$$C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \qquad D = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

The coefficients of the plane equation can also be obtained using a vector cross product calculation. Here, we have to select vertex positions V_1, V_2 and V_3 in counter clockwise order when viewing the surface from outside to inside in a right handed cartesian system. With three vertex positions we can form two vectors : V_1 and V_2 and V_1 to V_3. The cross product of these vectors is

$$N = (V_2 - V_1) \times (V_3 - V_1)$$

This cross product generates values for coefficients A, B, and C for the plane equation. We can then obtain the value for parameter D by substituting these values and the co-ordinates for one of the polygon vertices in the plane equation.

It is important to note that the resultant vector N is the normal vector to the surface. The plane equation can be expressed in vector form using the normal N and the position P of any point in the plane as

$$N \cdot P = -D$$

For any point (x, y, z) with parameters A, B, C, D we can say that -

- Point is not on the plane if $A_x + B_y + C_z + D \neq 0$
- Point is inside the surface if $A_x + B_y + C_z + D < 0$
- Point is outside the surface if $A_x + B_y + C_z + D > 0$

6.4.1.3 Polygon Meshes

A polygon mesh is a collection of edges, vertices, and polygons connected such that each edge is shared by at most two polygons. An edge connects two vertices, and a polygon is a closed sequence of edges. An edge can be shared by two adjacent polygons and a vertex is shared by at least two edges. A polygon mesh can be represented in several ways. In the following sections we are going to discuss three polygon-mesh representations : explicit, pointers to a vertex list, and pointers to an edge list.

The application programmer has to apply space and time criteria to evaluate different representations and select the best suitable representation. Typically, we have to perform following operations on the polygon mesh :

- Finding the vertices connected by an edge,
- Finding the edges of the polygon
- Displaying the mesh, and
- Identifying errors in representation

When the relations among polygons, vertices, and edges are explicitly defined then the time required for above operations is less; however the explicit definition requires more space.

Explicit Representation : In the explicit representation, each polygon is represented by a list of vertex co-ordinates :

$$P = ((x_1, y_1, z_2), (x_2, y_2, z_2), \dots, (x_n, y_n, z_n))$$

The vertices are stored in the sequence such that there are edges between successive vertices in the list and between the last and first vertices. For single polygon this method of representation is space efficient; however for polygon mesh it requires more space because the co-ordinates of shared vertices are duplicated. This representation does not represent shared edges and vertices explicitly. Therefore, it is necessary to compare polygon co-ordinates with other polygons to get the information about shared edge and vertices which is time consuming. In this representation, each polygon is displayed separately, thus shared edges are drawn twice. This causes problems on pen plotters, film recorders, and vector displays due to the overwritting. A problem may also be created on raster displays if the edges are drawn in opposite directions.

Pointer to Vertex List Representation

In this representation, polygons are defined with pointers to a vertex list. Each vertex in the polygon mesh is stored once in the vertex list $V = ((x_1, y_1, z_1), (x_2, y_2, z_2), \dots (x_n, y_n, z_n))$. The individual polygons are then defined by the list of pointers into the vertex list. A polygon made up of vertices 2, 3 and 5 is represented as P = (2, 3, 5). Here, each vertex is stored once, so considerable space is saved, and it is easier to change the co-ordinates of vertex easily. However, in this method it is still difficult to find polygons that share an edge, and shared edges are still drawn twice when all polygon outlines are displayed.

Pointer to Edge List Representation

The difficulties in the above two representation can be eliminated by representing edges explicitly. In this representation, polygons are defined by pointers to an edge list. Each edge in the edge list occurs just once and it points to the two vertices in the vertex list defining the edge, and also to the one or two polygon to which the edge belongs. Here, polygon is defined as $P = (E_1, E_2, \dots E_n)$ and an edge as $E = (V_1, V_3, P_1, P_2)$.

6.4.1.4 Consistency of Polygon-Mesh Representation

In all the three representations discussed above, we have to make sure that all polygons are closed, all edges are used at least once, and each vertex is referenced by at least two edges. This check is called consistency check. Amongst three representations, the explicit edge scheme is the easiest to check for consistency, because it contains the most information.

When the object surfaces are to be tiled it is more convenient to specify the surface facets with a mesh function. One type of polygon mesh is the **triangle strip**. This function produces n – 2 connected triangles, given the co-ordinates for n vertices. Other function is quadrilateral mesh, which generates a mesh of (n – 1) or (m – 1) quadrilaterals, given the co-ordinates for an n by m array of vertices. When polygons are specified with more than three vertices, it is possible that the vertices may not at all lie in one plane. This can be due to numerical errors or errors in selecting co-ordinate positions for the vertices. One way to handle this situation is simply to divide the polygons into triangles. Another approach is that sometimes taken is to approximate the plane parameters. A, B and C. We can do this by averaging methods or we can project the polygon onto the co-ordinate planes. Using the projection method, we take A proportional to the area of the polygon projection on the yz plane, B proportional to the projection area of on the xz plane and C proportional to the projection area on the xy plane.

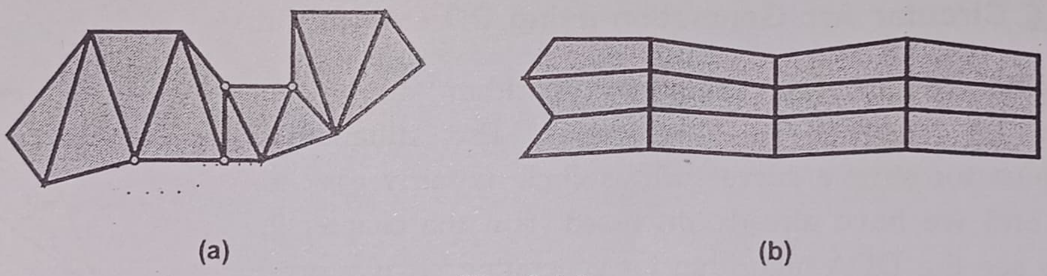

Fig. 6.4.2

The Fig. 6.4.2 (a) and (b) show a triangle strip formed with 11 triangles connecting 13 vertices. A quadrilateral mesh containing 12 quadrilaterals constructed from a 5 by 4 input vertex array.

6.4.2 Curved Lines and Surfaces

Three-dimensional curved lines and surfaces can be generated and displayed -

- From an input set of mathematical functions defining the object or
- From a set of user specified data points.

When functions are specified, a graphics package defines the equations for a curve and plots pixel positions along the path of the defined equation. In case of surfaces, a functional description is used to produce a polygon-mesh approximation to the surface. Usually, this is done with triangular strips to ensure that all vertices of any polygon are in one plane.

When a set of discrete co-ordinate points is used to specify an object shape, a spline representation or curve-fitting methods are used to display objects.

The equations used to represent curves or surfaces can be expressed in either a parametric form or nonparametric form. In computer graphics applications, parametric representations are used because they are more convenient.

6.4.3 Quadric Surfaces

Quadric surfaces are most commonly used class of objects. The reasons for frequent use of quadric surface are :

- Ease of computing the surface normal,
- Ease of testing whether a point is on the surface,
- Ease of computing z if x and y are given and
- Ease of calculating intersections of one surface with other.

Quadric surfaces are described with second degree equations. They include spheres, ellipsoids, tori, paraboloids and hyperboloids. Let us see how to represent commonly used quadric surfaces, spheres and ellipsoids, in parametric form.

Sphere

In cartesian co-ordinates, the equation of spherical surface with radius r centered on the co-ordinate origin is given as

$$x^2 + y^2 + z^2 = r^2 \qquad \text{... (6.4.1)}$$

In parametric form, the spherical surface using latitude and longitude angles can be described as

$$\begin{aligned} x &= r \cos\phi \cos\theta, & -\pi/2 \le \phi \le \pi/2 \\ y &= r \cos\phi \sin\theta, & -\pi \le \theta \le \pi \\ z &= r \sin\phi \end{aligned} \qquad \text{... (6.4.2)}$$

This is illustrated in Fig. 6.4.3.

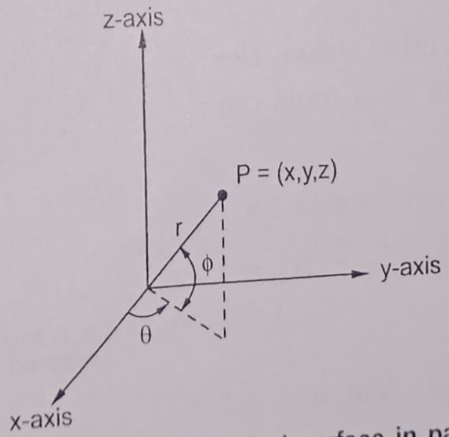

Fig. 6.4.3 Representation of spherical surface in parametric form

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, as shown in the Fig. 6.4.4.

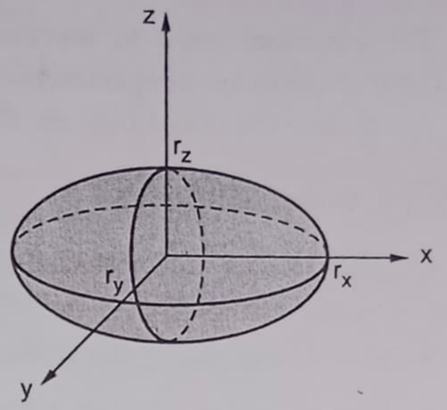

Fig. 6.4.4

In cartesian co-ordinate, the ellipsoid surface can be represented as

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

A parametric representation for the ellipsoid in terms of the latitude angle ϕ and the longitude angle θ can be given as

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

Torus

The Fig. 6.4.5 shows torus. It is a doughnut-shaped object generated by rotating a circle or other conic about a specified axis.

In cartesian co-ordinate, equation for points over the surface of a torus is given as

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

where r is any given offset value. The parametric representations for a torus can be given as

$$x = r_x(r+\cos\phi)\cos\theta, \quad -\pi\le\phi\le\pi$$
$$y = r_y(r+\cos\phi)\sin\theta, \quad -\pi\le\theta\le\pi$$
$$z = r_z \sin\phi$$

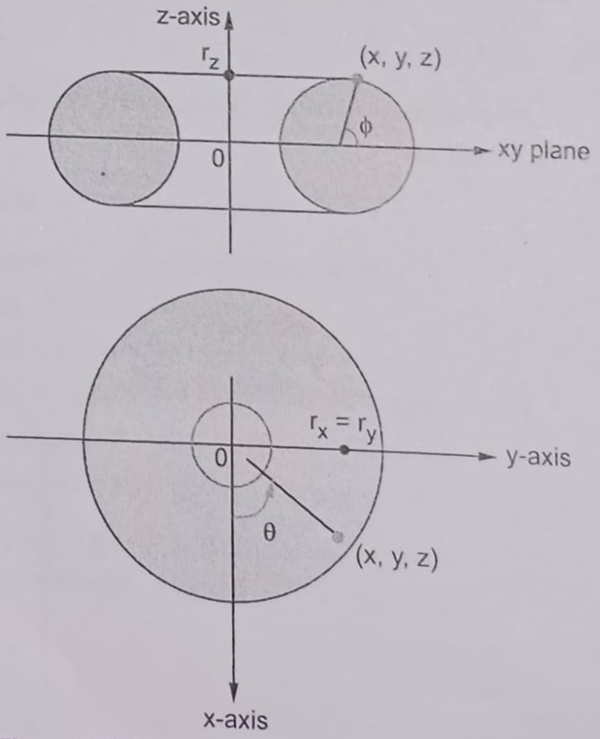

Fig. 6.4.5 A torus with a circular cross-section centered on the coordinate origin

where ϕ and θ are latitude and longitude angles, respectively and angle θ extends over 360°.