Curved Surfaces

Curved Surfaces

- Equations for objects with curved boundaries can be expressed in either a parametric or a nonparametric form.
- The various objects that are often useful in graphics applications include quadric surfaces, superquadrics, polynomial and exponential functions, and spline surfaces.
- These input object descriptions typically are tessellated to produce polygon-mesh approximations for the surfaces.

Quadric Surfaces

- A frequently used class of objects are the *quadric surfaces*, which are described with second-degree equations (quadratics).
- They include spheres, ellipsoids, tori, paraboloids, and hyperboloids.
- Quadric surfaces, particularly spheres and ellipsoids, are common elements of graphics scenes, and routines for generating these surfaces are often available in graphics packages.

Sphere

- In Cartesian coordinates, a spherical surface with radius *r* centered on the coordinate origin is defined as the set of points *(x, y, z)* that satisfy the equation

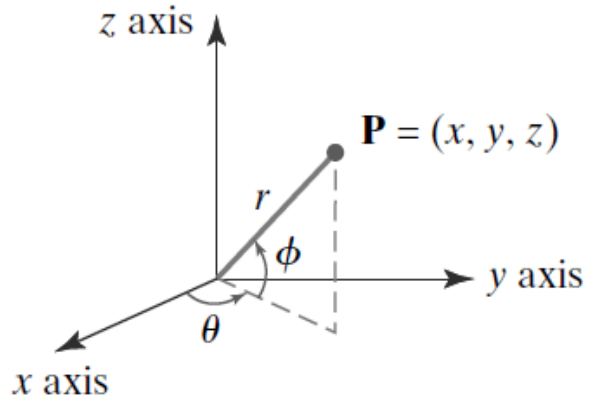
$$x^2 + y^2 + z^2 = r^2$$

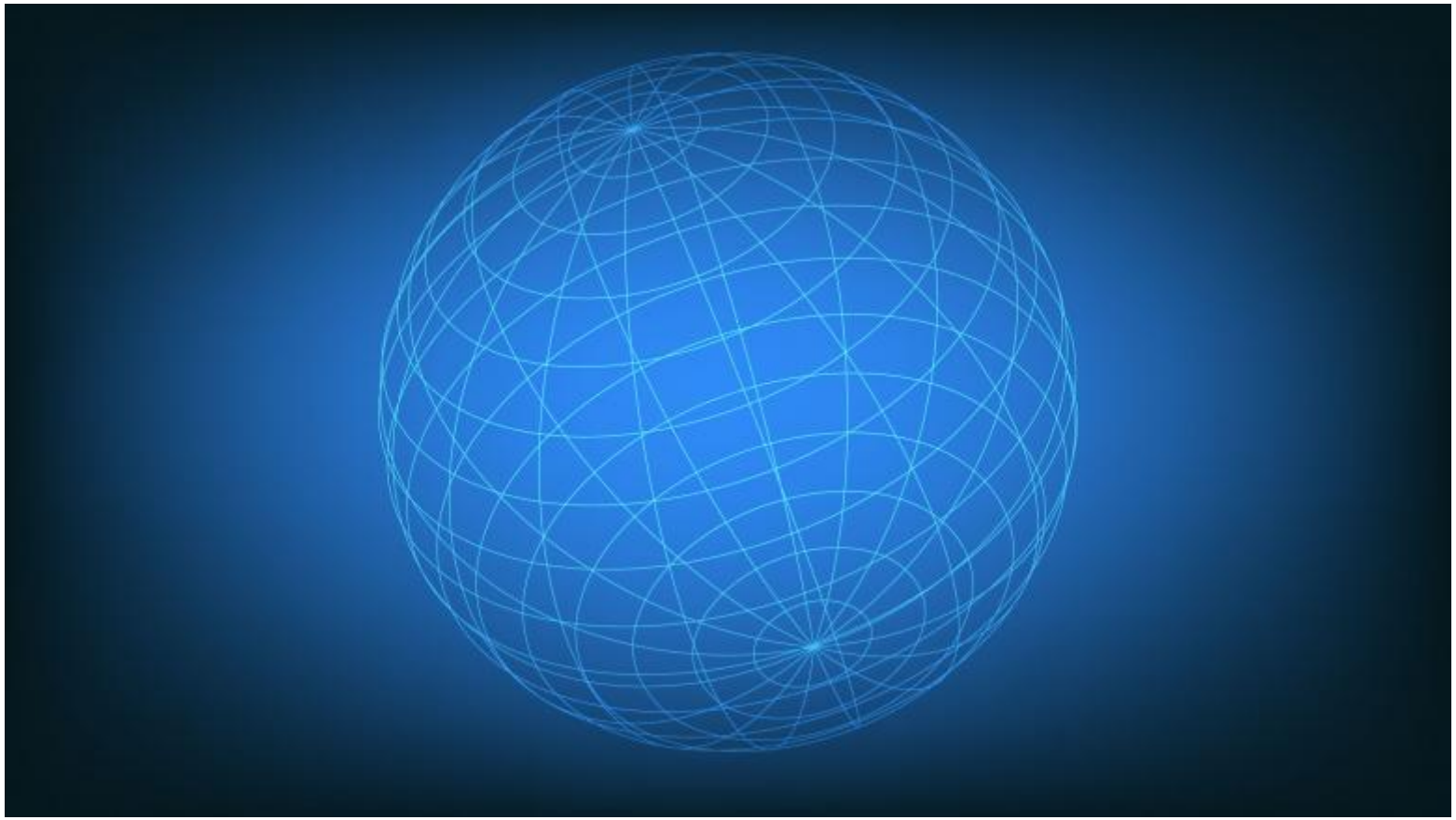
- the spherical surface in parametric form, using latitude and longitude angles

$$x = r\cos\phi\cos\theta, \quad -\pi/2 \leq \phi \leq \pi/2$$

$$y = r\cos\phi\sin\theta, \quad -\pi \leq \theta \leq \pi$$

$$z = r\sin\phi$$





GLUT Quadric-Surface Functions

- We generate a GLUT sphere with either of these two functions:

  - **glutWireSphere (r, nLongitudes, nLatitudes);**

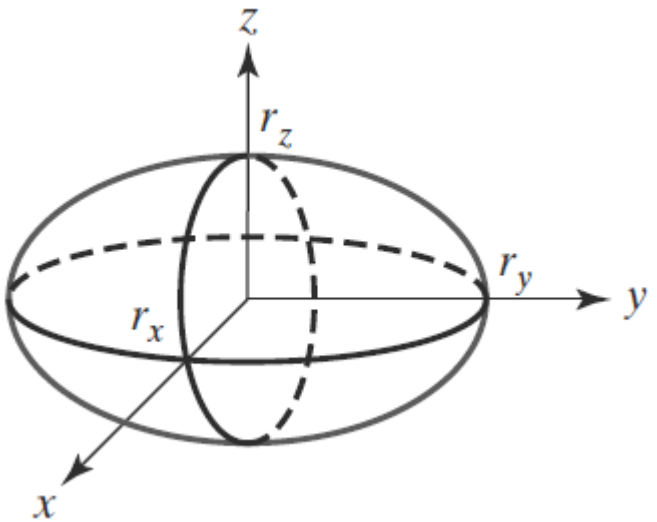
  - **glutSolidSphere (r, nLongitudes, nLatitudes);**

- where the sphere radius is determined by the double-precision floating-point number assigned to parameter **r**.
- Parameters **nLongitudes** and **nLatitudes** are used to select the integer number of longitude and latitude lines that will be used to approximate the spherical surface as a quadrilateral mesh

Ellipsoid

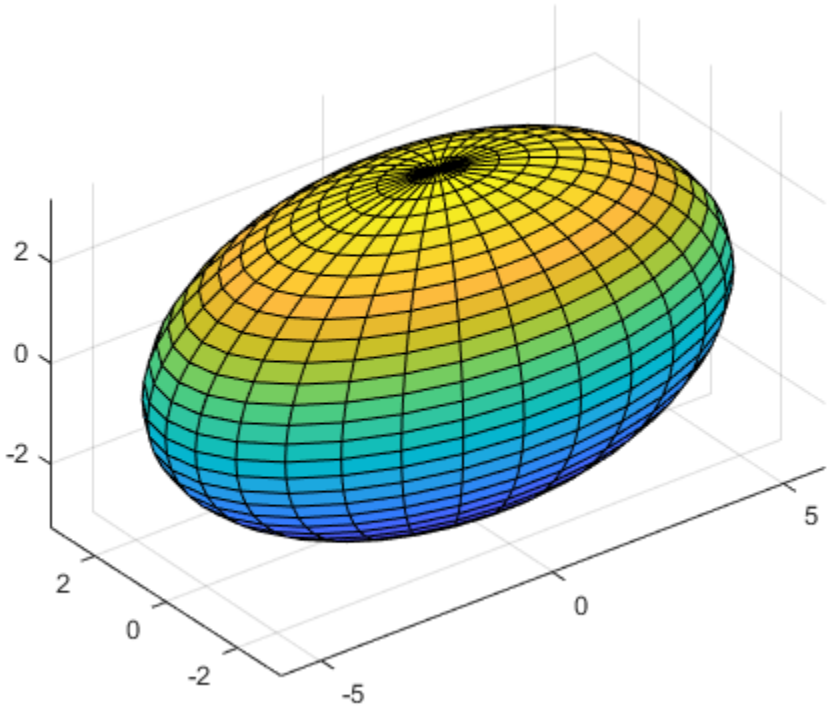
- An ellipsoidal surface can be described as an extension of a spherical surface where the radii in three mutually perpendicular directions can have different values (Figure 4). The Cartesian representation for points over the surface of an ellipsoid centered on the origin is

$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 = 1$$



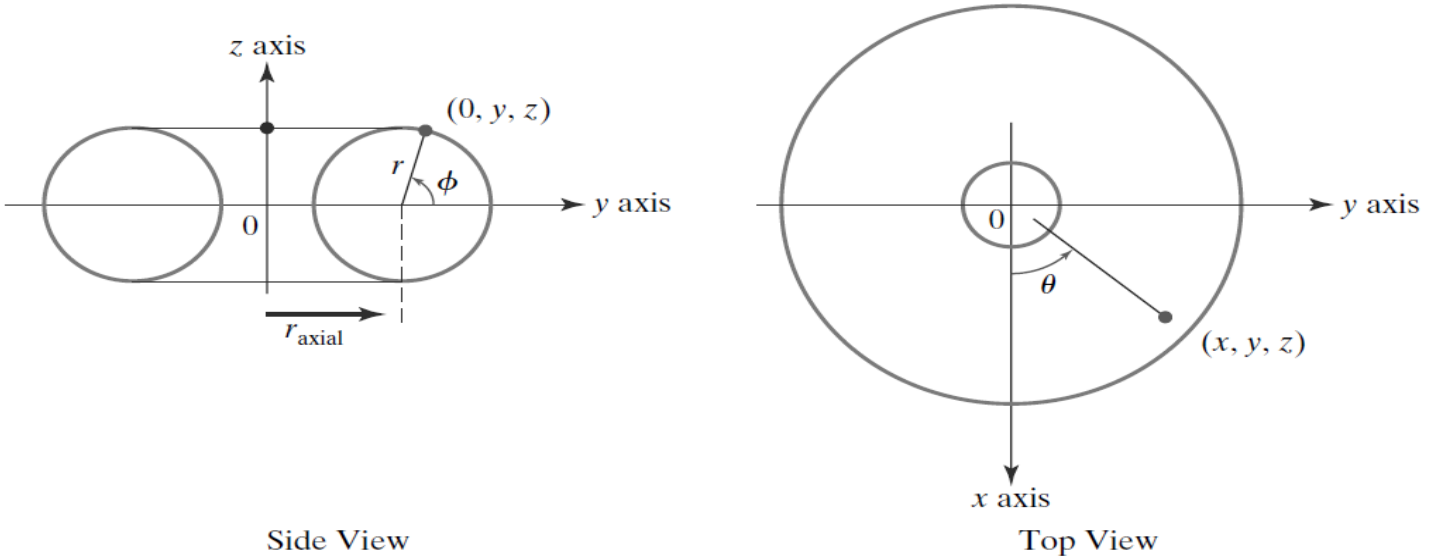
- And a parametric representation for the ellipsoid in terms of the latitude angle φ and the longitude angle ϑ in Figure 2 is

$$x = r_x \cos\phi \cos\theta, \qquad -\pi/2 \le \phi \le \pi/2$$
$$y = r_y \cos\phi \sin\theta, \qquad -\pi \le \theta \le \pi$$
$$z = r_z \sin\phi$$



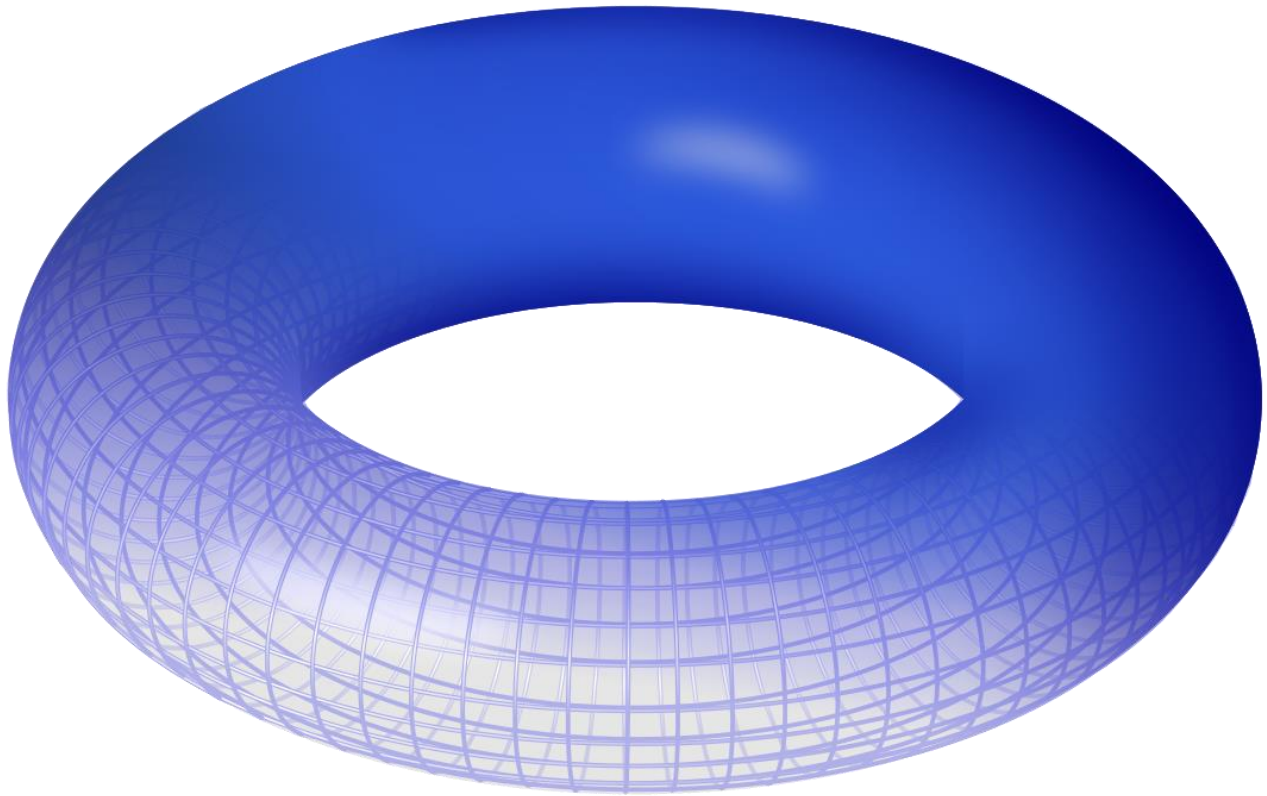
Torus

- A doughnut-shaped object is called a *torus* or *anchor ring*.
- Most often it is described as the surface generated by rotating a circle or an ellipse about a coplanar axis line that is external to the conic.
- The defining parameters for a torus are then the distance of the conic center from the rotation axis and the dimensions of the conic.
- A torus generated by the rotation of a circle with radius *r* in the *yz* plane about the *z* axis is shown in Figure.
-



Side View

Top View



- **glutWireTorus (rCrossSection, rAxial, nConcentrics, nRadialSlices);**

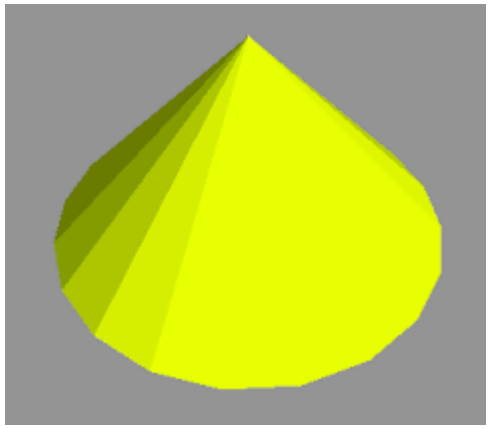
- **glutSolidTorus (rCrossSection, rAxial, nConcentrics, nRadialSlices);**

- The torus obtained with these GLUT routines can be described as the surface generated by rotating a circle with radius **rCrossSection** about the coplanar *z* axis, where the distance of the circle center from the *z* axis is **rAxial**
- Parameter **nConcentrics** specifies the number of concentric circles (with center on the *z* axis) to be used on the torus surface
- parameter **nRadialSlices** specifies the number of radial slices through the torus surface.

- **glutWireCone (rBase, height, nLongitudes, nLatitudes);**

- **glutSolidCone (rBase, height, nLongitudes, nLatitudes);**

- We set double-precision, floating-point values for the radius of the cone base and for the cone height using parameters **rbase** and **height**, respectively.
- parameters **nLongitudes** and **nLatitudes** are
- assigned integer values that specify the number
- of orthogonal surface lines for the quadrilateral
- mesh approximation.



- We can display the teapot, as a mesh of over 1,000 bicubic surface patches, using either of the following two GLUT functions:
    - **glutWireTeapot (size);**
    - **glutSolidTeapot (size);**
- The teapot surface is generated using OpenGL B´ezier curve functions
- Parameter **size** sets the double-precision floating-point value for the maximum radius of the teapot bowl. The teapot is centered on the world-coordinate origin coordinate origin with its vertical axis along the *y* axis.