify to ratios of heights. Another way of looking at barycentric coordinates is therefore as the components *u*, *v*, and *w* corresponding to the normalized height of the point *P* over each of the edges *BC*, *AC*, and *AB* relative to the height of the edge's opposing vertex. Because the triangle *ABC* is the intersection of the three 2D slabs — each slab defined by the infinite space between two parallel lines (or planes) at height 0 and height 1 of a given triangle edge — it also directly follows why $0 \leq u, v, w \leq 1$ is required for a point to be inside the triangle (Figure 3.9).

The lines coinciding with the edges of a triangle can also be seen as dividing the triangle plane in seven barycentric regions based on the signs of the barycentric coordinate components: three edge regions, three vertex regions, and the triangle interior (Figure 3.10). These regions are relevant to both mesh traversal and various containment algorithms.

An important property of barycentric coordinates is that they remain invariant under projection. This property allows for a potentially more efficient way of computing the coordinates than given earlier. Instead of computing the areas from the 3D coordinates of the vertices, the calculations can be simplified by projecting all
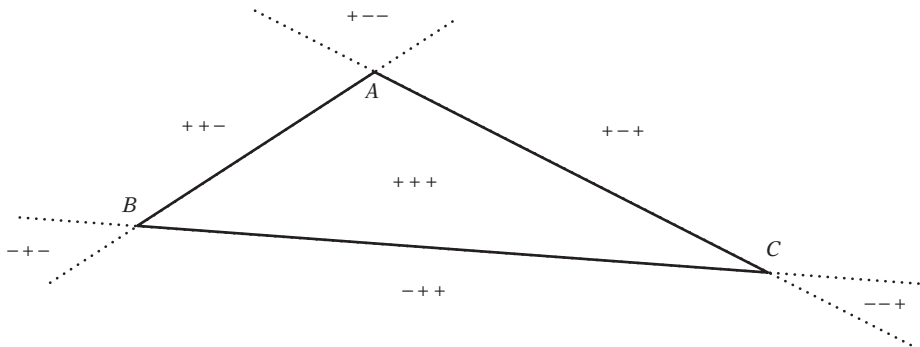
Figure 3.10 Barycentric coordinates divide the plane of the triangle *ABC* into seven regions based on the sign of the coordinate components.

vertices to the *xy, xz,* or *yz* plane. To avoid degeneracies, the projection is made to the plane where the projected areas are the greatest. The largest absolute component value of the triangle normal indicates which component should be dropped during projection.

```
inline float TriArea2D(float x1, float y1, float x2, float y2, float x3, float y3)
{
    return (x1-x2)*(y2-y3) - (x2-x3)*(y1-y2);
}

// Compute barycentric coordinates (u, v, w) for
// point p with respect to triangle (a, b, c)
void Barycentric(Point a, Point b, Point c, Point p, float &u, float &v, float &w)
{
    // Unnormalized triangle normal
    Vector m = Cross(b - a, c - a);
    // Nominators and one-over-denominator for u and v ratios
    float nu, nv, ood;
    // Absolute components for determining projection plane
    float x = Abs(m.x), y = Abs(m.y), z = Abs(m.z);

    // Compute areas in plane of largest projection
    if (x >= y && x >= z) {
        // x is largest, project to the yz plane
        nu = TriArea2D(p.y, p.z, b.y, b.z, c.y, c.z);  // Area of PBC in yz plane
        nv = TriArea2D(p.y, p.z, c.y, c.z, a.y, a.z);  // Area of PCA in yz plane
        ood = 1.0f / m.x;                              // 1/(2*area of ABC in yz plane)
```

```
    } else if (y >= x && y >= z) {
        // y is largest, project to the xz plane
        nu = TriArea2D(p.x, p.z, b.x, b.z, c.x, c.z);
        nv = TriArea2D(p.x, p.z, c.x, c.z, a.x, a.z);
        ood = 1.0f / -m.y;
    } else {
        // z is largest, project to the xy plane
        nu = TriArea2D(p.x, p.y, b.x, b.y, c.x, c.y);
        nv = TriArea2D(p.x, p.y, c.x, c.y, a.x, a.y);
        ood = 1.0f / m.z;
    }
    u = nu * ood;
    v = nv * ood;
    w = 1.0f - u - v;
}
```

Barycentric coordinates have many uses. Because they are invariant under projection, they can be used to map points between different coordinate systems. They can be used for point-in-triangle testing. Given a vertex-lit triangle, they can also find the corresponding RGB of a specific point within the triangle, which could be used to adjust the ambient color of an object at that position on the triangle. For triangle clipping, they can be used to interpolate any quantity, including colors (Gouraud shading), normals (Phong shading), and texture coordinates (texture mapping). The following code illustrates how barycentric coordinates can be used to test containment of a point P in a triangle ABC.

```
// Test if point p is contained in triangle (a, b, c)
int TestPointTriangle(Point p, Point a, Point b, Point c)
{
    float u, v, w;
    Barycentric(a, b, c, p, u, v, w);
    return v >= 0.0f && w >= 0.0f && (v + w) <= 1.0f;
}
```

A generalized form of barycentric coordinates for irregular n-sided convex polygons is given in [Meyer02]. For $n = 3$, it reduces to the traditional formula for barycentric coordinates. See also [Floater04].

When the barycentric coordinates of a point P ($P = a_0 P_0 + a_1 P_1 + \cdots + a_n P_n$, $a_0 + a_1 + \cdots a_n = 1$) with respect to the points $P_0, P_1, \ldots, P_n$ also satisfies $a_0, a_1, \ldots, a_n \geq 0$, P is said to be a *convex combination* of $P_0, P_1, \ldots, P_n$.

I. Yöntem

$$d = \frac{|(x_0 - x_1) \times (x_0 - x_2)|}{|x_2 - x_1|}$$

```
Vertex X0_X1 = iPoint - S->V0;
Vertex X0_X2 = iPoint - S->V1;
Vertex X2_X1 = S->V1  - S->V0;
float dv = (X0_X1.CrossProduct(X0_X2)).Length() / X2_X1.Length();
float  v = dv / (S->V2 - S->V0).Length();

X0_X2 = iPoint - S->V2;
X2_X1 = S->V2  - S->V0;
float du = (X0_X1.CrossProduct(X0_X2)).Length() / X2_X1.Length();
float  u = du / (S->V1 - S->V0).Length();
```

II. Yöntem

$$u = \frac{(iPoint - V_0) * (V_1 - V_0).Normalize()}{(V_1 - V_0).length()}$$

$$v = \frac{(iPoint - V_0) * (V_2 - V_0).Normalize()}{(V_2 - V_0).length()}$$

```
Vertex L    = iPoint - S->V0;
Vertex Rdu  = (S->V1 - S->V0).Normalize();
float  Su   = L * Rdu;
float  u    = Su / (S->V1 - S->V0).Length();

Vertex Rdv  = (S->V2 - S->V0).Normalize();
float  Sv   = L * Rdv;
float  v    = Sv / (S->V2 - S->V0).Length();
```

III. Yöntem

$$P = A + u(B-A) + v(C-A)$$

$$u\underbrace{(B-A)}_{e0} + v\underbrace{(C-A)}_{e1} = \underbrace{P-A}_{e2}$$

$$u*e0 + v*e1 = e2$$

$$(u*e0 + v*e1)*e0 = e2*e0$$
$$(u*e0 + v*e1)*e1 = e2*e1$$

$$u\underbrace{(e0*e0)}_{d00} + v\underbrace{(e1*e0)}_{d01} = \underbrace{e2*e0}_{d20}$$
$$u\underbrace{(e0*e1)}_{d01} + v\underbrace{(e1*e1)}_{d11} = \underbrace{e2*e1}_{d21}$$

$$v*d00*d11 + v*d01*d11 = d20*d11$$
$$u*d01*d01 + v*d11*d01 = d21*d01$$

$$u(d00*d11 - d01*d01) = d20*d11 - d21*d01$$

$$u = \frac{d20*d11 - d21*d01}{d00*d11 - d01*d01}$$

```
Vertex e0 = S->V1 - S->V0;
Vertex e1 = S->V2 - S->V0;
Vertex e2 = iPoint - S->V0;
float d00 = e0 * e0;
float d01 = e0 * e1;
float d20 = e2 * e0;
float d11 = e1 * e1;
float d21 = e2 * e1;
float u = (d20 * d11 - d21 * d01) / (d00 * d11 - d01 * d01);
float v = (d20 * d01 - d21 * d00) / (d01 * d01 - d11 * d00);
```

IV. Yöntem

$$v = \frac{S_1}{S} \qquad u = \frac{S_2}{S}$$

* bu iPoint için v'yi   * bu iPoint için u'yu

```
float v = ((S->V1 - S->V0).CrossProduct(iPoint - S->V0)).Length() / ((S->V1 - S->V0).CrossProduct(S->V2 - S->V0)).Length();
float u = ((S->V2 - S->V0).CrossProduct(iPoint - S->V0)).Length() / ((S->V1 - S->V0).CrossProduct(S->V2 - S->V0)).Length();
```