Lecture 19: Surfaces II -- Simple Surfaces

He hath put down the mighty from their seats, and exalted them of low degree. Luke 1:52

1. Simple Surfaces

Some surfaces are easy to describe without equations. A *sphere* is the locus of points at a fixed distance from a given point; a *cylinder* is the locus of points at a fixed distance from a given line. Thus a sphere can be represented simply by a center point and a radius; a cylinder by an axis line and a radius. For these simple surfaces, we shall see that we can compute surface normals and ray-surface intersections directly from their geometry, without resorting to implicit or parametric equations. Thus ray tracing these surfaces is particularly easy.

In this lecture we will investigate ray tracing strategies for surfaces defined geometrically rather than algebraically. These surfaces include the plane and the natural quadrics -- the sphere, the right circular cylinder, and the right circular cone. We shall also explore ray tracing for general quadric (second degree) surfaces as well as for the torus, which is a surface of degree four. General surfaces of revolution will also be investigated here. These surfaces are among the simplest and most common surfaces in Computer Graphics, so they deserve special attention.

2. Intersection Strategies

Special strategies exist for ray tracing simple surfaces. One of the most common tactics is to apply a transformation to reduce the ray-surface intersection problem to a simpler problem either by projecting to a lower dimension, or by repositioning to a canonical location, or by rescaling to a symmetric shape.

Consider intersecting a curve C and a surface S. If a point P lies on the intersection of the curve C and the surface S, then for any affine or projective transformation M the point $P*M$ lies on the intersection of the curve $C*M$ and the surface $S*M$. If the transformation M is cleverly chosen, in many cases it is easier to intersect the curve $C*M$ with the surface $S*M$ rather than the original curve C with the original surface S. Moreover, if the transformation M is invertible, then

$$P \in C \cap S \;\Leftrightarrow\; P*M \in C*M \cap S*M.$$

Therefore, when M is invertible, we can find the intersection points of the the original curve C and the original surface S by computing the intersection points of the transformed curve $C*M$ with the transformed surface $S*M$ and then mapping back by M^{-1}. This method is essentially the deformation technique discussed in the previous lecture, though M may also be a rigid motion as well as a shear or a scale.

But what if the transformation M is not invertible? For example, what if M is an orthogonal or perspective projection? Remarkably, for parametric curves and surfaces the same strategy can still work.

Consider intersecting a parametric curve $C = C(t)$ and a parametric surface $S = S(u,v)$. The curve C and the surface S intersect whenever there are parameter values t^*, u^*, v^* for which

$$C(t^*) = S(u^*,v^*).$$

Applying an affine or projective transformation M yields

$$C(t^*) * M = S(u^*,v^*) * M.$$

Thus if the original curve and surface intersect at the parameters t^*, u^*, v^*, so will the transformed curve and surface. The intersection points may change, but the parameter values are unaffected. This observation is the key to our intersection strategy.

If no new intersection points are introduced by applying the transformation M, then we can find the intersection points of the original curve and surface by solving for the parameters t^* or (u^*,v^*) of the intersection points of the transformed curve and surface and then substituting these parameter values back into the original parametric equations $C = C(t^*)$ or $S = S(u^*,v^*)$ to find the intersection points of the original curve and surface.

Moreover, even if we do introduce some new spurious intersection points by applying a singular transformation, we never lose any of the old intersection points. So, at worst, we would need to check whether the parameters representing intersection points on the transformed curve and surface also represent intersection points on the original curve and surface. But we can always verify potential intersections by substituting the intersection parameters back into the parametric equations of the original curve and surface and checking whether or not we get the same point on the curve and the surface -- that is, by verifying that $C(t^*) = S(u^*,v^*)$. We shall see many examples of this general transformation strategy for computing intersections throughout this lecture.

3. Planes and Polygons

A plane can be described by a single point Q on the plane and a unit vector N normal to the plane. A point P lies on the plane if and only if the vector $P - Q$ is perpendicular to the normal vector N or equivalently if and only if $N \bullet (P - Q) = 0$.

A line $L(t) = P + tv$ intersects the plane determined by the point Q and the normal vector N at the parameter value t^*, where

$$N \bullet \left(L(t^*) - Q\right) = N \bullet \left(P + t^* v - Q\right) = 0.$$

If $N \bullet v = 0$, then the line is parallel to the plane, so the line and plane do not intersect. Otherwise, solving for the parameter t^* yields

$$t^* = \frac{N \bullet (Q - P)}{N \bullet v}. \tag{3.1}$$

Substituting the parameter t^* back into the equation of the line, we find the intersection point

$$R = L(t^*) = P + t^* v. \tag{3.2}$$

Thus we have a simple algorithm to intersect a line with an infinite plane.

Often, however, we want to find the intersection of a line with a finite polygon, rather than with an infinite plane. For example, when we ray trace a cube, the faces of the cube are squares, not infinite planes. Even if a line intersects the plane containing the polygon, the line might not intersect the plane at a point inside the polygon.

For convex polygons like triangles and rectangles, there is a straightforward test for determining if a point R on a plane of the polygon lies inside the polygon. Consider a closed convex polygon with vertices P_i, $i = 1, \ldots, m+1$, where $P_{m+1} = P_1$. Let $N_{i,i+1}$ be the vector normal to the edge $P_i P_{i+1}$ and pointing into the polygon (see Figure 1).

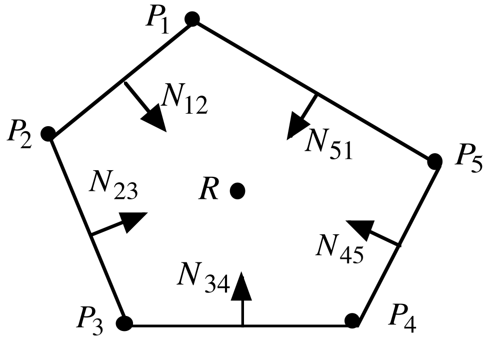


Figure 1: A point R in the plane of a convex polygon lies inside the polygon if and only if R lies on the positive side of the inward pointing normal vector to each boundary line.

A point R in the plane of the polygon lies inside the polygon if and only if

$$N_{i,i+1} \bullet (R - P_i) \ge 0 \qquad i = 1, \ldots, m. \tag{3.3}$$

To find the normal vectors $N_{i,i+1}$, observe that if the vertices of the polygon are oriented counterclockwise with respect to the normal N to the plane of the polygon, then

$$N_{i,i+1} = N \times (P_{i+1} - P_i) \qquad i = 1, \ldots, m$$

and Equation (3.3) reduces to

$$\det(N,\ P_{i+1} - P_i,\ R - P_i) \ge 0 \qquad i = 1, \ldots, m; \tag{3.4}$$

if the vertices are oriented clockwise with respect to the normal N, then

$$N_{i,i+1} = (P_{i+1} - P_i) \times N \qquad i = 1,\ldots,m$$

and Equation (3.3) reduces to

$$\det\left(R - P_i,\ P_{i+1} - P_i,\ N\right) \geq 0 \qquad i = 1,\ldots,m. \tag{3.5}$$

The tests in Equations (3.3)-(3.5) work only for convex polygons. For arbitrary, possibly non-convex, planar polygons there are more general inside-outside tests based either on ray casting or on winding numbers. The details of these tests are provided in Lecture 11, Section 7. In most cases, however, we shall be interesting only in convex polygons, so we shall not generally require these more sophisticated tests.

4. Natural Quadrics

The *natural quadrics* consist of the sphere, the right circular cylinder, and the right circular cone. These three second degree surfaces are among the most common surfaces in Computer Graphics, so we shall develop special ray tracing algorithms for each of these surfaces. Moreover, for these three surfaces, we shall reduce the ray-surface intersection computation to a single algorithm for computing the intersection of a line and a circle.

4.1 Spheres. Next to planes, spheres are the simplest and most common surfaces in Computer Graphics. Therefore we would like to have a simple way to represent spheres as well as a fast, robust ray-sphere intersection algorithm.

A sphere S can be represented simply by a center point C and a scalar radius R. The vector N from the center point C to a point P on the surface of the sphere S is normal to the surface of the sphere at the point P. Therefore the outward pointing normal vector N at the point P is given by

$$N_{sphere} = P - C.$$

To simplify the ray-sphere intersection calculation, we shall reduce the 3-dimensional ray-sphere intersection problem to a 2-dimensional ray-circle intersection problem. Suppose we want to intersect the line L with the sphere S. Consider the plane containing the center point C and the line L (see Figure 2). This plane intersects the sphere S is a circle with center C and radius R, and the line L intersects this circle in the same points that the line L intersects the sphere.

4.1.1 Intersecting a Line and a Circle. To find the intersection points of a line and a circle, we can take a simple geometric approach. Let the line L be determined by a point P and a direction vector v. First we check the distance from the center point C to the line L. If this distance is greater than the radius R, then there is no intersection between the line and the circle. Thus we have a fast test for rejection. Otherwise, we proceed by first finding the point Q on the line L closest to the center point C. We then use the Pythagorean theorem to calculate the distance from Q to the intersection points A,B of the line and the circle. Below is pseudocode for this Line-Circle intersection algorithm.

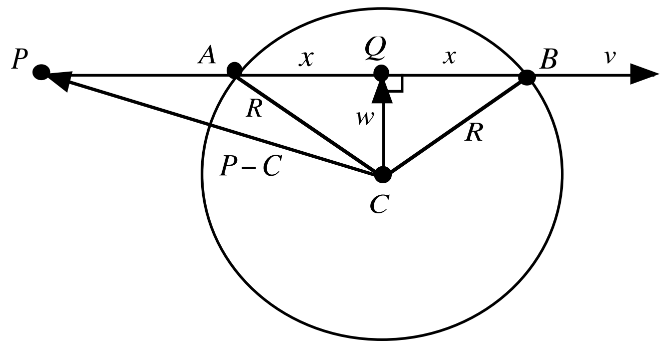


Figure 2: A line determined by a point P and a vector v intersecting a circle with center C and radius R.

Line-Circle Intersection Algorithm

```
Input
    $C$ = Circle Center
    $R$ = Circle Radius
    $P$ = Point on the Line
    $v$ = Vector in the Direction of the Line
Algorithm
    If $Dist^2(C,L) > R^2$, there are no intersection points (fast reject)
    Otherwise
        Normalize the direction vector $v$ to a unit vector
            $v \to \frac{v}{|v|}$
        Find the orthogonal projection $w$ of the vector $P-C$ on the line vector $v$
            $w = (P-C)_\perp = (P-C) - \big((P-C) \bullet v\big)v$
        Find the point $Q$ on the line $L$ closest to the center point $C$
            $Q = C + w$
        Using the Pythagorean theorem, find the distance $x$ from the point $Q$ to the
        intersection points $A,B$.
            $x^2 = R^2 - |w|^2$
        Compute the intersection points $A,B$
            $A = Q - x\,v$
            $B = Q + x\,v$
```

4.1.2 Inversion Formulas for the Line. The preceding algorithm finds the intersection points of a line and a circle or equivalently a line and a sphere. But in ray tracing we also need to know the parameter values along the line corresponding to these intersection points. A formula for finding the parameter value corresponding to a point along a line is called an *inversion formula* for the line. The exact form of the inversion formula depends on the particular form of the parametric equation of the line.

Consider, for example, a line L determined by a point P and a vector v; then $L(t) = P + t\,v$. If Q is a point on the line $L(t)$, then for some value t

$$Q = P + t\,v.$$

Subtracting P from both sides, dotting with v, and solving for t yields:

Inversion Formula (Linear Parametrization)

$$t = \frac{(Q - P) \bullet v}{v \bullet v} \tag{4.1}$$

When v is a unit vector, then $v \bullet v = 1$ and this inversion formula simplifies to:

Inversion Formula (Linear Parametrization with Unit Vector v)

$$t = (Q - P) \bullet v \tag{4.2}$$

If the line L is determined by two points P_0, P_1 rather than a point P and a vector v, then

$$L(t) = (1 - t)P_0 + t\,P_1 = P_0 + t(P_1 - P_0).$$

Therefore to find the parameter value corresponding to a point Q on the line $L(t)$, we can use the first inversion formula (Equation (4.1)) with $v = P_1 - P_0$.

Suppose, however, that L is the image under perspective projection of a line in 3-space. Perspective projection maps points and vectors into mass-points. Therefore if the original line is represented either by a point and a vector or by two points, then the image L is represented by two mass-points (X_0, m_0) and (X_1, m_1). Here if $m_k \neq 0$, then $X_k = m_k P_k$, where P_k is a point in affine space, whereas if $m_k = 0$, then $X_k = v_k$, where v_k is a vector. In either case, since L is determined by mass-points, to find the affine points on the projected line $P + t\,v \to (X_0, m_0) + t(X_1, m_1)$, we must divide by the mass. Thus L has a rational linear parametrization

$$L(t) = \frac{X_0 + t\,X_1}{m_0 + t\,m_1}.$$

To find an inversion formula for a line $L(t)$ with a rational linear parametrization, let Q be a point on the line $L(t)$. Then for some value of t

$$Q = \frac{X_0 + t X_1}{m_0 + t m_1}.$$

Multiplying both sides by $m_0 + t m_1$ yields

$$(m_0 + t m_1)Q = X_0 + t X_1$$

or equivalently

$$(m_0 Q - X_0) = t(X_1 - m_1 Q).$$

Dotting both sides with $X_1 - m_1 Q$, and solving for t, we arrive at:

Inversion Formula (Rational Linear Parametrization)

$$t = \frac{(m_0 Q - X_0) \bullet (X_1 - m_1 Q)}{(X_1 - m_1 Q) \bullet (X_1 - m_1 Q)}. \tag{4.3}$$

If $m_0, m_1 \neq 0$, then $(X_k, m_k) = (m_k P_k, m_k)$ and this formula reduces to:

Inversion Formula (Rational Linear Parametrization with $m_0, m_1 \neq 0$)

$$t = \frac{m_0 (Q - P_0) \bullet (P_1 - Q)}{m_1 (P_1 - Q) \bullet (P_1 - Q)} \tag{4.4}$$

We shall have occasion to apply these inversion formulas for rational linear parametrizations of the line in Section 4.3 when we ray trace the cone.

4.2 Cylinders. A sphere is the locus of points equidistant from a given point; a cylinder is the locus of points equidistant from a given line. Thus a cylinder can be represented simply by an axis line L and a scalar radius R. Typically the axis line L is itself represented by a point Q on the line and a unit direction vector A parallel to the line (see Figure 3).

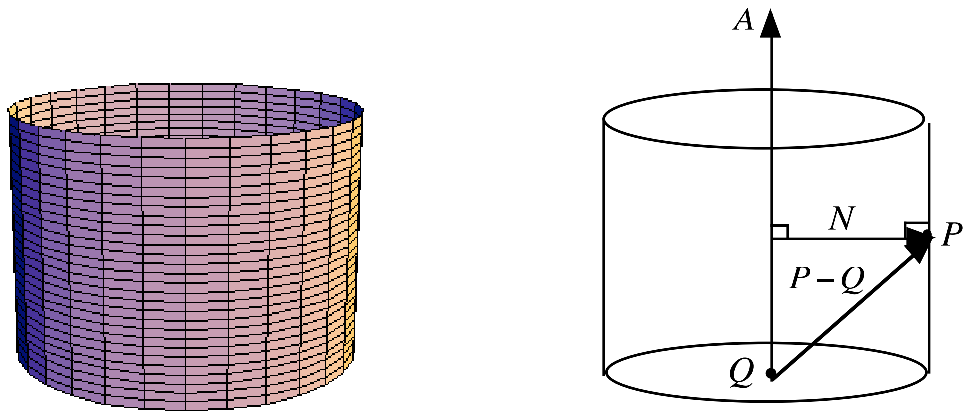


Figure 3: A right circular cylinder with an axis line L determined by a point Q and a unit direction vector A. The normal vector N to the surface of the cylinder at the point P is the orthogonal component of the vector $P - Q$ with respect to the axis vector A.

The normal N to the cylinder at a point P on the surface is just the orthogonal component of the vector $P-Q$ with respect to the axis vector A (see Figure 3). Therefore,

$$N_{cylinder} = (P-Q)_{\perp} = (P-Q) - \big((P-Q) \bullet A\big)A \, .$$

4.2.1 Intersecting a Line and an Infinite Cylinder. As with the sphere, we can reduce the ray-cylinder intersection problem to a ray-circle intersection problem. The strategy here is to project both the cylinder and the line orthogonally onto the plane determined by the point Q and the axis vector A. The cylinder projects to the circle with center Q and radius R; the line projects to another line. Now we can use the algorithm in the previous section to intersect the projected line with the circle. However, orthogonal projection is not an invertible map, so we cannot use the inverse map to recover the actual intersection points of the line and the cylinder.

What saves us here is our observation in Section 2 that the parameters of the intersection points of the projected line and the projected cylinder (the circle) are the same as the parameters of the intersection points of the unprojected line and the unprojected cylinder.

Thus to intersect a line and a cylinder, we can proceed in the following manner. Let the line L be determined by a point P and a unit direction vector v. If the distance from the cylinder axis to the line L is greater than the cylinder radius R or if the direction of the line vector v is parallel to the direction of the cylinder axis vector A, then there is no intersection between the line and the cylinder. Thus, once again, we have fast tests for rejection. If the line and the cylinder do intersect, then we project the line L orthogonally into the plane determined by the point Q and the axis vector A, and we intersect the projected line with the projection of the cylinder, the circle with center Q and radius R. We then apply the inversion formula for the projected line to find the corresponding parameter values. Finally, we substitute these parameters values into the equation of the original line L to find the intersection points of the line and the cylinder. Below is pseudocode for this Line-Cylinder intersection algorithm. Notice that this algorithm computes both the intersection points and the parameter values along the line corresponding to these intersection points.

Line–Cylinder Intersection Algorithm (Infinite Cylinder)

```
Input
    $Q$ = Point on the Cylinder Axis
    $A$ = Unit Direction Vector Parallel to the Cylinder Axis
    $R$ = Cylinder Radius
    $P$ = Point on the Line $L$
    $v$ = Unit Vector in the Direction of the Line $L$
Algorithm
    If $Dist(Axis\ Line, L) > R$ or $v \parallel A$ ($|v \bullet A| \approx 1$), there are no intersection points (fast
    reject).
```

```
Otherwise
    Project the line $L$ into the plane determined by the point $Q$ and the axis vector $A$, and
    find the equation of the projected line $L_\perp$.
        $L_\perp(t) = P_\perp + t v_\perp$
        • $v_\perp = v - (v \bullet A) A$
        • $P_\perp = P - \big((P - Q) \bullet A\big) A$
    Intersect the projected line $L_\perp$ with the circle with center $Q$ and radius $R$.
        Apply the line-circle intersection algorithm in Section 4.1.1 to find the
        intersection points $D_1, D_2$.
    Use the inversion formula in Section 4.1.2 for the line $L_\perp(t) = P_\perp + t v_\perp$ (or
    alternatively apply Exercise 8) to find the corresponding parameter values $t = t_1, t_2$.
        $t_i = \frac{(P_\perp - D_i) \bullet v_\perp}{v_\perp \bullet v_\perp} = \frac{(P - D_i) \bullet v_\perp}{v \bullet v_\perp} \qquad i = 1,2.$
    Substitute the parameters values $t = t_1, t_2$ into the equation of the line $L$ to find the
    intersection points $R_1, R_2$ of the line and the cylinder.
        $R_i = L(t_i) = P + t_i v \qquad i = 1,2.$
```

4.2.2 Intersecting a Line and a Bounded Cylinder. The preceding algorithm intersects a line with an infinite cylinder, but most of the cylinders that appear in Computer Graphics are closed and bounded. Thus additional computations are needed to find the intersections of the ray with the disks at the top and the bottom of the cylinder. Fortunately these computations are easy to perform:

To intersect a line with a bounded cylinder, first intersect the line with the planes of the two bounding disks; then test to see whether or not these intersections points lie within a distance R of the centers of these disks. If both intersection points lie inside the bounding disks, then the line cannot intersect the surface of the bounded cylinder; otherwise use the algorithm in the preceding section to find the intersection points of the line with the infinite cylinder and discard those intersection points that are not within the bounds of the finite cylinder. Below is pseudocode the Line-Cylinder intersection algorithm for bounded cylinders.

Line–Cylinder Intersection Algorithm (Bounded Cylinder)

```
Input
    $Q$ = Point on the Cylinder Axis at the Base of the Cylinder
    $A$ = Unit Direction Vector Parallel to the Cylinder Axis
    $R$ = Cylinder Radius
    $H$ = Cylinder Height
    $P$ = Point on the Line $L$
    $v$ = Unit Vector in the Direction of the Line $L$
```

Algorithm

If the line L is parallel to the plane determined by the point Q and the unit normal vector A -- that is, if $v \bullet A \approx 0$ -- then:

If the distance from the line L to the plane determined by the point $Q + (H/2)A$ and the normal vector A is greater than $H/2$, then there are no intersections between the line L and the bounded cylinder.

If $\left|(P - (Q + (H/2)A)) \bullet A\right| > H/2$, then the line L does not intersect the cylinder.

Otherwise use the algorithm in Section 4.2.1 to intersect the line L with the infinite cylinder and store the results.

Otherwise intersect the line L with the planes of the two disks bounding the cylinder.

Use the results in Section 3 to find the intersection points E_1, E_2 of the line L and the the two planes determined by the points Q and $Q + H\,A$ and the unit normal vector A.

If the distance of these intersection points E_1, E_2 from the centers $Q, Q + H\,A$ of the corresponding disks are less than the radius of the cylinder, then save these intersection points and stop.

Test: $Dist^2(Q, E_1) < R^2$ and $Dist^2(Q + H\,A, E_2) < R^2$

Otherwise intersect the line $E(t) = E_1 + t(E_2 - E_1)$ with the infinite cylinder.

Use the algorithm in Section 4.2.1 for intersecting a line and an infinite cylinder to find the intersection points D_1, D_2 and the corresponding parameter values t_1, t_2.

Discard any intersection point D_i with parameter value $t_i < 0$ or $t_i > 1$, $i = 1,2$, since any intersections of the line L with the bounded cylinder must lie between E_1 and E_2.

Now there are two cases:

Case 1: *Both points E_1, E_2 lie outside the corresponding disks.*

Either both D_1 and D_2 lie on the bounded cylinder or the line L does not intersect the bounded cylinder.

Case 2: *One of the points E_1 or E_2 lies inside the corresponding disk and the other point lies outside the corresponding disk.*

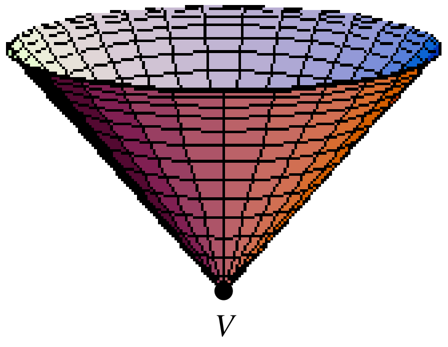
Exactly one of the points D_1 or D_2 lies on the bounded cylinder and one of the points E_1 or E_2 lies on one of the bounding disks.

Notice that in both case we need to test only one of the parameters t_1, t_2 to see if the parameter lies in the interval $[0,1]$. Whether we will need to keep or discard the other parameter is completely determined by the case; no further testing is required.

4.3 Cones. A cone can be represented simply by a vertex V, a unit axis vector A, and a cone angle α (see Figure 4). Notice that a cylinder is a special case of a cone, where the vertex V is located at infinity.

The normal N to the cone at a point P on the surface is the sum of a component along the cone axis A and a component along the vector $P - V$ from the cone vertex V to the point P (see Figure 4). By straightforward trigonometry, the length of component of N along A is $\frac{|P-V|}{cos(\alpha)}$. Therefore,

$$N_{cone} = (P - V) - \left\{\frac{|P-V|}{cos(\alpha)}\right\} A .$$



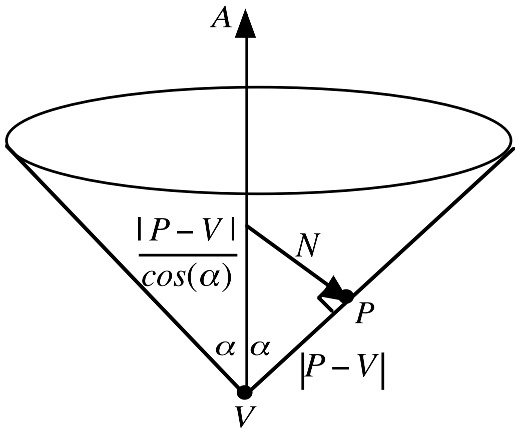


Figure 4: A right circular cone with a vertex V, a unit axis vector A, and a cone angle α. The normal vector N to the surface of the cone at the point P can be decomposed into the sum of two components: one along the axis of the cone and the other along the vector from the cone vertex V to the point P. By trigonometry, the length of component of N along A is $|P - V| / cos(\alpha)$.

As with the cylinder, we can reduce the ray-cone intersection to a ray-circle intersection by projecting both the cone and the line onto the plane determined by the point $Q = V + A$ and the axis vector A. Here, however, instead of using orthogonal projection, we use perspective projection from the vertex of the cone. Under perspective projection, the cone projects to the circle with center Q and, since $Dist(V,Q) = 1$, with radius $R = Tan(\alpha)$. The line projects to another line, but since perspective projection introduces mass, the projected line has a rational linear parametrization (see Section 4.1.2). Nevertheless, we can now proceed much as we did in the ray-cylinder intersection algorithm. We intersect the projected line with the circle to find the parameter values corresponding to the intersection points, and we substitute these parameter values back into the equation of the original line to find the intersection points of the line with the cone. Below is pseudocode for this Line-Cone intersection algorithm. As usual this algorithm computes both the intersection points and the parameter values along the line corresponding to these intersection points.

Line–Cone Intersection Algorithm (Double Infinite Cone)

```
Input
    $V$ = Cone Vertex
```

A = Unit Direction Vector Parallel to the Cone Axis

α = Cone Angle

P = Point on the Line L

u = Unit Vector in the Direction of the Line L

Algorithm

Project the line L into the plane determined by the point $Q = V + A$ and the normal vector A, and find the rational parametrization of the projected line $L^*(t)$.

$$L^*(t) = \frac{X_0 + tX_1}{m_0 + tm_1}$$

$$X_0 = \big((V - Q) \bullet A\big)P + \big((Q - P) \bullet A\big)V = -P + \big((Q - P) \bullet A\big)V$$

$$m_0 = (V - P) \bullet A = (Q - P) \bullet A - 1$$

$$X_1 = \big((V - Q) \bullet A\big)u - \big(u \bullet A\big)V = -\big(u + (u \bullet A)V\big)$$

$$m_1 = -u \bullet A$$

Intersect the projected line L^* with the circle with center $Q = V + A$ and radius $R = Tan(\alpha)$.

Apply the line-circle intersection algorithm to find the intersection points D_1, D_2.

There are four cases to consider, depending on the masses m_0, m_1:

Case 1: $m_0 = m_1 = 0$. The line L projects to the line at infinity, so there are no intersection points between the circle and the line $L^*(t)$.

Case 2: $m_0, m_1 \neq 0$. The line $L^*(t)$ is determined by the two affine points $\frac{X_0}{m_0}, \frac{X_1}{m_1}$ or equivalently by the point $P = \frac{X_0}{m_0}$ and the vector $v = \frac{X_0}{m_0} - \frac{X_1}{m_1}$.

Case 3: $m_0 \neq 0,\ m_1 = 0$. The line $L^*(t)$ is determined by the affine point $P = \frac{X_0}{m_0}$ and the vector $v = X_1$.

Case 4: $m_0 = 0,\ m_1 \neq 0$. The line $L^*(t)$ is determined by the affine point $P = \frac{X_1}{m_1}$ and the vector $v = X_0$.

Use the inversion formula for the rational linear parametrization of the line $L^*(t)$ (or alternatively apply Exercise 8) to find the corresponding parameter values $t = t_1, t_2$:

$$t = \frac{(m_0 D_i - X_0) \bullet (X_1 - m_1 D_i)}{(X_1 - m_1 D_i) \bullet (X_1 - m_1 D_i)} \qquad i = 1,2.$$

Substitute the parameters values $t = t_1, t_2$ into the equation of the line L to find the intersection points R_1, R_2 of the line and the cone.

$$R_i = L(t_i) = P + t_i v \qquad i = 1,2.$$

This algorithm intersects a line with a double infinite cone. To intersect a line with a finite cone bounded by two disks (or one disk and the cone vertex), proceed as in the algorithm for intersecting a line with a bounded cylinder: first intersect the line with the planes of the two bounding disks; then test to see whether or not these intersections points lie inside these disks. If both intersection points lie inside the bounding disks, then the line cannot intersect the surface of the finite cone; otherwise use the preceding algorithm to find the intersection points of the line with the infinite cone and discard those intersections that are not within the bounds of the finite cone. Details are left as an exercise for the reader (see Exercise 3).

4.4 Ellipsoids, Elliptical Cylinders, and Elliptical Cones. The ellipsoid is a scaled sphere; the elliptical cylinder and the elliptical cone are scaled variants of the right circular cylinder and right circular cone. One simple strategy for ray tracing these elliptical surfaces is to apply non-uniform scaling to transform these surfaces to the corresponding natural quadrics, perform the surface normal and ray-surface intersection calculations on these natural quadrics, and then map the normal vectors and the ray-surface intersection points back to the elliptical quadrics.

For example, the ellipsoid can be represented by a center point C, three orthogonal unit axis vectors u,v,w, and three scalar lengths a,b,c. Scaling non-uniformly about the center point C by a in the u direction, b in the v direction, and c in the w direction maps the sphere with center C and unit radius to the ellipsoid (see Figure 5). The inverse transformation that maps the ellipsoid to the unit sphere scales non-uniformly about the center point C by $1/a$ in the u direction, $1/b$ in the v direction, and $1/c$ in the w direction.

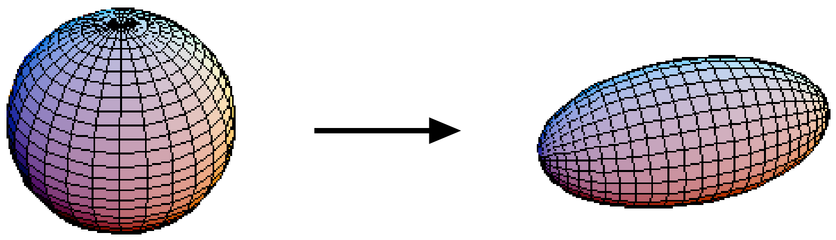


Figure 5: The ellipsoid is the image of the sphere under non-uniform scaling.

To find the normal vector to the ellipsoid at an arbitrary point, let M be the non-uniform scaling transformation that maps the sphere to the ellipsoid, and let N_E, N_S denote the normal vectors to the ellipsoid and the sphere at corresponding points. We know for any point P on the sphere how to compute $N_S = P - C$. Now we can easily compute N_E, since by the results in Lecture 18,

$$N_E = N_S * M_u^{-T},$$

where M_u is the upper 3×3 submatrix of M.

Intersection points between a ray L and an ellipsoid E are similarly easy to compute. Simply intersect the ray $L \bullet M^{-1}$ with the unit sphere centered at C using the algorithm in Section 4.1. If R_1, R_2 are the intersection points of $L \bullet M^{-1}$ with the unit sphere, then $R_1 * M, R_2 * M$ are the intersection points of L with the ellipsoid; note that the parameter values of the intersection points are the same for the sphere and the ellipsoid.

Analogous strategies can be applied to find the normal vectors and the ray-surface intersection points for the elliptical cylinder and the elliptical cone by applying non-uniform scaling transformations to the right circular cylinder and right circular cone. We leave the details as simple exercises for the reader (see Exercises 2,4).

5. General Quadric Surfaces

A general quadric surface is the collection of points satisfying a second degree equation in x, y, z -- that is, an implicit equation of the form

$$Q(x,y,z) \equiv Ax^2 + By^2 + Cz^2 + 2Dxy + 2Eyz + 2Fxz + 2Gx + 2Hy + 2Iz + J = 0.$$

We can rewrite this equation in matrix notation. Let

$$P = (x,y,z,1)$$

$$Q = \begin{pmatrix} A & D & F & G \\ D & B & E & H \\ F & E & C & I \\ G & H & I & J \end{pmatrix}.$$

Then it is easy to verify that

$$P * Q * P^T = Ax^2 + By^2 + Cz^2 + 2Dxy + 2Eyz + 2Fxz + 2Gx + 2Hy + 2Iz + J.$$

Thus in matrix notation, the equation of a quadric surface is

$$P * Q * P^T = 0.$$

For this reason, we will typically represent quadric surfaces by symmetric 4×4 matrices Q.

The normal vector to an implicit surface is parallel to the gradient. For a quadric surface

$$Q(x,y,z) = Ax^2 + By^2 + Cz^2 + 2Dxy + 2Eyz + 2Fxz + 2Gx + 2Hy + 2Iz + J$$

the gradient

$$\nabla Q(x,y,z) = 2(Ax + Dy + Fz + G,\ Dx + By + Ez + H, Fx + Ey + Cz + I).$$

Therefore the normal vector

$$N(x,y,z) \parallel (Ax + Dy + Fz + G,\ Dx + By + Ez + H, Fx + Ey + Cz + I).$$

Equivalently in matrix notation,

$$N(x,y,z) \parallel P * Q_{4\times 3}, \tag{5.1}$$

where $Q_{4\times 3}$ consists of the first three columns of Q.

To intersect a line $L(t) = P + tv$ with a quadric surface Q, we can use the quadratic formula to solve the second degree equation

$$L(t) * Q * L(t)^T = 0.$$

The real roots of this equation represent the parameter values along the line $L(t)$ of the intersection points. The actual intersection points can be computed by substituting these parameter values into the parametric equation $L(t) = P + tv$ for the line.

Affine and projective transformations map quadric surfaces to quadric surfaces. To find the image of a quadric surface Q under a nonsingular transformation matrix M, recall that for any implicit surface $F(P) = 0$, the equation of the transformed surface is

$$F_{new}(P) = F(P * M^{-1}). \tag{5.2}$$

For a quadric surface represented by a symmetric 4×4 matrix Q, Equation (5.2) becomes

$$P * Q_{new} * P^T = (P * M^{-1}) * Q * (P * M^{-1})^T = P * (M^{-1} * Q * M^{-T}) * P.$$

Therefore

$$Q_{new} = M^{-1} * Q * M^{-T} \tag{5.3}$$

is the matrix representation of the transformed surface.

There are many different types of quadric surfaces. In addition the the natural quadrics -- the sphere, the right circular cylinder, and the right circular cone -- and their elliptical variants, there are also parabolic and hyperbolic cylinders, elliptical and hyperbolic paraboloids, and hyperboloids of one or two sheets. In Table 1, we list the equations of the different types of quadric surfaces in canonical position -- that is, with center at the origin and axes along the coordinate axes.

Special ray tracing algorithms for the natural quadrics and their elliptical variants are presented in Section 4. The reason that these special ray tracing techniques are effective is that it is easy to represent the natural quadrics in arbitrary positions geometrically without resorting to equations. But it is not so natural to represent general quadric surfaces such as paraboloids and hyperboloids in arbitrary positions geometrically without resorting to equations, and even if such geometric representations were feasible, straightforward intersection algorithms are not readily available. Rather than start with a quadric surface in general position, typically one starts with a quadric surface in canonical position and then repositions the surface by applying a rigid motion M.

Now there are two ways we can proceed to ray trace a quadric surface in general position. We can compute the equation of the surface and the surface normal in general position from the equation of the surface in canonical position using Equations (5.1) and (5.3). We can then solve a general second degree equation to find the intersection of the ray and the quadric surface. Alternatively, if we store the transformation matrix M along with the equation of the surface in canonical position, then we can compute the normal to the surface in arbitrary position from the normal to the surface in canonical position from the formula

$$N_{new} = N_{old} * M_u^{-T} .$$

The normal N_{old} to the surface in canonical position is parallel to the gradient, and this gradient is easy to compute because the canonical equation is simple. Similarly, to intersect a line L with a quadric surface in general position, we can intersect the line $L * M^{-1}$ with the quadric surface in canonical position and then map these intersection points back to the desired intersection points by the transformation M. Again since the equation of a quadric in canonical position is simple, the quadratic equation for the intersection of a line and a quadric in canonical position is also simple.

Sphere $x^2 + y^2 + z^2 - R^2 = 0$	*Right Circular Cylinder* $x^2 + y^2 - R^2 = 0$	*Right Circular Cone* $x^2 + y^2 - c^2 z^2 = 0$
Ellipsoid $\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0$	*Elliptical Cylinder* $\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0$	*Elliptical Cone* $\frac{x^2}{a^2} + \frac{y^2}{b^2} - z^2 = 0$
Parabolic Cylinder $x^2 - 4py = 0$	*Hyperbolic Cylinder* $\frac{x^2}{a^2} - \frac{y^2}{b^2} - 1 = 0$	*Elliptical Paraboloid* $\frac{x^2}{a^2} + \frac{y^2}{b^2} - z = 0$
Hyperbolic Paraboloid $\frac{x^2}{a^2} - \frac{y^2}{b^2} - z = 0$	*Hyperboloid of One Sheet* $\frac{x^2}{a^2} - \frac{y^2}{b^2} - \frac{z^2}{c^2} - 1 = 0$	*Hyperboloid of Two Sheets* $\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} - 1 = 0$

Table 1: Equations of quadric surfaces in canonical position.

6. Tori

The torus is the next most common surface in Computer Graphics, after the plane and the natural quadrics. A torus can be represented geometrically by a center point C, an axis vector A, and two scalar radii d,a (see Figure 6). Cutting the torus by the plane through C orthogonal to A generates two concentric circles with center C on the surface of the torus. The scalar d represents the radius of a third concentric circle midway between these two circles -- that is, a circle inside the torus. We call this circle the *generator* of the torus. The surface of the torus consist of all points at a fixed distance a from this generator. Thus the scalar a is the radius of the toroidal tube.

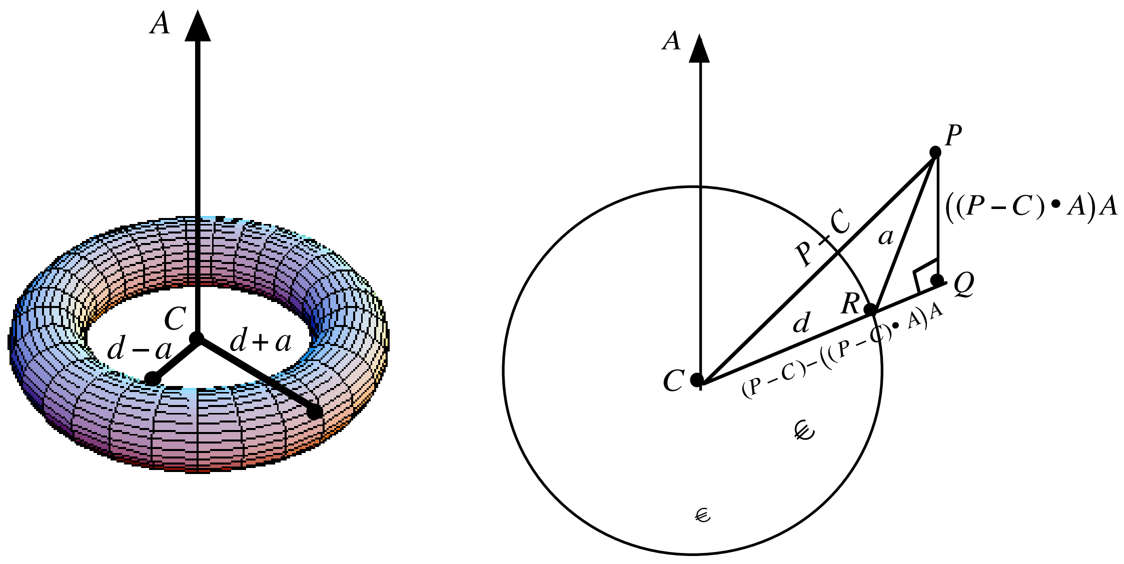


Figure 6: The torus. The circle at a distance *d* from the center of the torus in the plane perpendicular to the axis *A* is the generator for the torus. The radius of the inner tube is *a*.

To find the implicit equation of the torus, project an arbitrary point *P* on the torus orthogonally into the plane of the generator. Then the point *P*, the image point *Q*, and the point *R* lying on the generator between *Q* and *C* form a right triangle whose hypotenuse has length *a* (see Figure 6). Thus by the Pythagorean theorem

$$|P-Q|^2+|Q-R|^2=a^2. \tag{6.1}$$

But $|P-Q|$ is just the distance from the point *P* to the plane determined by the center point *C* and the axis vector *A*, which in turn is just the length of the parallel projection of the vector $P-C$ on *A*. Therefore

$$|P-Q|=\left|(P-C)\bullet A\right|. \tag{6.2}$$

Moreover,

$$|Q-R|=|Q-C|-d, \tag{6.3}$$

and $|Q-C|$ is just the length of the perpendicular component of the vector $P-C$ with respect to *A*. Therefore

$$|Q-C|=|(P-C)_{\perp}|=\sqrt{|P-C|^2-\left((P-C)\bullet A\right)^2}. \tag{6.4}$$

Substituting Equations 6.2-6.4 into Equation 6.1, we arrive at the following implicit equation for the torus in general position:

$$\left(\underbrace{(P-C)\bullet A}_{\text{Dist. to Plane}}\right)^2+\left(\underbrace{\sqrt{|P-C|^2-\left((P-C)\bullet A\right)^2}}_{\text{Dist. of Projection to Center}}-\underbrace{d}_{\text{Radiaus of Generator}}\right)^2=a^2. \tag{6.5}$$

Because of the square root, this equation is actually degree four in the coordinates (x,y,z) of the point P, since to clear the square root we must square the equation.

We can also find the normal N to the surface of the torus at the point P by a geometric argument. Cut the torus by the plane determined by the point P and the axis of the torus. The intersection of this plane and the torus consists of two circle; one of these circles passes through the point P. The center of this circle is the point R on the generator of the torus, and the vector $P-R$ from R to P is orthogonal to the surface of the torus. Thus we need to find the point R.

To locate the point R, observe that R is on the line from Q to C, and

$$Q-C=(P-C)-\big((P-C)\bullet A\big)A.$$

Let q be the distance from Q to C. Then by the Pythagorean theorem

$$q=|Q-C|=\sqrt{|P-C|^2-\big((P-C)\bullet A\big)^2}.$$

Thus

$$R=C+(d/q)(Q-C).$$

Therefore the normal to the torus is given by

$$N \parallel (P-R)=(P-C)-(d/q)(Q-C). \qquad (6.6)$$

To intersect a line $L(t)=P+tv$ with a torus, we simply substitute the parametric equations of the line into the implicit equation for the torus and solve the fourth degree equation

$$\big((L(t)-C)\bullet A\big)^2+\left(\sqrt{|L(t)-C|^2-\big((L(t)-C)\bullet A\big)^2}-d\right)^2=a^2. \qquad (6.7)$$

The real roots of this equation represent the parameter values along the line of the intersection points. As usual, the actual intersection points can be computed by substituting these parameter values into the parametric equation $L(t)=P+tv$ for the line.

The implicit equation for the torus in general position is somewhat complicated. Often, as with general quadric surfaces, it is easier to begin with a torus in canonical position -- with center at the origin and axis along the z-axis -- and then apply a rigid motion to relocate the torus to an arbitrary position and orientation. This approach also works for an elliptical torus, since we can include non-uniform scaling along with the rigid motion.

To find the implicit equation of the torus in canonical position, substitute $C=(0,0,0)$ and $A=(0,0,1)$ into Equation (6.5), the implicit equation of the torus in general position. Squaring to eliminate the square root and simplifying the algebra, we arrive at the fourth degree implicit equation for the torus in canonical position:

$$T(x,y,z)\equiv\left(x^2+y^2+z^2-d^2-a^2\right)^2+4d^2z^2-4a^2d^2=0. \qquad (6.8)$$

The normal vector $N(x,y,z)$ to the surface $T(x,y,z)$ is given by the gradient. Therefore

$$\begin{aligned} N(x,y,z) &= \left(\frac{\partial T}{\partial x}, \frac{\partial T}{\partial y}, \frac{\partial T}{\partial z}\right) = \left(4xG(x,y,z),\ 4yG(x,y,z),\ 4z(G(x,y,z)+2d^2)\right) \\ G(x.y,z) &= x^2 + y^2 + z^2 - d^2 - a^2 \end{aligned} \qquad (6.9)$$

To reposition the torus so that the center is at C and the axis is oriented along the direction A, rotate the z-axis to A and then translate the origin to C. These transformations are given explicitly by the matrices (see Lecture 13)

$$trans(C) = \begin{pmatrix} I & 0 \\ C & 1 \end{pmatrix}$$

$$rot(u,\theta) = cos(\theta)I + \left(1 - cos(\theta)\right)(u^T * u) + sin(\theta)(u \times \_)$$

where $u = \dfrac{k \times A}{|k \times A|}$ and $\cos(\theta) = k \bullet A$.

Now we can find normal vectors on the surface of the torus in general position simply by transforming the corresponding normal vectors from canonical position. Thus

$$N_{general\ \ position} = N_{canonical\ \ position} * rot(u,\theta).$$

(Recall that $rot(u,\theta)^{-T} = rot(u,\theta)$, so we do not have to take the inverse transpose.) Similarly, we can intersect lines with the torus in general position by transforming the lines to canonical position via the inverse transformation

$$M^{-1} = trans(-C) * rot(u,-\theta),$$

intersecting the transformed lines with the torus in canonical position, and then transforming the intersection points back to general position using the forward transformation matrix

$$M = \begin{pmatrix} rot(u,\theta) & 0 \\ trans(C) & 1 \end{pmatrix}.$$

6.1 Bounding the Torus. Ray-torus intersections are expensive to compute because to find the intersection points of a line and a torus, we must solve a fourth degree equation. However, the torus is a bounded surface, so many lines fail to intersect the torus. Thus to speed up our computations, we would like to have fast rejection tests to determine when a line does not intersect the torus. We can develop such fast tests by bounding the torus with simpler surfaces.

Consider a torus represented by a center point C, an axis vector A, and two scalar radii d,a as in Figure 6. We can bound this torus on the outside by a sphere with center C and radius $d+a$ (see Figure 7, left). Any line that fails to intersect this sphere will surely fail to intersect the torus. Since we have fast tests for determining whether or not a line intersects a sphere, this approach provides a fast rejection test when intersecting a line and a torus.

Nevertheless, lines that do intersect this sphere may still fail to intersect the torus by passing above or below the planes bounding the torus or by going through the hole at the center of the torus. To eliminate many of the lines passing through the hole in the torus, consider the cylinder of radius $d - a$, whose axis line is the axis of the torus and whose height above and below the center point C is a (see Figure 7, right). Any line that passes through the interior of this cylinder will fail to intersect the torus. To detect this case, consider the disks at the top and the bottom of this bounded cylinder. Any line that intersects both of these disks cannot intersect the surface of this bounded cylinder and hence cannot intersect the torus. Since we have fast tests for determining if a line intersects these disks (see Section 4.2.2), this approach provides a fast test for rejecting many of the lines that pass through the hole in the torus.

We can also use cylinders to bound the torus on the outside. Consider the cylinder of radius $d + a$, whose axis line is the axis of the torus and whose height above and below the center point C is a. Any line that fails to intersect this bounded cylinder will surely fail to intersect the torus (see Figure 7, middle). To test whether or not a line intersects this bounded cylinder, we first compute the intersection points of the line with the surface of the infinite cylinder. If both intersection points lie above the plane bounding the cylinder from above or below the plane bounding the cylinder from below, then the line cannot intersect the bounded cylinder. Since there are fast tests for determining if a point lies above or below a plane, this approach provides another fast test for rejecting lines that fail to intersect the torus. This cylinder is often a tighter bound to the torus than the sphere, so this approach may speed up the ray tracing computation by eliminating more lines. Below is pseudocode for the fast rejection tests for the Line-Torus intersection algorithm.

<u>Fast Rejection Tests for Line–Torus Intersection</u>

```
Input
    $C$ = Center Point of the Torus
    $A$ = Unit Direction Vector Parallel to the Axis of the Torus
    $d$ = Radius of the Generator of the Torus
    $a$ = Tube Radius
    $P$ = Point on the Line $L$
    $v$ = Unit Vector in the Direction of the Line $L$
Algorithm
    If there are no intersections between the line $L$ and the sphere with center $C$ and radius
    $d + a$ bounding the torus, then there are no intersections between the line $L$ and the torus.
        If $Dist^2(C,L) > (d + a)^2$, then the $L$ line does not intersect the torus.
    Otherwise if the line $L$ is parallel to the plane determined by the point $C$ and the unit
    normal vector $A$ -- that is, if $v \bullet A \approx 0$ -- then:
        If the distance from the line $L$ to the plane determined by the point $C$ and the normal
        vector $A$ is greater than $a$, there are no intersections between the line $L$ and the torus.
```

```
If $v \bullet A \approx 0$ and $|(P - C) \bullet A| > a$, then the line $L$ does not intersect the torus.
Otherwise, find the intersections $E_1, E_2$ of the line $L$ with the planes determined by the
points $C - aA$ and $C + aA$ and the normal vector $A$ that bound the torus above and below.
  If the points $E_1, E_2$ lie inside the disks bounding the inner cylinder above and below,
  then there are no intersections between the line $L$ and the torus.
    If $Dist^2(E_1, C - aA) < (d - a)^2$ and $Dist^2(E_2, C + aA) < (d - a)^2$, then the line $L$
    does not intersect the torus.
  Otherwise, if the points $E_1, E_2$ lie outside the disks bounding the outer cylinder above
  and below:
    If $Dist^2(E_1, C - aA) > (d + a)^2$ and $Dist^2(E_2, C + aA) > (d + a)^2$, then
      Find one intersection point $D$ and the corresponding parameter $t$ of the line
      $E(t) = E_1 + t(E_2 - E_1)$ with the cylindrical surface bounding the torus on the
      outside -- that is, the infinite cylinder with radius $d + a$ and the same axis line
      as the torus.
      If $t < 0$ or $t > 1$, then the line $L$ does not intersect the torus.
```

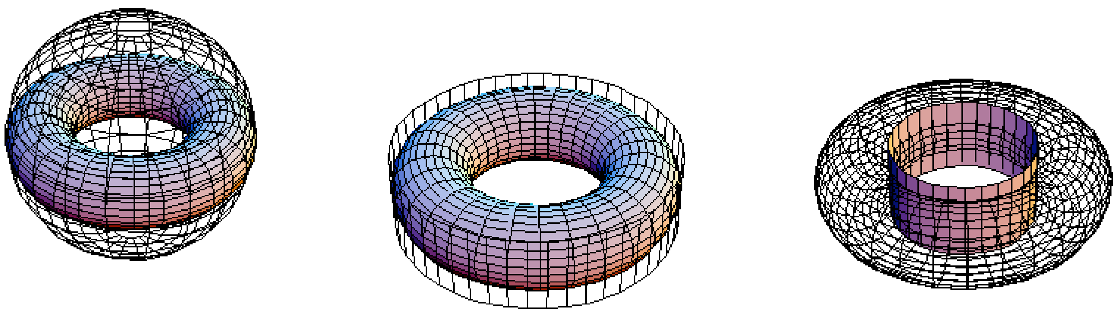


Figure 7: A sphere (left) and a cylinder (center) bounding a torus on the outside, and a cylinder (right) surrounding the hole to the torus on the inside. Notice that the cylinder is a tighter fit than the sphere to the torus on the outside.

If these fast rejection tests all fail, then you will need to intersect the line with the torus. To intersect a line and a torus, you must solve Equation (6.7), an equation of degree four in the parameter *t*. There are two ways to proceed: either solve analytically or perform a binary search. Solving analytically requires using the formula for the roots of a degree 4 equation; this formula can be found in standard algebra texts. To perform a binary search, first find the parameter values where the line intersects the bounding sphere or the bounding cylinder. Then using substitution, search for intermediate parameter values where the implicit equation (Equation (6.7)) changes sign. An intersection point must occur between any two parameter values where Equation (6.7) takes on opposite signs. You will need to pick a tolerance at which to terminate the search. Note that for each line you may find anywhere between zero and four intersection points.

7. Surfaces of Revolution

A *surface of revolution* is a surface generated by rotating a planar curve called the *directrix* about a straight line called the *axis of revolution*. Thus the cross sections of a surface of revolution by planes orthogonal to the axis of revolution are circles. Spheres, right circular cylinders, right circular cones, and tori are all examples of surfaces of revolution (see Table 2).

So far we have studied special surfaces using either a geometric approach or implicit equations. Surfaces of revolution, however, are modeled most easily using parametric equations.

Consider a parametric curve

$$D(u) = \big(f(u), 0, g(u)\big)$$

in the xz-plane. Rotating this curve about the z-axis generates the parametric surface

$$R(u,v) = \big(f(u)\cos(v), f(u)\sin(v), g(u)\big) \qquad (7.1)$$

(see Figure 8). If we fix a parameter $u = u^*$, then the plane $z = g(u^*)$ intersects the surface $R(u,v)$ in the circle

$$C(v) = \Big(f(u^*)\cos(v), f(u^*)\sin(v), g(u^*)\Big)$$

with center $(0,0,g(u^*))$ and radius $f(u^*)$, since $x^2 + y^2 = f^2(u^*)$. Thus cross sections of the surface $R(u,v)$ by planes perpendicular to the z-axis are indeed circles, so the parametric surface $R(u,v)$ truly is a surface of revolution. Table 2 provides some examples of simple directrix curves and the corresponding surfaces of revolution. Using this table together with Equation (7.1) allows us to generate simple parametric equations for cylinders, cones, spheres, and tori.

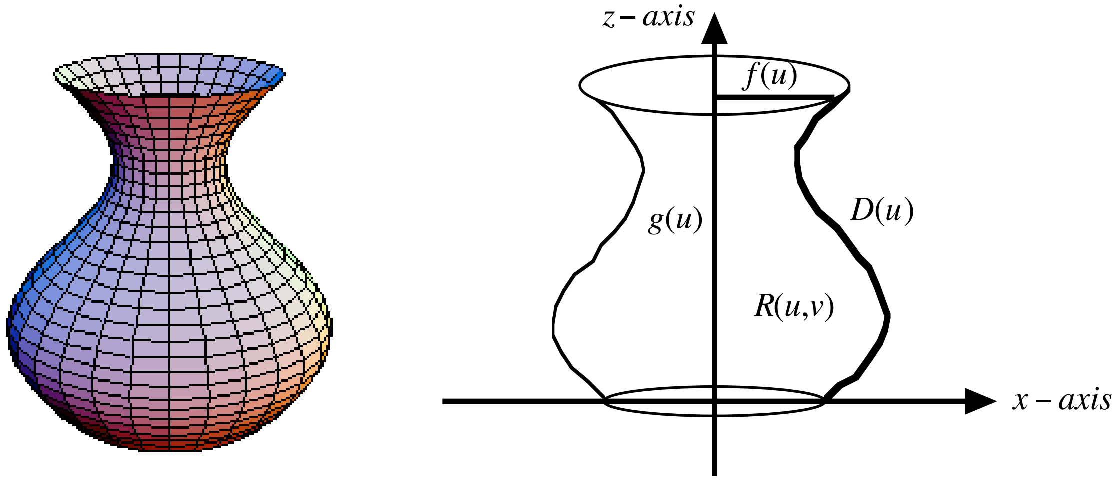


Figure 8: Surface of revolution $R(u,v)$ with directrix $D(u)$.

<u>Directrix</u>	<u>Surface of Revolution</u>
Line: $D(u) = (1,0,u)$	Cylinder: $R(u,v) = (\cos(v), \sin(v), u)$
Line: $D(u) = (u,0,u)$	Cone: $R(u,v) = (u\cos(v), u\sin(v), u)$
Circle: $D(u) = (\cos(u),0,\sin(u))$	Sphere: $R(u,v) = (\cos(u)\cos(v), \cos(u)\sin(v), \sin(u))$
Circle: $D(u) = (d + a\cos(u),0,a\sin(u))$	Torus: $R(u,v) = \begin{pmatrix} (d + a\cos(u))\cos(v), \\ (d + a\cos(u))\sin(v), a\sin(u) \end{pmatrix}$

Table 2: Directrix curves that are lines and circles together with the corresponding surfaces of revolution. The axis of revolution is the z-axis.

We are not restricted to surfaces of revolution whose axis of revolution is the z-axis. To extend our parametric equations to general position, we first rewrite the parametric equations of the directrix and the surface of revolution in vector notation:

$$D(u) = Origin + f(u)i + g(u)k$$

$$R(u,v) = Origin + f(u)\cos(v)i + f(u)\sin(v)j + g(u)k .$$

Rotating the unit coordinate axis vectors i,j,k to three arbitrary orthogonal unit vectors C,B,A and then translating the origin to an arbitrary point E, we arrive at the parametric equations of the directrix and the surface of revolution in general position:

$$D(u) = E + f(u)C + g(u)A$$

$$R(u,v) = E + f(u)\cos(v)C + f(u)\sin(v)B + g(u)A .$$

Notice that here the axis of revolution is parallel to the vector A, and the directrix lies in the plane of the vectors A and C.. The vectors B and C are orthogonal unit vectors perpendicular to the axis A.

There is not much difference between working in canonical position and working in general position for surfaces of revolution, since we can always transform from canonical position to general position by the rigid motion

$$M = \begin{pmatrix} C & 0 \\ B & 0 \\ A & 0 \\ E & 1 \end{pmatrix} .$$

Therefore, for the remainder of this discussion, we will consider surfaces of revolution in canonical position, where the axis of revolution is the z-axis.

The normal vector to a parametric surface is parallel to the cross product of the partial derivatives. For the surface of revolution in Equation (7.1),

$$R(u,v) = (f(u)\cos(v), f(u)\sin(v), g(u))$$

so

$$\frac{\partial R}{\partial u} = \left(f'(u)\cos(v), f'(u)\sin(v), g'(u)\right)$$

$$\frac{\partial R}{\partial v} = \left(-f(u)\sin(v), f(u)\cos(v), 0\right).$$

Therefore,

$$N(u,v) \parallel \frac{\partial R}{\partial u} \times \frac{\partial R}{\partial v} = \left(-f(u)g'(u)\cos(v), -f(u)g'(u)\sin(v), f(u)f'(u)\right)$$

To ray trace a surface of revolution, we need to intersect a surface of revolution with straight lines. A surface of revolution

$$R(u,v) = \left(f(u)\cos(v), f(u)\sin(v), g(u)\right)$$

intersects a line

$$L(t) = At + B = \left(A_x t + B_x, A_y t + B_y, A_z t + B_z\right),$$

at parameters where

$$R(u,v) = L(t)$$

or equivalently where

$$\left(f(u)\cos(v), f(u)\sin(v), g(u)\right) = \left(A_x t + B_x, A_y t + B_y, A_z t + B_z\right).$$

Thus to find the intersection of a line and a surface of revolution, we need to solve three simultaneous nonlinear equations with three unknown t,u,v. The simplest of these three equations is the *z*-equation, since only two parameters u,t appear in this equation. Solving the *z*-equation

$$A_z t + B_z = g(u)$$

for *t* yields

$$t = \frac{g(u) - B_z}{A_z}, \tag{7.2}$$

so we can eliminate one of the variables. {If $L(t)$ is horizontal, then $A_z = 0$ and we can solve $g(u) = B_z$ directly for *u*.} Next, summing the squares of the x,y equations and invoking the trigonometric identity

$$\cos^2(v) + \sin^2(v) = 1$$

yields

$$f^2(u) = (A_x t + B_x)^2 + (A_y t + B_y)^2. \tag{7.3}$$

Since we already know *t* as a function of *u* from Equation (7.2), we can reduce Equation (7.3) to one unknown. We then solve Equation (7.3) for *u*, usually by numerical methods. The corresponding *t* parameters can now be computed from Equation (7.2). Finally taking the ratio of the *x* and *y* equations yields

$$\tan(v) = \frac{A_y t + B_y}{A_x t + B_x},$$

so

$$v = \tan^{-1}\left(\frac{A_y t + B_y}{A_x t + B_x}\right). \qquad (7.4)$$

The only bottleneck in this intersection algorithm is solving Equation 7.3. If $f(u), g(u)$ are trigonometric functions, as is the case for the sphere and the torus in Table 2, then this equation contains transcendental functions and may be difficult to solve.

There is a trick, however, for replacing sines and cosines by rational functions. Let $s = \tan(u/2)$. Then

$$\cos(u) = \cos^2(u/2) - \sin^2(u/2) = 2\cos^2(u/2) - 1 \Rightarrow \cos^2(u/2) = \frac{1+\cos(u)}{2}$$

so

$$s^2 + 1 = \tan^2(u/2) + 1 = \sec^2(u/2) = \frac{1}{\cos^2(u/2)} = \frac{2}{1+\cos(u)}.$$

Solving for $\cos(u)$, we find that

$$\cos(u) = \frac{1-s^2}{1+s^2} \quad \text{and} \quad \sin(u) = \frac{2s}{1+s^2}.$$

Replacing sine and cosine with rational functions often leads to faster computations. Moreover, it is generally easier to solve polynomial equations than to solve trigonometric equations.

If the directrix is an unbounded curve, then the method outlined above intersects a line with an unbounded surface of revolution. But just like cylinders and cones, most surfaces of revolution that appear in Computer Graphics are closed and bounded. To intersect a line with a surface of revolution bounded above and below by a pair of disks, we must discard any intersections that are above or below the planes that contain these disks. We must also intersect the line with the planes of the two bounding disks and then test to see whether or not these intersections points lie inside these disks. The details are similar to the ray-tracing algorithms for the bounded cylinder and the bounded cone and are therefore left as an exercise for the reader (see Exercise 17).

8. Summary

In this lecture we have concentrated on the simplest mathematical surfaces, those surfaces most common in Computer Graphics:

- planes and convex polygons
- natural quadrics and their elliptical variants

    - spheres and ellipsoids
    - right circular cylinders and elliptical cylinders
    - right circular cones and elliptical cones
- general quadric surfaces both parabolic and hyperbolic
    - parabolic cylinders and hyperbolic cylinders
    - paraboloids and hyperboloids
- tori and surfaces of revolution

For each of these surfaces we have shown how to compute their surface normals as well as how to calculate their intersections with straight lines. With these results in hand, ray tracing these surfaces is a straightforward task.

In the next lecture we shall extend our discussion from surface modeling to solid modeling. Solids are bounded by surfaces, so many of the surfaces that we have discussed here will reappear in our discussion of solid modeling. Ray tracing which is a powerful technique for rendering surfaces will also reappear as a potent technique for analyzing solid models.

Exercises:

1. Show that a point R in the plane of $\Delta P_1P_2P_3$ lies inside $\Delta P_1P_2P_3$ if and only if all three of the barycentric coordinates of R with respect to $\Delta P_1P_2P_3$ are positive. (See Lecture 4, Exercise 24 for a definition of barycentric coordinates.)

2. Explain in detail how to ray trace an elliptical cylinder. In particular, explain how you would:
    i. model the elliptical cylinder;
    ii. compute the surface normals to the elliptical cylinder;
    iii. find ray-surface intersections for the elliptical cylinder.

    Treat both the bounded and unbounded elliptical cylinder.

3. Develop an algorithm to intersect a straight line with a right circular cone bounded by:
    a. two disks
    b. one disk and the cone vertex.

4. Explain in detail how to ray trace an elliptical cone. In particular, explain how you would:
    i. model the elliptical cone;
    ii. compute the surface normals to the elliptical cone;
    iii. find ray-surface intersections for the elliptical cone.

    Treat both the bounded and the unbounded elliptical cone.

5. Let $D(s) = (x(s), y(s), z(s))$ be a parametrized curve lying in a plane in 3-space, and let $v = (v_1, v_2, v_3)$ be a fixed vector in 3-space not in the plane of $D(s)$. The parametric surface

$$C(s,t) = D(s) + t\,v$$

is called the *generalized cylinder* over the curve $D(s)$ (see too Lecture 18, Exercise 2). Suppose that you already have an algorithm to compute the intersection points of any line in the plane of $D(s)$ with the curve $D(s)$ Based on this algorithm, develop a procedure to find the intersection points of an arbitrary line in 3-space with the surface $C(s,t)$.

6. Let $D(s) = (x(s), y(s), z(s))$ be a parametrized curve lying in a plane in 3-space, and let $Q = (q_1, q_2, q_3)$ be a fixed point in 3-space not in the plane of $D(s)$. The parametric surface

$$C(s,t) = (1-t)\,D(s) + t\,Q$$

is called the *generalized cone* over the curve $D(s)$ (see too Lecture 18, Exercise 3). Suppose that you already have an algorithm to compute the intersection points of any line in the plane of $D(s)$ with the curve $D(s)$ Based on this algorithm, develop a procedure to find the intersection points of an arbitrary line in 3-space with the surface $C(s,t)$.

7. Let $U_1(s), U_2(s)$ be two curves in 3-space. The parametric surface

$$R(s,t) = (1-t)U_1(s) + t\,U_2(s)$$

is called the *ruled surface* generated by the curves $U_1(s), U_2(s)$ (see too Lecture 18, Exercise 4). Explain how you would find the intersection points of a ruled surface $R(s,t)$ with an arbitrary straight line $L(t)$.

8. Let $R(s) = R + s\,u$ and $P(t) = P + t\,v$ be two intersecting lines in 3-space. Let $v_\perp$ be the component of v perpendicular to u.

a. Show that the lines $R(s)$ and $P(t)$ intersect at the parameter value

$$t = \frac{(R-P) \bullet v_\perp}{v \bullet v_\perp}.$$

b. Use the result of part a to find the parameter value where the line and the cylinder intersect without resorting to the inversion formula for the line $L_\perp(t)$.

c. Use the result of part a to find the parameter value where the line and the cone intersect without resorting to the inversion formula for the rational linear parametrization of the line $L^*(t)$

9. Let $R = (x_0, y_0, z_0, 1)$ be a point on a quadric surface represented by a symmetric 4×4 matrix Q. Show that the equation of the plane tangent to the quadric Q at the point R is given by $R * Q * P^T = 0$.

10. Find the symmetric 4×4 matrices representing each of the quadric surfaces in Table 1.

11. Compute the mean and Gaussian curvature (see Lecture 18) for each of the quadric surfaces in Table 1.

12. Compute the mean and Gaussian curvature (see Lecture 18) for a torus in canonical position in two ways:

   i. from the implicit equation;
   ii. from the parametric equation.

13. A *cyclide* is a surface whose lines of constant curvature are all either circles or straight lines. (The torus is a special case of a cyclide.) The equation of a cyclide in canonical position -- centered at the origin with axes aligned along the coordinate axes -- is

$$F(x,y,z) = \left(x^2 + y^2 + z^2 - d^2 + b^2\right)^2 - 4(ax - cd)^2 - 4b^2y^2 = 0$$

where a,b,c,d are constants such that $c,d \geq 0$, $b > 0$, and $a^2 = b^2 + c^2$.

   a. For the cyclide in canonical position, explain how you would:
      i. compute the normal vector at the point (x,y,z),
      ii. find the intersection points with an arbitrary straight line.
   b. For a cyclide in arbitrary position -- a cyclide with center at the point C and axes aligned along three mutually orthogonal unit vectors u,v,w -- explain how you would:
      i model the cyclide,
      ii. compute the normal to the cyclide at the point (x,y,z),
      iii. find the intersection points of the cyclide with an arbitrary straight line.

14. Consider a surface whose cross sections by planes perpendicular to a fixed line L are ellipses whose centers lie on the line L and whose major axes are all parallel to each other. Explain how you would:

   a. model this surface,
   b. compute the normal to this surface at an arbitrary point on the surface,
   c. find the intersection of this surface with an arbitrary straight line.

15. Consider a surface of revolution

$$R(u,v) = \big(f(u)\cos(v), f(u)\sin(v), g(u)\big).$$

Suppose that there is a function $F(u)$ such that

$$F\big(g(u)\big) = f^2(u).$$

   a. Show that the implicit equation of $R(u,v)$ is given by

$$x^2 + y^2 = F(z).$$

b. Find the function $F(u)$ for each of the surfaces of revolution in Table 2.
c. Using the results of part *b*, find the implicit equation for each of the surfaces of revolution in Table 2.

16. Consider the parametrizations in Table 2 of the cylinder, the cone, the sphere, and the torus.
a. Using the substitution $t = \tan(v/2)$, find parametrizations for the cylinder and the cone that do not involve trigonometric functions.
b. Using in addition the substitution $s = \tan(u/2)$, find parametrizations for the sphere and the torus that do not involve trigonometric functions.

17. Develop an algorithm to intersect a straight line with a surface of revolution bounded by two disks.

Programming Projects:

1. *Recursive Ray Tracing*
Implement a recursive ray tracer in your favorite programming language using your favorite API.
a. Include at least the following surfaces:
   - Tetrahedra and Cubes
   - Natural Quadrics and their Elliptical Variants
   - Tori
b. Use several light sources along with the following lighting models:
   - Ambient Light
   - Diffuse Reflection
   - Specular Reflection
c. Incorporate both reflection and refraction.
d. Incorporate a spotlight (see below)
A *spotlight* is a focused point light source at a finite distance from the scene. The light cone of the spotlight is specified by the following parameters (see Figure 9)
   - C = cone vertex = location of light source
   - A = cone unit axis vector
   - γ = cone angle

The lighting formulas for a spotlight are the same as the lighting formulas for a point light source, except that the diffuse and specular components are attenuated by a factor c_{spot}^{e}. Thus for a spotlight:

- $I_{diffuse} = c_{spot}^{e} I_p k_d (L \bullet N)$
- $I_{specular} = c_{spot}^{e} I_p k_s (R \bullet V)^n$

where

- L = direction to light source
- N = normal vector to surface
- V = direction to eye
- R = direction of reflection vector = $2(L \bullet N)N - L$
- A = spotlight cone axis vector
- γ = spotlight cone angle
- $$c_{spot} = \begin{cases} -L \bullet A & -L \bullet A \geq \cos(\gamma) \\ 0 & \textit{otherwise} \end{cases}$$
- e = exponent controlling falloff in intensity from center of spotlight (user defined)

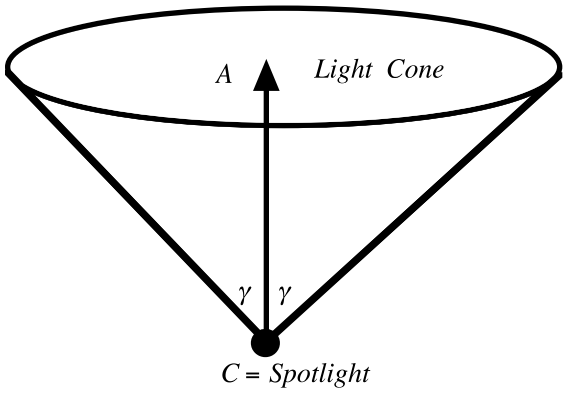


Figure 9: The light cone for a spotlight.

.2. *Camera Obscura*

A *camera obscura* is the technical term for a pinhole camera. In a camera obscura the film is behind the aperture. Placing the film behind the aperture is equivalent to placing the screen behind the eye, rather than in front of the eye.

a. Use recursive ray tracing to simulate a camera obscura.

b. How does the image differ from the image generated by standard ray tracing?