General Quadric Surfaces

These surfaces are so named because the variables *x, y, and z* take on at most the power of two. The general equation for the quadric is given below:

$$Ax^2 + By^2 + Cz^2 + Dxy + Exz + Fyz + Gx + Hy + Iz = J$$

We have already encountered two special cases:

The sphere:

$$x^2 + y^2 + z^2 = r^2$$

The plane:

$$Gx + Hy + Iz = J$$

where the plane normal is *(G, H, I)* and J is chosen so that the plane passes through the selected point.

Quadric surfaces are "nice" in a raytracing environment because the intersection of a ray with the surface may always be found by solving at worst a quadratic equation.

Adding a cylinder object to the ray tracer

Finite length cylinder will be specified in the following way:

```
22              cylinder

0.2 0.2 0.2     ambient
2.0 2.0 2.0     diffuse
1.0 1.0 1.0     specular

4  3 -4         center point of finite cylinder base.
1 0 0           centerline vector
2 5             radius and height
```

The cylinder definition is:

```
typedef struct cyl_type
{
   double  center[3];
   double  centerline[3];
   double  radius;
   double  height;
   double  rotmat[3][3];
   double  irot[3][3];
}  cyl_t;
```

Determining if a ray hits a cylinder.

Assume the following:

B = base or start of the ray
D = a unit vector in the direction the ray is traveling
C = center of the base of the cylinder
r = radius of the cylinder.
h = height of the cylidner

The arithmetic is much simpler if the center of the cylinder is at the origin. So we start by moving it there! To do so we must make a compensating adjustment to the base of the ray.

B′ = B - C = new base of ray
C′ = C - C = (0, 0, 0) = new center of cylinder
D does not change

The next step in the operation is to rotate the center line of the cylinder so that it lies on the y axis. Begin by constructing an identity matrix. If the centerline *already* lies on the y-axis there is nothing more to do. If not, procede to construct a rotation matrix as follows.

Rotate the centerline of the cylinder into the positive y axis
Rotate a vector orthogonal to both the centerline and the postive y axis into the positive x axis.

Apply the rotation to *B′* getting a new *B′* and to *D* getting a new *D*. Since *C′* is zero the rotation would not change it.

All points on the ray may be expressed in the form

$$P = B' + t\,D = (b'_x + td_x,\ b'_y + td_y,\ b'_z + td_z) \qquad (1)$$

A point P on the infinite cylinder which is aligned with the y-axis whose center passes through (0, 0, 0) necessarily satisfies the following equation:

$$p_x^2 + p_z^2 = r^2 \qquad (2)$$

Thus we need to find a value of *t* which yields a point that satisfies the two equations. To do that we take the *(x, y, z)* coordinates from equation (1) and plug them into equation (2). We will show that this leads to a quadratic equation in *t* which can be solved via the quadratic formula.

$$(b'_x + td_x)^2 + (b'_z + td_z)^2 = r^2$$

Expanding this expression by squaring the three binomials yields:

$$b'^2_x + 2tb'_x d_x + t^2 d_x^2 + b'^2_z + 2tb'_z d_z + t^2 d_z^2 = r^2$$

Next we collecting the terms associated with common powers of *t*

$$(b'^2_x + b'^2_z) - r^2 + 2t\,(b'_x d_x + b'_z d_z) + t^2(d_x^2 + d_z^2) = 0$$

We now make the notational changes:

$$a = (d_x^2 + d_z^2)$$
$$b = 2\,(b'_x d_x + b'_z d_z)$$
$$c = (b'^2_x + b'^2_z) - r^2$$

to obtain the solution in the standard form of the quadratic formula:

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The *hits_cyl()* function should also fill in the coordinates of the *hit* in the *obj_t* structure.

The *(x, y, z)* coordinates of the hit point are on the translated and rotated cylinder are then computed as follows.

$$H = B' + t\,D$$

Important items to note are:

> The the values used here are the *translated and rotated* base point of the ray and the *rotated* direction.

Next it is necessary to check to see if the *y* component of H is less than 0 or greater that *h*. In these cases we have a *miss* and -1 should be returned.

For the cylinder it is best to compute the surface normal from the coordiates of the hitpoint $H = (h_x, h_y, h_z)$ *now.*

$$N = (h_x, 0, h_z)$$
$$N /= ||N||$$

The final step is to transform both the normal and the hitpoint back to the original coordinate space. To do this you will need the *inverse* rotation matrix which is simply its transpose.

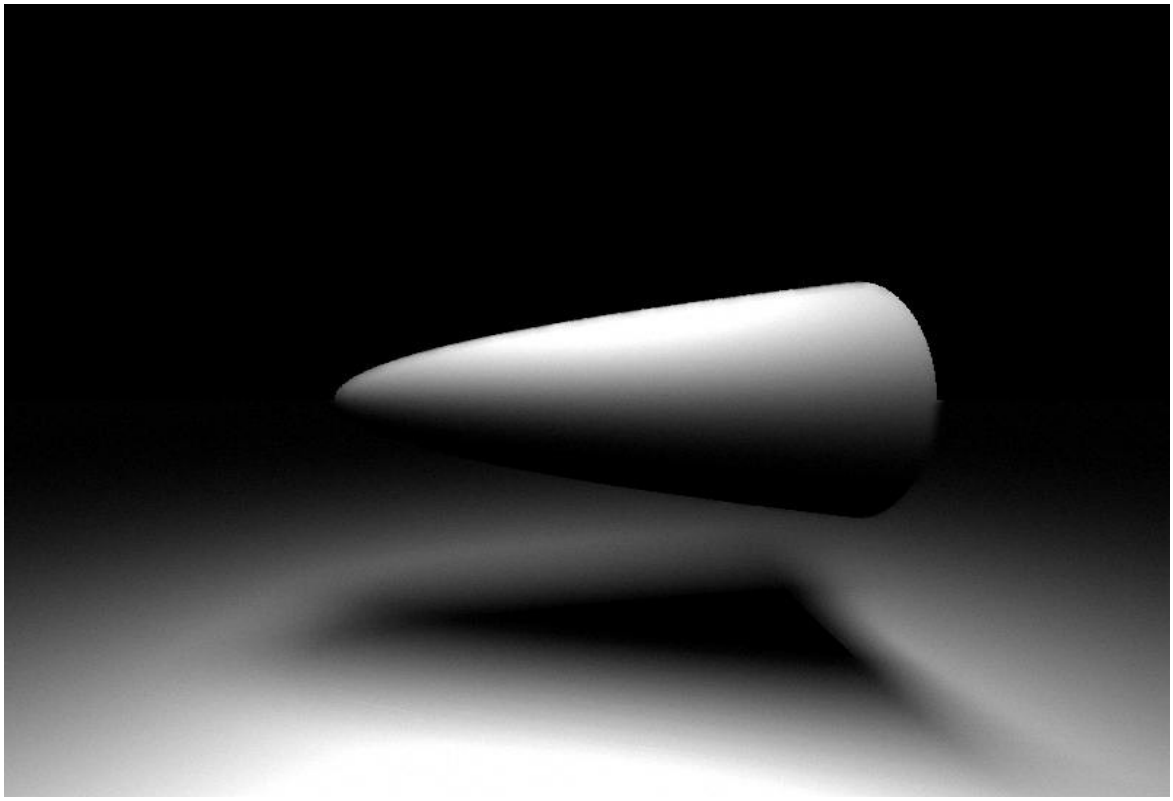
In finding the location of the hitpoint we *first* translated the cylinder to the origing *and then* rotated in to the proper orientation. To *undo* this effect we *first* rotate back and then transate.

Transform the vector *N* with the inverse rotation to get the correct normal vector.

Transform the hitpoint *H* with the inverse rotation matrix to get the rotated hitpoint.

Finally add the coordinates of the center of the base of the cylinder to translate the hitpoint back to the correct location in the original coordinate space.

Adding a finite circular paraboloid object to the ray tracer



The finite length paraboloid will be specified in the following way:

```
21              paraboloid

0.2 0.2 0.2     ambient
2.0 2.0 2.0     diffuse
1.0 1.0 1.0     specular

4  3 -4         center point of finite circular paraboloid base.
1 0 0           centerline vector
2 5             radius at base and height

typedef struct parab_type
{
   double  center[3];       /* center of base */
   double  centerline[3];
   double  radius;          /* radius of base */
   double  height;
   double  scale;           /* r^2 / h        */
   double  rotmat[3][3];
   double  irot[3][3];
}  parab_t;
```

Determining if a ray hits a circular paraboloid.

Assume the following:

B = base or start of the ray
D = a unit vector in the direction the ray is traveling
C = center of the base of the paraboloid
r = radius of the paraboloid.
h = height of the parabloid

The arithmetic is much simpler if the center of the paraboloid is at the origin. So we start by moving it there! To do so we must make a compensating adjustment to the base of the ray.

B′ = B - C = new base of ray
C′ = C - C = (0, 0, 0) = new center of cone
D does not change

The next step in the operation is to rotate the center line of the cone so that it lies on the y axis. Begin by constructing an identity matrix. If the centerline *already* lies on the y-axis there is nothing more to do. If not, procede to construct a rotation matrix as follows.

Rotate the centerline of the cone into the positive y axis
Rotate a vector orthogonal to both the centerline and the postive y axis into the positive x axis.
Exercise: How should you handle the upside down paraboloid??

Apply the rotation to *B′* getting a new *B′* and to *D* getting a new *D*. Since *C′* is zero the rotation would not change it.

All points on the ray may be expressed in the form

$$P = B' + t\,D = (b'_x + td_x,\ b'_y + td_y,\ b'_z + td_z) \qquad (1)$$

A point P on the infinite paraboloid which is aligned with the y-axis and whose vertex is at (0, 0, 0) necessarily satisfies the following equation:

$$p_x^2 + p_z^2 = kp_y \qquad (2)$$

Since the *radius* at the top of the paraboloid is what we want to specify, when $p_y = h$ we want $kh = r^2$. Thus $k = r^2/h$. We compute the *k* value at object initialization time and save it in *parab->scale.*

$$a = (d_x^2 + d_z^2)$$
$$b = 2\,(b'_x d_x + b'_z d_z) - kd_y$$
$$c = (b'^2_x + b'^2_z) - kb'_y$$

to obtain the solution in the standard form of the quadratic formula:

$$t = \frac{-b +/- sqrt(b^2 - 4ac)}{2a}$$

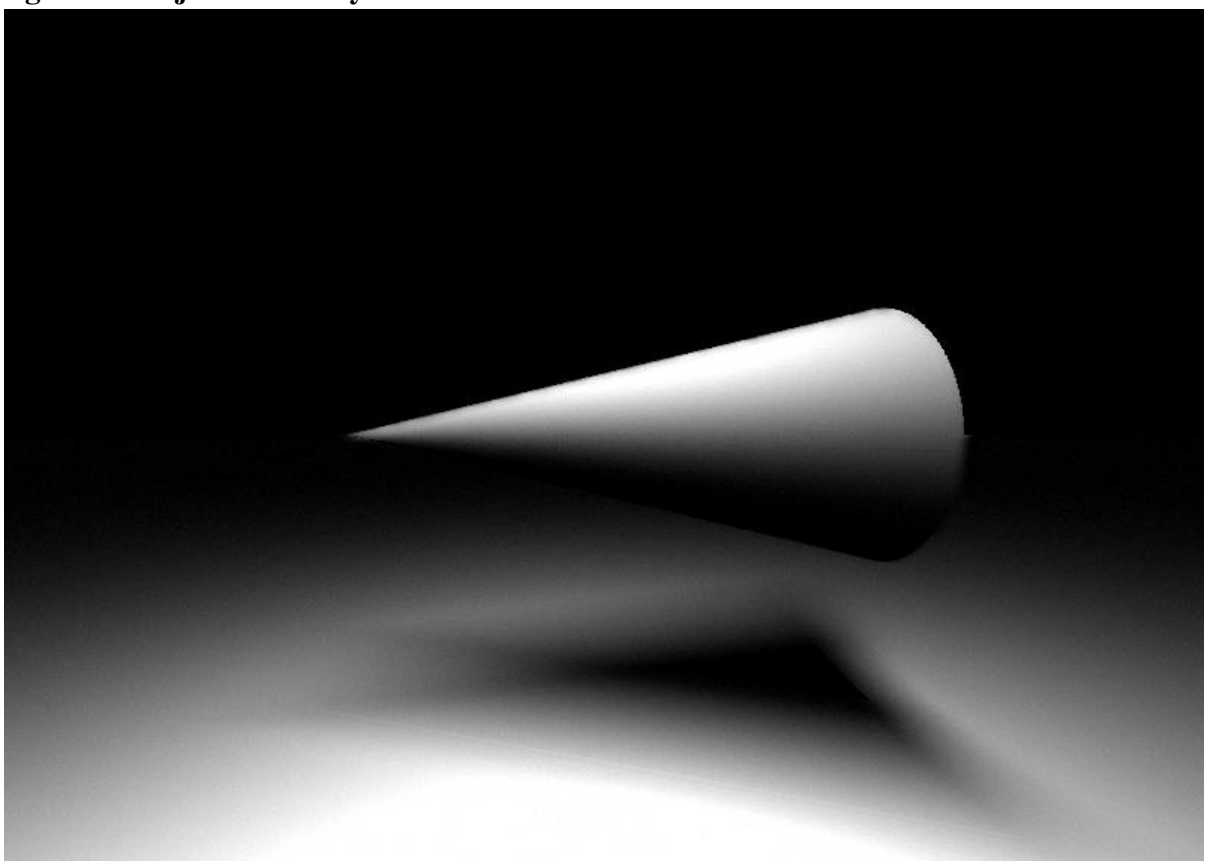
Once *t* has been identified, proceed as with the cylider to compute the *hitloc* and determine if the *hitloc* is within the constraints of the object.

Computing the normal vector

Given a surface specified as $f(x,y,z) = 0$. The *normal* at the point is found by evaluating the partial derivative of f with respect to x, y, and z at the *hitloc.* : $(\sqrt{x}, \sqrt{y}, \sqrt{z})$. Therefore the normal is:

$$N = (2h_x, -k, 2h_z)$$

Adding a cone object to the ray tracer



The finite length cone will be specified in the following way:

```
23              cone

0.2 0.2 0.2     ambient
2.0 2.0 2.0     diffuse
1.0 1.0 1.0     specular

4  3 -4         center point of finite cone base.
1 0 0           centerline vector
2 5             radius at base and height
```

```
typedef struct cone_type
{
   double  center[3];       /* center of base */
   double  centerline[3];
   double  radius;          /* radius of base */
   double  height;
   double  rotmat[3][3];
   double  irot[3][3];
}  cone_t;
```

Determining if a ray hits a cone.

Assume the following:

B = base or start of the ray
D = a unit vector in the direction the ray is traveling
C = center of the base of the cone
r = radius of the cone.
h = height of the cone

The arithmetic is much simpler if the vertex of the cone is at the origin. So we start by moving it there! To do so we must make a compensating adjustment to the base of the ray.

B′ = B - C = new base of ray
C′ = C - C = (0, 0, 0) = new center of cone
D does not change

The next step in the operation is to rotate the center line of the cone so that it lies on the y axis. Begin by constructing an identity matrix. If the centerline *already* lies on the y-axis there is nothing more to do. If not, procede to construct a rotation matrix as follows.

Rotate the centerline of the cone into the positive y axis
Rotate a vector orthogonal to both the centerline and the postive y axis into the positive x axis.
Exercise: How should you handle the upside down cone??

Apply the rotation to *B′* getting a new *B′* and to *D* getting a new *D*. Since *C′* is zero the rotation would not change it.

All points on the ray may be expressed in the form

$$P = B' + t\,D = (b'_x + td_x,\ b'_y + td_y,\ b'_z + td_z) \qquad (1)$$

A point P on the infinite cone which is aligned with the y-axis whose vertex is at (0, 0, 0) necessarily satisfies the following equation:

$$p_x^2 + p_z^2 = (rp_y/h)^2 = kp_y^2 \qquad (2)$$

That is, when $p_y = h$, $p_x^2 + p_z^2 = r^2$ as it should at the top of the cone. The presence of the *y* term on the right side of the equation does make the algebra a bit more messy than in other cases. Letting $k = (r/h)^2$ and doing the usual algebra we find:

$$a = d_x^2 + d_z^2 - k\,d_y^2$$
$$b = 2\,(b'_x d_x + b'_z d_z - k d_y\,b'_y)$$
$$c = b'^2_x + b'^2_z - kb'^2_y$$

to obtain the solution in the standard form of the quadratic formula:

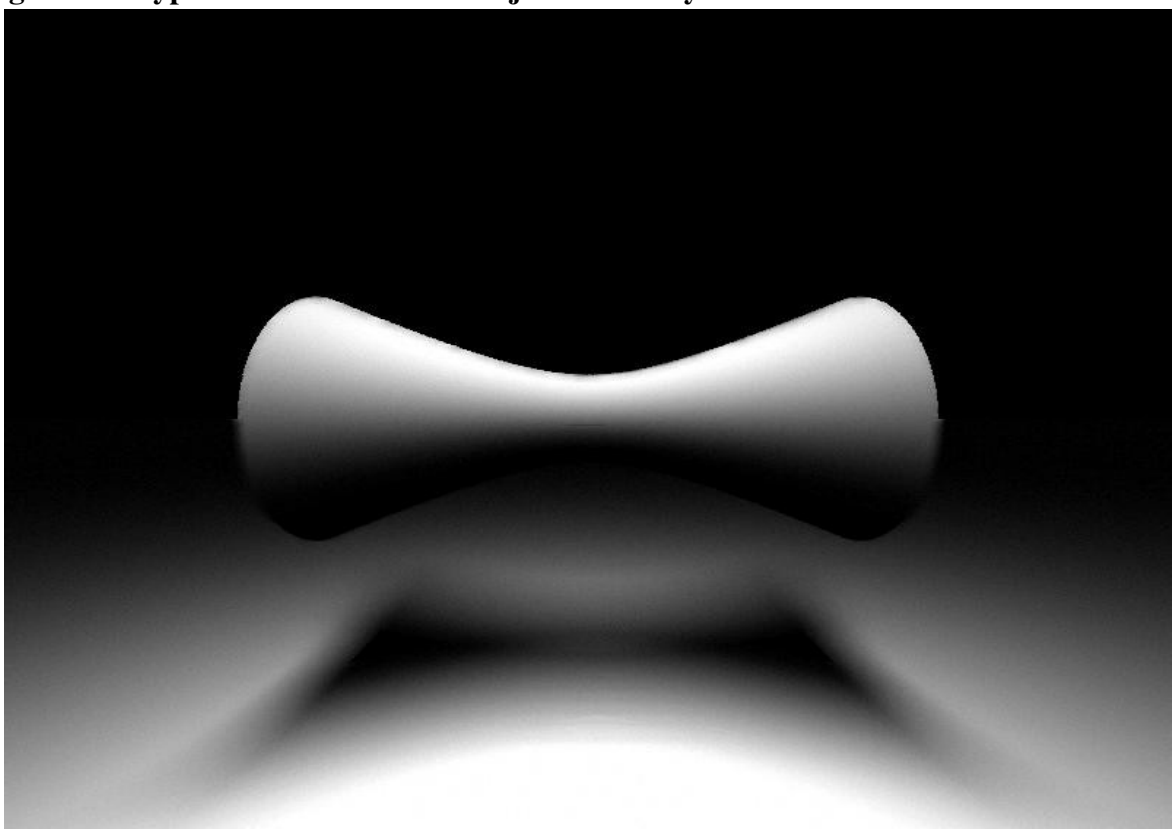
$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Computing a normal to the cone

Given a surface specified as $f(x,y,z) = 0$. The *normal* at the point is found by evaluating the partial derivative of f with respect to x, y, and z at the *hitloc.* : $(\sqrt{x}, \sqrt{y}, \sqrt{z})$. Therefore the normal is:

$$N = (2h_x, -2kh_y, 2h_z)$$

Adding a finite hyperboloid of one sheet object to the ray tracer



The finite length hyperboloid will be specified in the following way:

```
21              hyperboloid

0.2 0.2 0.2     ambient
2.0 2.0 2.0     diffuse
1.0 1.0 1.0     specular

4  3 -4         center point of finite circular hyperboloid base.
1 0 0           centerline vector
2 5             radius at ends and half-height
0.5             radius at center

typedef struct hyperb_type
{
   double  center[3];       /* center of base */
   double  centerline[3];
   double  radius;          /* radius of base */
   double  height;
   double  scale;           /* r^2 / h        */
   double  rotmat[3][3];
   double  irot[3][3];
}  hyperb_t;
```

Determining if a ray hits a circular hyperboloid.

Assume the following:

B = base or start of the ray
D = a unit vector in the direction the ray is traveling
C = center of the base of the hyperboloid
r = radius of the hyperboloid.
h = height of the hyperbloid

The arithmetic is much simpler if the center of the hyperboloid is at the origin. So we start by moving it there! To do so we must make a compensating adjustment to the base of the ray.

B′ = B - C = new base of ray
C′ = C - C = (0, 0, 0) = new center of cone
D does not change

The next step in the operation is to rotate the center line of the cone so that it lies on the y axis. Begin by constructing an identity matrix. If the centerline *already* lies on the y-axis there is nothing more to do. If not, procede to construct a rotation matrix as follows.

Rotate the centerline of the cone into the positive y axis
Rotate a vector orthogonal to both the centerline and the postive y axis into the positive x axis.
Exercise: How should you handle the upside down hyperboloid??

Apply the rotation to *B′* getting a new *B′* and to *D* getting a new *D*. Since *C′* is zero the rotation would not change it.

All points on the ray may be expressed in the form

$$P = B' + t\,D = (b'_x + td_x,\ b'_y + td_y,\ b'_z + td_z) \qquad (1)$$

A point P on the infinite hyperboloid which is aligned with the y-axis and whose vertex is at (0, 0, 0) necessarily satisfies the following equation:

$$p_x^2 + p_z^2 = kp_y^2 + r_c^2 \qquad (2)$$

Where r_c is the radius of the surface when $y = 0$. We want to specify the *radius, r,* at the top of the hyperboloid. When $p_y = h$ we want $kh^2 + r_c^2 = r^2$. Thus $k = (r^2 - r_c^2)/h^2$. We compute the *k* value at object initialization time and save it in *hyperb->scale.*

$$a = (d_x^2 + d_z^2 - kd_y^2)$$
$$b = 2\,(b'_x d_x + b'_z d_z - kb'_y d_y)$$
$$c = (b'^2_x + b'^2_z) - kb'^2_y - r_c^2$$

to obtain the solution in the standard form of the quadratic formula:

$$t = \frac{-b +/- \sqrt{b^2 - 4ac}}{2a}$$

Once *t* has been identified, proceed as with the cylider to compute the *hitloc* and determine if the *hitloc* is within the constraints of the object.

Computing the normal vector

Given a surface specified as $f(x,y,z) = 0$. The *normal* at the point is found by evaluating the partial derivative of f with respect to x, y, and z at the *hitloc.* : $(\sqrt{x}, \sqrt{y}, \sqrt{z})$. Therefore the normal is:

$$N = (2h_x, -2kh_y, 2h_z)$$