UNIT-4 (Curves and Surfaces)

In computer graphics, we often need to draw different types of objects onto the screen. Objects are not flat all the time and we need to draw curves many times to draw an object. A curve is an infinitely large set of points. Each point has two neighbors except endpoints.

The curves are of two types:
1. Non-parametric Curve
2. Parametric Curve

1. **Non-parametric Curve:** Curves can be defined mathematically by equations. Nonparametric equations can be explicit or implicit. For a nonparametric curve, the coordinates y and z of a point on the curve are expressed as two separate functions of the third coordinate x as the independent variable.
   The curve can be represented as a relationship between x, y and z.
   f(x,y,z) = 0

   Example:
   For circle: $ax^2 + by^2 + 2kxy + 2gx + 2hy + c = 0$

   For ellipse: $\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0$

   Implicit Curves: Implicit curve representations define the set of points on a curve by employing a procedure that can test to see if a point in on the curve. Usually, an implicit curve is defined by an implicit function of the form –

   f(x,y) = 0

   It can represent multivalued curves *multiple y values for an x value*. A common example is the circle, whose implicit representation is

   $$x^2 + y^2 - r^2 = 0$$

   Explicit Curve: A mathematical function y = f(x) can be plotted as a curve. Such a function is the explicit representation of the curve. The explicit representation is not general, since it cannot represent vertical lines and is also single-valued. For each value of x, only a single value of y is normally computed by the function.

   Disadvantages using non-parametric curve: There are three problems with describing curves using nonparametric equations:
   a. If the slope of a curve at a point is vertical or near vertical, its value becomes infinity or very large.
   b. Shapes of most engineering objects are intrinsically independent of any coordinate system.
   c. If the curve is to be displayed as a series of point or straight-line segments, the computations involved could be extensive.

   Limitations of non-parametric Curves:
   - It cannot be used for representing close curves like circle and multivalued curves like ellipse, parabolas and hyperbolas.
   - If the straight line is vertical, the slope is infinity. Such values are difficult to handle in computation.

- Equations are required to be solved simultaneously.
- The equation of the curve depends upon the coordinate system.

2. **Parametric Curve:** The explicit and implicit curve representations can be used only when the function is known. In parametric form, each point on a curve is expressed as a function of a parameter u. In practice the parametric curves are used.

A two-dimensional parametric curve has the following form –

$$p(t) = f(t), g(t)$$
$$\text{or } p(t) = x(t), y(t)$$

The functions f and g become the x,y coordinates of any point on the curve, and the points are obtained when the parameter t is varied over a certain interval [a, b], normally [0, 1].

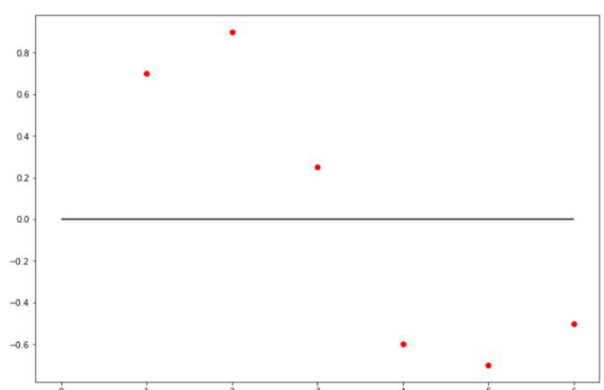
In general, the curve is not represented as the relationship between x, y and z rather it is represented as a function of independent parameters such as: u or ɵ.

Form: $x = x(u)$, $y = y(u)$, $z = z(u)$
Example:
Circle: $x = r\cos\theta$
$y = r\sin\theta$

Advantages of Parametric Curve: The advantages of parametric curves are as follows:
- It is used for closed and multivalued curves.
- It can replace slope by tangent vectors.
- For conic and curves, the parametric equation uses polynomial rather than solving roots.
- It is suitable for partial curve.

Interpolation:

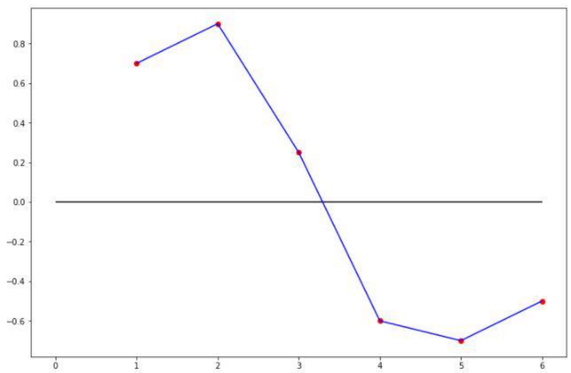
We estimate *f(x)* for arbitrary *x*, by drawing a smooth curve through the x_i. If the desired *x* is between the largest and smallest of the x_i then it is called interpolation, otherwise, it is called **Extrapolation**.



Random Points

1. **Linear Interpolation:**

   Linear Interpolation is a way of curve fitting the points by using linear polynomial such as the equation of the line. This is just similar to joining points by drawing a line b/w the two points in the dataset.



Linear Interpolation

2. **Polynomial Interpolation:**

   Polynomial Interpolation is the way of fitting the curve by creating a higher degree polynomial to join those points.

3. **Spline Interpolation:**

   Spline interpolation similar to the Polynomial interpolation *x'* uses low-degree polynomials in each of the intervals and chooses the polynomial pieces such that they fit smoothly together. The resulting function is called a spline.

Spline Specifications:
There are three equivalent methods for specifying a particular spline representation:

1. We can state the set of boundary conditions that are imposed on the spline
2. We can state the matrix that characterizes the spline
3. We can state the set of blending functions (or basis functions) that determine how specified geometric constraints on the curve are combined to calculate positions along the curve path.

To illustrate these three equivalent specifications, suppose we have the following parametric cubic polynomial representation for the x coordinate along the path of a spline section:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x, \qquad 0 \le u \le 1 \text{ --------------------------(1)}$$

Boundary conditions for this curve might be set, for example, on the endpoint coordinates *x(0) and x(1)* and on the parametric first derivatives at the endpoints *x'(0)* and *x' (1)*. These four boundary conditions are sufficient to determine the values of the four coefficients a_x, b_x, c_x and d_x.

From the boundary conditions, we can obtain the matrix that characterizes this spline curve by first rewriting Eq. 1 as the matrix product

$$x(u) = [u^3 \ \ u^2 \ \ u \ \ 1] \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix} \text{ ------------------------- (2)}$$

$$=U.C$$

Where U is the row matrix of powers of parameter **u,** and C is the coefficient column matrix. Using **Eq.** 2, we can write the boundary conditions in matrix form and solve for the coefficient matrix C as

$$C = M_{spline}.M_{geom} \text{ -------------------------- (3)}$$

Where M_{geom} is a four-element column matrix containing the geometric constraint values (boundary conditions) on the spline; and M_{spline} is the 4-by-4 matrix that transforms the geometric constraint values to the polynomial coefficients and provides a characterization for the spline curve. Matrix M_{geom} contains control point coordinate values and other geometric constraints that have been specified. Thus, we can substitute the matrix representation for C into Eq. 2 to obtain

$$x(u) = U.M_{spline}.M_{geom} \text{ ------------------------------------ (4)}$$

The matrix, M_{spline}, characterizing a spline representation, sometimes called the basis matrix, is particularly useful for transforming from one spline representation to another.

Finally, we can expand Eq. 4 to obtain a polynomial representation for coordinate x in terms of the geometric constraint parameters

$$x(u) = \sum_{k=0}^{3} g_k.BF_k(u) \text{ -------------------------------- (5)}$$

where g_k are the constraint parameters, such as the control-point coordinates and slope of the curve at the control points, and $BF_k(u)$ are the polynomial blending functions.

1. **CUBIC SPLINE INTERPOLATION METHOD:**
   This class of splines is most often used to set up paths for object motions or to provide a representation for an existing object or drawing, but interpolation splines are also used sometimes to design object shapes. Cubic polynomials offer a reasonable compromise between flexibility and speed of computation. Compared to higher-order polynomials, cubic splines require less calculations and memory and they are more stable. Compared to lower-order polynomials, cubic splines are more flexible for modeling arbitrary curve shapes.

   Cubic spline interpolation is a way of finding a curve that connects data points with a degree of three or less. Splines are polynomials that are smooth and continuous across a given plot and also continuous first and second derivatives where they join.

$$p_k = (x_k, y_k, z_k), \qquad k = 0,1,2, \dots \dots \dots . n$$

Fig 1: A piecewise continuous cubic spline interpolation of n+1 points

Cubic interpolation fit of these points is illustrated in Fig. 1. We can describe the parametric cubic polynomial that is to be fitted between each pair of control points with the following set of equations:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$
$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y$$

$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$$

For each of these three equations, we need to determine the values of the four coefficients a, b, c, and ***d*** in the polynomial representation for each of the n curve sections between the ***n*** + 1 control points. We do this by setting enough boundary conditions at the "joints" between curve sections so that we can obtain numerical values for all the coefficients. Common methods for setting the boundary conditions for cubic interpolation splines are as follows:

a. **Natural Cubic Splines:**

One of the first spline curves to be developed for graphics applications is the **nat**ural cubic **spline. This** interpolation curve is a mathematical representation of the original drafting spline. We formulate a natural cubic spline by requiring that two adjacent curve sections have the same first and second parametric derivatives at their common boundary. Thus, natural cubic splines have C^2 continuity.

If we have n + 1 control points to fit, as in Fig. 1, then we have n curve sections with a total of 4n polynomial coefficients to be determined. At each of the n - 1 interior control points, we have four boundary conditions: The two curve sections on either side of a control point must have the same first and second parametric derivatives at that control point, and each curve must pass through that control point. This gives us 4n - 4 equations to be satisfied by the 4n polynomial coefficients. We get an additional equation from the first control point p_0 , the position of the beginning of the curve, and another condition from control point p_n, which must be the last point on the curve. We still need two more conditions to be able to determine values for all coefficients. One method for obtaining the two additional conditions is to set the second derivatives at p_0 and p_n, to 0. Another approach is to add two extra "dummy" control points, one at each end of the original control-point sequence. That is, we add a control point p_{-1} and a control point p_{n+1}, Then all of the original control points are interior points, and we have the necessary 4n boundary conditions.

Although natural cubic splines are a mathematical model for the drafting spline, they have a major disadvantage. If the position of any one control point is altered, the entire curve is affected. Thus, natural cubic splines allow for no "local control", so that we cannot restructure part of the curve without specifying an entirely new set of control points.

b. Hermite Interpolation:

Hemite spline (named after the French mathematician Charles Hermite) is an interpolating piecewise cubic polynomial with a specified tangent at each control point. Unlike the natural cubic splines, Hermite splines can be adjusted locally because each curve section is only dependent on its endpoint constraints.

If P(u) represents a parametric cubic point function for the curve section between control points p_k and p_{k+1}, as shown in fig 2, then the boundary conditions that define this Hermite curve section are:

$$P(0) = p_k$$
$$P(1) = p_{k+1}$$
$$P'(0) = Dp_k$$

$$P'(1) = Dp_{k+1}$$

with Dp_k and Dp_{k+1} specifying the values for the parametric derivatives (slope of the curve) at control points p_k and p_{k+1}, respectively.

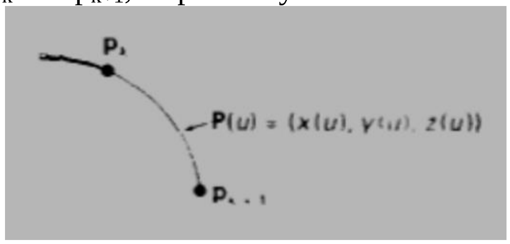


Fig 2: Parametric point function P(u) for a Hermite Curve section between control points p_k and p_{k+1}

We can write the vector equivalent of Eqs. 1 for this Hermite-curve section as:

$$P(u) = au^3 + bu^2 + cu + d \qquad 0 \le u \le 1$$

Where the x component of P(u) is $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$ and similarly for the y and z components. The equivalent matrix is

$$P(u) = [u^3 \;\; u^2 \;\; u \;\; 1] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

and the derivative of thin point function can be expressed as

$$P(u) = [3u^2 \;\; 2u \;\; 1 \;\; 0] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Substituting endpoint values 0 and 1 for parameter u Into the previous two equations, we can express the Hermite boundary conditions in the matrix form:

$$\begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Solving this equation for the polynomial coefficients, we have

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}$$

$$= \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}$$

$$= M_H \cdot \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}$$

where M_H, the Hermite matrix, is the inverse of the boundary constraint matrix. Equation can thus be written in terms of the boundary conditions as

$$P(u) = [u^3 \;\; u^2 \;\; u \;\; 1].M_H.\begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}$$

Finally, we can determine expressions for the Hermite blending functions by carrying out the matrix multiplications.

Hermite polynomials can be useful for some digitizing applications where it may not be too difficult to specify or approximate the curve slopes. But for most problems in computer graphics, it is more useful to generate spline curves without requiring input values for curve slopes or other geometric information, in addition to control-point coordinates.

c. **Cardinal Splines:**

As with Hermite splines, cardinal splines are interpolating piecewise cubics with specified endpoint tangents at the boundary of each curve section. The difference is that we do not have to give the values for the endpoint tangents. For a cardinal spline, the value for the slope at a control point is calculated from the coordinates of the two adjacent control points.

A cardinal spline section is completely specified with four consecutive control points. The middle two control points are the section endpoints, and the other two points are used in the calculation of the endpoint slopes. If we take **P(u)** as the representation for the parametric cubic point function for the curve section between control points p_k and p_{k+1}**,** as in Fig. 3, then the four control points from p_{k-1}, to p_{k+1}, are used to set the boundary conditions for the cardinal spline section as

$$P(0) = p_k$$
$$P(1) = p_{k+1}$$
$$P'(0) = \frac{1}{2}(1-t)(p_{k+1} - p_{k-1})$$
$$P'(1) = \frac{1}{2}(1-t)(p_{k+2} - p_k)$$

Thus, the slopes at control points pk and p,,, are taken to be proportional, respectively, to the chords $p_{k-1}p_{k+1}$, and p. Parameter t is called the tension parameter since it controls how loosely or tightly the cardinal spline fits the input control points.

2. **Bezier Curve:**

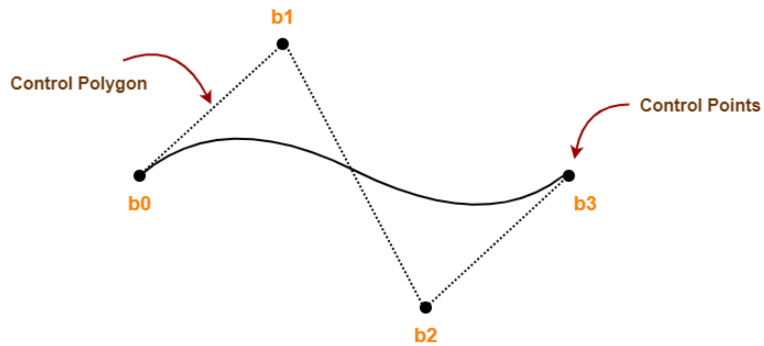
Bezier Curve may be defined as-

- Bezier Curve is parametric curve defined by a set of control points.
- Two points are ends of the curve.
- Other points determine the shape of the curve.

The following curve is an example of a Bezier curve-

Here,

- This bezier curve is defined by a set of control points b_0, b_1, b_2 and b_3.
- Points b_0 and b_3 are ends of the curve.
- Points b_1 and b_2 determine the shape of the curve.

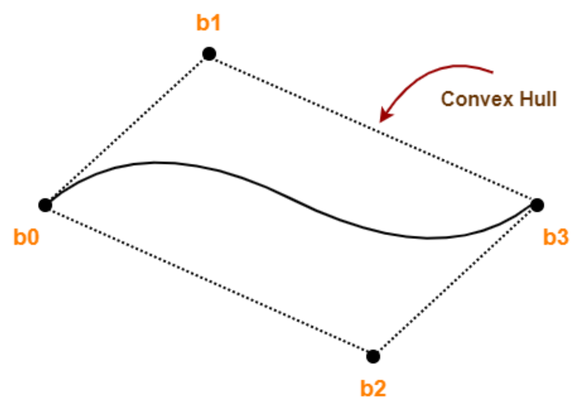


Bezier Curve Example

Bezier Curve Properties:

Few important properties of a Bezier curve are-

a. **Property 1:**
   - Bezier curve is always contained within a polygon called as convex hull of its control points.



Bezier Curve With Convex Hull

b. **Property 2:**
   - Bezier curve generally follows the shape of its defining polygon.
   - The first and last points of the curve are coincident with the first and last points of the defining polygon.

c. **Property 3:**
   The degree of the polynomial defining the curve segment is one less than the total number of control points.

   Degree = Number of Control Points - 1

d. **Property 4:**
   The order of the polynomial defining the curve segment is equal to the total number of control points.

e. **Property 5:**
   - Bezier curve exhibits the variation diminishing property.
   - It means the curve do not oscillate about any straight line more often than the defining polygon.

Bezier Curve Equation: A bezier curve is parametrically represented by-

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t)$$

Bezier Curve Equation

Here,

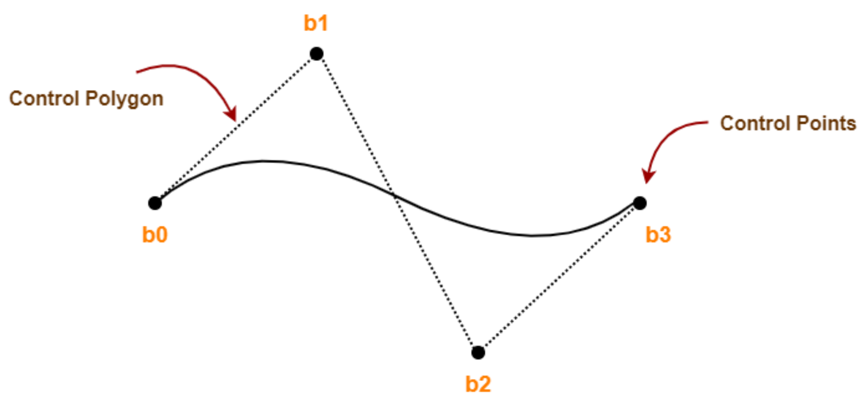
- t is any parameter where 0 <= t <= 1
- P(t) = Any point lying on the bezier curve
- B_i = i^{th} control point of the bezier curve
- n = degree of the curve
- $J_{n,i}(t)$ = Blending function = $C(n,i)t^i(1-t)^{n-i}$ where C(n,i) = n! / i!(n-i)!

Cubic Bezier Curve-

- Cubic Bezier curve is a Bezier curve with degree 3.
- The total number of control points in a cubic Bezier curve is 4.

EXAMPLE:

The following curve is an example of a cubic Bezier curve-



Cubic Bezier Curve

Here,

- This curve is defined by 4 control points b_0, b_1, b_2 and b_3.
- The degree of this curve is 3.
- So, it is a cubic Bezier curve.

Cubic Bezier Curve Equation:
The parametric equation of a Bezier curve is-

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t)$$

Bezier Curve Equation

Substituting n = 3 for a cubic Bezier curve, we get-

$$P(t) = \sum_{i=0}^{3} B_i J_{3,i}(t)$$

Expanding the above equation, we get-

$P(t) = B_0J_{3,0}(t) + B_1J_{3,1}(t) + B_2J_{3,2}(t) + B_3J_{3,3}(t)$ ……….(1)

Now,

$$J_{3,0}(t) = \frac{3!}{0!\,(3-0)!}\, t^0 (1-t)^{3-0}$$

$$J_{3,0}(t) = (1-t)^3 \qquad \text{............(2)}$$

$$J_{3,1}(t) = \frac{3!}{1!\,(3-1)!}\, t^1 (1-t)^{3-1}$$

$$J_{3,1}(t) = 3t(1-t)^2 \qquad \text{............(3)}$$

$$J_{3,2}(t) = \frac{3!}{2!\,(3-2)!}\, t^2 (1-t)^{3-2}$$

$$J_{3,2}(t) = 3t^2(1-t)^2 \qquad \text{............(4)}$$

$$J_{3,3}(t) = \frac{3!}{3!\,(3-3)!}\, t^3 (1-t)^{3-3}$$

$$J_{3,3}(t) = t^3 \qquad \text{............(5)}$$

Using (2), (3), (4) and (5) in (1), we get-

$$P(t) = B_0(1-t)^3 + B_1 3t(1-t)^2 + B_2 3t^2(1-t) + B_3 t^3$$

This is the required parametric equation for a cubic bezier curve.

Applications of Bezier Curves-
Bezier curves have their applications in the following fields-

a. **Computer Graphics:**
   - Bezier curves are widely used in computer graphics to model smooth curves.
   - The curve is completely contained in the convex hull of its control points.
   - So, the points can be graphically displayed & used to manipulate the curve intuitively.

b. **Animation:**
   - Bezier curves are used to outline movement in animation applications such as Adobe Flash and synfig.
   - The application creates the needed frames for the object to move along the path.
   - Users outline the wanted path in Bezier curves.
   - For 3D animation, Bezier curves are often used to define 3D paths as well as 2D curves.

c. **Fonts:**
   - True type fonts use composite Bezier curves composed of quadratic Bezier curves.
   - Modern imaging systems like postscript, asymptote etc use composite Bezier curves composed of cubic Bezier curves for drawing curved shapes.

EXAMPLE:

Given a bezier curve with 4 control points-
$B_0[1\ \ 0]$, $B_1[3\ \ 3]$, $B_2[6\ \ 3]$, $B_3[8\ \ 1]$
Determine any 5 points lying on the curve. Also, draw a rough sketch of the curve.

SOLUTION:

We have-

The given curve is defined by 4 control points.

So, the given curve is a cubic bezier curve.

The parametric equation for a cubic bezier curve is-

$$P(t) = B_0(1-t)^3 + B_1 3t(1-t)^2 + B_2 3t^2(1-t) + B_3 t^3$$

Substituting the control points B_0, B_1, B_2 and B_3, we get-

$P(t) = [1\ 0](1-t)^3 + [3\ 3]3t(1-t)^2 + [6\ 3]3t^2(1-t) + [8\ 1]t^3$ ……..(1)

Now,

To get 5 points lying on the curve, assume any 5 values of t lying in the range 0 <= t <= 1.

Let 5 values of t are 0, 0.2, 0.5, 0.7, 1

For t = 0:

Substituting t=0 in (1), we get-

$P(0) = [1\ 0](1-0)^3 + [3\ 3]3(0)(1-t)^2 + [6\ 3]3(0)^2(1-0) + [8\ 1](0)^3$

$P(0) = [1\ 0] + 0 + 0 + 0$

$P(0) = [1\ 0]$

For t = 0.2:

Substituting t=0.2 in (1), we get-

$P(0.2) = [1\ 0](1-0.2)^3 + [3\ 3]3(0.2)(1-0.2)^2 + [6\ 3]3(0.2)^2(1-0.2) + [8\ 1](0.2)^3$

$P(0.2) = [1\ 0](0.8)^3 + [3\ 3]3(0.2)(0.8)^2 + [6\ 3]3(0.2)^2(0.8) + [8\ 1](0.2)^3$

$P(0.2) = [1\ 0] \text{ x } 0.512 + [3\ 3] \text{ x } 3 \text{ x } 0.2 \text{ x } 0.64 + [6\ 3] \text{ x } 3 \text{ x } 0.04 \text{ x } 0.8 + [8\ 1] \text{ x } 0.008$

$P(0.2) = [1\ 0] \text{ x } 0.512 + [3\ 3] \text{ x } 0.384 + [6\ 3] \text{ x } 0.096 + [8\ 1] \text{ x } 0.008$

$P(0.2) = [0.512\ 0] + [1.152\ 1.152] + [0.576\ 0.288] + [0.064\ 0.008]$

$P(0.2) = [2.304\ 1.448]$

For t = 0.5:

Substituting t=0.5 in (1), we get-

$P(0.5) = [1\ 0](1-0.5)^3 + [3\ 3]3(0.5)(1-0.5)^2 + [6\ 3]3(0.5)^2(1-0.5) + [8\ 1](0.5)^3$

$P(0.5) = [1\ 0](0.5)^3 + [3\ 3]3(0.5)(0.5)^2 + [6\ 3]3(0.5)^2(0.5) + [8\ 1](0.5)^3$

$P(0.5) = [1\ 0] \text{ x } 0.125 + [3\ 3] \text{ x } 3 \text{ x } 0.5 \text{ x } 0.25 + [6\ 3] \text{ x } 3 \text{ x } 0.25 \text{ x } 0.5 + [8\ 1] \text{ x } 0.125$

$P(0.5) = [1\ 0] \text{ x } 0.125 + [3\ 3] \text{ x } 0.375 + [6\ 3] \text{ x } 0.375 + [8\ 1] \text{ x } 0.125$

$P(0.5) = [0.125\ 0] + [1.125\ 1.125] + [2.25\ 1.125] + [1\ 0.125]$

$P(0.5) = [4.5\ 2.375]$

For t = 0.7:

Substituting t=0.7 in (1), we get-

$P(t) = [1\ 0](1-t)^3 + [3\ 3]3t(1-t)^2 + [6\ 3]3t^2(1-t) + [8\ 1]t^3$

$P(0.7) = [1\ 0](1-0.7)^3 + [3\ 3]3(0.7)(1-0.7)^2 + [6\ 3]3(0.7)^2(1-0.7) + [8\ 1](0.7)^3$

$P(0.7) = [1\ 0](0.3)^3 + [3\ 3]3(0.7)(0.3)^2 + [6\ 3]3(0.7)^2(0.3) + [8\ 1](0.7)^3$

$P(0.7) = [1\ 0] \text{ x } 0.027 + [3\ 3] \text{ x } 3 \text{ x } 0.7 \text{ x } 0.09 + [6\ 3] \text{ x } 3 \text{ x } 0.49 \text{ x } 0.3 + [8\ 1] \text{ x } 0.343$

$P(0.7) = [1\ 0] \text{ x } 0.027 + [3\ 3] \text{ x } 0.189 + [6\ 3] \text{ x } 0.441 + [8\ 1] \text{ x } 0.343$

$P(0.7) = [0.027\ 0] + [0.567\ 0.567] + [2.646\ 1.323] + [2.744\ 0.343]$

$P(0.7) = [5.984\ 2.233]$

For t = 1:

Substituting t=1 in (1), we get-

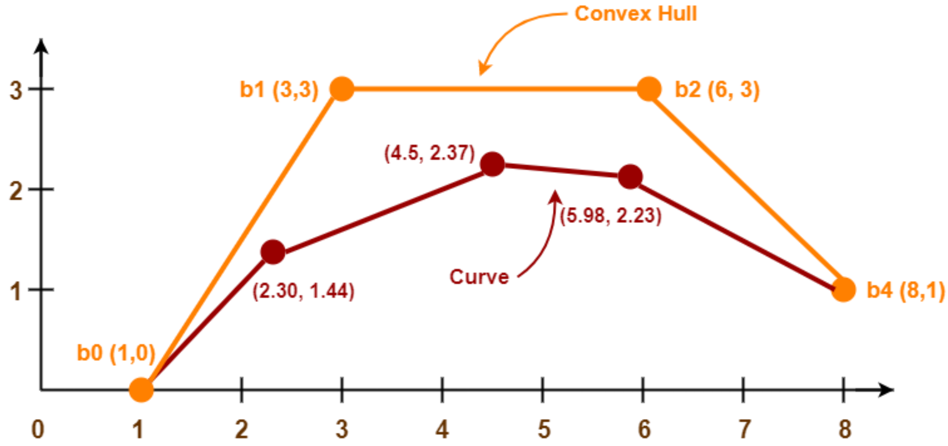
$P(1) = [1\ 0](1-1)^3 + [3\ 3]3(1)(1-1)^2 + [6\ 3]3(1)^2(1-1) + [8\ 1](1)^3$

$P(1) = [1\ 0] \times 0 + [3\ 3] \times 3 \times 1 \times 0 + [6\ 3] \times 3 \times 1 \times 0 + [8\ 1] \times 1$

$P(1) = 0 + 0 + 0 + [8\ 1]$

$P(1) = [8\ 1]$

Following is the required rough sketch of the curve-

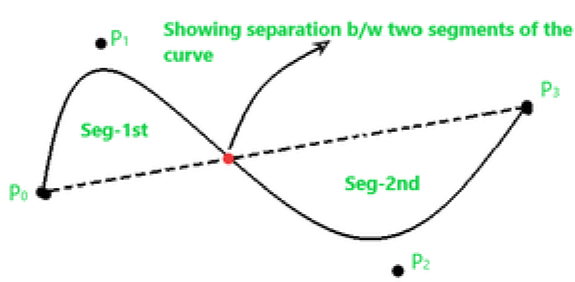


3. **B-Spline Curve:**

Concept of **B-spline** curve came to resolve the disadvantages having by **Bezier curve,** as we all know that both curves are parametric in nature. In Bezier curve we face a problem, when we change any of the control point respective location the whole curve shape gets change. But here in B-spline curve, the only a specific segment of the curve-shape gets changes or affected by the changing of the corresponding location of the control points.

In the **B-spline curve**, the control points impart local control over the curve-shape rather than the global control like **Bezier-curve.**

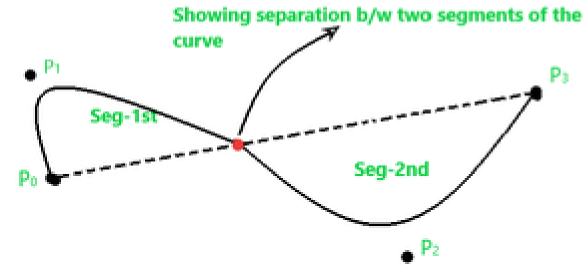
B-spline curve shape before changing the position of control point P_1 –



Control points { P0, P1, P2, P3}

B-spline curve shape after changing the position of control point P_1 –

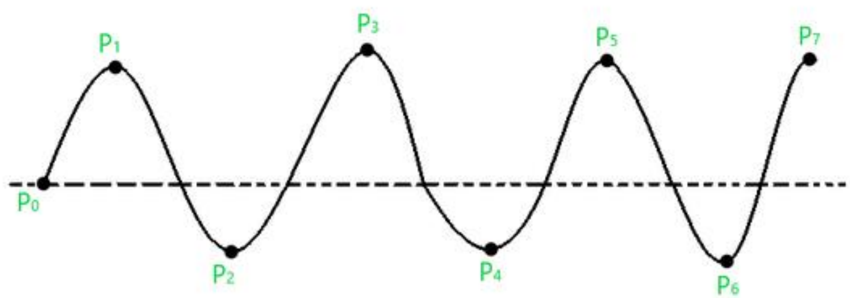
You can see in the above figure that only the **segment-1st** shape as we have only changed the control point P_1, and the shape of segment-2nd remains intact.



Control points { P0, P1, P2, P3}

B-Spline Curve:

As we see above that the B-splines curves are independent of the number of control points and made up of joining the several segments smoothly, where each segment shape is decided by some specific control points that come in that region of segment. Consider a curve given below –



Attributes of this curve are –

- We have "n+1" control points in the above, so, n+1=8, so n=7.
- Let's assume that the order of this curve is 'k', so the curve that we get will be of a polynomial degree of "k-1". Conventionally it's said that the value of 'k' must be in the range: $2 \leq k \leq n+1$. So, let us assume k=4, so the curve degree will be k-1 = 3.
- The total number of segments for this curve will be calculated through the following formula –

  Total no. of seg = n – k + 2 = 7 – 4 + 2 = 5.

Segments	Control points	Parameter
S_0	P_0,P_1,P_2,P_3	$0 \leq t \leq 2$
S_1	P_1,P_2,P_3,P_4	$2 \leq t \leq 3$
S_2	P_2,P_3,P_4,P_5	$3 \leq t \leq 4$

Segments	Control points	Parameter
S_3	P_3,P_4,P_5,P_6	$4 \le t \le 5$
S_4	P_4,P_5,P_6,P_7	$5 \le t \le 6$

Knots in B-spline Curve :

The point between two segments of a curve that joins each other such points are known as knots in **B-spline curve.** In the case of the cubic polynomial degree curve, the knots are **"n+4".** But in other common cases, we have **"n+k+1"** knots. So, for the above curve, the total knots vectors will be –

Total knots = n+k+1 = 7 + 4 + 1 = 12

These knot vectors could be of three types –

- Uniform (periodic)
- Open-Uniform
- Non-Uniform

B-spline Curve Equation : The equation of the spline-curve is as follows –

$$Q(t) = \sum_{i=0}^{n} Pi * N_{i,k}(t)$$

Where $N_{i,k}$ is the Basis function of B-spline curve.

Where **P_i, k, t** correspondingly represents the control points, degree, parameter of the curve.

$$N_{i,k}(t) = \frac{(t - x_i) * N_{i,k-1}(t)}{{x_{i+k-1}}^{-1}} + \frac{(x_{i+k} - t) * N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}}$$

And following are some conditions for $\mathbf{x_i}$ are as follows –

$$x_i = 0; if\ ik;$$

$$x_i = i - k + 1; if\ k \le i \le n$$

$$x_i = 0; if\ in$$

Some Cases of Basis Function:

$$N_{i,k}(t) = \begin{Bmatrix} 1;\ \ if\ x_i \le t \le x_{i+1} \\ 0;\ \ else \end{Bmatrix}$$

$$where\ t_{min} \le t \le t_{max}$$

Properties of B-spline Curve :

- Each basis function has 0 or +ve value for all parameters.

- Each basis function has one maximum value except for k=1.
- The degree of B-spline curve polynomial does not depend on the number of control points which makes it more reliable to use than Bezier curve.
- B-spline curve provides the local control through control points over each segment of the curve.
- The sum of basis functions for a given parameter is one.

Parametric Surface:

A **parametric surface** is a surface in the Euclidean space which is defined by a parametric equation with two parameters. Parametric representation is a very general way to specify a surface, as well as implicit representation. Surfaces that occur in two of the main theorems of vector calculus, Stokes' theorem and the divergence theorem, are frequently given in a parametric form. The curvature and arc length of curves on the surface, surface area, differential geometric invariants such as the first and second fundamental forms, Gaussian, mean, and principal curvatures can all be computed from a given parameterization.

Examples:

- The simplest type of parametric surfaces is given by the graphs of functions of two variables:

$$z = f(x, y), \quad r(x, y) = (x, y, f(x, y))$$

- A rational surface is a surface that admits parameterizations by a rational function. A rational surface is an algebraic surface. Given an algebraic surface, it is commonly easier to decide if it is rational than to compute its rational parameterization, if it exists.

- Surfaces of revolution give another important class of surfaces that can be easily parameterized. If the graph z = f(x), a ≤ x ≤ b is rotated about the z-axis then the resulting surface has a parameterization

$$r(u, \emptyset) = \big(u\, cos\emptyset, u\, sin\emptyset, f(u)\big), \quad a \le u \le b, \;\; 0 \le \emptyset \le 2\pi$$

Surface of Revolution:

A **surface of revolution** is a surface in Euclidean space created by rotating a curve (the generatrix) around an axis of rotation. Examples of surfaces of revolution generated by a straight line are cylindrical and conical surfaces depending on whether or not the line is parallel to the axis. A circle that is rotated around any diameter generates a sphere of which it is then a great circle, and if the circle is rotated around an axis that does not intersect the interior of a circle, then it generates a torus which does not intersect itself (a ring torus).

Properties:

The sections of the surface of revolution made by planes through the axis are called meridional sections. Any meridional section can be considered to be the generatrix in the plane determined by it and the axis.

The sections of the surface of revolution made by planes that are perpendicular to the axis are circles.

Some special cases of hyperboloids (of either one or two sheets) and elliptic paraboloids are surfaces of revolution. These may be identified as those quadratic surfaces all of whose cross sections perpendicular to the axis are circular.

Area Formula:

If the curve is described by the parametric functions x(t), y(t), with t ranging over some interval [a,b], and the axis of revolution is the y-axis, then the area Ay is given by the integral

$$A_y = 2\pi \int_a^b x(t) \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}\, dt,$$

provided that x(t) is never negative between the endpoints a and b. This formula is the calculus equivalent of Pappus's centroid theorem. The quantity

$$\sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}$$

comes from the Pythagorean theorem and represents a small segment of the arc of the curve, as in the arc length formula. The quantity $2\pi x(t)$ is the path of (the centroid of) this small segment, as required by Pappus' theorem.

Likewise, when the axis of rotation is the *x*-axis and provided that *y*(*t*) is never negative, the area is given by

$$A_x = 2\pi \int_a^b y(t) \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}\, dt.$$

If the continuous curve is described by the function y = f(x), a ≤ x ≤ b, then the integral becomes

$$A_x = 2\pi \int_a^b y\sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx = 2\pi \int_a^b f(x)\sqrt{1 + \left(f'(x)\right)^2}\, dx$$

for revolution around the x-axis, and

$$A_y = 2\pi \int_a^b x\sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx$$

for revolution around the y-axis (provided a ≥ 0). These come from the above formula.

For example, the spherical surface with unit radius is generated by the curve y(t) = sin(t), x(t) = cos(t), when t ranges over [0, π]. Its area is therefore

$$\begin{aligned} A &= 2\pi \int_0^{\pi} \sin(t)\sqrt{\left(\cos(t)\right)^2 + \left(\sin(t)\right)^2}\,dt \\ &= 2\pi \int_0^{\pi} \sin(t)\,dt \\ &= 4\pi. \end{aligned}$$

For the case of the spherical curve with radius r, y(x) = √r2 − x2 rotated about the x-axis

$$\begin{aligned} A &= 2\pi \int_{-r}^{r} \sqrt{r^2 - x^2}\,\sqrt{1 + \frac{x^2}{r^2 - x^2}}\,dx \\ &= 2\pi r \int_{-r}^{r} \sqrt{r^2 - x^2}\,\sqrt{\frac{1}{r^2 - x^2}}\,dx \\ &= 2\pi r \int_{-r}^{r} dx \\ &= 4\pi r^2 \end{aligned}$$

A minimal surface of revolution is the surface of revolution of the curve between two given points which minimizes surface area. A basic problem in the calculus of variations is finding the curve between two points that produces this minimal surface of revolution.

There are only two minimal surfaces of revolution (surfaces of revolution which are also minimal surfaces): the plane and the catenoid.

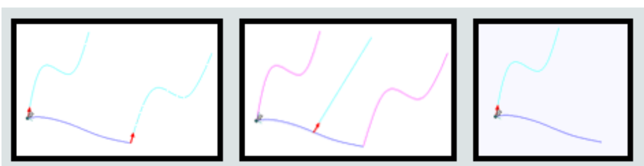
Applications:

The use of surfaces of revolution is essential in many fields in physics and engineering. When certain objects are designed digitally, revolutions like these can be used to determine surface area without the use of measuring the length and radius of the object being designed.

Sweep Surfaces:

Sweep Surfaces are surfaces that are generated from a section curve positioned along a path. Various Sweep types are available to extend the sweep definition as follows:

- ***Sweep by Two Guide Curves*** is defined as moving and adjusting a section curve along two guide curves.
- ***Bounded Sweep*** is created by moving a section curve along a guide path while bounding the section curve by two bounding curves.
- ***Fixed Sweep*** is a section curve moving along a guide path while fixing the orientation of the section curve.
- ***Parallel Sweep*** is defined as a section curve moving along a guide path while maintaining the section curve to be parallel to the guide path.



Two Guide Rails Sweep, Bounded Curve Sweep, and Parallel/Fixed Sweep

Quadric Surface:

Quadric surfaces are defined by quadratic equations in two dimensional space. Spheres and cones are examples of quadrics. The quadric surfaces of RenderMan are surfaces of revolution in which a finite curve in two dimensions is swept in three dimensional space about one axis to create a surface. A circle centered at the origin forms a sphere. If the circle is not centered at the origin, the circle sweeps out a torus. A line segment with one end lying on the axis of rotation forms a cone. A line segment parallel to the axis of rotation forms a cylinder. The generalization of a line segment creates a hyperboloid by rotating an arbitrary line segment in three dimensional space about the Z axis. The axis of rotation is always the z axis. Each quadric routine has a sweep parameter, specifying the angular extent to which the quadric is swept about z axis. Sweeping a quadric by less than 360 degrees leaves an open surface.

Quadrics

Many common shapes can be modeled with quadrics. Although it is possible to convert quadrics to patches, they are defined as primitives because special-purpose rendering programs render them directly and because their surface parameters are not necessarily preserved if they are converted to patches. Quadric primitives are particularly useful in solid and molecular modeling applications.

All the following quadrics are rotationally symmetric about the z axis. In all the quadrics *u* and *v* are assumed to run from 0 to 1. These primitives all define a bounded region on a quadric surface. It is not possible to define infinite quadrics. Note that each quadric is defined relative to the origin of the object coordinate system. To position them at another point or with their symmetry axis in another direction requires the use a modeling transformation. The geometric normal to the surface points ``outward" from the z-axis, if the *current orientation* matches the orientation of the *current transformation* and "inward" if they don't match. The sense of a quadric can be reversed by giving negative parameters. For example, giving a negative *thetamax* parameter in any of the following definitions will turn the quadric inside-out.

Each quadric has a *parameterlist*. This is a list of token-array pairs where each token is one of the standard geometric primitive variables or a variable which has been defined with **RiDeclare**. Position variables should not be given with quadrics. All angular arguments to these functions are given in degrees. The trigonometric functions used in their definitions are assumed to also accept angles in degrees.

Bilinear Surfaces:

In mathematics, **bilinear interpolation** is a method for interpolating functions of two variables (e.g., *x* and *y*) using repeated linear interpolation. It is usually applied to functions sampled on a 2D rectilinear grid, though it can be generalized to functions defined on the vertices of (a mesh of) arbitrary convex quadrilaterals.

Bilinear interpolation is performed using linear interpolation first in one direction, and then again in the other direction. Although each step is linear in the sampled values and in the position, the interpolation as a whole is not linear but rather quadratic in the sample location.

Bilinear interpolation is one of the basic resampling techniques in computer vision and image processing, where it is also called **bilinear filtering** or **bilinear texture mapping**.

B-spline surface:

The surface analogue of the B-spline surface (patch). This is a tensor product surface defined by topologically set of control points $p_{ij},\ 0 \le i \le m$ and two knots vector $U = (u_0, u_1, \dots\dots\dots.. u_{m+k})$ and $V = (v_0, v_1, \dots\dots\dots\dots. v_n)$ associated with each parameter u, v. The corresponding integral B-spline surface is given by

$$r(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} p_{ij} N_{i,k}(u) N_{j,l}(v)$$

Parametric line on a B-spline surface are obtained by letting u = const. a parametric line u = u_0 is a B-spline curve in v with V as a its knot vector and vertices q_j, $0 \le j \le n$ given by $q_j = \sum_{i=0}^{m} p_{ij} N_{i,m}(u_0)$.

Properties:

- Geometry invariance property
- End points geometric property
- Convex hull property
- B-spline to Bezier property

Bezier surfaces:

Bézier surfaces are a species of mathematical spline used in computer graphics, computer-aided design, and finite element modeling. As with Bézier curves, a Bézier surface is defined by a set of control points. Similar to interpolation in many respects, a key difference is that the surface does not, in general, pass through the central control points; rather, it is "stretched" toward them as though each were an attractive force. They are visually intuitive, and for many applications, mathematically convenient.

Bézier surfaces were first described in 1962 by the French engineer Pierre Bézier who used them to design automobile bodies. Bézier surfaces can be of any degree, but bicubic Bézier surfaces generally provide enough degrees of freedom for most applications.

A given Bézier surface of degree (n, m) is defined by a set of (n + 1)(m + 1) control points $k_{i,j}$ where i = 0, ..., n and j = 0, ..., m. It maps the unit square into a smooth-continuous surface embedded within the space containing the $k_{i,j}$ s – for example, if the $k_{i,j}$ s are all points in a four-dimensional space, then the surface will be within a four-dimensional space.

A two-dimensional Bézier surface can be defined as a parametric surface where the position of a point p as a function of the parametric coordinates u, v is given by:

$$\mathbf{p}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u)\, B_j^m(v)\, \mathbf{k}_{i,j}$$

evaluated over the unit square, where

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}$$

is a basis Bernstein polynomial, and

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

is a binomial coefficient.

Properties:

- A Bézier surface will transform in the same way as its control points under all linear transformations and translations.
- All u = constant and v = constant lines in the (u, v) space, and – in particular – all four edges of the deformed (u, v) unit square are Bézier curves.
- A Bézier surface will lie completely within the convex hull of its control points, and therefore also completely within the bounding box of its control points in any given Cartesian coordinate system.
- The points in the patch corresponding to the corners of the deformed unit square coincide with four of the control points.
- However, a Bézier surface does not generally pass through its other control points.

Generally, the most common use of Bézier surfaces is as nets of bicubic patches (where m = n = 3). The geometry of a single bicubic patch is thus completely defined by a set of 16 control points. These are typically linked up to form a B-spline surface in a similar way as Bézier curves are linked up to form a B-spline curve.

Simpler Bézier surfaces are formed from biquadratic patches (m = n = 2), or Bézier triangles.

Generalized Cylinders:

A ruled surface is called a generalized cylinder if it can be parameterized by x(u, v) = vp + y(u), where p is a fixed point. A generalized cylinder is a regular surface wherever $y' \times p \neq 0$. The above surface is a generalized cylinder over a cardioid. A generalized cylinder is a developable surface and is sometimes called a "cylindrical surface" (Kern and Bland 1948, p. 32) or "cylinder surface" (Harris and Stocker 1998, p. 102).

A generalized cylinder need not be closed (Kern and Bland 1948, p. 32).

Kern and Bland (1948, p. 32) define a cylinder as a solid bounded by a generalized cylinder and two parallel planes. However, when used without qualification, the term "cylinder" generally refers to the particular case of a right circular cylinder.

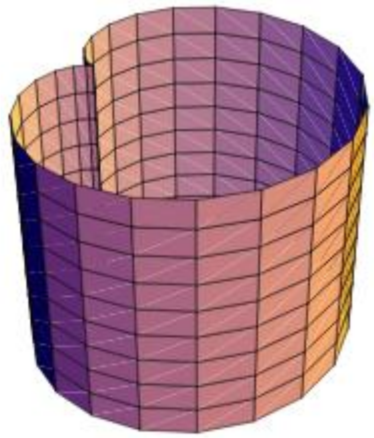


Fig: Generalized Cylinder

Generalized Cone:

A ruled surface is called a generalized cone if it can be parameterized by x(u, v) = p + v y(u), where p is a fixed point which can be regarded as the vertex of the cone. A generalized cone is a regular surface wherever $vy \times y' \neq 0$. The above surface is a generalized cone over a cardioid. A generalized cone is a developable surface and is sometimes called "conical surface."

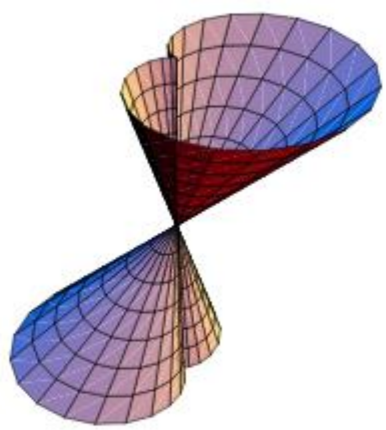


Fig: Generalized Cone

Polygon Mesh:

3D surfaces and solids can be approximated by a set of polygonal and line elements. Such surfaces are called polygonal meshes. In polygon mesh, each edge is shared by at most two polygons. The set of polygons or faces, together form the “skin” of the object.

This method can be used to represent a broad class of solids/surfaces in graphics. A polygonal mesh can be rendered using hidden surface removal algorithms. The polygon mesh can be represented by three ways –

- Explicit representation
- Pointers to a vertex list
- Pointers to an edge list

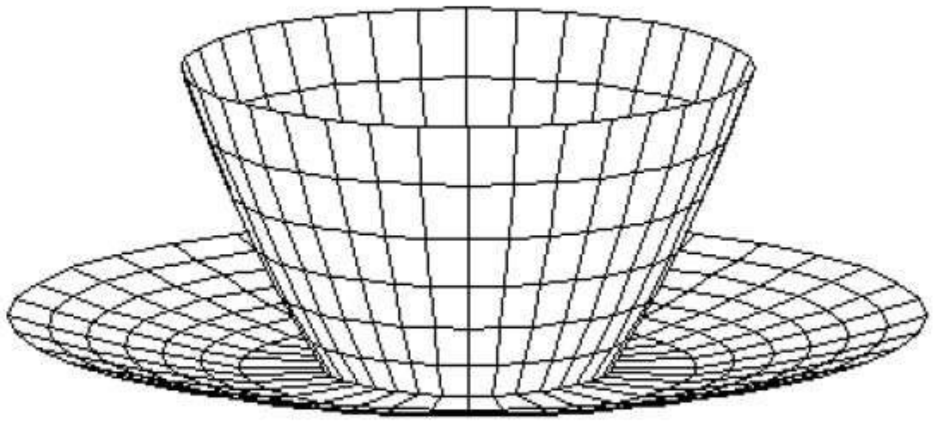


Fig: Polygon Mesh

Advantages

- It can be used to model almost any object.
- They are easy to represent as a collection of vertices.
- They are easy to transform.
- They are easy to draw on computer screen.

Disadvantages

- Curved surfaces can only be approximately described.
- It is difficult to simulate some type of objects like hair or liquid.

Wireframe:

It has a lot of other names also i.e.

1. Edge vertex models
2. Stick figure model
3. Polygonal net
4. Polygonal mesh
5. Visible line detection method

Wireframe model consists of vertex, edge (line) and polygons. Edge is used to join vertex. Polygon is a combination of edges and vertices. The edges can be straight or curved. This model is used to define computer models of parts, especially for computer-assisted drafting systems.

Wireframe models are Skelton of lines. Each line has two endpoints. The visibility or appearance or look of the surface can be should using wireframe. If any hidden section exists that will be removed or represented using dashed lines. For determining hidden surface, hidden lines methods or visible line methods are used.

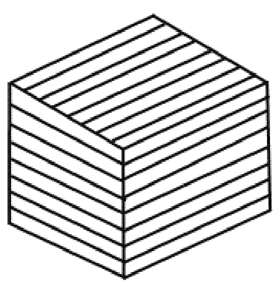
Advantage

1. It is simple and easy to create.
2. It requires little computer time for creation.
3. It requires a short computer memory, so the cost is reduced.
4. Wireframe provides accurate information about deficiencies of the surface.
5. It is suitable for engineering models composed of straight lines.
6. The clipping process in the wireframe model is also easy.
7. For realistic models having curved objects, roundness, smoothness is achieved.

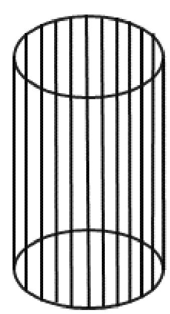
Disadvantage

1. It is given only information about the outlook if do not give any information about the complex part.
2. Due to the use of lines, the shape of the object lost in cluttering of lines.
3. Each straight line will be represented as collections of four fold lines using data points. So complexity will be increased.

Following figure shows wire frame model



(a) Wireframe of Block



(b) Wireframe model of cylinder